Sprint Planning Meeting – 2 hours

1:00pm – Intro by each developer

* Alan – Planning and design, also enjoys the puzzle of the backend. Building tests to make sure things work.
* Bryan – Jack of all trades, project management, loves to learn
* Brandon – Looking to be a player and help it go
* Cory – Looking for a new process, excited about user stories, better understanding and communication, get to the end (goals) for the project
* John – Sys admin, coding, business logic, learning more about the system by working with others, likes the frontend
* Wayne – He loves coding. He doesn’t like the UI stuff. He loves the backend stuff. Testing and dev ops.
* Eric – Design, database modeling, business analysis and design process, process improvement projects, building things that work – likes the design portion of the development cycle

1:05pm – Training

Intro to Scrum (videos/discussion)

* Backlogs, sprints, ceremonies
* YouTube – How Scrum Works – Agile Coach (2018)
  + Small discussion from the team
  + 3 artifacts – product backlog (to do list), sprint backlog (what you are currently working on), increment (what is done from the sprint)
  + 4 ceremonies – sprint planning, daily stand up, sprint reviews, sprint retrospective
* YouTube – Scrum Roles Explained – Agile Coach (2018)
  + Product owner – manages the product backlog
  + Scrum master – servant leader to help it happen and go forward
  + Development team – core team who make it work
* YouTube – Sprint Planning Basics – Agile Coach (2018)
  + What we are doing?
  + How are we going to do it?
* YouTube – Sprint Planning Meetings – Agile Coach (2018)
  + Retrospective recap from prior sprints – recap and moving forward
  + Update from the product owner – any new updates and priorities
  + Sprint goal and velocity – why? And How much can you get done?
  + Make sure and review or recap before moving forward

1:20 – Retrospective Recap – Alan

* For this first one, we don’t have one. Next time.
* Product and Market Update – Cory
  + There are some up and coming projects for a company called AWH
  + Vendor Catalog – known need for companies that want a master catalog or master list of items.
    - Enterprise systems (master systems)
    - Transaction systems (smaller corps that want to use the master lists or catalog entries)
    - Data standardization
    - There is already an existing catalog called the vendor catalog – done by Alan earlier in 2021

1:25pm – Planning Conversation

* Sprint Goal
  + Train and get exposure to scrum
    - Backlog management
      * Current process – Cory
      * Our existing processes are:
        + EOT – community funded projects

Someone (admin or a client) presents a project and she creates a project in EOT.

She asks Steve or Brandon about it and gets a rough idea on what it may take.

We sometimes don’t create a new project unless we really get a “yes” from admin or a client

When we are ready, it would go into a queue, get more info, and get it assigned to a developer

Cory would then get follow-up, updates, and info and would pass it on to the client

Cory did a small demo on her current process, inside of adilas. Search, add new, setup budget, assign if known, add small notes, upload documents and photos. She then lets the developer know the project id and then adds it to the project queue.

* + - * + Project Queue
    - Testing (unit, integration, manual, dev server)
      * Wayne really wants to get this process buttoned down and more standard
    - Code review / coding practices
      * Option to learn from others while doing this
      * John brought up the adilas docs – aka the adilas style guild and code snippets
    - Code management (bitbucket)
      * We used to just do a branch and then merge it in… now we are all working on a single project and how does that work?
      * Work on the “git” to
      * Making the modules of our code working independent as much as possible. One person responsible for a single function. Trying to not step on each other. Coordinating tasks.
        + For example: Database pieces, services, DAO’s, functions, business logic, etc.
      * Rough overview… main project branch, as many sub branches as needed, merge into testing as needed, eventually merge into master
    - Transitioning/handing off stories
      * Layers within your code
        + UI/UX – GUI (display and user interface)
        + Business logic
        + Database access objects
      * Passing things over to testing
    - Scrum board using EOT – Brandon
      * This will be our phases and virtual columns
      * We will show how to use this in one of the daily scrum meetings
        + Sprint To Do
        + Planning/Design
        + Development
        + Testing
        + Code Review
        + Accepted/Approved
        + Deployed
  + Wish List items
    - Build out EOT to do more of what we need
    - Help get other’s projects across the finish line

1:35 pm – Velocity

* How to determine story points?
  + Risk
    - Unknowns???, outside parties, bottlenecks, distractions, bugs, time management, health and family issues, etc.
    - Is that project or change going to open up a can of worms?
    - Who is in charge of what and does that create a bottleneck?
  + Complexity
    - Understand, explain, implement, design, testing
    - Teams – one, more, etc. Coordinating
    - Skill levels
  + Repetition/Familiarity
    - Have you been there before? Can you get back in to it? Do you need some training before you start?
    - The whole adilas project keeps building, growing, and moving.
    - Testing may help here as well
    - Standards – eventually, we are going to be more on the same lines and doing things over and over again. We will become more efficient and doing things in a standard process.
  + Analogy of Chickens and Pigs (imagine a breakfast – a chicken makes a contribution, a pig makes a sacrifice) – who gets to make those calls?
* How to manage technical debt (bugs)
  + Any type of fixes that need to made to existing code.
  + How much time, of our current sprint, will be used to fix past pieces (not part of the current sprint)?

1:40 – User Stories

* What is a user story?
  + As <user role> I want to <do something> so that <value statement>
  + Who, what are they doing, what benefit or outcome is accomplished?
* INVEST
  + Independent
    - Reduced complexity
    - Code once, use many
    - Easier to test
    - Able to do in any order
    - Can it stand on its own
  + Negotiable
    - Flexible – think settings… we want to reuse as much as possible
  + Valuable
    - What does this bring to the table?
    - Think about other options – who else may want or need this function or process?
  + Estimate-able
    - Small enough to figure out the cost and time (story points)
    - Breakdown unknowns into smaller known pieces
  + Sized Appropriately
  + Testable
* EPIC
  + “As a manager with multiple corps, I want to manage items from a single corp, so that I don’t have to duplicate work.”
  + A large (scope) user story – overview or bigger picture type statement
  + Super broad, general, and wish list level
* Story Slicing
  + What are the benefits?
    - Able to assign different developers
    - Details
    - Known start and ends – parts of the bigger whole but independent
    - Bite size pieces vs the whole meal at once
    - Refinement – helping to get the backlog more and more ready to go and/or implement
      * There is a process of breaking things down, refining it, refining it again, etc.
      * Say you had a small piece that was assigned 15 story points… We want to keep breaking it down until it has 1-3 points per section. So, say we had 5 steps of 3 points a piece. Or 5 steps with 2 points a piece and 5 other steps with 1 point a piece.
  + Best Practices?
    - Bad practices – we don’t want to split it up into layers such as the gui will do…, the business logic will do…, and the database will do… - hard to coordinate and certain things may require the other pieces (dependencies and making things independent as much as possible)
    - Break things up based on features
      * CRUD – Create, Read, Update, Delete/Destroy
      * Mixings GUI, business logic, and database pieces
      * You could break things up based on acceptance criteria
      * Break it up by pages, screens, processes
      * It’s ok to use defaults, static data, mock-ups, prototypes
        + We will mix and blend later on to bring it all together
  + Acceptance Criteria
    - What is acceptable and wanted?
    - MVP – minimal viable product (plan, person, etc.)
    - TDD – test driven design – minimum necessary (process to use to create the MVP)
    - Drawing lines in the sand – known scope and boundaries (get rid of creep)
* Slice OUR epic Together
  + Our epic – create an item catalog
  + We know that we need global lists of part categories, unit of measure, vendors (already done), parent attributes, photos/images, media/content
  + What is wanted? Acceptance criteria
  + Bryan – brought up a past project from invoices to PO’s to cross between corps
  + We need a master list (with an API)
  + Per server for now – MVP
    - Next step – would be master/master list and then allow for a subscribe level
  + Wayne – maybe write an API socket that is server independent
  + Cory – future project or phase – the master image list and product image catalog
  + Bryan – would like to see (some training) how the vendor catalog works and is there parts and pieces that we could harness from that project?
    - Alan wrote this and may be able to do and give some training on that, to let them see the pieces.
    - We know that eventually we will be parts, items, customers, chart of accounts (expense types and deposit types), etc.
  + Push from master to transactional systems
  + Pull from enterprise to transactional systems
  + Permissions – who can do this… and locks and un-locks
    - We can create new permissions as needed
    - As a note, the vendor catalog looked at some cross corp settings (enterprise level settings) and then checked permissions after that.
  + We need new items (every time) vs just a cross corp mapping
  + Which pages will be exposed to these changes?
  + At the top level, login based, we need to look at is the current corp part of a bigger enterprise system
  + Let’s break these pieces into their own mini user stories – We eventually want a full master item list, these are some smaller pieces of the whole: global lists of part categories, unit of measure, vendors (already done), parent attributes, photos/images, media/content
  + Note from Alan – Agile and scrum is not a top down command type approach. It is a more team based approach based on MVP requirements. Ownership comes from the team. This may take a little bit to change our mindset.
* Create scrum board and pick tasks

2:55 pm – Recap