**Member/s: Dylan, Jackson, Joshua Cory, Philip, Isiah, Tanner, Shaun, Stephen**

**PO: Shaun**

**SM: Stephen**

**Comms: discord**

**Tasks: Trello**

**UI Standards:**

**Coding standards:**

* **Document attached.**

**Repository - GitHub**

**Def of Done**

* **Code review**
* **Happy path and unhappy.**

**Define UI standards to follow (e.g. display must fit on phones or computers; displays must resize automatically)**

**Define coding standards to follow**

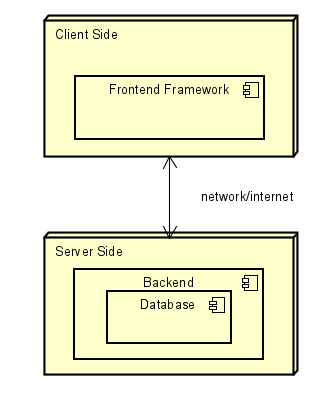
**What is the architecture?**

**What is the production environment? Blazor visual studio**

**What is the development environment (GitHub, Trello?, ide?, )? Github**

**What is the language? C#**

**What is the technology stack? Visual Studio, Sql express 2019**

****

**Define a Definition of Done (e.g. designed{e.g. sequence or interaction diagram}, built, reviewed {meets coding standards, fully implements design, follows good coding practices}, tested {happy path and all known unhappy paths}, committed {the code and the tests})**

**Define the team’s sprint velocity (the duration is 1 week, 2 class periods plus at least 2 hours so that’s a minimum of 6 hours each person on the development team each week; eg. 9\*6=54).**

**Steps for each sprint:**

**PO adds the user stories to the product backlog and the whole team grooms them based on priority**

**Sprint duration is 1 week; ends the last hour of the last class session of that calendar week.**

**Sprint planning: Recurring 1st day of each sprint, time box for 15 minutes the first day of the sprint**

**Revisit sprint goal; update as necessary**

**Select the Product Backlog user stories to be delivered in the sprint (includes a review/discussion of the stories/estimation)**

**Break them down as necessary to <4 hour tasks**

**Daily Scrum/Standup: Recurring daily during sprint, time box for 15 minutes.**

**Add a 16th minute for additional collaboration, time box for 15 more minutes and dismiss any team members not necessary for discussion**

**The last hour of the last class day of a sprint**

**Backlog grooming: max 30 minutes, (not limited to the scope of a single sprint;**

**meant to ensure the highest priority backlog items are detailed enough and estimated so they can be used in Sprint Planning;**

**ideally, enough stories are detailed for at least 2 sprint’s worth of work so you have enough for the next sprint**

**plus more if you can pull additional work into the next sprint)**

**Sprint review: max 15 minutes; this is akin to acceptance testing with the customer/product owner.**

**Sprint retrospective: Remainder of the last hour (min 15 minutes).**

**team discusses changes to daily routines, definition of done, ways to improve (e.g. training and or pair programming participants).**

**Backlog Grooming (at the end of a sprint):**

**The product backlog should have already been prioritized by the PO going into this exercise.**

**Do you have enough stories detailed out for the next sprint? If yes, do enough for one more sprint or till time runs out.**

**If no, detail out at least enough for the next sprint (even if you go over on time some). The PO and SM should take action to improve the backlog.**

**Talk through each story: (also see INVEST https://www.knowledgehut.com/tutorials/scrum-tutorial/user-stories)**

**What are the words behind the words**

**Risks/Constraints/relationship to other stories in list**

**What is the acceptance criteria (may only need this for the main story but helpful to have it for the subtasks/substories as well)**

**Can we estimate to be < 4 hours (if no, keep discussing or break into multiple stories)**

**Estimate it (in hours based on definition of done)**

**Sprint Planning (at the start of the next sprint)**

**Revisit the sprint goal and adjust as necessary based on the highest priority product backlog items**

**Make sure all stories associated with the sprint’s goal are estimated,**

**Prioritize them within the sprint**

**Add stories to the sprint backlog until the target performance estimate is reached (and no more)**

**Sprint Review -**

**PO/Scrum Master should have a list of acceptance criteria available based on the stories selected for the sprint during sprint planning**

**Step through each acceptance criteria; any feedback can be captured as a new task/story in the product backlog for the PO to prioritize.**

**These then get pulled into the next sprint's planning if high enough priority.**

**Once approved, deploy the software**

**Sprint Retrospective -**

**Everyone writes down up to 3 things the team did well and then discusses them (eliminating duplicates)**

**Everyone writes down up to 3 things the team could do better and then discusses them (eliminating duplicates)**

**The Scrum Master and the PO do not get to contribute but they do attend (this varies among Scrum implementations)**

**The SM then tracks these items and helps the team remember them going forward.**

**Ways to exceed expectations: You don't need to do all of these, or even any of these, but the following will**

**hopefully give you some idea of what I consider exceeding expectations during the semester project:**

**1) work more than the 6 hours per week I expect**

**2) Be proactively collaborative (that is, answering questions others have is expected but checking around with**

**teammates or even other teams to see if there is anything you can help them with; that's exceeds)**

**3) Follow the processes and document that you did and why you did (following the process is expected;**

**documenting that you did and why is exceeds).**

**4) Remember that ideally another semester class will build on top of what your team does. Make it easier**

**for them to be able to do so (commenting code, documenting designs, build readme files, how to deploy, etc).**

**Anything your team struggles with should be documented for the next class so they don't. That is exceeds.**

**5) Solicit feedback from others outside of class on what you are doing and how, e.g. show the gui and get**

**feedback on it from someone unfamiliar with the project. Even if the gui is partially done (like only the**

**tasks are displayed or you can only manually update the tasks being completed).**

**6. If your team (as a team) does code reviews, testing, and documentation; that is exceeds.**

**None of the above items requires the product to be complete in order to do it. Of course, completing all the**

**functionality would be exceeds (as long as you correctly applied Agile and Scrum to do it).**