**SWOT Analysis with Notes**

To-do:

Add a logo to the doc

Add an intro

We are going to go through the whole thing for A and for B - help us separate it out and see where we are at

Eventually we may want to prioritize them - put the power statements and most important points first

We also want to do this without being repetitive - don’t have to keep everything - let’s add and remove as needed

Credit to John - he was a great help here

Create a small outline/layout - you can see quickly at a glance - a one pager - SWOT with bulleted lists underneath - let’s try to boil it down to a top 10 in each section

Then we can create a fuller version with deeper details - sections - bigger chapter type breaks between the sections

Step 1 was brainstorming and getting it into a doc, step 2 we organize it, step 3 refine it, step 4 polish it, step 5 finish it

Multiple layers at work here - are we dealing with current or future Adilas, are we dealing with our team or other people?? - keep breaking it into where things are at and where things are headed

Separate this out for Ship A (current) and Ship B (future)

**STRENGTHS**

Outsourcing servers to Data centers -

A & B

On the sort order... I'd really like to get the first few points to really be power statements.

Leave on business and server pages - do we have our own hosting services? what benefits do we gain by farming that out?

Uniqueness - we do not follow standard concepts that our competitors avoid -

A & B

This is both a strength and a weakness. I like the word uniqueness. We may need to change the verbiage about the standards. We do it the right way but we also spin it in our own way.

Smaller code base -

Could be A & B, more heading toward B

The new code base will have a smaller foot print and still be more dynamic and powerful. This could be on all three pages - biz, server, and dev.  
  
Shorter timeline to train new developers -

B side

biz and dev pages

Standard Package and willingness to offer custom code -

A & B - one of our big selling points - priority

leave on all three - if we want we can tweak it per page. This is part of our elevator pitch.  
  
All in one solution (POS, ERP, CRM, etc) without having to use multiple software packages (QuickBooks, POS, etc) -

A & B - priority

keep this on business. Systems (interconnected) vs independent mashups (smaller systems joined together)  
  
Pioneering since early web days (20+ years) -

A & B

biz page - solving pain points and listening to our clients. multiple versions under our belt.  
  
20 + years of records requested from clients -

A & B - change the verbiage here to more how we help people  
biz - need to change the verbiage. We have all kinds of lessons learned, features that have been requested, changes that were made, improvements, etc. Lots in the developers notebook - over 10K entries of what we have been doing and learning over time.

The name ADILAS (All Data is Live and Searchable) -

A & B  
biz - should be way up towards the top. Add some verbiage about the acronym of ADILAS

Service businesses of all sizes (mom & pop to enterprise/publicly traded customers) -

A & B - we are going to keep building a generic tool to service multiple industries and peoples. Value add-on core is the very next layer, we really want to ramp this up in B and the subsequent layers.

biz - maybe change customer to corporations or business entities.

support multiple industries that are highly regulated (MMJ, Firearms, Alcohol, Auto Dealerships) -

A & B, may want to change the verbiage here to say the things we have serviced, then with B we would like to go here …. We are pioneering some of these pieces of where we are headed - we can do it and we can do it well  
biz and dev sides - both strengths and weaknesses - we do it, but it also causes more changes. It really helps us have a niche but it also costs upkeep and maintenance costs.

Pricing -

A & B - this one constantly has some controversy, cover for A & B, this is one of our huge assets but it also has… Priority.

Investors will want a return, what do they get back. You could charge more or you could charge less and have more clients. Maybe different options are available for different investors. We think you could have both - all size of investors that want to play at different levels.  
should be on all of them - maybe cater the info to the section. it is on all of the swot columns. some clients think that we are too cheap, some love it, we get judged based on price. Yet, we can compete with almost anyone on price and win, if needed.

Educate clients/consultants about accounting -

A & B - this is a huge piece of what we do. The gap still exists and we are working to fill this gap. Possible priority. We have a relational tech model that allows us to fill this gap.   
biz - We help do over 75% of the accounting by just using the system. We educate the clients/consultants about the flow of the data. Excellent customer services - this could go above or below but part of the same thing. We take care of the customer

Working/Live prototypes that solve real-world business problems -

A & B - we do this right now, want to change the verbiage, break that into those different sections - it works right now! Even if it look old, basically no one complains about the functionality. Everything we want for B we are prototyping right now.   
biz - this is one of our biggest assets. We have a fully working prototype. We really do and have expirimented with tons of functionality and almost everything that we want to do on the next round, has at least a few places where we have done it in the current model. The current model just doesn't have those same features and options built throughout the entire application like we are planning. Basically, taking the best of class from the current and spreading it all around and making it even better and more integrated into the system.

$7.8+ Million in revenue since 2001 with projections to surpass 8M by 2024 -

A & B - we’ve been around way before this was cool. We’ve been out pioneering and pushing on this for a long time.   
Other stats - We have over 200+ active clients. Over 83K reoccurring per month. We have 12 our top clients that have paid over $100K to use our system. Our top client has paid over $300K to use the system. We clear over a 1M per year.

-------- Start back here --------- Of gaps and gains, sometimes we measure ourselves inbetween the gaps, but the gains help you look back to where you started - the gain says where did you start? Are you making progress? If you measure the gain and where you started - then you can really see the progress.

Minimal Debt as compared to competitors who are millions in debt -

A currently very minimal debt, B we want to plan so there is minimal debt for what is to come in B as well - we want investors. The debt is both a strength and a weakness,   
biz - strength and a threat - it's hard to compete with some of our competitors on how much they are willing to outlay in order to bring a client on. If we are limited by funding, it may take us longer to produce certain features. However, we are burdened down by debt.

Dedicated long term staff -

A currently what we have. B will probably bring many of these people forward because of how we are setup. Both A & B portions.  
Maybe under all three - biz, server, and dev - change as needed.

Employee/Co-owned company (MMLLC) -

A, possibilities for B - lots of possible options for how B might be, EOT 10235 has some different ideas on this subject  
Could go under all three - if you own something, ideally you take better care of it.

Multiple versions of the system(classic, snow owl) -

Need more from Brandon - A side has a bunch of different existing ones, B is going toward fracture and white label options, we are hoping the system will go more like the intel chip and then any sort of company could use this technology. Adilas Features and Versions - is an Excel file that shows how things have been added over the years and how Adilas has grown. We have some simple things listed out in our databases as well. One great strength is that Steve and Brandon are still here. That can be so helpful because they know why they started where they did, why they added what they added. That is an amazing resource that the creators and still here to help us understand the manner and form of creation. A pro and a con that we have pioneered and built this up from the beginning to solve needs. Slow steps to solve needs. Someone now will come and just cherry pick what they want from the system but the method of our building was to solve the next problem. We have knowledge. We have dealt with so many different things, so many different industries, it is so diverse. We have some cool data, we have some amazing success stories, we have helped some companies grow to the point where they could be sold for millions.

We have a spreadsheet with some breakdowns of what was developed and when. Brandon has that. Here is a URL: [https://www.adilascontent.biz/top\_secret/images/corp748\_EA81FBBDE6/pdf/doc\_25878\_A5D9A46913.xlsx](https://www.adilascontent.biz/top_secret/images/corp748_EA81FBBDE6/pdf/doc_25878_A5D9A46913.xlsx" \t "_blank)  
  
Since inception there has been a positive revenue curve (YOY increased revenue) -

This is A. B would be building a new curve, it would have its own projections but we are improving from something that already works. We would try to show the projections here.

Here is a simple PDF of the revenue - [https://www.adilascontent.biz/top\_secret/images/corp748\_EA81FBBDE6/pdf/doc\_171058\_2F81EBB8CC.pdf](https://www.adilascontent.biz/top_secret/images/corp748_EA81FBBDE6/pdf/doc_171058_2F81EBB8CC.pdf" \t "_blank)  
  
50 page document that explains what Adilas can do -

This was an outline form for the presentation gallery. This was one portion of the whole application. We believe this is the Business Functions portion. There is an A side and a B side to this too.

Here is a link to this outline - [https://data0.adilas.biz/adilas\_presentation\_gallery.pdf](https://data0.adilas.biz/adilas_presentation_gallery.pdf" \t "_blank) - also, here is the link to the presentation gallery (visual for the outline) - [https://news.adilas.biz/sales-gallery/adilas-advert-index.html](https://news.adilas.biz/sales-gallery/adilas-advert-index.html" \t "_blank)  
  
Customers have been using Adilas consistently since 2008 -

biz - Some of our biggest or largest paying customers have generated over $380K+ in reoccurring revenue for our company. We have other clients that have been paying since 2008 and are still actively using the system.

12+ clients have spent over $100,000 -

What is trying to be said here is that there are many companies that have paid hundreds of thousands of dollars just to keep using our system. There is a great amount of value.

Link to a screenshot - [https://data0.adilas.biz/top\_secret/images/corp748\_EA81FBBDE6/large/time-10222-1.jpg](https://data0.adilas.biz/top_secret/images/corp748_EA81FBBDE6/large/time-10222-1.jpg" \t "_blank)  
  
Satisfied clients year over year -

This is A. We believe that many of our existing clients would move to B and we would have good retention. We have people that have been waiting for this.

We did some quick queries - we found that over 84 clients have had and paid over 120 invoices (10 years x 12 months = 120 periods or invoices) - this assumes one invoice per month for the 10 years. If needed, here is the SQL query that we ran. select count(\*) as inv\_count, customer\_id from invoices\_22 group by customer\_id having count(\*) > 120 order by customer\_id  
  
Juicy data - needs to be re written!! -

Need to be careful here. There is a target on this and it needs to be protected. This is what people have with us that is of value.

We have huge data analysis companys that use our API's for their clients to pull data from - We hold all of the transactional data. We would love to get it out to a BI - business intelligence level (aggregates, counts, sums, averages, maxes, mins, etc.). We have the transactions, we just need to aggregate the data and we'd have it all. Small side note, we had a client one time that offered to buy 300 licenses for some convenience stores, if they could get the data. They were willing to pay, they just wanted the data. That is worth something.

Multi corp and enterprise grade solution -

In our value add-on core this is the farthest ring out. This is a B side - to come plan.

Needs to be on all three - almost a world building level with transactional worlds and enterprise or roll-up, roll-down, levels. This is partially built out and more is planned in the value add-on core model. See this URL for an image - [https://data0.adilas.biz/adilas\_for\_business/images/photo\_gallery/ideas\_plans/value\_add\_on\_core\_model.jpg](https://data0.adilas.biz/adilas_for_business/images/photo_gallery/ideas_plans/value_add_on_core_model.jpg" \t "_blank)  
  
Storing various types of documents -

Side A already does store documents, but we keep needing to up and up our storage. We are storing more and more content. This is something we forsee could bring more revenue on the B side.

Photos, images, scans, media/content, files, and paperless office. Tons of options here.  
  
No contracts -

That is a strength of A, it has bit us a few times. We have lost some people this way but also gained clients this way. The plan is to continue this with B. When there is a detailed contract it can actually give people somewhere to stand on to fight things. We like having no contract, straightforward month by month.

We have had some that worry about this but most clients really like it. On the down side, we can't count on anything long-term. On the positive side, it help with sales and people feel like they can try it out easier, without committing the farm. Low pressure sales. This is a different side to this, when we first started, our attourney said you either need a bomb proof contract or don't give them anything to stand on. If they have nothing to stand on, they can't really sue you for anything. This needs help - verbiage but the concept is - if they aren't happy, we're not happy or we have to make it good so that they will keep paying. If we had a pain in the but customer - we could drop or fire them, if needed.

We listen to feedback from clients and are proactive to change the system to accommodate those requests -

This has been a strength and we want to keep building and harvesting this. It is still our goal to create this community and invest in different pieces they are interested in. We have been an idea farm for years and we want to keep improving on this. We want to create a community.

Add this to all 3 sections - biz, server, and dev. This is both a threat and a weakness on the dev side. Sometimes it can create a can of worms or we change something and then people want it the way it was. We've had multiple run in with this. It is awesome that we listen, but we have gotten in trouble with it. The main solution has been to add settings and let the settings control changes.  
  
Subject matter experts -

We’ve been in this thing a long time and we have so many clients that are an invaluable resource and expert on using this system. This is also part of sponsoring, who can train on certain things, who could we recruit as subject matter experts to help us with certain pieces. This is a future thing in B we would like to see.

We would love to go to business leaders and industry experts and utilize their expertise and experience to help build the system. That would be so helpful and handy and add so much value to what is built.

People will cherry pick what we have, which is great. We are trying to create enough of a flavor that they will want to partake.

We've been on this project since 2001. We've been around the ball park a few times. We love talking with and working with industry masters. On a different note, we have also seen that power users inside of adilas have really built up a strong expertise in how things work and flow, clear to the accounting side of things.  
  
Clients push the team to accommodate their business needs (tech support, coding requirements) -

We allow people to ask for custom. It is part of the value add-on core but it is going to become even more in B as we have more and more settings that allow people to create what they want.

Similar to # 30, listen to feedback. We love it when a client asks for something. Sometimes we can't see what they need and it allows them to at least reach out and ask. That tends to start a good discussion. This may be something dealing with custom code and listening to user/client feedback. Maybe group #30, #32, and custom code together on a new line. This tends to generate revenue - yes, we can do that, it will cost x.  
  
Just in time planning and project management -

This is a strength for A. But we would really like this to expand for B. We hope that we can get things planned out more and in place and then we can better work toward the bigger plan instead of just one little piece, or petal, at a time.

One of the benefits of this for A has been to save some major time. We can fit it in, we can retrofit it. This is not all positive though, it has really cost us some times. We have plans that B will be more structured in this area.

This is a strength and a weakness. it is very fast but sometimes the guys need more than just a quick one-liner or an hour with someone who knows what the project is about. John was talking about a triangle of cost, quality, and speed. On the strength side, reword it to say something like nimble and flexible project management. Positive spin on it. On the weakness side, maybe leave it at the just in time level.  
  
Tons of data and information -

That exists for A, it will exist for B. We do not have plans to own, or harvest the data still. We have told our clients that it is there data and we have honored that. That is still the plan for B.

The data is so important - that is absolutely one of the greatest strengths.

Strength and a threat - the bigger you get and more data you are holding, the bigger target you become as well. On a security level - we do hash sensitive data for storage. At a future point, we would like to take it to a full encryption/decryption protocol.  
  
Cutting edge -

We were cutting edge as we came up but we have fallen behind. With B we are chasing that next cutting edge. The cutting edge just keeps changing. Planning to go to the next level of cutting edge with B.

Cutting edge can cut both ways. It can cut through barriers but it could also cut us as well. We may have to spin the verbiage a bit to make it say what we want.

Passion to help create an all in one solution (not just a 9-5 job) -

We want to do this because we believe in these ideas, this is a dream. We have wanted to build and create this system that is an all in one solution. This has helped build A. But think of where this could go for B. We just have so many dreams of where the next level could go.

This applies to our internal team. Maybe add the word internal or other light tweaks.

Are there any other strengths we need to cover or that we missed????

There may be some other valuable strengths we want to cover or approach in a different manner.

**WEAKNESSES**

Lack of communication channels due to budget constraints -

Definitely a weakness of A, especially as we grew beyond the initial small team. Heading forward to B we would really love to acknowledge the weakness but we want to find solutions to overcome the weaknesses. Dedicate our resources to help the weak places more.

We have a remote team and certain communications can struggle, at times. Everything isn't broken or bad, but we do struggle some times.

Support multiple industries that are highly regulated (MMJ, Firearms, Alcohol, Auto Dealerships) -

This was also on the strength side but also here. Some of these industries have really high demands and some people don’t want to touch these because they are so demanding. Going forward to B we really want to have industry specific skins so that it is easier and more straight forward to deal with industry needs.

See # 12 for the strengths. Certain industries are in a constant flux and thus require more attention. We are at the mercy of those who are changing things - regs, government, states, etc. If regs change, we could be out of compliance pretty easily. Most of the time our client let us know.

Pricing -

This has been a weakness for us. The companies that use our system love the rates they get from us. But we have struggled to bring in the revenue we need or that matches the value of what people are getting. We tend to get the short end of the stick. It has been a weakness but at times we have been able to turn it into a strength.

We have plans for B to have more structure in pricing and more specific ways to track usage and gather revenue from clients and users. Different versions, different prices, pay for your features, or certain features exist at different levels or layers so that you can choose the features you need. You would have to pay for the product at the feature level you want. Brandon and Alan have created some documents in this regard. It has a date of 7/5/23 of fracture planning with Alan. Has some planning on what could be done at what levels of the product you have. Still planning to tie it to actual prices but there is progress and planning here.

See #13 for the strength. Currently, there is no way to do price increases other than manual changes. The price of everything keeps going up. It would be good to set some standards and then play along with that. We get judged hard on this... sometimes we come across as being so cheap, we can't do what we are saying.  
  
Lack of a competitive marketing budget -

A currently has a $0 marketing budget. It just is what it is, and has been. Marketing is expensive and has a need for more people. So on one hand we have just been so busy building that we keep pushing and refining we don’t take the time to stop for marketing.

B going forward we want to for sure allocate a certain marketing budget. Run campaigns, or adds, we don’t really know exactly what format this might be in. We are good at building, we want to team up with someone else who is a good promoter. We pioneer, we prototype, we build but we need help with marketing. We may create commissions or campaigns that can have returns for help with marketing.

This whole thing has been done through word of mouth so far. That only goes so far. It worked great until we started competing with bigger money (big corporations). We don't have a professional person just to do marketing. We've been using who we have and at whatever level they are at. It's been very slim thus far.

Legacy code -

A currently works, which is good. As we keep going and try to get other developers on, it is hard to add other people and get other help. There was no master plan, we were self taught, it has been hodge podge, build as we go. The strength is that we have a 100% working prototype for everything we want to build.

B would have a plan. It would also have standard code practices, procedures, sign-offs, checks. This would allow other developers to come on easier and we would be building in more specified ways.

It still works... but there may be better ways. Changes get tougher, scale of the application pages (thousands), lots of copy and paste and duplicating of code. DRY - Don't repeat yourself Database structure - unnormalized Mixed - logic and views Naming conventions on CFC's and methods - for example - assets, liabilities, security, search, maintenance, etc. It does really tell us what it is doing. Frameworks and dependencies Deprecated code and libraries Patches and updates - keeping things up to date - pulled from a CDN rather than hosting our own versions.

Depending on external code libraries -

Side A currently does have certain external libraries that we depend on. We pull those in from outside sources. That is a potential weakness that we have some of these dependencies.

B will probably remain the same. We could combat it by picking and choosing which dependencies are the most stable and helpful. There is really just no way around some dependencies unless you had unlimited resources, time and money, to build all your own things. The more external pieces you have the slower things run. Going forward there may be ways we can cache, or shortcut things, to help it run faster.

See # 41. Similar topics.

Function over form (look and feel) -

As a weakness, we have had more complaints about our look and feel more than anything else. We basically never get complaints about our functionality. We have so much functionality that most people don’t even use it all. But we chose form over function. Form was most important to us though some people would fault us on that, that it is not pretty, or new looking. This is how it went with A.

With B we would like to add look and feel to play throughout the whole thing. We want to add that piece and we plan to add that piece but not sacrifice any of our function. People judge us very harshly on what it looks like. In some ways we have been able to go as long as we have, maybe a blessing in disguise or camouflage, that people have discounted us because we have not been as pretty as people want. If people knew what we were doing and what function we had available we probably would have had a lot more competition.

So many of these well known products have used older practices but made things look so pretty and user friendly. It has been a mixed blessing in this regard for us. There has been a game going with this over the years.

This may need to be on the strength side as well. We have a lot of function. We need some more form (look and feel). It hard to do both at the same time. This has been a huge weakness on the sales side of things.

Complexity requires training (time and willingness) -

A we have to train people on the code side and the user side. We have to train everybody on all the pieces. We even need to train people on the value the system has and how much it can do. How to interact with it and how to get so much out of it. Once you start to get the generalities and the flavor of the application you can start to go through things much better. It is not as intuitive yet as we would like it to be.

With B, we are hoping that we can use the new look and feel to help create a more intuitive platform. To help people use and thrive more in this. We also are planning to change the code so that it follows better standards and that people will be able to participate on the coding side much easier as well.

Some of what we do is deep. That takes time and training. No way around that. In ship B, we would like to hide a lot of this in the future. Based on settings. We use Adobe ColdFusion. That is not being taught in schools. We have to train our developers or help them convert other skills into what is needed. More time and training. If we were starting all over... it may be PHP but we already know CF and it is working. Just talking about options. It seems that a form of JavaScript, CSS, HTML, and SQL are standard across the board.  
  
Allow co-owners to work remotely - somewhat decentralized system -

This is how we built A. We do save a lot of money without having to pay rent, internet, utilities, snacks, furniture. We save money by having people all over the place. But it can make communication far more difficult. You also lose some time on commute. If everyone is in one central spot you can see who is working, check in on things, track progress on projects. When everyone is remote you can have guys disappear for a month and re-emerge with a large coding bill and we have no idea what has been done. Having people in different time zones has sometimes been a plus and sometimes a drawback. Sometimes people have been able to help while others are sleeping but vice versa, sometimes needing to get ahold of someone can’t happen because they are sleeping when you need them.

We use to have a daily meeting, and when we switched to moving forward with B, we cut out that meeting entirely. This has caused our communication to go from bad to worse in some ways. We have no consistent communication channels and constructive ways to work together. We have spent a lot of money from not having good communication channels. This has really caused some of our planning and projects to suffer. We don’t have things mapped out, no plans made on one line plans, because we don’t have enough people to manage that. We don’t have enough people to manage this.

We really need to go to the next level and have more money to put toward more managers so that we can have more oversight on development, on planning, on project management. Currently we have one person that is kind of keeping this together at the level it is operating at. But we are not efficient or effective because we don’t have enough management or oversight on these things. Brandon gets pulled away to help these other guys. Who is in charge of code sign-offs? Who is charge of managing these inter-dependencies? How is this code going to affect other things? Who is doing what? It all really comes down to funding.

The very first project that needs to happen is the Adilas jellyfish model. We are all running around like chickens with our heads cut off. It is not a coordinated effort which makes us less efficient and able to handle projects well. If we keep doing it on our thing we may slowly get there. If someone came in from the outside they would immediately want to structure things and tighten down channels so that we could accomplish goals. Money doesn’t solve everything but we have to have more resources to be able to do the things we need.

We don’t have an actual leader who is making decisions on the things that need to happen and assigning people to watch over certain things. It is day by day, fly by the seat of your pants, put out the next fire. We have a lot of need to find solutions in this area. It is hard to prioritize what is most important and we are all so busy we just try to keep going. We really need a plan! And it is hard to make a plan when you don’t feel like you have the time and money to plan. We get tempted into seeing the next little step and then we build it. Then we keep walking that path and don’t sometimes see the bigger picture.

We need lines of communication and authority. Taking things to the next level. Take that to your supervisor, get this signed off, get authorization to move this to the next level. It gets a little crazy. There are levels of management but not the structure to really play at the next level right now. We were really good at certain pieces when we were playing in a smaller arena. But as we get bigger and bigger and things get bigger and bigger we haven’t adapted to play at that bigger level. We are still playing our small based game but we are playing in a much bigger arena. We don’t have documentation, we have backup, we don’t have other people that can take over other people’s roles or parts. We are all running so fast there is not a known structure, there are no known operating procedures. It has worked thus far but it depends how big or efficient we want to be. We all end up clipping our own wings in our own way because we don’t want to touch the pieces we don’t want to and thus far we have been able to get away with that. But it limits us in many other ways.

When things get rough and tough we tend to just sweep it under the rug and move on. We tend to just build the next little piece to move forward instead of planning, addressing the issues, etc. We trend to build the next little piece we can see ahead, or that people are asking for every time.

This approach has created some good things. But it also creates a debt that becomes insurmountable. These repetitive cycles go over and over again but how do you break out of these cycles in a way that makes it truly different, truly constructive, that truly helps you make the changes you want for the long-term game. There is another aspect of age as well. How long of a game do you want this to be? How fast do you want things to go? When does it change from building, learning, having fun, wanting to build to when this changes to this must produce or else. Decisions are being made but they are being made quickly and with what end in mind? What is the master plan we are building to? Do we have a master plan we are building to?

When you get to the survival, completely overwhelmed state, you do not care about what is the best long term plan. You are clawing and scratching for whatever will help you survive. It becomes survival living instead of long term goal planning and living.

See # 37 - lack of communication channels. Having said that, lots of who do this like it and don't really want to change that.

Biased about our own products - eyes half closed -

This is a true story, we have our eyes half closed. For A we didn’t even know where we were going, we just stumbled upon some stuff.

With B, we have now seen where A has gone. We now see where things go. We see that we have some things that no one else has so this keeps pushing us to move forward. But because we still keep moving forward with our eyes half closed we are rejecting some of the structure or pieces that would help us move forward to the next level or make our organization more productive and smoother.

These are notes from meeting with Jonathon Johnson, a business consultant about our business structure. We were talking about dysfunctions and trying to reinvent the wheel on different topics. Jonathan was using a word that kept hitting me in the face. It was "rejecting", meaning rejecting or not allowing certain primary functions to take place or do their jobs... We talked about rejecting structure, rejecting responsibility, rejecting proper pricing configuration, rejecting executive time management, and rejecting a coherent form (what, who are we).

Here is a link to this EOT concerning some business consulting notes:

[https://data0.adilas.biz/top\_secret/time\_web\_gallery.cfm?corp=748&id=5295](https://data0.adilas.biz/top_secret/time_web_gallery.cfm?corp=748&id=5295" \t "https://app.goto.com/meeting/_blank)

For B, we really want to make changes on this. We really want to look at those things we have been rejecting, or discounting, and make them intentional pieces of our company structure and planning.

\*\*\*Gap and gain scenario - if you measure in the gap you measure what still is lacking. That is a hard thing and it hurts. If you measure the gain, then you measure from where you started and see how far you’ve come. We would like to measure off the gain because that helps our perspective and hope to keep moving forward. There are problems but we cannot focus all on the gaps, it almost leaves us all immobilized. We have to find the right balance of this. We don’t want to deny that there are gaps. But we do want to focus on the gains. There are good things happening and focusing here can help us move forward to the next steps.\*\*\*

We get drunk on progression and progress... We don't really get exposed to other competitors and their products. Most of the developers live in a cave. Since touchscreen, things have been changing so fast. Marketing and conventions - seeing what is new and available Budgets make it hard to get around as well. Things can be pretty tight.

Sometimes we do things without asking if we should -

There is a lot of history of doing this for the good and bad.

Going forward, I don’t know if we will overcome this. There may still be some of this that goes on in B. But if we have a plan then this can help inform where we are going and does this particular thing fit in with our plan. Having a plan will help us so much with determining if we should or should not do whatever the next certain thing is.

This comes back to authority and who is making the call (a set or known leader). We prototype things and then just move on. We don't really market our products after we build them. We are builders and just keep going and going. We don't have a lot of sellers or salespersons among us. We may want to change the verbiage. Often we use our existing knowledge and just do it like we always have. What about code standards and keeping that up to date. We don't like to say no. We like saying yes. I'll bet you that we could do that... off we go. We don't really have a true R&D company. We just build it and keep moving. We need market research, market analysis, and other ways of getting relative feedback. John was saying, we do so many different verticals. We almost need market research and feedback per business vertical.  
  
Busy, busy, busy.… -

YES!!! Everyone is too busy! This is how we have been operating, A is what it is. We are so busy we don’t even market our own stuff, not even our new products and features that we build and create.

What we want for B is to: Slow down! Try to sell it. Document it. Train on it. Education and maintenance and often more important than new features. It is easy to just want the next little piece or thing. However are we maintaining them, are we educating people on them, are people even using them. If you are constantly putting in new features that makes some people excited until there are too many to maintain. Or it makes it more difficult or cumbersome to use what is there. Everyone likes things that are new but sometimes the things that are well maintained and that people are able to use, that is what is smooth and useable.

We wear a lot of hats. Understaffed.

Gathering IP from contractors who have left the company -

We have all of these resources out there done by different people. We have not had a systematic way to collect or keep all of the things created for us as our assets. It is stored all over the place - on personal computers, different versions, different laptops, etc. How do we get this intellectual property as a resource, stored in an accessible environment, kept as a resource and asset of our company.

With B, one thing we could do knowing that this has been a weakness, is set this up in a standard practice. Push all projects into a centralized repository. This includes code, graphics, documents, all of the products created in the building and construction of Adilas. We have to have someone to manage and oversee this. Or it needs to be setup as part of the standard process so that the company has access and rights to all of the resources and the ability to access and retrieve any desired pieces. We still want to protect through sharing, but we do want to standardize where it is all kept so that we have access to it and can organize everything that we have.

We do have things (assets) spread around right now. We could use media/content to gather and organize things. Not only graphics, logos, images, and code - but we also have outside systems, 3rd party tools, and login stuff. Missing some standards and what is used for what. For example: vista print, banners, biz cards, stickers, promo items, etc. It was all done by individuals without us getting all of those pieces back in one place or spot. Training materials, SOP's, shortcuts, code snippets, etc. We are trying to build up the adilas docs and maybe that would be a good spot to put some of this info and data.

Cross training of co owners - (Subject matter experts) -

Cross training has not happened very much in A. Right now all of the knowledge exists in certain people or parties. We have a lot of knowledge within individual people, it just resides where it resides. But it is not a global asset right now or accessible to other people. It is limited in existing within that one person, or that person’s sphere or influence.

Where we would like to take this weakness of A and fight against this difficulty. We think having a plan for B will give everyone a broader view. We also want it be that when we sign off on new features or code, there will be a training as it is completed. We will have procedures, or documents of features shared. Gathering of the resources and skills that people have on the different pieces and making them into reusable resources and global assets instead of single user skills or assets.

This seems like all three biz, server, and dev topics. Just a general lack of education and training. On the server side, Wayne holds most of those keys. Nobody else knows what to do. Having said that, John is expected to support things without a full knowledge of subject. We may have to find someone who knows it - different subjects and topics. We don't have a list of what we use... We use x, y, and z - we don't really have that list. We all live on our own island, in a way. We only have a few that are full stack developers. The rest of us just fill in and use what we know. Sprints and scrum stuff could help solve this. Most of our current projects are single person projects. Lots of levels - server, backend, database, API sockets, frontend, general devs, tech support, etc. Almost the adilas Jelly Fish model (corporate structure). We tend to develop for a system that we don't use as a normal user. Some of that knowledge exists in or with a few key players.  
  
Such a diverse client base that hinders building pieces of the system to 100% support different verticals -

Lots of software companies choose one subject or industry and then they knock it out of the park. We are general and have been general. We do things so diversely that people complain. They wonder why they have build certain things they think should just be there for their industry.

With B we think that a lot of this will be solved in industry specific skins. How can we make this easier for specific industries? Once we make industry specific skins we can really hone in on pieces that that particular industry uses and is relevant to their world. So many people don’t even want to see or know that pieces are available that are not useful to their industry. We can show/hide whatever we need with industry specific skins and settings. We can create a flavor that fits for each person or industry that wants to participate.

Change such to such - This is a pro and a con. Because it is so diverse, certain clients get things that normally they wouldn't have access to. On the reverse, because it is not specific to a certain industry, naming, flow, and having to make do with something that was developed with a different purpose end up being the norm. We are really good at making general tools that can be configured and used in various ways. As far as the 100% complete levels - it might be good to go through each section and talk MVP and what percentage are we done/complete with certain pieces.  
  
Adilas is its own entity (Large up-front training when bringing on a new developer/salesperson/marketing) -

This deals with how diverse we are. We know the path that we have done but it is not standard or well known. A just exists, it has been built on the fly.

With B, we really want to show what we have and educate on this. If you like features you can turn them on and use them. If you don’t like or want certain features we can hide or stack it away in a shed of sorts. The ability to let them still use any feature that they want, it is all still adaptable, but you can see only what you want. Almost like a buffet line with access to a kitchen as well. It is beyond just being done and prepped and pick your pieces. There are more options than that, you can build and create with the pieces we have.

If we are using more standardized codes or languages we also want to code it in a way that we have more API sockets. More modular. Computers talking to computers instead of so hard coded. We had a company who skipped our entire interface but used all of our database, our relationships, our API, our functionality. They were able to use all the features and construct what they wanted and created their own creation. (Fly Hi)

We have our own verbiage, nomenclature, and vocab. We have the client side. We can start them out slowly and add as they are ready. On the dev side, it gets pretty deep and it's hard to just do a fully standalone project without it touching other pieces of the puzzle. On the marketing and sales side - where do we focus? That is hard to say. Not all of the assets are in one place either - they are all spread out. Just an idea from John - we may need to developer marketing materials for each sub set of system. People and material for POS's, people and material for CRM stuff, ERP stuff, scheduling, payroll, ecommerce, etc. Where is the best attack angle? We may need to do some exploring on this before we just jump. Research and making decisions.

There are so many options... maybe we could either do local options, sync up with the web, etc. A full local copy or backup. We could also gain by having someone work with a client and see what they actually need. That would be awesome. Figure it all out so that we could effectively market things.  
  
Employees are stretched due to the team size -

We read through the comments below and that pretty much captures the flavor of A. We have mainly had problems with structure, communication channels, picking up pieces, etc. Ship A was primarily just can we build it?

Ship B is okay, we know we can. What can we do to improve it, refine it, take it to the next level? What is its life cycle? It needs that next gen so that it doesn’t die. So many goals and dreams of where we want this to go. This is what fuels so much of this, this helps us to keep building and keep dreaming.

We didn’t know where we were going so it has been a hodge podge. Getting in and building and moving forward has helped build and create the vision. It has helped us see where this can go - getting the idea of the path and being able to visualize what can come next, then we can help people see the value of doing this. We know that some people would love it to be going faster but we are going to just have to keep chipping away at it. We don’t have the resources to pound it all out right now. There is also some value in time and some things need time to move to the next level or need to simmer.

Brandon’s goal is really to help plan this out, help people catch the vision, see the path/plan. Not necessarily be the primary programmer for building it all out.

That is a true story. Also, certain roles haven't been defined and so certain things tend to fall through the cracks. Currently, many of our internal team members wear multiple hats. We did make a graphic called the adilas jelly fish model - with some departments and light corporation structure ideas. It hasn't gone much further than that, as of mid June 2023. Part of our upcoming plan for fracture or adilas lite is to establish a more defined company structure. Based off of the jelly fish model. Here is a link to a document that we did internally - [https://www.adilascontent.biz/top\_secret/images/corp748\_EA81FBBDE6/pdf/doc\_145001\_6CEBC78C9D.docx](https://www.adilascontent.biz/top_secret/images/corp748_EA81FBBDE6/pdf/doc_145001_6CEBC78C9D.docx" \t "_blank) This limited team also tends to add pain points to our communication channels and cross training.

Clients push the team to accommodate their business needs (tech support, coding requirements) -

The notes really capture what has happened and happens. In A this is played such a huge role in how we got built. There was a need for us or a client and we filled the need. There was a pain and we found a solution for the pain. This process helped outline the environment and helped us grow into what we are today. A lot of this is operation led. If this is what happens in operations, how can we solve this problem, and how does that affect or play in to the financials. We did not set out to create a financial program, or an accounting program. But solving problems and following cause and effect is what has led us to the next solutions, led us to feed accounting.

With B we still anticipate this type of thing occurring with our company, advanced client requests. One we are still going to allow custom code to be created, second we are going to have and head toward industry specific skins, third we would really love to take these industry specific skins or pieces of the application to investors or sponsors, have people own this or sponsor and support it. Enough that the sponsors or investors are getting a residual revenue stream for helping promote and sell that piece of the product.

This is a strength and a weakness. Sometimes our customers really take advantage of us. Tech support becomes training without any fees or monetary switches. It can put strain on our tech support. It also makes us develop with more settings as not everyone will want a certain feature and/or will even use it. We end up building a foundation based off of permissions, settings, and templates. Many times we get compared to big, big, big boys in the market - IBM, Apple, Google, Amazon, Microsoft, etc. It is good that they want stuff and are using our products. It becomes a pain when we get forced to comply, without fair compensation.

Just in time planning and project management -

In A, originally Steve would come up with an idea, he would pitch it to Brandon and no one really cared when it was done. So we could plan out and figure out the whole things, we could figure out priorities, what is in front of us, what do we need or want to do. We had dedicated time to work on these specific items. We had really good efficiency at the beginning because we were building for ourselves and this just in time project management style worked really wonderfully. It has started failing as our team has grown. Now we also have so many clients that want things on a certain timeline, we have developers that need their hand to be held, that don’t have the ability to do certain pieces on their own. Or there is a lack of communication or …. We still have this mentality that we are tiny. If you look at our team size then we still do seem tiny. But if you look at our code we are NOT tiny. We have so many pieces, we are too big to keep playing at the level of the small team, just in time management we have always done.

The burnout has really happened and has cost us so much money. There are no benchmarks, stepping stones, thresholds, it doesn’t make ti to the finish line. This also affects developer’s confidence. They start to feel like they can’t accomplish any needful things, it can really start to affect other pieces because they get buried so deep.

Another stressful piece is that since no one is really managing this, this became Brandon’s role. So now he works with so many of the developers, there is a huge need here to fill, but it is not a direct money maker and it takes away from time on projects Brandon is supposed to be developing and coding for. If someone can be there helping to direct traffic, to keep things running smoothly, to help people know where to go next. But if it is not organized and run well then it really creates even MORE work in the end. It often takes more time and energy to clean up and help these things get back on track instead of investing more energy in the planning and outset of project managing.

We have also seen things where we build and build and build and because it wasn’t part of a master plan it maybe just sits on a shelf, it doesn’t get shared with our customer base, so others that could really benefit from it don’t get to use it because they don’t even know that it is there. That has been unfortunate to see for us and our clients.

We have some other notes on other pieces that help show what we plan for ship B. We plan to have more structure and resources allocated to help alleviate these pain points.

We can be super quick and nimble, but... that has bitten us before as well. Whether it is something doesn't fully get planned out, nobody knows how it gets connected to the whole, or we have different developers working on different pieces without proper communications. We end up spending a lot of time and money going back and saying, not quite right... Let's try this or that. We end up reworking things and sometimes totally missing the mark. We tend to build code that is needed, then reused, copy and paste style vs being in a nice function or method. Often we get a good one-liner - do x or y. We then have to plan it, get code resources, build it, test it, and deploy it. All off of a single one-liner. It makes better developers but it tends to cost a lot. Also, instead of breaking things into smaller pieces, we tend to give a big project to a single developer. That person is then booked or virtually taken for weeks and months on end. We have seen some burnout issues because of the lack of planning and super long development cycles - projects that never end or have no bounds set on them.

Just in time revenue -

A few things with A is that we started and we were scared to charge because we felt there were so many holes. Feeling like this wasn’t done and that wasn’t done so how could we charge a high price. Then once clients started it felt hard to up prices on them. Another thing about just in time revenue, we were doing well, then as we grew we needed more money to support things that weren’t directly bringing revenue in. Some developers have been pretty financially stable and have been able to float and roll with us if we need to pull back at times or gear up at times. But for developers that are primary bread winners that is really challenging to slim back or cut their hours and still expect them to be totally dedicated developers.

If you don’t have any reserves and you are just working with what you have coming in that week, you have no buffer, it all starts to feel like and emergency. It gets really stressful.

At times we have had budget meetings and at other times we haven’t had a focus here but it really needs to happen. There is a lot of value here and it will help us if we can do better on planning things out here.

We want certain things to happen but we have not really been willing to put certain things in place - that “rejecting” thing again. ;) We want it but it takes a lot. If we are stuck in just in time revenue, something we know will help us, often gets stuck to the side because we don’t have the resources to finish it.

Something that has been a strength has been that Steve has helped things get pushed out there that are mostly done and at least functional. Instead of getting it to a certain level that takes a lot more resources to get to that level, it has been a great strength to help just push things forward, take that risk and just work to move it to the next level. Very adaptable and using some of the strengths of that entrepreneurial mind and approach - we have been really good at this for a long time. But this is probably why it is so hard for us to change to the side that is becoming more and more needed, to have structure, to have plans, to have more boundaries so that we can successful in this bigger world and with a bigger product. We had a lot of fun when it was small, this is part of our style, rocking in this small to medium. Now that it is getting bigger, what do we really want? Do we want to make a plan and pass it off to someone else so that we can go back to a small/medium level? Or do we want to go to the next level? We really feel like we have been pushed in this direction. Is this the direction WE want though? Or is this just where we are getting pushed?

As you get more people involved you start to lose some of the influence you once had. Instead of being a leader and a key dreamer, vision maker, you become someone who just puts out fires, or just does the next reactive thing, instead of being able to dream and build and be on the offense. It all feels like being on the defense and that gets tiring.

This makes it really hard to budget and really set hard standards. This has affected teams, projects, future development, and often keeps the developer's on edge - the unknown. It often adds a strain on some of the services that we need internally to function as a business. Wage increases, bonuses, and even hours that are allowed and/or promised. One thing that we do is put whatever we can back into the system. That is awesome, but we don't really know how deep some of things are and everything just happens just in time. There are pros and cons to that. As a result, we've had to run slim at times. We try to stay minimum on the debt. Hard to market this product and really sustain it. The budgets are too weak and/or not sustained or planned out.

Money could really help us make better plans, dedicate people on certain tasks, put away a savings, and be able to roll better off of a cushion vs the just in time revenue model. Possible price increases, more sales, offer more services, debt, venture capital, investments, etc. We've done a great job getting it to here... it feels like it needs to change in order to get bigger. This is from a meeting with a business consultant - We were talking about dysfunctions and trying to reinvent the wheel on different topics. Jonathan (the consultant) was using a word that kept hitting me in the face. It was "rejecting", meaning rejecting or not allowing certain primary functions to take place or do their jobs... We talked about rejecting structure, rejecting responsibility, rejecting proper pricing configuration, rejecting executive time management, and rejecting a coherent form (what, who are we).

We do a lot of really good things and do things well. There are many places and things that we would like to do better and get a better handle on. The goal is improvement and change. We know that we have some weaknesses, let's see what we can do about them and go from there.

John would really like to see the morning meetings come back - to help with communication. A good leader, everybody knows what the plan is, small stand-up style reports, and keep it focused. Treat it as a business cost and put someone in charge of it.  
  
Knowledge transfer (in the minds of individuals rather than documented) -

We have built a generic platform and realized that everyone needs these business tools. But as we have done this, even more importantly, we have been learning lessons and learning principles that become even more important than the business tools. We are learning lessons and principles that every person needs and experiences in their life. There is more value in the principles and the lessons than there is in some of the exact tools or code. Sometimes it takes someone to challenge you or to have a specific reason that can help you learn and realize the knowledge and value that is really there.

Sometimes people want to be seen in just their best light but there comes a maturing where people realize they can be seen just as they are. You become more vulnerable and there is great value in being able to be seen as vulnerable in the appropriate settings. There is maturity in being okay with not being the super hero.

As far as A goes, knowledge transfer has primarily been from Steve and Brandon disseminating knowledge to other people. Steve helping to build up some of our best consultants. Brandon helping build up some of our best developers. With the budgets and team we had we feel we did a pretty decent job.

Moving forward to B we want to allocate a lot more resources here. Create a library to train and help people. Filter, search, send people to certain sections to really help them get to the things they want and need.

Another part of our plan in B is Adilas Cafe where people can contribute with their skills. A community based library and effort to help build and support the Adilas community.

This goes deep - on the business side, the server side, and the developer (code) level. It also goes for our clients. If they have a master power user and they leave, die, or get fired - the company is now in a tight spot. Some of the options for our clients - We take over and train them (with or without funding), they pay for a private consultant/trainer, or they leave us and go to a different product. We have some help files, but we don't really have a good learning environment. We would love to build out and make a thing called adilas university. We have to reverse engineer things. Nobody is locked into a certain role, we all get to make some decisions, but nobody is required to document anything and/or communicate to the other team members. We are literally all on our own little islands. It might be cool to help our clients build their own SOP (standard operating procedures) and/or help files. Currently, those are all on the side and not built in to the system.

Sadly, some of this comes down to money and structure and expectations. We all run as fast as we can right now. No one really thinks about what is coming next or what would happen if certain key players weren't part of the mix.  
  
Competitive wages and benefits -

Ship A we just started out on such a small micro budget, we tried to see how small and tight we could run. When you are trying to run so thin and low for long periods of time it has emotional, physical, and other ways it taxes you. It is not sustainable year after year, this causes lots of challenges and also burnout. Can you run thin? Maybe for a year or two, but how about 20? That hurts for a long time and really wears you down. Photo copy of a photo copy of a photo copy, you start to lose something of the original values.

We have this awesome product. We are scared to charge for it. We would love to keep it accessible for the mom and pop budget. So we would need way more clients to come on board but that means we would have to make it easier to use, easier for people to come on board. There is a small disconnect between these. Having said that if we got some major competition, we could even decrease our prices and live off of some of the byproducts. SAAS - software as a service, we are mainly doing the software part. Maybe we should be doing more of the service part - there is a lot of room to grow and a lot of abundance there. This ties into the Adilas Cafe, virtual market and services center.

For B our goal is to set up some structure, some bench marks, set up some plans and roll forward with those things. We have had lots of ideas, some even listed below, savings, revenue sharing - if we can set these up so they become official then we can start it. What do the members and owners want? Then we can move in that direction, even if it starts small. Consistency, stability, options, benefits, insurance, paid time off…. Everyone that works for Adilas LOVES it!! They have all expressed they love it more than the other jobs they have been working, or may currently be working at, they love the work - but the part that has been hard or makes them leave is the inconsistency and the concern of just in time revenue with the cycle it creates. We have had some great team members that leave because they feel they need to have the security and the stability even though they love the work. When it is tight there is too much stress and pressure.

It takes about 20,000 to float the team weekly, if we can keep double that in the bank then we can keep that lubed, that has helped it go better for the last little bit, that has helped it to not get as scary. Really has helped not to dip below some thresholds. This has really helped decrease the stress all around.

At some point we were so far behind we had to get an SBA loan and we have been able to maintain since then. We needed a bump to get out of that hole and it has felt like we have been a lot more stable for the past year and a half. Had to get to a threshold so we could create a buffer. If we have a buffer and a plan we could really start to do some of the next things, get investors, start creating some of these further opportunities and ideas.

Some of the most successful tech companies have the motto of - in order to be the best, you have to have the best - thus the competitive wages and benefits. This goes for developers, server/IT guys, marketing, project management, sales, tech support, admin/executives, etc. Just ideas... Take the monthly revenue and setup auto or reoccurring actions to put monies into savings, contingency plans, and co-owner distributions. Rates and percentages, and timing yet to be determined. Make some plans and then make it happen. John and I talked about education levels, high level skills, demand of/for those skills, available time, commitment levels, etc. What about setting standards for pay rates and/or salaries. We may need to set certain roles and specs per role. We then hire and/or place within those roles. We need a bit more structure. We may need to look at education levels, degrees, schooling. We also need to look at experience levels. We also need to look at relevance and other factors. If we have tons of turn over, we have to retain, retain, retain. That costs a lot of money and time. We've lost a few great people because they got better offers, options, salaries, benefits, etc.  
  
Too much data and information it can be hard to find -

What is the plan? Where are things going? If you don’t have a plan of where to put things they tend to end up in the junk drawer, or in the big pile where you heap all of the things that you don’t know where it goes. You need some planning and organizing but not so much structure and tightness that you lose the ability for some randomness and flexibility. Finding that right balance of structure.

With A we built as it came up and stashed it where we could and everything was not built with the idea to organize it. It was really, can we? Can we? Can we? There was not a thought of how does this all tie together we were just building what we could do. Now that we see what we have we really feel the need to organize more. As it gets bigger and bigger then organization really starts to matter. If you stay small then you don’t have as great of a need for organization. As we have gotten bigger we have begun to see more and more of our connections and realize that we really need to get organized with our people and things.

Also, some of the people that have been involved may have some knowledge about the pieces but no one else knows about it, or where it is. It is like so many things are hidden in secret places and it becomes inaccessible to others. We have come to a place where we need to mature to more structure and organization, a natural evolution of the process of growth from starting small. If you don’t have much then organization matters far less. Think of a really well organized gear shop versus an unorganized thrift store. It can be so overwhelming and hard to find anything in an unorganized thrift store - there could be some amazing deals but you can’t find anything or have any way to find it - really challenging and many people just won’t engage because it is just too challenging.

In software if you make something but it is not very obvious people will often give up - it becomes hard to find, people assume you don’t have it, they move on, there has to be a way to organize and put it away nicely if you don’t want it, but that it can be know, labeled, accessible if you do want to access it. This is part of our plan for B.

Dealing with all of this extra information - modern design has way more white space in it but it actually takes a lot more real estate that it takes up. It is really important for people to have it really pretty and cool but in some ways it makes it less efficient. Our biggest complaint has been look and feel. There is such a pressure to have it look new and modern, which keeps changing, so we will have to plan for tat too.

We have to separate the business logic from the display. There is something called MVC, model view controller - separating things into different entities. If I am the controller I am dealing with your data and your data access. The views can change often, like dressing a doll up into different outfits. This is related to object oriented programming. Ship A was linear programming. Object oriented programming you create standards, classes, sub classes, like a big giant tool box with multiple cubbies, organizers for data objects. Starting to organize it but you are creating objects. We didn’t even know we would have 12 main players. But now we could say we have 12 main players, they have all of these pieces, features. You could organize and get to all of the underlying logic and data that applies to this - this is related to the MVC. As far as the view this can change to so many different things, it becomes modular and very adaptable.

The reason we are trying to go to this next level is because we are trying to make it more consumable for others who are coming. How can we make it more organized, more accessible, more approachable, usable for all people. Start creating standards - like driving a car no matter what the car looks like or what the make and model are. This is why we need Adilas fracture and Adilas lite, everything you need is there but you can make or take it to whatever level you need. Tiny and easy, smoother, faster, pretty, so easy, etc.! So many options! Working to utilize the huge amount of tools and value that is already there.

One thing that we wanted to do eventually was have a standard package that we offer from Adilas. But if they want they can create their own interface, however limited to your tasks you want, create a mini app, but you could always fall back on to the baseline. Make your app as little or streamlined as you want, so many options here. Can use the powerful engine behind and create whatever you want to interact with. Underneath we can make so many things happen but there can be any shortcut apps or things created where you interact with or create data through a couple simple screens on an app. You could really shortcut a lot of things. One of millions of possibilities. If it is organized and detailed out enough we don’t care where the data comes from, just if you capture the needed info. Don’t even have to login, you can bypass all of that if you create an app that can make all of that simpler and designed to meet the client’s level. Ride on the backbone of this giant application. So many options and you wouldn’t have to build it from the ground up. Build applications off of our platform. Something created as a service and once it is documented and well known anyone can use it and play the game.

If you have a lot of stuff what are the pros and cons of that? We have so much that we have been able to help so many companies. We are a value, which has caused even more stuff to be added to keep taking on the next pieces. There is a lot of value in having all of the “stuff” we have because it really helps people with their needs. It is already being used very heavily and very hard. That is kind of a neat perspective and way to reflect on it. It does need organization and more accessibility but there is great value in what we do have.

We have tons of stuff... We just need to keep working on centralizing that data and then being able to search it, pull from it, and really use it. Currently, it is kinda spread out. It's there, just not all together which causes problems - time, energy, communications, clarity, etc. It would be nice if everything had it's place - like a well organized shop or garage. We have media/content, images, scans, developer's notebook, help files, raw files, code resources, etc. Time spent learning - some of our guys don't feel comfortable charging for things that they don't know but how do you learn it in order to be efficient? It would be awesome if we built in some time for training, experimenting, research, documentation, etc. Nobody likes to do free work on the system. This is from Wayne, but there is the work to get the work, the work before the work, the work, the work in between the work, the work after the work, and you get the idea. We need to plan for all of those pieces. There are a lot of settings, all over the place. Once again, too much data and information and it gets hard to find all of the pieces. Because we are not super organized (pre set or fully planned out) our application tends to have that same feel. Big companies spend big money to get their people all up to speed before they are allowed to really run with their responsibilities. They must see a value in that vs the money that gets spent to fix things after the fact.

Lack of testing and records of testing -

This is a known need, there are tons of options, it is going to come down to budgets, who is in charge of what and who can enforce it. There is a spectrum of how much our developers like testing their code, some are really great at it, some don’t test it as thoroughly. We also use our giant user base to be some of our testers on new projects - everyone uses it so differently that this becomes a huge strength for us. We could not pay for a testing crew to push as hard or as differently on our code as our users do. That is a huge asset. We have thousands and thousands of testers, they just don’t realize that is a role they are playing, they are called clients. It is also a weakness in its own way and at times that hurts a lot. But we do harness that usage of our clients. The downside is that clients sometimes feel frustrated or don’t like being the guinea pig per say. A double edged sword, they get an amazing product at a really cheap price but they contribute.

Not everything needs to be tested. There are things that are really interconnected and really need testing. But other pieces do not need to be tested as thoroughly. There really has to be testing all the way through building but unfortunately none of that is maintained. If we built tests, they could become repeatable, it is almost like a journal entry for testing what has been built. Once you get and establish your tests that can be great for recording and the computer can run that over and over and remember all of your pieces and scenarios. Especially when you involve all of your settings, building out tests can help us see if the path works because this leads you down different paths. Testing is really where we need to head.

Some people are so hard core that they will write their tests first - test driven design - before they even write the code. But that requires incredible project management and vision for what is going to be built. There is a balance that needs to happen between building and testing. We are not against testing but there needs to be a balanced option. Sometimes there is mocking and faking that feels like you don’t really know what is happening. It can also double the cost of the project right off the get go. That can make it go into the non-money contributing time for projects. That makes it really difficult. The flip-side is that sometimes we have developers create something and then we have to go back and fix all the mistakes and hold their hands through it which can also double the cost of the project and we have to foot that. There is value in testing but we feel there needs to be a balance because everything does not need to be covered or tested to that level.

We grew from an older background, Brandon was the original builder and had his style and technique for building. Now we have developers who use a lot of copy and paste and use minimal testing that it can create some real problems. People have different personalities and interpretations of this so it is maybe a bit more complex than just always have things tested.

The end goal would be a library of tests. You get into a certain part and then you can run all the tests for that area, if you pass, then you can have more confidence that what you developed will work well with the system. Eventually you have to have some one managing this and overseeing it, there has to be a standard and expected approach, someone has to make sure that all of these are set, standard, every developer has to go through these processes. This is a two-edged sword. There is more money and resources required to play at this level. There is a need for communication channels and expected procedures.

Another pro and con is that sometimes running testing over and over again can pollute the data. We had an experience where this happened where someone kept running their test and then added all the data to the system. Testing does have a value, it needs to be balanced, it needs to be managed. Expectations, budgets, etc. You are going to pay either way, is it in the front end with testing, or is it the backend with redoing and fixing code and mistakes. It would be nice to have better testing but still be able to use the incredible resource we have of our clients. If you go to either extreme on this subject you will suffer. By testing we mean documented testing. There is always some amount of testing that is necessary for building, so this is documented testing - it is good but more code. This is a tricky, challenging subject, you can’t do a blanket statement for what is needed.

Our current is something is better than nothing so we are running a hodge podge and we do some. We would love to do more here and if we bring on more developers we would have to employ more testing. The more developers you have the more need you have for testing. Too many cooks in the kitchen with no taste testers may result in some bad dishes - if the customers are your only taste testers you could have some problems. Sometimes as well those developers who get into testing tend to rely on it more and more, those that don’t know about it don’t really know what they are missing. So it really can reduce stress but it also has a cost.

When the system was first built, testing was only done by the developer on the project. As things get bigger, there is no way to remember what was tested, what to test, and how it did. The testing also flows into what else touches other parts of the system. Backend testing, frontend testing, and QC (quality control) and/or adilas super users (really use the product under different circumstances). Just a thought - if we had a plan that had a list of requirements, that would help with the testing specs, logic, decisions, and flow. It would help to define some standards and then fully play along those lines.  
  
Unnormalized database structure -

Any outside developer that comes in expects certain things or for certain things to do specific things. But we were self taught, kind of just did things on our own, we are also a general tool, so we have an unnormalized structure. This applies to how relational your databases are, this is really techy, there are things like first normalized, third normalized, etc.

Building from the ground up you just build. Now that we have what we have we could potentially go back and normalize things. We built the database as if it was a shelf. Databases really love to do all kinds of things but we really only used it as storage. It does what we want, we stick it in, we can pull it back out, etc. But aggregated data we have failed at doing, we did not build it to aggregate, to make some of these additional relationships that it can do. Databases can do all sorts of things. We would like to do some of this aggregating with our business intelligence plans. There really are so many possibilities and we have so much data that could be used to feed this. There is a real possibility here! We would love to go here and grow this area!

Some projects we have had multiple developers building on a project which can sometimes make for some detours and difficulties in the path. If we could smooth that out and clean up those divergences caused by the hodge podge planning or building, we could really open some further possibilities. Once again this can really depend on who is planning this, who is overseeing this, managing this.

The deeper your data and if you get into manufacturing, you get into more mini conversions, and things start to get less straight. So when we try to pull back all the data in certain ways or configurations it can be hard to pull back the data, because there is just so much! So if the path can get smoothed our or built better then you can have a greater data flow.

The internet is really built to limit records or flow, like a revolving door, so working with this on a cloud can make it hard to get all of the pieces. But of course clients want this. They want to be able to aggregate and get into certain parts and details. The data is all there but because it is running off of transactional data and it gets bigger and bigger that becomes a pain. You have to get into data warehouses and aggregates. Right now we use a relational database, but when you start going for speed and aggregates you on purpose break some of the relationships and dependencies so that you can speed up and group things. When you get to the next level you have to start changing some of the way it all interacts together. This is called ETL - extract, transform, load it back in.

To go to the next level you really need to know where you are going and have a plan. That is just part of the reason we feel the master plan is so important. To have some of these larger, overarching tools and abilities we have to plan better. We under use our databases right now, we could do a lot more here and it will take planning.

It may actually help to explain some of these things because people that don’t develop, and don’t understand the development world, may really benefit from gaining an understanding of how many pieces are actually involved in building or creating these certain things. Sometimes taking a little more time to illuminate the process or all the pieces really help people to see it is not as simple as they might suppose. A lot of pieces are more complex than people assume, or they touch a lot more things, etc. Big companies actually have database administrators. Right now all the developers just play a little part of this piece depending on what project they are working on or heading up. It is one thing to say can I do this. It is another thing to say what does this cascade into, what pieces does it touch and effect? Someone has to coordinate, oversee, and manage this stuff.

Just as an example we were talking about the tips project and how far it cascaded into financials, BS, statements, carts, invoices, deposits, eCommerce, A/R math, and more. The question seems so simple and easy, “Can we take tips?”. The project from that one question will touch over 40 pages and be over a $15,000 project. It seems like such a simple thing but the cost and effort to Adilas to create these additional features can blow up so huge, so quickly. Some of this stuff sounds so easy and it is so easy to say… But to actually build it and integrate it into the system is a huge amount of effort! For SO many things or pieces people want. Often people assume because it is easy to say or come up with the idea that it must be pretty easy to build it. But that is not the case. What is really wanted? How many pieces does it touch and need to be integrated with in the system? Who is going to pay for it? Clients really expect the company to have and provide so many of the pieces even though we are offering this incredible product at a fraction of the cost of other applications. Many, many challenges here. It ends up being stressful and frustrating at times for Adilas and for the clients.

If some of these things don’t seem to fit here we may need to find another spot for them to go. These are real problems and challenges that we face on a continual basis. We wanted to capture some notes on this point.

How do we fix this? There is not just one solution or part that will fix it all. We need a plan, we need additional resources, and there are a lot of pieces that would have to come into play to find a solution to these problems.

A huge strength of Steve is that he is willing to circle back to things and is willing to sell and use things where they are at. He can sell and share the virtues of things where they ARE at, at that moment. This has been such an incredible asset. He is willing to USE it even though it is not done, pretty, made up to the nines, and all packaged. He has been a great leader on this front and it is filling a need. It helps us see that there is a demand for this product and that people really want and need something like this. Also that people are willing to put up with a little bit of chaos to use this product at the prices we have. It is not a glamorous tool but it is a robust tool that is filling needs for people.

People with skills and that work at a certain level have certain expectations. High level developers sometimes look at our things like garbage because it is different, it is not according to expected, or their expected standards. We have found that people with more of the entrepreneur and business type mind can have more variables introduced and deal with the variables and things that don’t fit the exact form or expectations. Sometimes computer programmers that have great skills would want to throw our whole application to the curb because it doesn’t fit into the expectations they have. Programmers can tend to be more black and white thinkers. Even programming logic, you have to walk through if, then logic statements. You can’t have maybes, or unknowns, or more fuzzy logic? This tends to lead to more of the black and white type of thinking that can be a challenge when you are mixing and blending all the things that are happening in Adilas. We didn’t originally think of this as a weakness but we have heard this from our developers over and over again - we need to normalize our databases. Some of these developers will spend more time making their “tools”, than they would take to actually build the project. This is part of the mindset of some computer programmers. This becomes a real challenge when we have people with certain skills coming on and it becomes a real challenge when we mix this particular computer programming personality with the entrepreneur attitudes. Another personality conflict is that we have developers who comment on all the code, basically create a story to go along with it. While other developers never write comments on their code and believe they can make their code so clean and understandable that it doesn’t need any notes.

We really believe there are people that could come in and see what we have built here - keep in mind that it is a working product that is producing and bringing in revenue - and they would throw it all away and start over. You need to find the right people that see the potential and see what can be made with what we have. We didn’t have the plan from the outset, so it has produced a lot of challenges. But on the other side of the coin, it is working! People use it! It does so many things already and it has SO MUCH potential!

We believe if we have a plan, we could bring people on with both skill sets. Because if we have a plan, that can help developers with more of a conventional training or mindset to know how we can get to somewhere that they expect and want. But this is an interesting challenge.

As we get further into this and it gets bigger and bigger we look back at our decisions and realize we built things because it seemed like the easy choice or the solution that would work out. Steve and Brandon would just throw out ideas with each other and create the next step. Now it is so big that no one is steering the ship. This is becoming a bigger and bigger deal. No one is making the bigger decisions and overarching things as to the trajectory of Adilas. Management positions still get filled by people in some ways or roles. Like Brandon is still having developers that he helps and reach out to him to sign off and check code. Steve is still out in the field selling Adilas and actively helping clients. But no one is actively managing the direction of Adilas. We have had leadership roles that have fit the size and growth of Adilas to this point but now as we are trying to get bigger we need to adjust leadership styles or get some more directed leadership for things. We need some additional planning, direction, vision, those that are seeing where we want to go and figuring out the steps that will lead us to that destination. We need a plan!

We are all operating on the see a need and fill a need approach. Still chipping away at the dream and trying to keep the slow and steady wins the race mentality. We really need the bigger vision and plan though. We need those with the right vision that can help steer the ship to the destination.

It is a crushing role to put this all on one person but there is not a budget for multiple people to manage and spread and divide this work load. There is also no plan, we are just following requests from clients mostly. It’s amazing they are willing to jump in and play along side of us but there is no master plan. Then add to that a whole number of other hats and responsibilities - it is absolutely crushing to that person. There is no wonder that no one wants this position. It crushes everything and squashes all potential there.

Another challenge that we have discussed briefly in other places is how do we let people know where all these new features are. How do we let them know about what is available, train them, etc., etc.?? There are some interesting questions that go along with this as well. Some items that people want in their businesses also brings up the question of are we allowed to do that? Who holds the documentation for that? Who’s legal responsibility is that? Should we and could we do this?

Maybe a lack of training. We built this from a small system to a huge multi-tenant application. We tend to cram too much data into a single table (multi use). Possible data schema per client (per corp). Getting someone to really help be the database administrator role. Review, approvals, plans, etc. Standards, structure, communication, and reviews/approvals. We also need public/internal records of what is going on and how everything fits together.

Non clustered DB and server environment -

Just to define this, right now we have servers that have an instance of the application running on them. It is a single web server, cold fusion server, and a database server to run these. This is what you have to have on each of these just to run them. You can only serve so many clients per time with this setup. It has more limitations.

Imagine a robot with multiple hands compared to someone who just has two hands. The robot could facilitate so much more. This is what a cluster could accomplish, it could cluster and accomplish far more, more efficiently because the servers can work in tandem and can help share and manage bigger loads. It allows for better up time and heavy loads. But the downside is that you have to have your code created in a way that can work in a clustered manner. Here you have multiple things that are trying to serve the same thing - it’s like sharing the waiter’s responsibility at a restaurant. Wherever they are free they could serve and help but they have to keep the continuity enough that they would know what that customer needs/wants.

Clustering can grow big or small depending on what’s needed. But right now we are limited because many of our files or photos, or different things are stored on the particular server, so if that server were to be unavailable then you potentially could not access some of that corporation specific content and be able to use it in a clustered setting.

Normally when we talk about clustering we are talking about getting bigger. But the other side to this is that if you ever need to shrink or get smaller clustering is the solution to that as well. Clustering can allow you to grow bigger or smaller. Clustering allows you to mix databases, it allows you to be more flexible and adaptable. This means we could have the capacity to grow and handle more or that we could shrink if we need to shrink. It allows this needle to move either direction on scaling to the desired level. Then you don’t have to keep unnecessary servers up and going. You could combine servers and be able to load balance whichever way you needed. Scaling really needs to be built for growth but also for minimizing, or combining. This is really important and clustering could make it possible for us to roll up or down. Changes happen all the time and the way it is setup now there is a limited number of options and moves we can make to scale up and down. It is especially painful the way it is set up now to scale down.

Another challenge for clustering is with our databases. We have a number of shared database tables which are separated by key ids. If you start clustering things you need to be able to move database schemas really easily. To be able to use clustering we would have to reorganize things so that everything is set up on an individual basis and not setup in shared or public tables that could have the same key ID fields or duplicated database references. It is currently almost like having a shared channel or phone line, but we would need to reorient things so that everyone had an individual channel or phone line.

A number of the tables are corp specific so that no one else can ever stomp on or touch someone else’s data. But some of the other tables are what we call a shared type of table. The main difference is that we are trying to get rid of a dependency upon a single primary key id and turn it into a combo based key. This combo based key allows each entry to make it’s own value called a combo primary key. Where there is potential trouble is if the reference is based solely on the auto id, that is where we can get into duplication problems. Especially if we try to cross data bases and share other server’s loads. Once you start saying you want to mix things from different databases then you can have the possibility of overlapping or duplicating the auto ids. When we started coding no one knew this was even coming in the future, meaning multiple servers and multiple systems. Now that we know about these other possibilities we can adjust and build it without the dependencies and make it more future proof.

This is why we see there could be a great benefit to clustering for certain aspects because there is so much power there. But many of our clients could care less about clustering. This is a mixed bag. Certainly some of our clients would really value and be interested in what clustering could accomplish for them. While some of our other clients would not be interested in investing in clustering. There is also an aspect of this being something that developers think would be so great. Almost something new and snazzy in the development world that appeals to developers taste. It is something new and shiny in the developer world and we can all get distracted with the new and shiny. The ones who really see and want this right now are our developers and a few of our larger clients have expressed a desire to aggregate data. To do this it would need to be standardized and normalized, a hurdle discussed in the previous entry.

Scalability, this is something that has started to be a challenge for us. When we started we were doing stock/units and big ticket items, maybe also a few little parts or small pieces. Back in the day we also built in our own pagination, deciding what we felt we could handle/want at a time. This was built into the classic system. But now our new/modern snow owl theme is using a data table. This has some really cool features - you can export to CSV, filter, search things, and it has it’s own pagination. But the problem is that instead of serving up just a certain amount of records at a time, it has to serve up the entire record set at a time. All of the processing for the entire record set has to happen even if it is only showing a small subset at a time. This gets SO slow once you get over 500 records. It can’t bring back a small chunk or bite, it is trying to pull such a huge amount of data so that it can do all of its fancy features. This makes this hard for the slow return time on this. Clients really want the fancy data tables but it makes it a pain for such huge amounts of data.

Where we are headed is to get to the Business Intelligence level, the aggregates. Sometimes these clients that are pulling these thousands of records don’t really need all of these records. They pull them though because they want summaries, aggregates, so that they can pull some final numbers that just say here is the count and the sum. So we are working SO hard to make this return back to them and send it through the Internet's revolving door just to make such a simple count, or simple math of what the totals are. We know what people really want. If we had these already aggregated and in ways that people could access them, we could let people pull this data faster than they could pull for a single day. There is so much time, value, energy that could be saved here by aggregating and making people’s data available in aggregated ways but we are so strapped with everything we can’t slow down enough to make these changes. We see it could be amazing, we could do it, we just haven’t planned it out and made space for it in the crazy hustle of trying to just fill the next need. That is the pattern we are in. We have continually been in the mindset of having a problem, coming up with a solution. Now we are having problems related to efficiency and the ability to optimize the system. We are having problems meeting our own needs in that way, but we haven’t spent much time meeting ‘our’ needs. We don’t have enough people and enough resources to make it happen. We know what so many of these problems are, we know how we could solve them, it just comes down to being limited by the available resources.

Another thing that gets complicated is that without a vision of where you are going, the more you keep building without adding to the greater vision, you have a constant increase of technical debt. More to sort through, more to deal with and fix later. You keep building and building but if it is not really contributing in a way that can be useful for the future then you are continually working to keep building other problems that you will have to fix later. It duplicates and multiplies so fast. It creates far more work later letting things just get done in whatever seems the easiest way for the present but isn’t done to the level of thinking for the future or taking care of it in the best/right way. This is exacerbated with our model of having independent developers that all have a different way of doing things. It makes it so hard to put this all back into one application together. Yes we are getting help and we need help but at some point is has also created a huge amount of chaos that now has to be organized. Some pieces will have to be undone, some redone. It is a help and a hurt. There is some great power in just continuing to push on things and just keep trying to move forward in whatever way we can but there are also a lot of challenges in this place!

It would be a wish come true to have a company that came and said we have a crew of ten developers. We are ready to build and take this to the next level within the structure and organization that is able to accomplish things in a cohesive and constructive manner for the future. It is getting a little too chaotic at this point. The chaos really takes a toll and it really has an effect. It seems like it is having more and more of an effect on some of the people we have working on our team.

We know we need people, we cannot do this as one individual. But working with other people also makes it challenging for making changes, making decisions, or having goals and visions. At a certain point you almost become resigned. You can’t fight all the battles, it becomes too wearying, you have to decide where to put your energies. That is why we are putting effort into a plan. We are trying to create a plan in the hopes that having a plan can help build for the future and help us continue to keep building on this working application. We need a plan to push things forward if this is going to keep going to the next levels. It is coming, that is what we are doing and working on.

There is some value to some of these hardships and lessons. That is why we are trying to share our experience and change and improve upon it. If we were to only share the successes that would be a dishonest view of building something like we have built. What has been built has been built to solve problems and create solutions. This process has also created other problems and challenges throughout the entire journey. This also makes it a challenge as we are working on this as a team and there are varying priorities and preferences. But this is life! This is how things really go in life and we want to be honest about capturing what we can of all of these pieces.

It’s not all broken. There are so many good things we have, we just have to decide at what level we really want to play. We know how to fix this challenge, it would just be a bit of a project to get this all done and ready to play at this clustering level if we wanted to do so. We want to fix these weaknesses. We acknowledge them, we have all played a part in them. We want to work toward solutions to keep building our dream and take it to the next levels.

We want this... Currently, the system was built without clustering in mind. We just need to change a few things in order to really be able to do this. We are spending way too much money on servers and server administration. It would be nice to be able auto load balance and auto spin things up/down as needed. We are at the mercy of the hosting companies. Our servers are going to be attacked, that's just part of being a server on the web. We share space with government, schools, businesses, private, recreation, etc. Mirrors and regional redundancy - where is the data located? How quickly can you get over to those other servers? Uptime and reliability issues. Our clients range from small to big. As they get bigger, they have bigger and bigger demands and requirements. We need to be able to flex up and down based on traffic and clients per server. Where are extra files stored? How does that play into clustering?

Regional redundancy for servers -

This also deals with mirrors, backups, roll-overs, and up time. This is really for the big boys. At our level we don’t really have clients that care or need this. This starts getting into banks of servers. This suggestion was probably added by a developer. Once you start getting really big you have to spread out your footprint to have more stability and security. This could grow as we really need it. Right now we don’t seem to have this need. It may be desirable for the future and could be an add-on service if we decide to have that at some point.

This is not in the current field of vision. This would only come into play if we end up having some really huge customers that have a desire for this. As needed it could be implemented in stages or as paid upgrades. There are options to have this if it becomes necessary.

See number 67 for some other server things. There is a need for communication for when clients come on (new accounts and how big they are) and when clients leave a system (clients who stop using our services). The server guys don't know that information right now.

**OPPORTUNITIES**

We are one of a few SaaS companies that are in compliance with heavily regulated industries -

We like to build generic tools and have generic functionality but it can become very specialized by changing various settings, templates, and permissions. The same application can service a car dealership, a bowling alley, and a cannabis shop. We are headed more and more this way with Adilas Lite. Everything will be a setting. You can toggle things on and off. What do you want? Use what you want and what you need. You can really make this application fit whatever your business needs are.

Some of our future plan also includes different levels including: a transactional core, industry specific skins, custom code where needed, business intelligence level, and if needed clear out to the enterprise and multi-corp level.

SaaS - spelling - for software as a service Definitely a marketing plug and push. We can tell them and we can show them. Willing to keep building and staying current. We would love to see the salespeople go around and get that business. Here are a few industries that are fairly regulated. Firearms, alcohol, cannabis, etc. We are willing to take on others as needed.  
  
Settings and data driven application -

We have already been doing this and we are seeing more and more of a need for it. This is nothing new to us, but now we are trying to say let’s see how much better we can make this. How fast can we make this go? This is how you get dynamics. Adilas could be a chameleon, able to change and adapt to whatever the business and business needs. We want people to get to the point where they can control almost anything - look, feel, flow, colors, permissions, access, customize all your business flows and operations, etc., etc., etc. These are the next levels to helping people take full control.

Most companies pick a industry and go down that channel. They build something specific to the operations of that industry. They try to track down all of the details of the process and then they build it out. But then when they want to expand, they have to start at square one and rebuild things because the whole process was created down one linear path. It wasn’t meant to go in different directions with how it was built. We have built with keeping flexibility and options as our main channel. Imagine a smorgasbord with options to use the kitchen. What do you want? Let’s build that!

We are creating customizable, industry specific needs that are able to be created and adapted right from the get go. They can choose the tools desired and create what they need to accomplish the work of their business. The cool thing about the industry specific skins is that you say what industry you are and we can roll a bulk set of permissions and settings that give you a standard place to start. Of course you can get in deeper and tweak other parts as needed or desired, but the industry specific skin or template helps you start off with something that matches your industry needs.

We have hundreds of settings. We have settings at multiple levels. Corp (world), group (12 main players), page level settings, and user level settings. With our upcoming fracture project, we plan on doing even more settings and making everything data driven. As little hardcoded text or verbiage as possible. Even making help files, SOP's, and other flowchart (wireframe) documents dynamic and per corporation. We want every table in our database to have field level settings. This deals with things like naming, aliases, show/hide, sort order, special instructions, maxes, mins, rules, defaults, etc. Along with settings, we also deal with permissions and soon to be sub permissions. A sub permission deals with the functionality within a single permission. For example: We have a permission called basic invoices. It allows for new shopping carts, creating the invoices, printing inovices, taking payments, and small edits. The sub permission level would take all of those underlying sub functions and make them available as permissions as well. Can they edit, void, see other salesperson's invoices, see costs, profit, etc. The other thing that we are seeing that plays into this realm is templates or preset rules and instructions. Once they are set, or we have an example, we can have the computer/server/application play along and use those pieces as a basis for other decisions.  
  
Smaller code base -

This is just a progression of things. Imagine historically how big super computers were and now we can hold them in our hands. Once you know what something can do and needs to do then you can start the continual evolution of that thing, you can make it more efficient, smaller, more streamlined, etc. You can just keep making it better and better, smaller and smaller, more efficient, etc.

There are so many technologies that we used to think of in a certain way - stable, immobile, has to be in this certain location, needs to take up this much space, etc. It is interesting to see these ideas we get comfortable with continue to change and adapt. We have to change and adapt as a company as well. Things are advancing and we have to advance to keep playing in this SaaS world.

These notes below were really techy. We either need to dumb it down or just preface that we are going to speak some tech Greek/Geek for a minute here.

Currently, we have it fully working... Now that we know what it needs to do, as well as what we want it to do, we can make it more efficient. Less cost, more modern, faster, based off of agreed upon standards, full documentation, testing built-in, and a plan for where we are heading. Exciting! Possibly work off of one or more frameworks. Options for backend code, API sockets, and even frontend code. Less copy and pasting of code. More use of API's, services, DAO's (database access objects), functions, validation routines, special includes, etc. Full MVC (model, view, controller) and object oriented code set. New themes, less outside dependencies, stable asset management, etc.

Clients ask for a solution like what Adilas offers -

People want something that is integrated and able to pick up all of the pieces they need and want in their business. A new business often doesn’t have the full understanding of what they need or want. If we set them up and train them, we set them up for success. They don’t know what a pain it is to marry systems together and to get all of the pieces you end up wanting or needing. That can create so much headache. By the end some people end up buying a number of products to try to accomplish all of their tasks and they all still end up dead-ending. The most common is for people to have between 4-10 software systems that they mix and blend just to run their business. Once people realize you can have all of the pieces together in one system that interacts with all of the other pieces they don’t want to ever go back!

Because Adilas is a system you can start mixing and blending things you never could before. You can get your operations tracked well, then you can in real time have good numbers in accounting also. That is amazing! You don’t have to wait, batch things, or have unknowns. If there are problems you fix them and it fixes them real time at the source and then it is fixed anywhere else you need that information to show up.

We can show that people want to use this. We have had clients that have spent hundreds of thousands of dollars to use our product and clients that have been on our system for years. There is a definite need that we are filling and there is a demand for our product and services. If Google or Apple said you can do all of this from one spot, people would be buying this service or ability just because it comes from them. We are a small no name company but people want what we have to offer. That is amazing! This is the reason we are still around. We can help solve people’s pain and that is a strength. There are people who want this!

Another aspect of our reoccurring model is being a software as a service company. This model has become more of a common thing across many companies in the current business world. This can allow for easier entry, without the huge up front costs, and creates a continual revenue stream for the company. It also can help create buy in as they keep using this over the months and years. Subscriptions can be easier to keep or hold on to instead of making a huge one time decision. It feels freeing that there are no contracts and customers are able to come and go as they like. This has worked out well for us at this point. This may change if we get bigger or depending on certain clients that come on. We will see what the future brings but these unique combinations have helped make us what we are today.

Some of these companies that try to go with a big software package can get into the millions of dollars for setting up their software packages. We can handle so many of those pieces right out of the box. We have this platform already constructed. Even if companies want some custom code or special tweaks we could do that for the thousands of dollars instead of millions. Especially where we are headed with the transactional core, industry specific skins, custom code as desired, business intelligence, and enterprise level tools. We could handle so many business needs for so many different businesses at a fraction of the cost of large scale software packages. Even our custom code on top of the platform is at a fraction of the cost of normal custom code.

The potential is out the roof! There really is just no cap on where it can go or what we could do! There are even more possibilities than we can record or even think of now. People could interact through API sockets and build out whatever they wanted and use the functionality of our application. There is so much potential here and that is exciting!

We get requests everyday for new features. We've had people tell us, if you could just do this and that (we have it all written down) this would be amazing. We've had almost 100 of our clients use our products for over 10 years. For example: 1 invoice per month, times 12 months, times 10 years >> clients have over 120+ invoices that they have paid out to us. Some of our clients have paid over $300,000 to use our products. We've had requests to do white labels and offer our products and services under a different name. The common solutions out there, right now, are not fully integrated systems. They are what are called a "mashup" which means that they mix and blend 4-10 different products together to make an output and/or product. We are a true system, where things interact, flow from level to level, and from piece to piece. We have 12 main player groups and 12 main business functions that all work together. [https://news.adilas.biz/sales-gallery/bf-index.html](https://news.adilas.biz/sales-gallery/bf-index.html" \t "_blank) - 12 business functions [https://news.adilas.biz/sales-gallery/ap-index.html](https://news.adilas.biz/sales-gallery/ap-index.html" \t "_blank) - 12 system players

Pricing -

There is a lot of room to grow in our pricing. In some ways we don’t even realize ourselves what we have. In some ways there are some of us that are scared to increase our prices. We have started so small with mom and pops who were willing to take a chance on us and now that we have grown into something far more robust and able to service even more people we have had a hard time bringing our pricing scheme along with that. We have struggled with this for years. We do keep inching it up, but we are inching it up when it should be leaping by feet. We have never lost clients to this point over our price increases because they still know they are getting a great deal.

Another thing is that we have to determine what we really have? We have so much it is hard to get your head around. What is that worth to people? We have a really powerful engine but we don’t have the look and feel where we want it. So we tend to measure on the gap - what we lack - instead of on the gain - what we actually have and where we have started from. In part this is because the managers are also the developers so all we get presented with day in and day out are the problems. This isn’t working right, fix this, there is this bug, we want this new feature, etc. That lens and perspective informs our ideas on pricing.

If our prices remain low that does make it really hard for our competition. So there are pros and cons to whatever choice we make. If all else fails we could make it for zero cost and live off of the other services. Although, sometimes there is a stigma of something costing nothing. Like the idea that to get something of value you have to pay more money. If we could get into the white label area then we could actually service both markets. We could service industries or companies that want things to look fancier, or cost more. But we could still service people at the generic level. Like the difference between a dress shoe and a work boot. But both have the same power and engine, it just differs in the look and feel, the presentation, and the marketing.

We don’t think there is a magic bullet for pricing, this will have to be something that we determine depending on what opportunities become available and how we want to configure it. If we can get it to the point where we can configure all the pieces and you can turn things on and off. Then you could charge people according to the portions they could use. You can turn on and off features and therefore pay for features as you want them. Industry specific skins will really be a help with this as well. This can help people have a great starting place with what they may want or need. We really want to educate people and make it so user friendly that it is easy for them to pick up and use.

An analogy is that we are like a ski resort. The ski resort has to run the ski lifts to service their customers. We have to run the servers to service our clients. Whether we have a lot of clients or very few clients it still takes the same amount of money to turn on and run the servers, or in the analogy the ski lifts. We run the servers either way. We earn more money if more people are there, ‘using’ the servers but we still pay the same amount either way. This is why clustering could be a great asset for us because we could scale up and down as we needed.

There are really so many different ways people can use this application and so many ways we could configure pricing and structure. There is probably not a one size fits all for our pricing and our clients. We can probably have a variety of options that people can use. Brandon and Alan did do some work on some pricing structures. They do not have specific values but there has been some leg work done on what different levels may, or may not, include and involve. It is often easier to just pick a monthly number than trying to nickel and dime each piece. We may really need to outsource these marketing and pricing aspects. We may need someone to help us with the marketing psychology and how we approach our pricing strategy. We would also really love someone to come and take on servicing higher end companies with white labeling. We want people that know things for different industries that can take it to them. They can do the marketing and speaking for the product. Our job is to make a product that is solid and configurable. Allow companies to use our robust engine to cover their business needs but also make it easier for users to jump in and start using the system.

Currently our pricing is so low, it has room to go up. Sometimes that is tough, but most of our clients know that they are getting a steal of a deal. With where we are heading, we are planning on doing a number of different pricing tiers based on usage ranges. We are planning on having a free version, basic, standard, pro, and enterprise. Along with the new levels, we will also offer options for hosting plans such as shared hosting, semi-dedicated hosting, dedicated hosting, and clustered environments. We plan on offering certain features based on the plans and levels. We will also include options for storage levels and overall footprint of each world or corporation. Depending on the clients and their needs, we would like to offer other services that will help support them. This could be compliance, oversite, consulting, data entry, training, custom code, design work, etc. We will also be offering white labeling options where they (the other companies) get to control their pricing and we get part of the revenue based on negotiations and agreements.

All in one solution (POS, ERP, CRM, etc) without having to use multiple software packages (QuickBooks, POS, etc) -

These notes below really capture some great points and concepts. We can offer so much! If we want we could expand any of the concepts below to further explain or clarify. We have such an incredible ability to give people an all in one solution, no more needing to marry and hodge podge systems together. There is incredible value and capacity here!

Similar to what has been covered above. It really is an all in one solution. If we don't have it, let's work together and add it or tweak it out so that it works as needed. Once we have the main core done and finished, we could add on almost anything fairly easily. There is some pain for companies that have to mix and blend tons of different systems. They have to double, triple, enter data, possible user errors, certain things don't talk to other programs, they have to export and import tons of data. Everything isn't in the same place. They are paying multiple bills, and the list goes on. This is super common. Most businesses use between 4-10 different systems to get the numbers, values, and output that they want. If a company is audited, we can shine, the whole story is on display and all of the pieces connect together. This is called digital storytelling. World building - cool concept and super powerful - you have to have the whole system in order to play that game.

Enhance the system based on trials and tribulations -

The system is currently working. However, if we don’t improve, increase and optimize it, it is going to have a shelf life. We have to keep pushing it forward if we want to keep this product going in the current and future world.

We experiment and we prototype, this is how we learn and build. We will prototype a new thought or idea, but it only gets built out in one section. We haven’t had the time or resources to cascade that out into the whole application. We have to push those new features and those successful prototypes forward to make the whole application become better and improve it’s standards across the board. Because we are externally funded through clients and their needs, we follow the funding. These other projects that would really enhance Adilas get set by the way side because we have to follow the projects that the clients are actively funding.

In theory, everything we want to do we have a working prototype of. We have so many pieces working or built to a certain level, we just haven’t had the resources to cascade them out fully or take them to bigger levels. Things also keep changing. There is always a moving target it seems so it takes constant, active work to keep going and building.

We've been at this for over 20+ years. We've built pieces, fixed pieces, refined pieces, invented pieces, etc. The current platform is very stable and is used on a daily basis by tons of users from different industries. With the upcoming fracture build out, we have a list of all of the lessons learned, what to improve, what to enhance, etc. We will roll all of that info and knowledge into the next full version. We have a concept called the value add-on core that takes a transactional core (similar to what exists now) and then adding on other layers on top of that, including business intelligence (BI) levels. This takes the transactional core data and aggregates that data so that it is quick and available for stats, counts, sums, totals, maxes, mins, averages, etc. We can then take that info and data and use it as a faster version of the underlying transactional pieces. If the user wants, they can do a drill-down to deeper and deeper levels to get at the underlying real numbers and data.  
  
Outside funding -

Currently we are not funded externally other than client’s requests for custom code work. Which, this is great! This means people are saying, they like what they have and they are willing to pay for external code on top of their monthly subscription to help create a better tool for their uses.

We want to look at outside funding through the lens of people seeing the possibilities with Adilas and investing in future growth and development. We could even build it out like a real estate venture where we build out different parts of the product and then sell certain pieces or portions of it. This is an analogy that people are familiar with but we think a similar thing could happen with Adilas. This would take a lot of planning and preparation to be ready to offer portions of Adilas in this way. Another aspect to consider is that if we add investors, or sponsors, we would also have to add more contracts and legal pieces to satisfy all the parties involved.

We are also interested in looking at having sponsors for certain pieces, or portions of the system, where sponsors could make reoccurring monies based on people’s usage of that part of the application. One of the downsides is that we don’t know all the timelines on these things and what happens when we need to upgrade the pieces that have been completed. For instance, if someone sponsors a new shopping cart, what happens when that shopping cart needs to be updated? Who pays for that? What happens if you have to discontinue the use of what was previously built? Do you give the original sponsor the opportunity to play? But what if they don’t want to invest more? Do they lose revenue entirely for shopping cart use if a new one is built? Lots of questions to explore here.

Also if we are doing all of these industry specific skins and white labels, that could be a great opportunity to have people come and invest. We would love to have people help with custom skins and receive benefits from taking over some of the white labeling pieces or for their industry specific skin. This might be an easier way for investors to participate because then they don’t have ownership over the core Adilas pieces but they have investments and revenue returns on the industry specific pieces. You technically can have this on both levels as we have mentioned before - core pieces and white labeling at the industry level. But as we were talking today we were wondering if the industry level might be a really sweet spot for having investors participate.

Ideally it would be easier if we had a single person willing to invest a large amount of money to build or create something. But we may have to do sub levels of marketing on pieces, or possibly group funding on pieces. How do you pitch that? Do you start at the highest level and then try to work down? What kind of fish are we fishing for? That makes all the difference when you are creating your pitch? Who are we trying to get? Where will we find the kind of investors we need and/or want for these different pieces.

Loans are a possibility but they create their own heavy load. We would still have to have the plan. We would have to have the people in place to be able to accomplish our next steps and growth. We still need a plan! Right now we have a maxed out team. If you took our maxed out team and added a bit of funding to it, it doesn’t mean you get all of your dream pieces built, or possibly any. We need to have a plan and a way to achieve our goals. If you look at our list of goals we have new code as number six on the list. It is so easy to want to jump straight to that piece, but we have to have some of those underlying pieces in place first. For instance, we need people to manage and lead these teams so that we have a cohesive core team for developing and refining this application. You can’t build effective new code if you don’t have management making decisions that are leading you to your desired end goals.

If we cannot secure outside funding, or it is slow in coming. We need to form an alternative plan to start breaking into some of these projects on our own so that we can begin to show how great this will be. We can then take these pieces out to the public and help generate excitement for our clients and hopefully potential investors for other future pieces. Regardless, the crux of what we need right now is still to make a plan! The plan needs to be structured, formatted, organized, presentable, and made available for us to move forward in building the next stages of Adilas. The plan is the next, most important step for moving forward!

We are really hoping to show our existing product as a fully working prototype. That is huge. We could sell part of the existing company. We could build new companies and get capital that way. We could use VC's (venture capitalist) or private equity or angel investors. We could get a loan (not our intent, but it could work). We could increase sales of the existing model to fund the newer version. We have some other ideas on financial products that we could offer on element of time # 10235 in the shop. [https://data0.adilas.biz/top\_secret/time\_web\_gallery.cfm?corp=748&id=10235](https://data0.adilas.biz/top_secret/time_web_gallery.cfm?corp=748&id=10235" \t "_blank)

We already have a full proof of concept - This is a multi-decade proof of concept. This proof of concept has already generated close to $8 million in revenue. That is awesome! We can back up what we are pitching and selling. We have a working prototype of every piece we are trying to do. here are some fun ideas for financial products that we could offer. [https://data0.adilas.biz/top\_secret/time\_web\_gallery.cfm?corp=748&id=10235](https://data0.adilas.biz/top_secret/time_web_gallery.cfm?corp=748&id=10235" \t "_blank) People get excited when they see opportunity. When they see that we are already doing it, that really helps. Way past ideas or dreams on paper. We've already made it work in the current version.

Gathering IP from contractors who have left the company -

IP is an acronym for intellectual property. There are so many things out there that we have paid for developers to create. We have many of the finished products but we would like to collect all of the building source files and process plans into a library so that we can use and access the content we have paid to be created. This also holds the proof for why, when, and how we created these different projects. Having a library with this information can be so helpful for the next levels of development. It helps developers see the patterns, or understand why and how things were built a certain way.

Since we have used a number of independent developers building this project we have things spread all over the place. We need to gather the pieces we have built back together and have one centralized place to collect all of these assets and information. Creating a master library of all of the development pieces would be an incredible asset to future building projects and to telling the story of what happened and when. This library also allows us to create new standards and procedures for capturing and organizing all of the parts of new development going forward. We want it to become part of the regular routine to organize all development work in this master library.

One of our goals is to protect ourselves through sharing. We have had this vision for quite some time. We want to share the ideas and concepts that have helped to build our core. A development library contributes to this goal. We can protect what we have built by sharing it and making it accessible and available.

We've already paid for it. We don't have any bad feelings, like where someone is not willing to give it to us. We just haven't asked (been running too fast). It needs to be centralized. It needs to be stated up front that all authoring files, XD, Illustrator, PhotoShop, etc. need to be sent over to us for organization and storage of those assets. We do have some info on the non-compete doc and we will revisit it and make it tighter. We need to get the assets as they get built and/or created. After the fact, they, our guys/gals, don't have any motivation to give up those assets after the fact. We need it to be part of the project and project management. Possibly holding payment until the assets are delivered and categorized. We need the originals (authoring files with layers and workable parts). The final product is great, but we actually need the originals as well. We have been talking about other developers and designers... We also need to get files from Brandon. He has thousands of files and original authoring files.

Biased about our own products -

We keep building as if this is the greatest thing on earth. The opportunity here is that we keep chasing dreams and building toward those dreams. Often reality makes you more grounded and not always as likely to take the risks or approaches we have taken. Surely there are cons here as well but it has allowed us to create so much.

This view has also caused us to take on the whole application with a broad view instead of just a small piece of it, or pigeon holing ourselves. The beauty and opportunity there is that we have created a system based application that is immensely robust and all works together as one.

We started doing operations, which led us to accounting, which has led us to other things. We started everything we’ve done with trying to help solve needs and problems. This underlying approach has really been the catalyst for what we have created. We put the horse first and allowed the cart to follow. This put things in order for us to create what has been created. We never started with the intention to create this system but it has come because we have followed the next problem and expanded it into more and more pieces of the puzzle.

We are creating a dream. We are dreaming it up, wiring it up, and making it happen. Less trying to keep up with competition, just doing our own thing. There is a weakness to that as well, meaning we don't even know what our competitors are doing. We are really pioneering our own path and making our own products. We do get some requests or other outside influences but often, we are on an island of our own.

Willingness to help people and companies succeed -

Technically we are a software as a service (SaaS) company. We have the software part but we would really like to add more and more to the service portion. The software will keep growing and improving as well but we could really do some great helps by improving more of the servicing pieces.

Steve has consistently been one who travels, works onsite with clients, teachings, trains, explains, and helps clients solve their problems and pain points. Steve has been such a leader on this front. He knows how to serve clients and take care of them.

We have traveled and worked with people over and over. Any time we do this people share ideas, troubles, successes, which all help us grow and know how to better service our clients. We really want to be a software and service company. We know there are so many opportunities on the service side of the equation that we have not expanded on. There is also a small outlay on services. You don’t have to build out a whole bunch of new code or create new products. The service side typically constitutes an individual bringing themselves and helping the client with any needed pieces.

We are not entirely a core for profit company. We love helping people and businesses. Most of adilas has been built off of trying to solve a need or fix a problem. We have gone above and beyond to help our clients with their needs, requests, training, staffing, etc. We've added features, changed things, and hopefully do a great job listening to what they are wanting and asking for. We are willing to travel, meet with, and take the input that others are willing to pass our way. We want to help people succeed and by using our product, we have helped a ton of people and companies.

Cater to specific verticals -

We have a general system that can do all sorts of things and we can cater to all sorts of verticals using our system. We will just add in settings, templates, and permissions to help customize the system. That same system that will be servicing a car dealership can service the bank, or the car wash, or the doctor’s office, or whatever the industry vertical may be. Adilas is so diverse, it has all the tools, and any industry can use it for whatever their needs are. Out of the box we can help any industry or company with over 75% of their business needs. Adilas is just that deep. Industry specific skins are an area where we can stream line and refine some of the other particular pieces for a business or industry.

There are so many interconnected pieces and ways to connect them in the system that we can service various company’s needs. It has even been fun for us to see people use our tool set for things we didn’t even imagine them using it for. Adilas is a robust tool set and it is amazing to see the creativity displayed by clients as they use the tools, processes, and features that Adilas offers.

The Adilas core can already do so much but it can explode with possibilities if we get it built out to fracture level and ready for the value add-on core. We don’t even know all the places this platform could go using the Adilas core with the API (application programming interface). The possibilities become endless! Hybrid products, mobile apps, create your own flows, processes, build your own automation, etc. People can use different programming languages or programs. If you are interested in this contact us. We can share with you some of the projects that have already been done using the Adilas API.

What it can do is so amazing! But there are always more opportunities, more things people want built, and more ways to move the project forward. This is not just pie in the sky type stuff. Adilas is doing this and this is what people are wanting. Clients and developers keep asking for more and more to make the API more accessible and easier to use for outside developers. People are using the API and they are giving us wish lists, requests, suggestions, and more ideas from their developers and users. Their projects are important to them and therefore it becomes a huge asset to us to see how they are using the API and what they want us to do so they can use it better. On the other hand, this project has become so big that we give and give and give and it doesn’t even fill the void. It is always wanting and able to consume more. The opportunity to see here is that there is a continuing need. We can keep working to fill this void/need.

We would love to help each industry play the game as they need to. We already do a bunch of this. Ideally, we would love to offer possible white label options. From John - Not all business is good business We, as a company, under value our product. We work really hard for the sale but we don't really capitalize on those products and features that help our clients. On the new fracture or adilas lite build out, we are going to take every section and say - Do you want this? If yes, what parts and pieces do you want? How will it all play together? Hide everything that is not needed (right now). Everything is there, it just may be dormant or hidden. It is still fully there and ready. That is huge! They get the whole thing without paying an extra cost. There are some exceptions but for the most part, they get the whole meal deal. One of the levels on the value add-on core model is the industry specific skins. See this image to help show that level. (level 3). [https://data0.adilas.biz/adilas\_for\_business/images/photo\_gallery/ideas\_plans/value\_add\_on\_core\_model.jpg](https://data0.adilas.biz/adilas_for_business/images/photo_gallery/ideas_plans/value_add_on_core_model.jpg" \t "_blank)

Make the new application lighter than the current application -

We originally were talking about the code name fracture because everything needs to broken into smaller and smaller pieces. Then we determined to use the name Adilas Lite because it is more descriptive to what we hope to make happen. We want to make all of the pieces seem smaller, more manageable, and easier for client’s to use. In Russell’s words we are trying to make it powerful, easy, and pretty.

All of the points in the paragraph below are there own topics that could be expanded on for pages on their own. We didn’t want to open all of that up right here under this topic.

This could be code, servers, settings, permissions, testing, validation (client side and server side), API sockets, etc. We have been working on this for over 20 years. We have learned a lot of lessons from the past. If we could conquer this, that would be awesome! What is in the core? What is really needed? We have a lot of bloat (excess stuff), do we really need that? Let's build out the first or biggest priorities and then see what else if needed. For example: The trucking homepage, the google calendar interface, other super specialized parts of the system. Build out the main core, start selling it (the system), and then build on more as needs arise. Build it out with full testing, validation, documentation, settings, permissions, education, etc. Basically put it to bed so that we know it is fully done and fully built out. We are looking at some frameworks that may help with standards, coding, debugging, and basic pre-set standards.

Work with experts to build white labels for specific industries -

If you read the notes below we really want to do this. We need to have more of a plan for executing this, but we do want the expertise of those in certain industries. We want to plan well, work hard, and utilize the skills of other’s around us. We also want to make it mutually beneficial for all involved. Helping our clients get industry specific tools, ensuring industry experts helping us would be well compensated, and improving Adilas’ overall business and services.

Big value here. If we can really get an expert, in a certain field, to help us build it out for that industry, we can hit a home run. This is worth spending money and time and resources on. From John - Mine the miners, not the mine. Playing the one off, be the bank, the store, the hotel, the dinner. They sometimes did better than the actual miners. All miners needed those services. Mine the miners. Idea - It would be super cool to set it up so that we had a series of industry level experts that we worked with. Each one could basically either have or help us setup a new industry specific skin or white label solution. That would be awesome. We would welcome all kinds of relationships and even outside entities to play this game.

Build a dream -

We love dreams! We love to support people in their dreams! Dreams are what has motivated this application to be built. Far more than the money, the dreams are what keep this project moving forward!

We are going to keep dreaming, planning, and working to pass along the dream and the vision. This is why our next goal is the plan! We have to develop a usable plan to help the vision keep moving forward. Without a plan, “good freaking luck”! ;)

Recording notes and lessons learned is one of the greatest contributions we feel we can make to this project. We will keep recording the journey and the story.

This is what inspires our developers and co-owners. A system is a system, but a dream is worth following and supporting. When we first started, we didn't worry about the tech, we built towards the dream and said, we'll figure out the tech either along the way or as it is needed. Build towards the dream! Brandon has been keeping the developer's notebook to help keep the dream alive. That is the motivation behind that part of the application. Wouldn't it be cool if we could expose a digital blog, journal, or simple forum for all of our end users. This is what keeps us all going - we see the potential and also see the progress towards that. That is awesome!

Market the new product to interested parties before the product is finished and released -

Because we don’t have the funding to do all of the projects that we want to do it is hard to finish projects. We have so many ideas, dreams, vision, and goals but they can often get derailed because we run out of money or resources.

So many projects have had tens of thousands of dollars worth of investment in planning and development that have then had to be shelved when resources run out. It hurts really badly to have seen this pattern repeat over and over again.

We also have no marketing on any of our projects. We are good at building and fixing, but we are not good at communicating and marketing. We are too small of a team trying to go too fast. Our clients are completely unaware of most of the features we even offer and no one gets told about the cool new features developed. It would seem simple to just say, slow down, so you can market and be more effective in these ways. But it is hard to slow down when you are stuck in the survival cycle.

There are still lots of opportunities here as these projects are still needed. We can circle back around to them, and we will eventually circle back to these projects as resources and time permit. Our beginning motto has changed, it is no longer can we do this??? It has become, how well can we get it done? This new question requires management, planning, dedicated budgets, marketing, etc. There are lots of opportunities available but sometimes you need outside help, or a different catalyst to get things moving in a more desired trajectory.

We would love to see this... Similar to the movie or video game hype - small teasers - people are already sold on what it will be. This marketing piece has been hugely missing in all previous builds. We build and then move right on to the next project, feature, or whatever. We have never really marketed this product. Mostly word of mouth, thus far. We would love for it to gain some hype and then really deliver on that hype. We would like to spend enough money on marketing to really make it happen. We could dedicate more resources, pitching things as they get planned or coming down the pipelines, etc. Because it is not yet built, we could market it even more than something that already exists (in some ways). Pitch the pitch, get great feedback, keep improving the actual product. Feedback is another huge part of marketing. We do a lot of idea farming and trying to listen to our clients. We'd love to take that to the next level. We are planning on getting some funding as well. Marketing will be getting them to buy in even before the actual product is finished. We would love to have someone or multiple people, just dedicated to the marketing, campaigns, and then checking on ROI from those plans and campaigns.

Define a MVP -

MVP stands for minimal viable product. It can also stand for minimal viable person, plan, etc., or MVT for minimal viable team. As we look at things it seems to become more clear that we need a plan more than we need a product. It is easy to see the next step and then take it. If you don’t know where you are going, you just keep taking the very next thing in front of you but this can quickly lead you away from your ultimate goal. Eventually you cannot go where you want, or as fast as you want, unless you have a plan and a way to get there. What we have been doing has worked for us but it cannot be the permanent solution. A plan is critical to the long term success and future of Adilas.

If you don’t know where you are going then the smartest thing to do is to just take the next step you can see. But that is not our ultimate goal. We see something much bigger that we want. If we don’t plan and build to that specific end we will not make it there by just chasing the next requested project. We have done this for years and sure, we have seen progress, but it does not necessarily get us to our bigger goals. It is like creating a road. You may start with a trail in the woods, which you then widen, eventually you clear out more things from the path, then at some point it becomes used enough that it is ready for pavement, vehicles, and taking to the next step.

Form follows function. First we have needed to make it function correctly and then we need to make it pretty and easier to use. We feel that we have done a good job on the function, it is time for us to work on the form, ease of use, and look and feel.

It is so easy to see what we want to do for fracture and Adilas Lite because it is new code and development. But technically fracture is number 6 on our list of the 7 projects we see as the next steps. New code is awesome and the most fun to work on, but we will end up in the same spot if we don’t work on some of the underlying foundation pieces we need to work on first. We have to focus on some of these other goals to truly continue building the Adilas we dream of and be successful in the future.

If we want others to come along we have to start prepping and preparing the way. It is fun to explore, it has been great to just see where the next thing takes us. That can work for a small team and when it doesn’t really matter where you end up. But if you want to reach a certain destination, or if you have somewhere you want to make it to, or a large number of people you want to take with you, you need a plan, you need a trajectory, you need to keep making progress toward a desired goal and end. Size and scale matter depending on the nature of the venture you are taking. If we want others to use this product we can’t keep taking them exploring in the woods. They will naturally gravitate toward things that are easier, faster, shinier, better, quicker, prettier, etc. We have to offer that if we want to keep servicing clients.

Here is a brief run down of where we are headed. For the full outline please follow the link below. This is just the ten thousand view fly-over at the upcoming projects.

[https://data0.adilas.biz/adilas\_master\_plan\_outline\_round\_1.pdf](https://data0.adilas.biz/adilas_master_plan_outline_round_1.pdf" \t "https://app.goto.com/meeting/_blank)

1. Company Structure (Jellyfish Model)

a) Overview

b) Divisions & Departments

c) Knowledge Workers vs. Hourly Workers

d) Personality Types

e) Payment & Compensation

2. Product Development (Value Add-on Core)

a) Overview

b) Define The Current Core – Transactional Core Level

c) Industry Specific Skins Level – White labeling

d) Custom Code Level

e) Business Intelligence (BI) Level

f) Enterprise Level - Unlimited

3. Education & Training (Internal & External; Adilas University)

a) Overview

b) Video Library

c) Tracking Education & Progress Outline

d) SOP’s (Standard Operating Procedures)

e) Interface Changes - Easier Interactions

f) Internal & External Training Needs

g) Online & In-Person Training Events

h) Dynamics & White labeling Options

4). Community & Third-Party Solutions (Adilas Marketplace)

a) Overview

b) Discussion On Byproducts – Created By Adilas System Or Platform

c) Professional Services

d) Buying & Selling Products – Virtual Community Marketplace

e) 3rd Party Solutions & Outside Vendors

f) Sales, Advertising, & Marketing

g) Dynamics & White Labeling Options

h) Exploration Of The Marketplace, Lots Of Potential

5. Social & Efficiency Options (Adilas Café & Community)

a) Overview

b) Personal Profile – Standardized Starting Point – Virtual Data Portal

c) Sales & Creation Of New Accounts

d) Options To Work

e) Options To Play (Demo Sites)

f) Options To Buy & Sell

g) Options To Get Training

h) Options To Participate & Do Social

i) News & Updates

6. Deeper Product Development (Adilas Lite & Fracture)

a) Overview

b) Setup Wizard

c) Customize Everything – Full Interface & Functionality

d) Features & Functions – Modular Architecture

e) Code & Server Changes – Frameworks

f) Options To Get Training & Help

g) Use Of API Sockets & API Endpoints

h) Testing & Integration Plans

i) Client-Side & Server-Side Validation

j) Mobile Ready & Responsive Design

k) Lessons Learned From 20+ Years Of Doing SaaS (Software As A Service)

l) Life Cycles – Maintenance & Upkeep

m) Dynamics & White Labeling Options

7. Budgets, Finances, Marketing & Sales (Other Business Plans)

a) Overview

b) Business Plan & Mission Statement

c) SWOT Analysis

d) Funding & Repayment

e) Creative Documents

f) Technical Documents

g) Sales & Marketing Plans

h) Profit Sharing

i) Budgets & Timelines

j) Prototyping & R&D (Research & Development)

k) General Research & Market Research

l) Other Plans & Pitches

m) ROI (Return On Investment)

n) Dynamics & White Labeling Options

The current system is huge. We could get the main core finished and done and take away any of the bloatware or extra stuff. We could also do smaller MVP's such as employee time-clocks, project time-clocks, simple POS systems, simple CRM functionality, etc. Because we are somewhat starting from scratch... We know what the current system does and has, so... what do we want and need for an MVP? We can leave the other stuff out for good or add it back in as needed. MVP isn't just minimal viable product, it could be minimal viable plan, person, etc. Those are important as well. MVT - minimal viable team - We know we need marketing, a decision maker, and enough people to support and build this product. What is that team size or roles that need to be addressed? What about board meetings? Keeping everybody on the same page. Setup and have a monthly meeting so that everybody knows where we are at and how things are proceeding. This could be paid or non-paid. If you are a co-owner, you need to be there. We have an agenda and cover basic things. Strengthen those communication channels and let everybody know what is going on and where things are going. Focus on the key big pieces of what is going on with the company.

Build a new system using lessons learned from the current system (practice makes perfect) -

As we went through the notes below we saw a lot of the pieces needed to be expanded and broken out. So this section took on a different flavor. We expanded the original notes and added more thoughts where they seemed to fit.

Just for fun here are a few other things we were saying regarding the title. Practice makes permanent. Practice makes perfect …??? Mmm, or how about perfect practice makes perfect. :)

Make the code base smaller, more dynamic, fully data driven, good quality control, testing, SOP's, standards, structured for growth, sign-offs and requirements. Having the standards and then holding people accountable to those standards. Setting those standards will really help! Where have we done great? Where have we fallen short? Where are we heading? Start - Stop - Continue. Start - what do we want to add? What will help us if were to add these certain things? Stop - what things need to be taken away or removed so we can improve? Continue - what is working well and what things are good to just keep going? Start - Stop - Continue - make a plan and then work that plan. Working towards an MVP level product. Even if not finished, show the progress or a progress report.

User stories and virtual obstacle course criteria. User stories have a who and tell what they are trying to accomplish and the process of those things working to accomplish their tasks. We then try to make sure people can go through these different scenarios. So that is the obstacle course criteria aspect - running your who/subject through these actions so that you can have the proper effects follow the subject and character. This is what we do for code. We test different scenarios and run virtual obstacle courses. We have thought about taking one main player and breaking out every piece and trying to take it through the new code sets. If the data and/or code can successfully make it through the new code sets then we are on a good path. If not, then we would have to re-code, fix, debug, etc.

We would like to make a list of what we have to offer right now, an inventory list of our current system. Along with that, trying to layout the vast majority to what is needed, wanted, and required. Also, what options exist, settings, permissions, and cause and effect flow stuff.

On the development side, we have developers who are developing on a system that they have never used. That has been an issue and sometimes shortcuts the vision or how everything plays together. Sometimes our developers have no idea how to navigate to the area they are working on or fixing. They may not even be able to navigate to that area to test it again and see what happens with their code. We need our developers to be trained on the system and how it works or how our users use it.

Another aspect is that our clients use the product in so many different ways and can change their outputs with settings and flow, that it can really effect development and knowing how to look for and test all the alternate routes or possible end points. We often code things against test data but many times we really need to code and test things against real data. Sometimes just the sheer volume difference of fake data versus live data can cause the code to fail or not perform as intended.

Too much information is useless if you can't access it (search it, find it, or even know where to look). Currently we have all kinds of cool stuff but you have to go all over the place to access it or set it up. As a solution we would like to start organizing it by going to one place to find it (all of the connections and sub connections per section). The more we make that plan and get it organized, it makes it more accessible and usable for all users. We keep trying to chip away at this one piece at a time but we feel like it would be amazing if we had a larger amount of monies, or resources, to dedicate to these projects. Limited resources limits growth and potential growth.

John once again said, not all business is good business. Even if someone pays for something, do we have to jump and take that bait or build out that product/feature? We have a few pieces that are only a certain percentage complete. We'd like to really finish things up, see them all the way through, and have enough to keep supporting and maintaining them. Yes, we can, but should we? Good questions. Get things done and across the finish line. Then go to the next.

We also want to improve the marketing, training, testing, planning, and sign-off processes. These may or may not even fit here but they were in the original notes and may need some further exploration.

New positions are going to become available as the new application is built -

Figure out the budget so that we can fill those positions. Let's figure out what we want and need and then play and plan accordingly. Roles and responsibilities can and do change. We were talking about a 3-month probation period. We currently, don't have a ton of requirements, expectations, and review processes. Peer reviews, raises, incentives, other perks, etc. How can we help our team to keep pushing and producing? Setup other rules and structure to make all of this possible. Everybody needs to pull their weight and produce.

More money to distribute to the team (current and future) -

Boost morale, pay people for what they are worth, and keep good people. Not just increased wages. Also thinking about other benefits, perks, and upgrades. Competitive wages, sick pay, vacation pay, etc. Possible profit sharing options. If we make a better product, it should pay the company more and thus open up some doors to distribute those funds to our guys and gals. A better product deserves a higher price tag. Our pricing model really needs some loving and help. Spend the time to make it correct. We could also checkout Amazon, Adobe, Microsoft, and other bugger fish to help figure out some the pricing options. We will then make it fit into our model.

Incentives, budgeting projects and paying according to budget, not necessarily hours. See EOT # 10300 in the shop - [https://data0.adilas.biz/top\_secret/time\_web\_gallery.cfm?corp=748&id=10300](https://data0.adilas.biz/top_secret/time_web_gallery.cfm?corp=748&id=10300" \t "_blank) - We want to create plans to help incentivize our players and people with talent. - You get what you pay for - old saying  
  
Develop documentation that creates consistency throughout the system (GUI, testing, data dictionary,etc.) -

I love it! Let's do it! We have an opportunity to take all of the information that is spread throughout the entire application and bring it into one centralized location or place. We have tons of great info and content. Table of contents - master level list Style guide, code conventions, date conventions, almost to the OCD level. Be able to read your code and others need to be able to read your code. Readability - Consistency is the key. We will be adding a number of settings, setup standards, and even do some product feedback and market research. Once those standards are set (style guide, conventions, etc.) we need to hold the developers accountable to those standards.

Expand our services to include website building and logos.

This doesn't even have to be deep backend adilas stuff. We could build simple websites for people. Almost brochure level sites for our clients, other people, etc. If they need it, we could tie it into adilas but that is not required. We have a full design and dev shop, why not use that as a service that people could pay for. Adilas creates byproducts - that could web sites, consulting, marking opportunities, project management, data entry, bank reconciliation, hosting, IT, compliance, oversite, etc. All billable professional services. See this adilas marketplace link for some fun ideas... this is just scratching the surface. [https://data0.adilas.biz/adilas\_for\_business/adilas\_market.cfm](https://data0.adilas.biz/adilas_for_business/adilas_market.cfm" \t "_blank) Just because we offer it, doesn't mean that people will bite. We have to push it and market it. We offer tons of services, reoccurring, one off's, custom, just in time, etc. Word of mouth is great. We could also supplement that and really push the services - all part of SaaS (software as a service). Maybe focus on both parts - the software and the services. Adilas creates all kinds of byproducts. Harness those pieces.  
  
Data aggregation from the start

Make sure that all of our database tables are fully related (correctly and properly) as well as defining where things go and connect to. This is huge. If we are in there redesigning things, we can take and spend the time to also include some data warehousing type functions to get the aggregated data, right from the start. On the adilas value add-on core model - The entire level 4 ring is the BI or business intellegence part of the model. All kinds of aggregates. This includes totals, counts, sums, averages, mins, maxes, etc. [https://data0.adilas.biz/adilas\_for\_business/images/photo\_gallery/ideas\_plans/value\_add\_on\_core\_model.jpg](https://data0.adilas.biz/adilas_for_business/images/photo_gallery/ideas_plans/value_add_on_core_model.jpg" \t "_blank) We could then display all of that data, dashboard style or accessed through simple AJAX calls or data drill-downs.

Shorter timeline to train new clients or  
self training for clients

One, if they can configure what they are seeing (permissions and settings) it will be smaller and more manageable. Two, if we allow for an education mode to be virtually turned on/off, that could be huge. Currently is hardcoded. How cool would it be if we allowed it to be dynamic or data driven. Also, if a company wants to change certain things, for their company, we could allow that. We would love to see short and concise videos for each section. This could be on the marketing side (sales) or on the user side for just in time training. Along with those tiny videos (laser focused) we will need a way to search those video clips and see the overarching organization and layout (table of contents of the video clips). Adilas university, to a whole new level. Able to hire new employees/workers and get them trained quickly. Internally, if a developer needs to learn something, they could use the same video clips to learn what to do and/or how a section works or interacts with another section. Testing and levels of certification (if needed). That could be cool to track as well. Including allowing each company to setup requirements and then monitor progress and completion. Clear out to the HR level if needed. Allow for industry specific training and certification - this is just another marketable service. Training and education is huge for us and for our clients. John and I had a discussion about standards and conventions - for training. We tend to blast things out quickly to help us move along. Then, depending on funding, we sometimes aren't able to cirlcle back around to really make those things tight, solid, and professional level. We need to be profitable, earning more than we are burning. This applies to training videos as well. Sometimes we are so worried about building or pioneering a path that we get lost and don't end up circling them back around. If we build it into the plan, then we can knock it out of the park from the get go. Good plans and then following those plans makes sense (cents and dollars - being silly). John and I were talking about some needed internal training courses and events for our developers. John did the last training session and we had requests for other courses and classes such as: Future Training Session Ideas - Cfscript - Javascript/JQuery - Back End (Models, Services, DAOs) - API's - Internal API sockets and external API sockets - Ajax - Testing - Custom CSS & Print CSS

API to drive training internally, clients, and  
third party entities

API's for everything - play at the wall. This is huge and we want to build out everything in an API socket or API endpoint. We will be able to turn them on/off and then use them both internally or externally. By way of an opportunity, if we open up the API sockets or endpoints, that allows all kinds of other outside developers to build out code, interfaces, and sites or widgets that play with adilas data. Once again, each corporation will have to turn things on/off, but it opens up huge opportunities. Even with what we currently have, there is a company called FlyHi that built out an awesome project using adilas API endpoints. Here is a video link to show some of what they did - [https://www.adilascontent.biz/videos/adilas\_flyhi\_demo\_sept\_2021.mp4](https://www.adilascontent.biz/videos/adilas_flyhi_demo_sept_2021.mp4" \t "_blank)  
  
Go public (publicly traded company) 96

This is an option. We don't have to exercise this opportunity. We want to be careful there but there are options. We want to watch out and protect each other as well. We have spent time, money, blood, and sweat on this company. We want to protect those assets. Inviting others in has some advantages but there are some risks as well. Think it out, plan it out, and get feedback from the rest of the team (co-owners). There are some on the team that don't want anything to do with this option. They like the smaller more family like feel.

**THREATS**

METRC, Payroll, Taxes  
(Federal, State, Ecommerce)

Metrc - They keep chaning things all the time. It takes a fulltime person(s) to keep up with that. If they change and we don't follow, our clients could be out of compliance. Payroll - This requires annual updates and changes, every year. Things could also change mid year. Often our clients end up letting us know right now. We have federal, state taxes, and documents and forms that need to be done and maintained every year. One other thing to note about payroll is that there are thousands of lines of code and it's not really something that just anyone could jump into. There is some specific flow, tables, values, and options. It's huge. Ecommerce and taxes - states, cities, and counties keep changing their tax rates. No big deal, we push that on to the user and they maintain that through location and tax settings. If could get really funky if and when new tax rules come out where online sales require you to tax based on where the person lives (not our tax rates - known - but the customer's tax rates - unknown). That could be really crazy! It seems like you need a small team to keep up with all of the unsolicited maintenance that is required. Once again, once the developers know what to do, they can do it. If you took a new developer and dumped them in to it, they would be lost for hours before being able to do much. Maintenance is a huge thing and big part of the puzzle. Most developers don't like to do maintenance. They like to build and create. Maintenance feels like drudgery or pulling teeth.

METRC/Bio Track - API changes  
that are launched without knowledge or warning

Once again, no one is really watching for this. We tend to get caught with our pants down. Then we scramble to get things fixed. We really need someone to help watch out for this and stay on top of it. This may be someone who is not a developer but who could let the developers know that something is coming and/or needed. Basically a compliance officer or something like that. That would really help.  
  
Frequencies of the third party updates.

Similar to the above. Things change and we have no knowledge of that until our clients tell us or a third party solution get back with us and says that something is not working anymore. We tend to build and then let go. We don't have anything in place for routine maintenance or updates. There is no contract and no expectations. They may assume that we will do the work for free to keep things up to date. We may need to charge a monthly fee for this (3rd party solutions) and Metrc/Bio Track type solutions. If they are using some of these heavy maintenance and compliance type solutions, we need to charge for that. Otherwise we take it on the chops. Pay to play!  
  
Web based solution - cybersecurity

This is a constant threat. It could happen from any place, any person, any size, automated, manual, mixed, etc. Our sites and servers are under attack every day, every hour. There are even things upstream of us that have shut us down. We get SQL injections, XXS - cross site scripting, network (DDOS) type attaches. We have some monitors in place to help us know how things are going and flowing but someone needs to watch and manage those pieces. PCI compliance for merchant processing and credit card transactions. We offer all kinds of payment options - carts, in person, ecommerce, pay after the fact, automated billing, etc. Email threats, phishing, SPAM, spoofs, viruses, trojans, etc.

Outsourcing servers to Data centers

This is a pro and a con. We save hundreds of thousands of dollars by doing this every year. On the treat side of things, we don't have full and total control. If something happens at the data center, we may not know about it and can't do anything about it. If the data center gets hit with a natural disaster, DDOS attack, power issues, Internet and pipeline issues, or whatever, we are at their mercy. Even if we have the bat phone for emergency calls - we can't guarantee that they will pick up and/or be able to help us. We have tied paying more for certain services (top tier or white glove stuff) and that has only been marginally successful. As a matter of fact, we recently stopped the current service due to lack of priority attention when it was really needed. Communication has also been a huge issue. We want to know what is going on, no one tells us, our clients get upset, and when all is said and done, we look bad, even if it was completely out of our hands.

Uniqueness & Trends - Apple/Google/IBM - 102, 103????

These are the giants. They could dump tons of money into the same space where we live. Event if our products are superior to theirs, our clients or potential customers tend to trust these bigger players more quickly. Ideally, if they (these big dogs) really started pushing on what we do. We would hope that the tide rises all boats, and they would basically be helping us out by educating the public to what we offer. We just don't have the compacity to keep up with the big dogs. We are only able to run as fast as we can go. The way our code is built, it take time to make massive or global changes. We are heading to a more object oriented type model and creating real distinction between backend, frontend, logic, services, database access, etc. That should really help speed things up for future changes.

John was saying that once we get a bunch of this stuff done, we need to circle back around and show ways to improve and/or get rid of some of these pieces. They still exist, just what is our plans and how can we alter things to make it better and better.

Look and feel

The look and feel are soooo subjective... Everybody has a different idea of what is nice, cool, hip, stylish, and what not. We really want to allow our users all kinds of options for look and feel. Either presets, settings per corp, settings per user, and even future custom CSS themes (data driven from the database vs hardcoded on the screen). We would like to upgrade to Bootstrap 5. Currently on Bootstrap 3. We also want to build as much as possible that we fully control and don't require any other dependencies.  
  
Overworked/under-payed co-owners

Threat on multiple levels. Training time, burnout levels, new guys (full time needs and expectations) vs older guys (semi-retired or self sufficient - not fully reliant on adilas funding). Certain pieces need to be built in... They can't be maybe's. It cost a certain amount to really play the game. Some of the pieces need to be built in to the cost of doing business. We have certain mission critical departments that need to be fully staffed, all the time. Communication lines need to be setup as well. On call stuff - if we need them, we need to give them authorization to still handle the extra stuff. Basically, a flex number of hours that are part of the budget. Contingency plan stuff. Make it part of the budget. We need some savings and a cash buffer to help if anything goes crazy. Speaking about budgets, we need to charge our clients enough to meet all of the needs and costs. Plan in savings, emergency, growth, maintenance, etc. We like to cater to the little guys. They don't always pay the bills. We have to figure out a good mix. John likes to say - not all business is good business. We are semi independent co-owners. There is a small cross over between co-owners and employees. We need to look at who does what, what hours or time commitments, and who is controlling all of that. We are called co-owners but sometimes it feels like a hybrid employee type relationship. Co-owners need to be compensated enough to cover their needs. There are all kinds of other costs to play that game. Health insurance, benefits, taxes, computers, software, paid time off, rent/office space, Internet, gas, holiday pay, overtime, etc. Tax laws keep changing - there were a bunch of expenses that may no longer be covered as a write-off. The co-owner ends up eating those costs. All of these things need to be taken into consideration.  
  
Mashups - cherry picking technologies and  
mixing it together

On the front lines, this is a huge threat. Other companies can make it look like it does all kinds of stuff by mixing a number of existing pieces. It looks super awesome at the sales level. Then later on, when you want everything to really go full circle, you end up with problems because it is not all fully integrated. You need a full system for that. We sometime lose out on the sales side, they can make it look easy and powerful without doing much more than blending it together. John was saying, this is a strength for us, a weakness, an opportunity, and a threat. Depends on how you look at it. How do you show a simple, powerful, and pretty system that is as vast as adilas (fracture)? That gets really tough. Currently, our system feels big - massive or overwhelming system. We need a way to dumb this thing way down or put a bunch of it behind a curtain. We need the power and full capabilities, we just need a way to present it in smaller bite sized pieces. Once again, a big plug for fracture and where we are heading. We need to use technologies wisely. You can get in trouble if you use libraries that are dependent on other libraries. If we can control those pieces better, it actually makes us faster and more nimble in the future. Be careful of some of the code dependencies. If it is not easy to change it, we may want to look into not using it or coming up with other options. Things keep changing, assume that it will keep going and changing! John and I were talking about data driven pages. We put everything into the database and then have the pages pull and show what it needs to. Keep it really simple. Think - everything needs to be data driven. The actual pages (GUI or UI/UX) just loop over data and show what they need to. Everything is fully data driven. One other little thought, dealing with CSS and themes - we could just make our own and then standardize all of our pages to use our own CSS and themes. If something changes, no problem, it is already built in to cascade those changes.

Time to market

Lots of questions here - go fast and focus your efforts or go slower and really go deep and knock it out? Are we talking time to market on coding? projects? funding? products? services? plans? etc. It can get deep. Sometimes the theory is good but it may not have been built with a certain scale or other variable factors built in. It all comes back to time to market. Just being silly here - but we almost play the game in a building frenzy type mode... We just push and push and push and don't market anything. We may need to slowdown and really try to sell what we making and building. Ads, billboards, ad campaigns, social media, radio, TV, YouTube, banner ads, company vehicles, etc. Currently, we do some word of mouth and that's about it. This may require more budgets and managing those campaigns and budgets but it could really pay off in the long run. This may not go here, but we could outsource some of the pieces. There are pros and cons to both ways. If we are in there deep, we know it. That does take time and resources but we can virtually kick the tires and really make sure we get what we want. We are in an interesting place - we are a legacy start-up. Currently, we are allowing our guys to work on both ship A and ship B. We may need to setup some standards and some rules. ROI - return on investment - how quick or how long are we willing to wait to get this ROI? MVP levels, prototyping, and quick test files. This may help us decide where we want to go and/or jump. We need to play the game in order to really know what it will take.

Cutting edge can cut deep

It may look and sound cool but sometimes there are hidden bugs and/or features that can bite you in the butt. Market research and trying to listen to our clients. Sometimes it is better to be just behind the bleeding edge vs on the bleeding edge. We need to consider marketing, code, servers, databases, hosting, backup, etc. Where do we want to play - ideally find the sweet spot and play there. That sweet spot keeps moving as well, plan for that.  
  
inability or a lack of funding  
(can take a 1-3 year project and turn it into a  
decade project)

We have put a bunch of code on the shelf... That hurts... This is tied to funding, teams vs individuals, project managements or lack thereof, and priorities. If the funding was there, we could do so much more. Including cross training, standards, marketing, etc. Lots of our guys would like to do more than just code. We would really like to use some strategic small teams and sprints.