# Project Plan

**Task Estimation in Scrum (Adam & Darren)**

* Estimation Matrix: Time, Difficulty & Extent of unknowns
* Planning Poker: Each scrum member puts forward a value, average is taken as the story points for the story being estimated.

**Code Reviews (Connor & Adam)**

* Attention to traditional conventions and patterns
* Reusing previously created/tested methods rather than creating anew

**Additional Topic: Testing / Test Coverage (Darren & Connor)**

* Unit Testing
* Integration Testing
* Mocking
* End-To-End Testing

**Future Plan**

* Get a list of more specific topics under our main headings
* Split each section into tasks for individual contributors
* Execute on assigned tasks
* Merge changes with co-contributor
* Assess end result and apply formatting

2nd Meeting: Assigning Tasks

**Task Estimation (Adam / Darren)**

* Planning poker
* Estimation matrix
* Affinity Estimation
* Benefits: Improved planning and prioritization, transparency and collaboration
* Limitations: Time-consuming and lack of action. Lack of accuracy, unknowns obscuring time-frame.
* Break down larger tasks
* Roughly estimate items further down the backlog
* Use a variety of estimation techniques: top-down estimate, bottom-up estimate, analogous estimating, parametric estimate, three-point estimating, and what-if analysis
* 3 amigos, involve QA, front-end and back-end in full vertical slice meetings
* Don’t rely solely on estimates.
* Use story points
* “Just enough” documentation

**Code Review (Connor / Adam)**

* Reusing code
* Importance of code reviews
* Code-review checklist
* Code review tools and technologies
* Patterns and Architecture / Software Quality
* Process for fixing issues on code reviews.
* Proper communication: The code is wrong instead of you are wrong
* Effectiveness of code reviews
* Monthly check-ins to deal with recurring problems
* Standards on what the code review is trying to achieve.
* Best practices of code review
* Meetings around coding practices in a larger dev group, where outside teams can share personal experiences.

**Testing (Darren / Connor)**

* Test one thing at a time
* Test edge cases and flows
* Quality Control processes
* Clear and concise test cases
* Quick sanity checks before overnight regression testing
* Ensure smooth integration with other components
* Adhere to a test-coverage guideline.
* Ensure the tests accurately describe the function they are validating
* Ensure the tests do not have obsolete functionality
* Mock other aspects of the test which are not directly being tested currently