COMP4920 19T3 Project Plan

Interactive UNSW Degree Planner

Group Name: On Course

Alexander Rowell	z5116848
Eleni Dimitriadis	z5191013
Emily Chen	z5098910
George Fidler	z5160384
Kevin Ni	z5025098

Submission Date: 20th October, 2019

1. Introduction

The degree-planning process can be a pain-staking one. There are lots of prerequisites to keep in mind, UOC for different categories of courses, and a plethora of course choices. It can be difficult to navigate through the handbook to look for courses that fit with your current study progression. Depending on each individual, you may resort to trying to plan your degree in dot-point form on your phone or with a spreadsheet that takes you an unreasonably long time to format and get right.

It would be much clearer for many students if there was a more streamlined degree planning tool. This is where our “On Course” project comes in. We aim to build an interactive degree-planning tool that consolidates many of the resources you need to look at to make sure you graduate on time.

2. Aim

We wish to build a system that will allow students at UNSW to plan their courses of their degrees at a high level, taking into account course prerequisites. We would like to present a streamlined experience to the user, providing an interface that will display course descriptions and the set of available courses based on the courses that have already been completed.

3. Product Backlog and Release Plan

We have categorized our tasks into six broad epics.

Epic Summary	Name
As an administrator I want to be able to edit courses in the database in order to correct any mistakes encountered during parsing	Data Populating
As a student, when my degree offers me a choice in courses to take I would like to make that choice within the planner so the final schedule is complete and reflects my wants.	Course Choosing
As a student, I would like the system to make it clear when I have likely made a mistake in planning my degree so that I do not accidentally fail to meet my degree requirements	Mistake Handling
As a student, I would like to receive relevant information when using the timeline planner so I can make informed choices in planning my courses without doing extra research.	Relevant Information

As a student, I would like a degree plan to be generated for me based on my needs relating to my university experience so that I do not have to do so manually.	Plan Personalisation
As a student, I would like to have an intuitive timeline interface to make planning my degree easy.	Timeline Interface

Below we give our prioritised product backlog with corresponding release dates. Our sprints will be 1 week in duration. Only the role-goal-benefit statements and estimated size are given for each user story for brevity. The acceptance criteria can be seen by examining our Pivotal tracker.

Each story point is approximately equal to 1 day of work for 1 developer.

For our first release, we are restricting the scope of degrees to just the Computer Science (3778) degree to get more of our functionality set up. Then in the second release we will increase our degree offerings. We are allowing 2 weeks for our first release as we predict that our velocity will be slower as we need to set up more pieces of infrastructure at the beginning of the project. After that, we will have a release each week as we should be more productive in our established development environment.

Week	**Prioritised User Stories** [Estimated Size]
6	• As a student, I should be able to select a degree (Computer Science 3778), so that I can use the tool for the particular requirements of my degree. [3 SP] [Plan Personalisation] • As a student, I would like to view a generated schedule for my core units, so I can better understand the structure of my degree. [2 SP] [Plan Personalisation] • As a student, I would like to be able to access information about my degree, so that I can understand the requirements. [1 SP] [Relevant Information]
7	• As a student, I would like to be able to view the information about any given course, so that I can make an informed choice. [1 SP] [Relevant Information] • As a student, I would like to see my degree structure represented graphically on a timeline, so that I can better understand the progression of my program of study. [2 SP] [Timeline Interface] • As an administrator, I would like to populate our database of courses and degrees with simple requirements automatically for use in the rest of the system. [2 SP] [Data Populating]
Release 1: Minimum Viable Product	
8	• As a student, I would like to be able to save my course plan, so that I can

	view it offline later. [1 SP] [Timeline Interface] • As a student, I would like to be able to add courses to my timeline, so that I can build upon the base plan for my degree. [1 SP] [Timeline Interface] • As a student, I would like to be able to remove courses from my schedule on the fly, so that I can correct any mistakes. [1 SP] [Timeline Interface] • As a student, I would like to be able to customise my schedule on the fly so that I can experiment with different options. [2 SP] [Timeline Interface] • As a student, I would like to be made aware if my current proposed degree structure violates any unit of study requirements, so that I know when a selection is not allowed. [1 SP] [Mistake Handling] • As an administrator, I want to be able to edit courses in the database, so I can correct any mistakes encountered during parsing [2 SP] [Data Populating] • As a student, I would like to be able to input previously studied courses so that I can plan a degree which is already partially complete. [1 SP] [Plan Personalisation]
Release 2: Interactivity and Increased Customizability	
9	• As a student, when my degree offers a choice between core units (choose X of Y for X < Y), I should be able to choose which I take, so I am aware of my choices and can tailor my core units to my preferences. [1 SP] [Course Choosing] • As a student, I would like to see a summary of the degree requirements remaining assuming all units currently on my schedule have been completed, so that I am aware of which units I still need to account for. [1 SP] [Mistake Handling] [Relevant Information] • As a student who is in a slightly different version of a program to what is currently offered, I should be able to select the year I started to get a plan that fits that version of the program. [1 SP] [Plan Personalisation] • As an administrator, I would like to populate our database of all courses and degrees automatically for use in the rest of the system. [4 SP] [Data Populating] • As a student outside of the CSE faculty, I would like to be able to plan my degree, so that I can reap the benefits of having an interactive tool as well. [2 SP] [Plan Personalisation] • As a student in a degree with multiple possible majors, I should be able to select a specific major and possibly minors that I want for my degree. [2 SP] [Plan Personalisation] • As a student, I would like to be able to select to include summer term study so that I can plan a degree that utilises this term. [1 SP] [Plan Personalisation]
Release 3: Advanced Course Requirements	
10	Reserved for bug-fixing and a buffer for a slower velocity than predicted.

Below we detail the rest of our prioritised backlog. These are features that we would like to include in the product, but we do not think we have time to fit into the rest of the term. If, as our sprints progress, we find that our velocity is faster than predicted, then we can move user stories from this backlog to our releases.

RGB Statement	Estimated Size
As a student, I would like to be able to indicate the number of UoC I want to study per term, so that I can plan a program that is compatible with my external schedule. [Plan Personalisation]	1 SP
As a student, I would like to be able to indicate any exemptions which have been granted by my faculty so that these can be accounted for when planning my degree structure. [Mistake Handling]	2 SP
As a transfer student from another university, I should be able to enter credit received that does not match a specific course (e.g. 12 units of computer science electives). [Plan Personalisation]	1 SP
As a student doing a combined degree, I should be able to select the combined degree and have a plan that handles both at once so that I don't have to separately plan each part of my degree. [Plan Personalisation]	4 SP
As a student, I would like to view the options I have available for elective units so that I can decide which options suit me. [Course Choosing] [Relevant Information]	2 SP
As a student, I would like to filter my elective options by area of study so that I can more quickly access the options that are relevant to my interests. [Course Choosing] [Relevant Information]	2 SP
As a student, I would like to know which classification of elective a given course would be in regards to my degree so I can accurately plan electives needed to complete my degree. [Course Choosing] [Relevant Information]	1 SP

The full details of our project plan can be viewed via our project on Pivotal Tracker (https://www.pivotaltracker.com/n/projects/2401184).

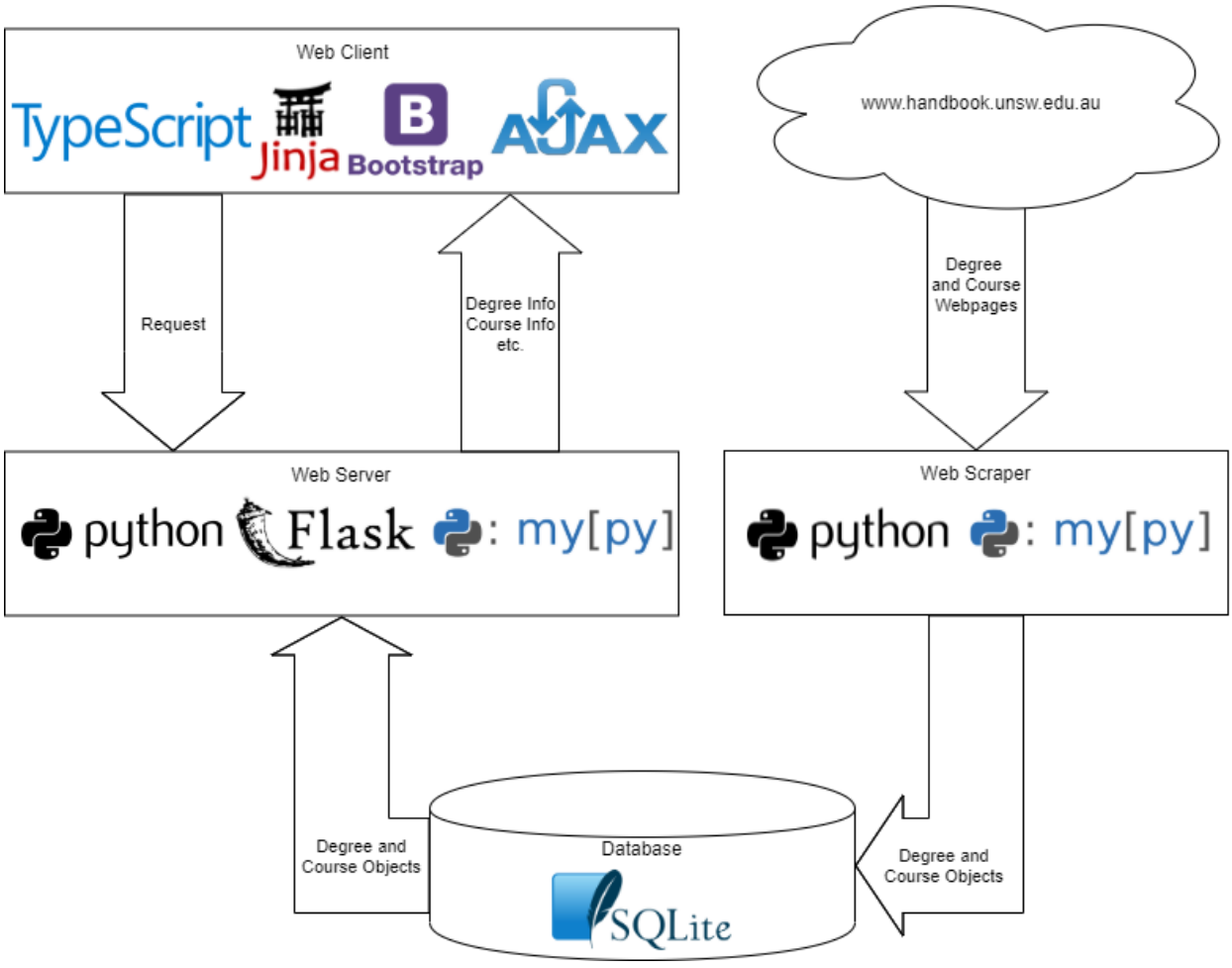
4. System Architecture for MVP

In the next few sections, we will detail the system architecture we have derived. This system architecture was designed mainly with the MVP in mind. However, extensibility to our full envisioned requirements was kept in mind in the design.

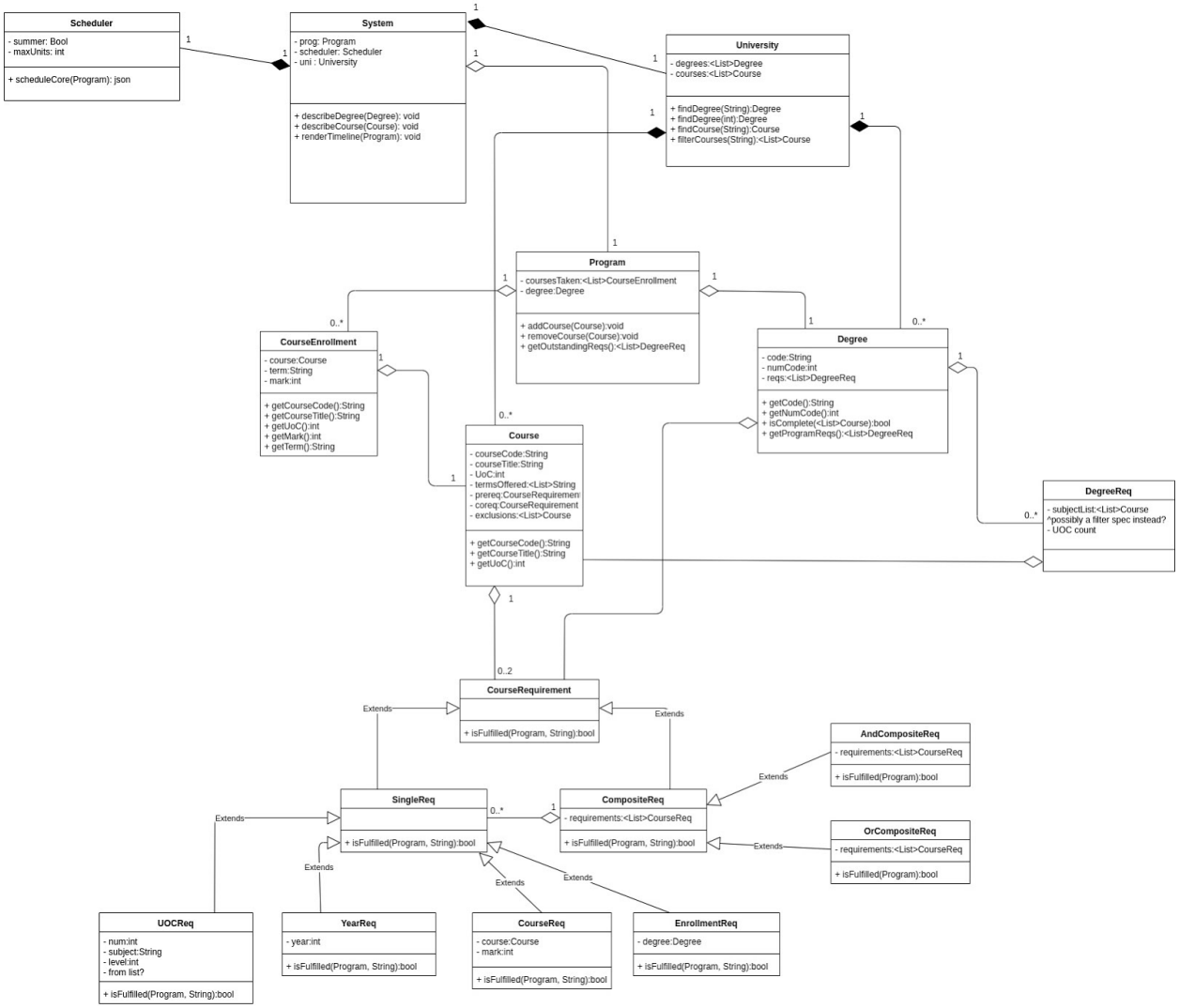
4.1 Technology Stack

Back-end: Python (Mypy), Flask
Front-end: TypeScript (React), Bootstrap, Jinja, Ajax
Database: SQLite3
Web Server: CSE machines

4.2 System Architecture

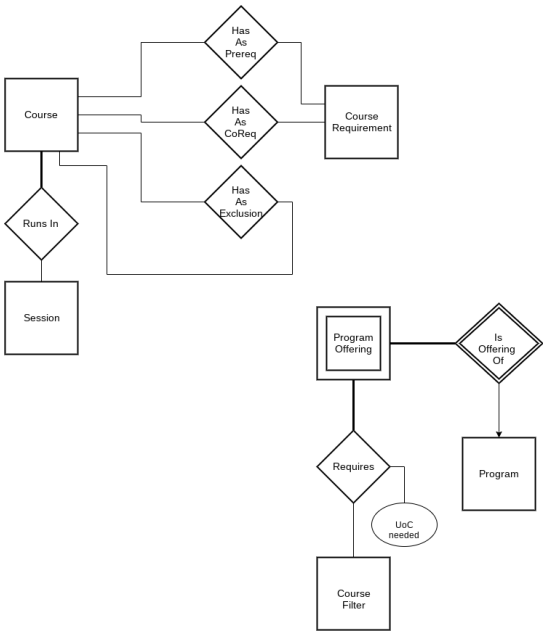


4.3 UML Class Diagram

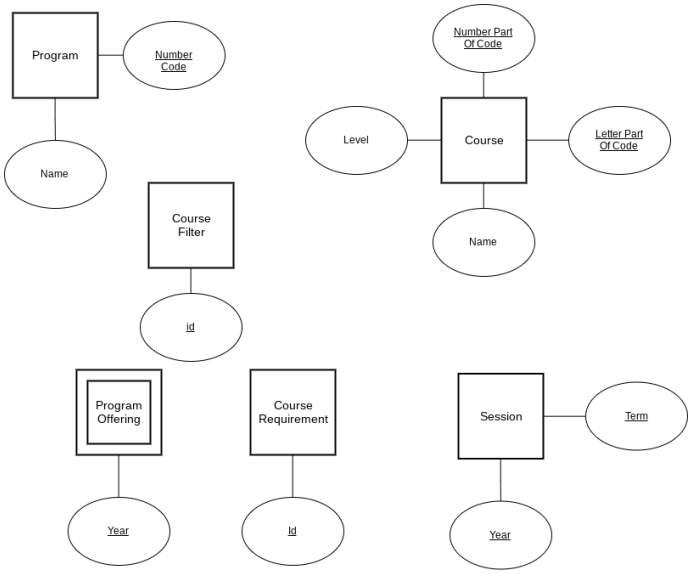


4.4 ER Diagram

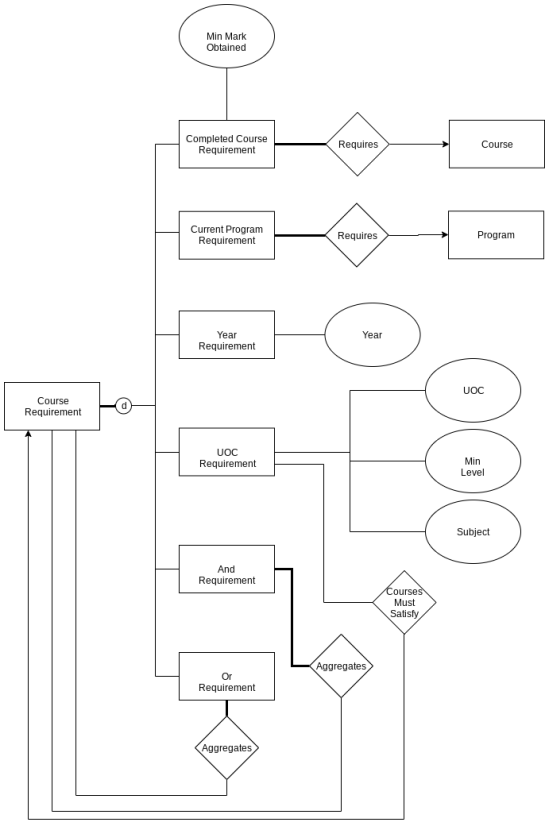
Main Entities and Relationships:



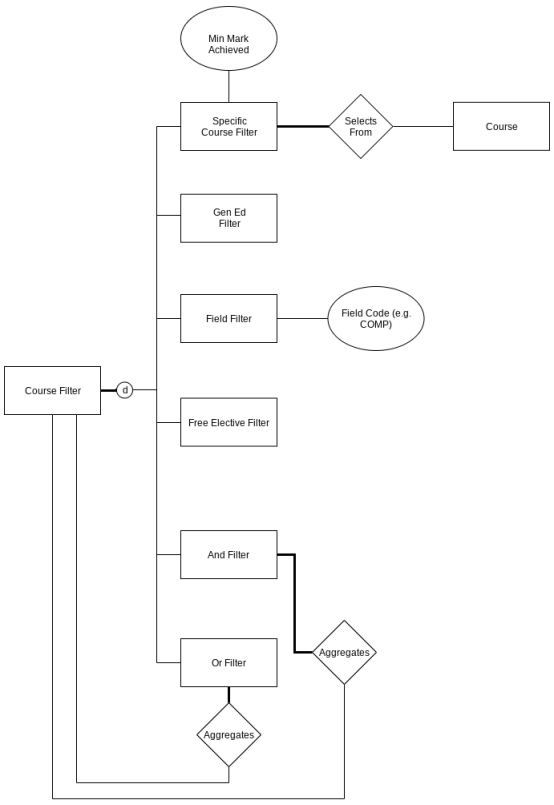
Properties:



Course Requirements:



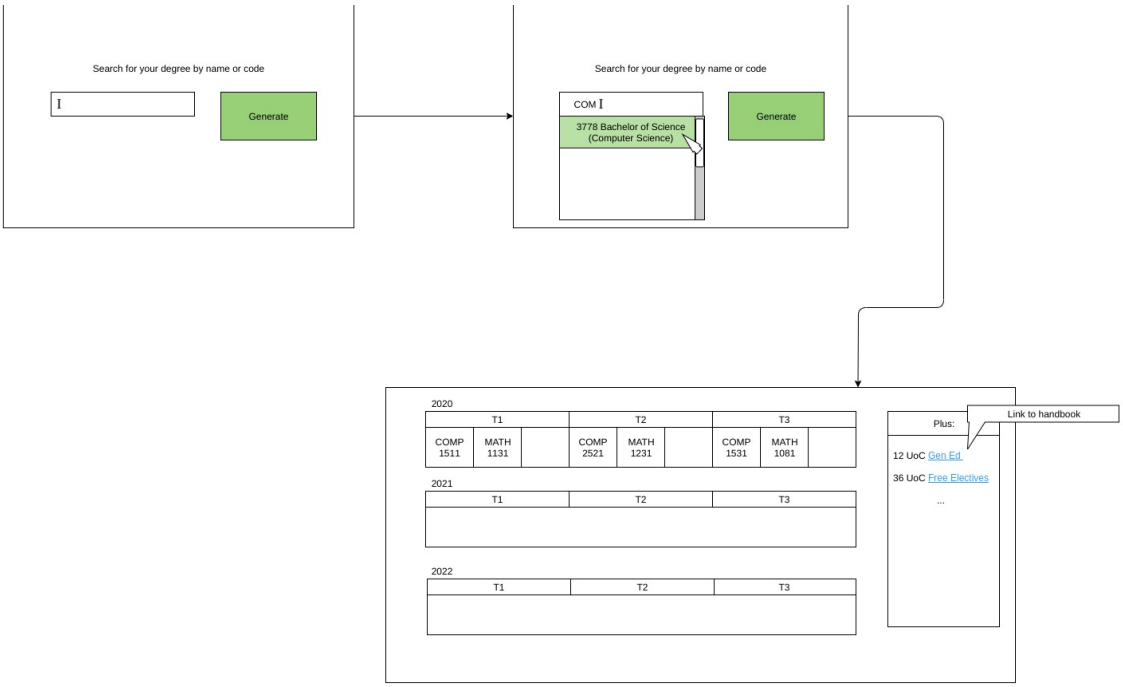
Course Filters:



4.5 Wire-Frames

4.5.1 Release 1 Wire-Frame

This is a simpler interface to reflect the simpler requirements of release 1.



4.5.2 Release 2 Wire-Frame

This interface has more functionality that will be implemented in release 2. This diagram has been included at the end of the document.

5. Team Management

5.1 Allocation of Roles

Scrum Master: Emily Chen (z5098910)
Product Owner: Eleni Dimitriadis (z5191013)
Developers: all team members

5.2 Meeting Schedule

We will have 2 main types of meetings, daily standups and sprint-oriented meetings. This is in addition to the sprint reviews in our allocated class time.

5.2.1 Daily Standups

Daily standup meetings will occur on Messenger chat. All members will answer the 3 questions “what did I do yesterday?”, “what will I do today?”, and “is anything in my way?” in the morning. All members will have to acknowledge that they have read each member’s standup. The standup will also be recorded in our personal diaries. Online standups are used in place of in-person standups since the constraints of our schedule means that it is not practical for us to meet in person everyday.

5.2.2 Sprints

Week	Weekly Check-in	Sprint Review	Sprint Retrospective, Planning
6	Thursday 10 - 11 in person	Friday 1 pm with the meeting group	Monday 2-3 pm call
7	Thursday 10 - 11 in person	Friday 1 pm with the meeting group	Monday 2-3 pm call
8	Thursday 11 - 1 in person during seminar	Friday 1 pm with the meeting group	Monday 2-3 pm call
9	Thursday 11 - 1 in person during seminar	Friday 1 pm with the meeting group	Monday 2-3 pm call
10	Thursday 11 - 1 in person during seminar	Friday 1 pm with the meeting group	Saturday call

5.3 Extreme Programming Practices

We will abide by the following Extreme Programming practices to ensure our deliverables are of high quality.

5.3.1 Test-Driven Development

We will require that our code is tested as it is developed. This means that all back-end functionality should have test cases with `pytest` developed before the functions are implemented. This will also allow us to easily check that any modifications during refactoring maintain the correctness of our functions. We will also use `Selenium` and `Jest` to test our front-end.

5.3.2 Small Releases

Our releases do not encompass many features at a time and we have many of them (the second and third are equivalent to our one week sprints). This allows us to get feedback often and adapt as needed.

5.3.3 Refactoring

Our initial design is focused on our MVP only. As we increase functionality, we will refactor the code as necessary to streamline operations.

5.3.4 Keeping it Simple

As mentioned above, our initial design is only for the MVP. We leave pathways to extend our design, but we will only implement what we need for the initial release when we start.

5.3.5 Continuous Feedback

We will have daily standups to ensure that everyone is informed of the progress made so far. We will also meet in person weekly to discuss and pair-program through any issues that we may have. We will also have weekly calls to plan our sprints and reflect on past sprints. This allows many opportunities for critical reflection on our practices and to adapt as necessary.

5.3.6 Style Guide

We will use the following style guide to streamline our operations:
Python/mypy

- Follow Google's Python style guide: http://google.github.io/styleguide/pyguide.html
- Python functions should be type annotated and checked with `mypy`

React/typescript

- Follow airbnb's React style guide: https://github.com/airbnb/javascript/tree/master/react
- JS functions annotated and checked with typescript

5.3.7 Velocity Tracking

We will track our velocity on pivotal and inspect the burnup charts weekly in our sprint retrospective meetings. Based on this and our individual velocities, we will figure out how to distribute the workload to maximize overall velocity.

Final Release Wireframe Diagram