



Quality Assurance Manual

QAM/1.0

MEng Year 3

**Department of Electronics
University of York**

Software Engineering Group 4

Document Control

| Version | Modified By | Date | Section Modified | Remarks |
|---------|-------------------------|------------|-------------------------|---------|
| 1 | J Pearson | 05/12/2019 | All | Updated |
| 1.2 | Entire Team | 31/01/2020 | All | Updated |
| 2 | J Pearson | 11/03/2020 | 4.4, 4.5 and Appendix A | Updated |
| 2.1 | J Pearson, Louis Newman | 10/04/2020 | 4.7 | Updated |
| 2.1.1 | J Pearson | 10/04/2020 | 2.2.1 | Updated |
| 2.1.2 | N Billis | 13/04/2020 | 2.6.1 | Updated |

Table of Contents

| | |
|--|-----------|
| 1. Introduction | 5 |
| 1.1. Company Profile | 5 |
| 1.2. Vision | 5 |
| 2. Roles and Responsibilities | 5 |
| 2.1. Organizational Structure | 5 |
| 2.2. Project Leader | 6 |
| 2.2.1. Role Description | 6 |
| 2.2.2. Risk Management | 7 |
| 2.2.3. QA Metrics | 7 |
| 2.3. Financial Manager | 8 |
| 2.3.1. Role Description | 8 |
| 2.3.2. Risk Management | 9 |
| 2.3.3. QA Metrics | 9 |
| 2.4. Design & Specification Manager | 10 |
| 2.4.1 Role Description | 10 |
| 2.4.2. Risk Management | 11 |
| 2.4.3. QA Metrics | 11 |
| 2.5. Lead Software Developer | 12 |
| 2.5.1. Role Description | 12 |
| 2.5.2. Risk Management | 13 |
| 2.5.3. QA Metrics | 13 |
| 2.6. Testing & Integration Manager | 14 |
| 2.6.1. Role Description | 14 |
| 2.6.2. Risk Management | 15 |
| 2.6.3. QA Metrics | 15 |
| 2.7. Marketing Manager | 16 |
| 2.7.1. Role Description | 16 |
| 2.7.2. Risk Management | 17 |
| 2.7.3. QA Metrics | 17 |
| 2.8. XML & Server Manager | 18 |
| 2.8.1. Role Description | 18 |
| 2.8.2. Risk Management | 19 |
| 2.8.3. QA Metrics | 19 |
| 3. Deliverables | 20 |
| 4. Project Management Methodology | 21 |
| 4.1. Requirements and Specifications | 21 |

| | |
|--|-----------|
| 4.1.1. User Stories | 22 |
| 4.1.2. User Story Process | 22 |
| 4.1.2.1. Estimating Value and Risk for each User Story | 22 |
| 4.2. Design for each User Story | 23 |
| 4.2.1. Flexible implementation | 24 |
| 4.2.1.1. Skeletons and Initial/repeated tests | 24 |
| 4.2.1.2. Interfaces | 24 |
| 4.3. Implementation of each User Story | 24 |
| 4.4. Testing of each User Story | 25 |
| 4.5. Quality Auditing Reviews | 25 |
| 4.6. Labour Payments and Time Sheets | 25 |
| 4.6.1. Clockify | 26 |
| 4.6.2. Paying the Development Team | 26 |
| 4.7. Team Programming Procedure | 27 |
| 4.8 Weekly Meetings | 28 |
| 5. Appendix A: Document Templates | 29 |
| Meeting Agenda | 29 |
| Meeting Minutes | 30 |
| Test Report | 31 |
| Peer Review for stories | 32 |
| 6. Appendix B: Design Methodologies | 33 |
| Kanban Board | 33 |
| Gantt Chart | 33 |

1. Introduction

1.1. Company Profile

This company was set up as a developer of applications to be used on mobile and tablet devices. Our focus is to deliver well developed and functional applications of the highest quality to an intended target market.

Based out of the University of York, we are a team of highly dedicated and enthusiastic software developers providing high quality Java Android applications.

1.2. Vision

To be a leader in application development, creating ideas and solutions to differentiate our products from a crowded market while aiming to develop quality reliable applications with intuitive controls dedicated towards a specific target market. Our applications will produce pieces of interactive software that play/display multimedia presentations on Android devices.

2. Roles and Responsibilities

2.1. Organizational Structure

As a whole, the company has experts in each of their own fields, this allows a structure where each member of the team has a valid and respected input. The project leader oversees the team, over the term of the project, to make sure all specifications and requirements are met, ensuring processes are followed.

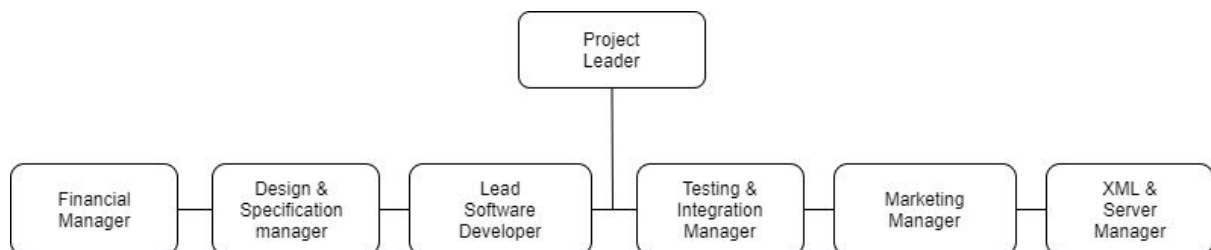


Figure 1 Structural organisation of the development team.

2.2. Project Leader

2.2.1. Role Description

To maintain and promote the product vision while organising the development team toward the desired goals. Accountable for making sure the design and development process runs smoothly by keeping development goals clear allowing the team to focus on the tasks at hand while tracking workloads and redirecting resources if needed.

Help develop and maintain processes and procedures while making sure they are followed, including setting and monitoring a management system for changes in documentation. Alongside this role the Project leader will also have a hands on involvement with the creation of user stories and programming of all applications.

Key duties:

- Arrange regular meetings with the group/development teams
- Providing agendas and completing meeting minutes (see Appendix A for templates)
- Maintaining a positive work environment that promotes creativity and instils confidence into the development team
- Provide updated plans which outline clear tasks for the development team and monitor the progress
- Provide QA plan that explains what processes should be followed and what procedures should be used
- Carry out review processes at the end of each phase of development to ensure the quality is maintained
- Track all the updates related to the project
- Keep track of delivery deadlines
- Monitor development process to make sure the requirements and standards are met at every stage
- Ensure all documents are up to date and are available to be used by the development team
- Deliver reports to the customer and act as the connection between them and the development team
- Produce and maintain project plan
- Work Alongside Financial Manager monitoring weekly timesheets, checking hours
- Work With Financial Manager and Lead Programmer to develop any and all contracts for work outsourcing.

2.2.2. Risk Management

| Risk | Possible solution |
|--|--|
| Group members don't get along or have a disagreement | Understand the issues that caused the problem, then try to remedy the issue |
| Prolonged absence | Make sure all team members are comfortable in all aspects of development to maximise flexibility |
| Deadline overdue | Have regular meetings and keep track of progression to allow movement of resources or adjustment of scheduling |
| Failure to meet requirements | Regular meetings to make sure standards and conformity is met. Make sure project is managed in a way that makes changes to meet requirements easy and as efficient as possible |
| Missing or corrupted documents | Keep fully updated backups of all documents in use |
| QA metric not met | Monitor the development team's adherence to the QA procedures |

2.2.3. QA Metrics

| Metric | How measured |
|-------------------------------|--|
| Customer requirements clarity | Hold weekly meetings to make sure that the customer understands their own requirements and that these have been understood properly by the development team. This then gives the go ahead to proceed. |
| Complete specifications | Monitor company workflow so that it continues to adhere to the specified structure laid out in the QA Manual by making sure all relevant paperwork is being kept and sign off timesheets. |
| Deadlines met | Monitor development tasks on Trello to make sure features are kept within the timeframe and all deadlines are met. If deadlines are not being met, meetings will be held with members of the development team to determine the problems and resources will be redirected if necessary. |

2.3. Financial Manager

2.3.1. Role Description

No company is successful without active financial management, in most cases the financial side becomes the key assessor of the success of a company.

The finance manager is to take lead on all company finances ensuring the company runs smoothly, makes a profit and doesn't go bankrupt. Alongside this role, the financial manager will support the team with vision, ideas and programming capability.

Key duties during project include:

- Provide financial insight to the team where necessary.
- Formulate financial business plan for investment necessary for the project. Research monetary possibilities and potential.
- Formulate financial reports showing actual against planned spending, explaining any variance along with revising plan for review if variance is significant.
- Provide professional guidance and advice on all monetary decisions.
- Manage company assets and budgets.
- Pay all members of the development team to timesheets that are overseen and ensure these are approved through the project leader.
- Manage financial payments of rent, utilities and IT infrastructure.
- Calculate overhead recovery rate, (submitting with business plan) based on the full absorption costing method.
- Management of company debts and interest owed on these debts.
- Keep a record of all company payments and revenue.
- Check and approve all company contracts, ensure contracts are valued in pounds sterling and comply with the following schedule:
 - 25% of contract value payable upon placement of contract.
 - 50% of contract value payable upon handover of module code.
 - 25% of contract value payable upon purchaser group acceptance of module.
- Adhere to forced transactions from the financial advisor, take these financial changes into account. If appeal is deemed necessary, submit this in writing to the financial advisor stating all pertinent details of disagreement.
- Formulate financial performance review along with profit and loss statement.
- Propose price for product created by the company.
- Propose requirements (features) for the application.
- Assist with representing requirements as user stories.
- Assist with product vision and product operating mechanics.
- Assist with team programming effort.
- Design back end data model.
- Advise the team about how to structure Firebase.
- Take the lead on the social side of application.
- Design pages of the application where no design instruction has been previously drawn up.

2.3.2. Risk Management

| Risk | Possible Solution |
|--------------------------------------|---|
| Credit risk | Assist and advise team lead to make detailed contracts with other teams and investors |
| Poor budgeting | Make accurate budgets, monitor these budgets throughout project |
| Poor team timesheet management | Check all hours, ensure payments are paid in ¼ hour intervals only and pass through project leader for final sign off |
| Poor monetisation of the application | Assess and analyse all possible ways to monetise on the application to make an informed decision on how to proceed with market testing. |

2.3.3. QA Metrics

| Metric | How Measured |
|---|--|
| General financial health | <p>Active financial management at every stage of the project. Ensure all financial information, in and out of the company, is recorded and reviewed against predictions made in the Financial Business Plan.</p> <p>Metric measures percentage deviation of finances assessed and predicted as laid out in the Financial Business Plan against actual costs detailed in the Financial Reports.</p> |
| Return of investment, profitability and yield | <p>Final product pricing that will make 'reasonable' profits after development, which is based on a realistic assessment of the market.</p> <p>Metric measured in future profit potential when seeking further financial backing.</p> |
| Time-sheet management | <p>Active timesheet management, ensuring timesheets are completed with justification, checked and signed off each week by team lead.</p> <p>Metric measured in average time taken to check off timesheets and send to team lead.</p> |

2.4. Design & Specification Manager

2.4.1 Role Description

Whilst the project on paper may have all the elements required for a great launch, if the user interaction, functionality and overall design of the software is poor this may not translate into enough user downloads or traffic to make the software sustainable and ultimately generate enough revenue.

In order to provide a good user interaction the Design and Specification manager develops a link between the feedback given by the users during testing (user stories) and the appropriate actions to be taken by the project team in order to make sure the UI elements within the project are both appropriate and appealing to the correct target audience and equally function as intended. The functional implementation of the user interaction and layout of the entire project is also considered to make sure this fits.

In addition to this feedback on the overall design and implementation of the software must be taken from all other group members and implied from user stories and sketches/diagrams to be put into an overall design specification for the team that can be updated throughout the week.

Additional responsibilities

- Divide up available time between the users and the project team effectively.
- Research and where possible use extensive existing knowledge of best design practices to tailor the product to the correct target audience
- Check the requirements in implementing the design on the current platform(s) effectively and make sure that these requirements are consistently satisfied without any loss in quality.
- Able to quickly iterate on the design and produce multiple versions if appropriate for the context.
- Provide a balanced and realistic requirement of what is possible for UI from feedback from the software group, users feedback and the limitations of the development platform and hardware.
- Interview users on the design implementation using mock-up paper prototypes or partially finished interactive GUI elements through a similar device as the one used in development.
- Work in close collaboration with all additional team members involved in coding an element that may be related to the feedback from the associated users story.
- Work in collaboration with the lead testing managers/lead developers to create appropriate UML project diagrams where appropriate to be displayed in the design specification which can be referenced by all team members.
- Attend the meeting every week to talk with all other members of the team on the progress of the project

- Create/add to an overall, well formatted Design specification for the entire project each week (or bi-weekly where appropriate) whilst in possible collaboration with a Lead Programmer.
- Propose requirements (features) for the application
- Assist with representing requirements as user stories
- Assist with product vision and product operating mechanics

2.4.2. Risk Management

| Risk | Possible Solution |
|---|--|
| Delays in producing code to fit a partial interactive prototype of the design | Present ideas on paper instead and make sure to explain elements that may not be immediately clear without feedback from the application. |
| Delays in completing design elements | Re-assign roles according to a best guess of who would be better suited if time permits or increase the size of the subset of the main group working on that element. |
| Incoherent design ideas | Provide additional guidance if the issue is they don't understand the bigger picture of the design and how it works within the existing framework we have currently implemented. |

2.4.3. QA Metrics

| Metric | How it's measured |
|-----------------------------|---|
| Faults/Issues in design | Amount of issues related to the UI design or functionality reported either through Github and(or) Trello from the user or from other team members for the given week along with their severity. |
| Cost | Based on whichever is more important, the basic working functionality or the design of an element. Same cost analysis done whenever possible. Determined based on a best guess from the project team and responses from the user story at each iteration. |
| Quality of Design | Ratio of positive to negative comments on the design at each iteration and where possible the implied weighting of each comment out of 10 combined into an overall average at each iteration. |
| Quality of user interaction | *Same as above but for UI elements only. |

2.5. Lead Software Developer

2.5.1. Role Description

Find the most effective way of delivering the stories in the plan, provided effort estimates, suggest alternatives and help customers create an achievable plan by planning in advance

- Joint effort between multiple programmers
- Establishes coding standards that allow shared responsibility for the code
- Spends most of the time pair programming
- Using test-driven development:
 - Writes tests
 - Supervises implementation of code
 - Refactors code
 - Incrementally design and architect the application
- Monitor and supervise design quality
- Oversee technical debt and its impact on development time and future maintenance costs
- Support customer choice with software releases and at the end of development iterations
- Avoid and manage technical bugs/errors in completed software
- Maintain ten-minute builds that can build a complete release package at any time
- Use version control and continuous integration, keeping all but recent work integrated and passing tests
- Responsible for fixing any problem they see, regardless of which part of the application is affected
- Rely on customers for information regarding the software
- Request customer feedback instead of guessing on direction of software
- Customer conversations enabled by programming in a ubiquitous language
- Assist in customer testing by automating customer examples
- Provide documentation to help ensure long-term maintainability of the product
- Propose requirements (features) for the application
- Assist with representing requirements as user stories
- Assist with product vision and product operating mechanics
- Work in collaboration with the UI/Graphical designer in producing an easy to read master design specification

2.5.2. Risk Management

| Risk | Possible Solution |
|---|---|
| Bugs/errors in program | Continuous testing before integration of any code |
| Application does not reflect customers vision | Constant contact with customer during development, especially after any iteration is finished |
| Story overruns allotted time | Supervise programming team and ensure time targets are being met. If a story does overrun, readjust future time schedule to mitigate the effect on the timing |

2.5.3. QA Metrics

| Metric | How Measured |
|-------------------------------|--|
| Issue Logging | Log issues using GitHub/Trello to track current progress |
| Commenting code | Monitor code and ensure an appropriate amount of commenting is present |
| Time scheduling of code | Compared time logged for each programming task against the initial predictions given. From this, produce a percentage to measure the timing effectiveness of each task |
| Errors in compiling/execution | Produce a report after each compilation after completion to indicate the errors or lack thereof |

2.6. Testing & Integration Manager

2.6.1. Role Description

The Testing and Integration Manager (T&I Manager) will be in charge of the processes behind testing the code and integrating it together with other modules in the project. They must understand the project fully and how each module works together, as well as what each module must do standalone. The T&I Manager must create effective and complete plans when testing the project. An example test report will be in *Appendix A*. The report(s) will be delivered to the group leader for analysis to review before the weekly meetings.

Key Responsibilities include:

- Create effective and appropriate testing and integration plans.
- Manage the testing of components amongst the team.
- Define test plans, test specifications, and test cases.
- Agree testing scope, requirements and exit/entry requirements for each module.
- Find and identify bugs in the code and help prevent bugs from being sent to final release and make appropriate GitHub errors.
- Create CI build tests for the code which is pushed to GitHub - using Travis CI. (<https://travis-ci.org/github/SWEngGroup4/scrانplan>)
- Create unit testing using JUnit and focused integration tests to test singular classes or methods throughout the development process.
- Work with the team creating espresso testing.
- Create automated testing for the code alongside programmers designing the code.
- Produce reports and documentation in a timely manner covering the results from the tests.
- Merge Pull Requests made by the Team to GitHub, and update related documentation.
- Maintain the master branch on github to ensure features are continuously added - keeping with CI, CD and Agile philosophies.
- Working following the agile development method.
- Track errors using Sentry, and open GitHub issues for any recurring bugs.
- Log and report test failures to the development teams.
- Propose requirements (features) for the application.
- Assist with representing requirements as user stories.
- Assist with product vision and product operating mechanics.
- Assist with team programming effort.

2.6.2. Risk Management

| Risk | Solution |
|---|---|
| Error occurs with no obvious cause in the design. | Communicate with the lead programmers about issues |
| Continuous Integration builds fails | Investigate the cause and communicate issues with the group leader and programmers. |
| Testing fall behinds the agreed schedule | Reschedule the tests. |
| User test fails, but another tests pass | Create new user tests and investigate what failed initially. |

2.6.3. QA Metrics

| Metric | How to measure |
|--|---|
| Number of modules tested | Collected from error reports |
| Testing and integration plans followed | Monitor progress of the testing throughout the development progress, and collect any issues with plans not being followed. |
| Number of errors and issues | Collected from error reports. |
| Time testing | Compared time logged for each code review task against the initial predictions given. From this, produce a percentage to measure the timing effectiveness of each task. |

2.7. Marketing Manager

2.7.1. Role Description

The marketing manager is an essential role to provide a marketing strategy that displays the company's vision and is relevant while being in line with the customer's specification.

Roles of the marketing manager include taking lead and responsibility of the marketing of the product. Ensuring the marketing is effective while working alongside other members to ensure the product and marketing work well together.

Responsibilities Include:

- Conduct market research on current similar products and competitors.
- Conduct research on what the customer is trying to achieve with the product.
- Decided the marketing strategy to fill the gap in the market and find our niche.
- Decided target audience for marketing to be directed at.
- Conduct surveys to find out about what our target market wants from our product. Use this to influence decisions with product design.
- Work alongside product development to ensure marketing fits the product.
- Manage launches and demonstration campaigns.
- Decide on how the product will be advertised through different mediums, for example, campaign launches, outreach events and social media
- Ensure the marketing team has the ability to explain marketing design and ideas.
- In charge of branded marketing, through marketing media, communication etc...
- Keep reviewing the effectiveness of the marketing strategy.
- Organise and keep track of customer feedback, marketing surveys and market research.
- Marketing budget control - estimate pricing of how much each marketing campaign will need
- Propose requirements (features) for the application.
- Assist with representing requirements as user stories.
- Assist with product vision and product operating mechanics.
- Assist with team programming effort.

2.7.2. Risk Management

| Risk | Solution |
|------------------|--|
| Lack of interest | Research the current market and see how engaged people are. Record feedback and included in changes. Ensure an effective marketing campaign. |
| Competition | Ensure there is a niche in the market. Ensure quality of products. Provide good advertisements. |
| Copyright | Do extensive research into content used. Check all sources are correctly cited. |

2.7.3. QA Metrics

| Metric | How measured |
|---|---|
| Brand awareness | Market survey with our target audience. Compare the results of the survey to estimated survey responses |
| Customer satisfaction | Customer feedback, positive or negative against each other. When an important user story/feature is finished, send a video to the customer and receive feedback and suggestions on the product. |
| Customer marketing satisfaction | The interaction and feedback after demonstrations focused on the marketing section. Positive comments against negative ones. |
| Produce pricing and estimation of users | The interaction from the financial advisor and the judging panel on if the estimates are reasonable |

2.8. XML & Server Manager

2.8.1. Role Description

As a database designer, my main job is to take on the database structure. In fact, the changes are a natural, inescapable and desirable aspect of software-development projects, and should be planned for, instead of attempting to define a stable set of requirements. Achieve the goal of reducing the cost of change by introducing concepts such as basic values, principles, and methods. After creating the database successfully, storing and recording the customers' information is vital.

Key duties:

- Conduct research on cloud computing platforms.
- Decide which cloud storage platform offers the cheapest and design friendly experience for developing the back end of the application.
- Investigate which platform offers the fastest data retrieval.
- Ensure, based on the change of data, the database can also be updated in real-time.
- Ensure, based on the request of users, the database can increase or delete some functions.
- Design data structures within the cloud storage platform, including how users will be stored, multimedia and all information databases.
- Decide on the most appropriate multimedia formatting within the cloud storage platform with regards to storage, speed and quality.
- Decide on the most appropriate user authentication methods with the cloud storage platform.
- Decide on the most appropriate format to integrate advertising into the application through the cloud storage platform.
- Design notification methods for users sent from the cloud storage platform.
- Record user engagement metrics from the cloud storage platform and deliver to the testing manager.
- Propose requirements (features) for the application.
- Assist with representing requirements as user stories.
- Assist with product vision and product operating mechanics.
- Assist with team programming effort.

2.8.2. Risk Management

| Risk | Solution |
|---|---|
| Choose an inappropriate platform to create database | By research on the different cloud storage platforms, compare the cost and functions of each one and then choose the most suitable one. |
| Information missing | To satisfy the demand of customers, list all the items and classify them into categories. After choosing firebase as the platform, create a new collection to collect one kind of item with its values. And then, repeat this operation to include all the items. |
| Cannot fit the customer's using habits | Choose the most common user authentication methods to sign in our application. Let the user choose how notification works. |
| Cannot update to follow customers' request | Contact with teammates immediately and then update the items or add and delete more items on demand. |

2.8.3. QA Metrics

| Metrics | How to measure |
|---------------------------------|---|
| Suitable cloud storage platform | Compare the pricing of them and see some feedback of the developers |
| Concise storage of xml files | Feedback after using the firebase storage to save the xml files |
| Total spending | Use the calculation of firebase and make an estimation |

3. Deliverables

Table 1: List of all Deliverables expected throughout the duration of the project

| Deliverable | Producer | Recipient | Due |
|---------------------------------|---------------------------------|-------------------------------------|-----|
| Functional Specification | Project Leader | Customer | |
| QA Manual | Project Leader | All Company Personnel | |
| Financial Business Plan | Finance Manager | Financial Backer | |
| Tender Presentation | Development Team | Chief Customer and Financial Backer | |
| Financial Report I | Finance Manager | Financial Backer | |
| Financial Report II | Finance Manager | Financial Backer | |
| First Iteration Complete | Project Leader | Customer | |
| Final Test and Integration Plan | Testing and Integration Manager | Customer | |
| Financial Report III | Finance Manager | Financial Backer | |
| Financial Summary Report | Finance Manager | Financial Backer | |
| Sales Presentation | Development Team | Customer and Financial Backer | |

4. Project Management Methodology

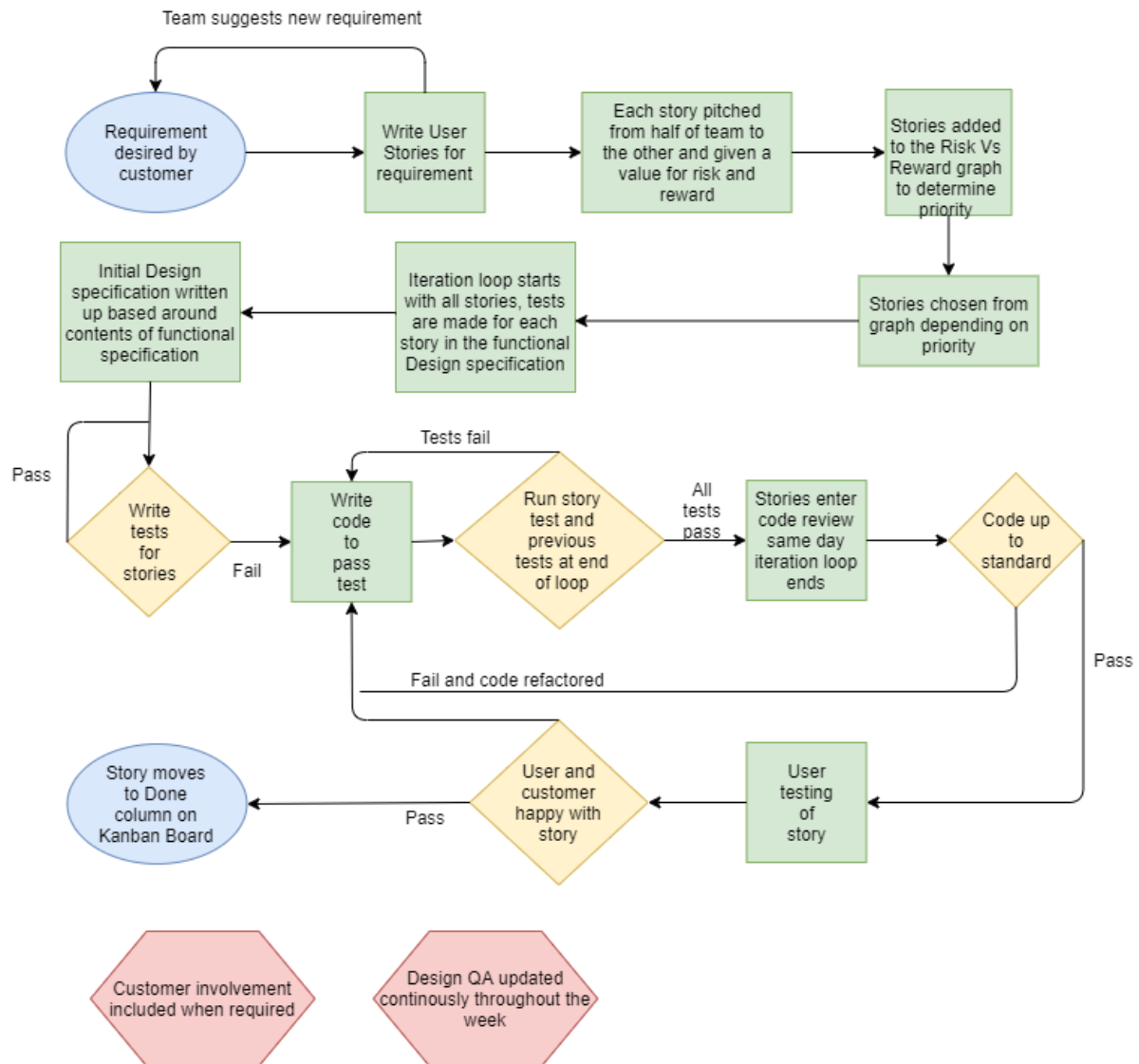


Figure 2 Project life cycle

4.1. Requirements and Specifications

The requirements of a project being completed by the company will be provided by a customer statement highlighting their requirements and needs from the project. Given these initial requirements, the company will undertake a comprehensive requirement capture analysis to produce a design specification of the project by writing 'User Stories' fulfilling the customer requirements. At this stage, inconsistencies and errors present between the company specification and the customer requirements should be established

and discussed with the customer, forming the basis of the project design phase. To summarise the basic steps:

- The customer provides a statement outlining their desired requirements of the product
- This statement will be examined and evaluated by the Design & Specifications manager and the QA & Documentation manager to produce a product specification document implementing the customers' requirements as accurately as possible – adding or removing requirements to better fit the functionality of the product. Additionally, the company can assess whether the end product is feasible, and an initial budget can be estimated
- Changes to the initial requirements can be discussed with the customer by the Group Leader
- Once an agreement between the company and customer is made, a specification document can be made by the Design & Specification manager and a project time schedule can be created by the Group Leader in congregation with the company managers and the customer
- At each stage of the development process, the Design and Specification manager will provide the other company managers with the specification document, starting the implementation phase

4.1.1. User Stories

User stories will be made to represent the requirements. All major User Stories will be called Epics and assigned as projects within the Clockify timesheets, individual smaller user stories will be assigned timesheet tasks.

For each user story there will be a specific **who** does **what** and **why**. If the user story is not obvious from the requirement it is an indication that the team and customer must talk further. Each user story is a token for work to be done. Each user story will be numbered in order to track them and will have a risk and value score assigned to it. Each iteration loop will have stories allocated to it starting with the most risk and value and working down. If a story needs to be broken down into smaller stories and if the smaller stories on their own have no value these stories can be broken down into tasks that fall under a specific story where all tasks together hold the value of the story.

4.1.2. User Story Process

From the list of requirements all team members will assist with representing them as user stories. Where there is a specific requirement that a team member has suggested, the team member that has suggested the requirement will write the story. Where a requirement is not immediately obvious, the team and customer must talk further to ensure maximum value of the user story.

4.1.2.1. Estimating Value and Risk for each User Story

In the weekly team meeting, new user stories will be pitched from one half of the team to the other half of the team, with the author on the pitching side. The side of the team not pitching will then assign this story and value between 0 and 100 along with a risk rating of between 0 and 100. A number will then be assigned to the user story in order to keep track of it. These User Stories will be graphed risk versus reward to allow the development team to pick the highest value User stories to implement first allowing the Project Leader to plan accordingly and assign tasks to the team.

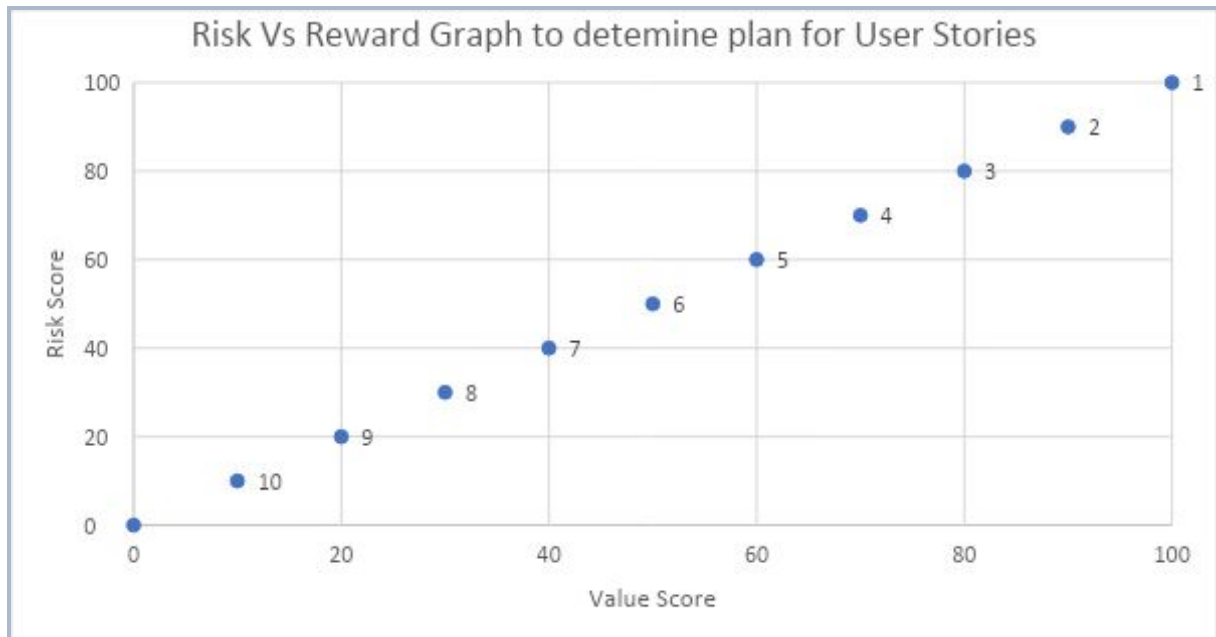


Figure 3 Graph to demonstrate how user stories will be numbered and then given value between 1 and 100 for risk and reward. The higher the risk and reward, the higher the priority.

4.2. Design for each User Story

After the functional specification is defined a weekly design specification is created based on the user requirements and user stories for the week. The document takes no particular format or template as it does not need to be presentable to the user and it will never be officially finished since new user/programmer requirements may mean the document has to be edited later. For this reason, google docs is used to produce fast iterations by providing real-time editing support and history for all devices with stable internet access.

On each weekly iteration a new design specification document is created for the desired user story. Once we are happy that the user story has been successfully satisfied in its current state this document is collated into a larger, master design specification which is formatted better by the Design and Specifications manager in possible collaboration with a Lead programmer depending on if the specification for the week is mainly based around user input/graphical elements or java/xml code. The document is reviewed when appropriate by other members of the team when it is required to reference it.

Elements in the design specification may include, but are not limited to:

- Pseudo code descriptions
- Flow Diagrams
- Java Class diagrams and naming schemes
- Hand-made drawings or android studio design screenshots/gifs/videos for UI and graphical elements.

4.2.1. Flexible implementation

4.2.1.1. *Skeletons and Initial/repeated tests*

A skeleton framework is created in Java code which consists of a collection of functions that will serve as the basis for the outline of the user story and how it should be implemented. Nothing is initially put into the functions and therefore when the program runs if the test fails, we know the code is written as intended. This process is repeated as more elements are gradually added and the same process is implied. If we expect the test to fail and it does fail, we are successful in achieving what we want the program to do and then we can add code that will satisfy this condition with the minimal amount of code required at each stage.

4.2.1.2. *Interfaces*

Generic Java interfaces are used as a way to bring objects we define under a common structure without having any conflicts due to the way an object is implemented. For example, if we have a test class that checks for user accounts in a list for login purposes there are multiple ways of implementing a list but the way this list is implemented shouldn't be explicitly defined.

By doing this it allows to prevent wasting time refactoring (re-writing) code and instead focus on more important aspects of the project.

4.3. Implementation of each User Story

Provided with the design documents created by the Design & Specification manager, the Software managers will assess time schedules and resource requirements in compliance with the outlined design presented in the documentation. Additionally, the design requirements will be split into User Stories and further split into individual tasks, which then will be added to a Kanban board using Trello, after which tasks will be assigned to the development team by the Project Leader. The software team will work across various types of IDEs and programs in order to produce a result in the desired format. The software managers will be responsible for ensuring iteration loops are consistent in timing and the appropriate amount of user stories are generated by each iteration. If complications arise with tasks or stories, it is the Project leaders' job to ensure the timing schedule and Trello plan is adjusted accordingly and the implementation plan is revised to mitigate the problems. After each successful story implementation, which will have included testing, each story implementation will be peer reviewed by designated members of the development team to review the produced code to ensure a standard is met, before signing off the task as done.

4.4. Testing of each User Story

For each module it must be tested individually and, when integrated with the project. Test driven development will be used with writing Junit/Espresso tests alongside the design of the code. White box testing will be used for each unit of code then grey box for integrating it together with other modules of code (testing units together), and finally black box testing when the code is complete. The testing and integration manager is responsible for checking the reports from testing code and collecting any results of errors and communicating it with the lead programmer so they can implement a fix. Regression testing should be used for any changed unit to ensure it doesn't adversely affect any other part of the code.

- Throughout the phase every member should follow the testing and integration plan.
- Each member should report any and all errors encountered.
- Code will be checked using Continuous Integration software to check each revision can be built.

4.5. Quality Auditing Reviews

Generating a quality product cannot be achieved without first implementing Quality Control (QC) checks of the highest quality. When a user story has been completed, the planner will be updated, the User Story will be placed in the peer review section and a Pull Request to merge with the master branch on GitHub will be submitted.

When a pull request has been submitted, members of the development team who were not a part of writing the user story will be assigned to check the code. That team member will check that the code will not crash when merged with the master by using Travis CI which runs a simulation and flagging errors. The reviewer will also check that the relevant testing reports are present making sure that the unit testing has been done correctly and has tested what the user story requires.

Once all checks have passed, the appropriate form must be filled out (see Appendix A) and then the story can be merged and be moved on the planner to Done.

4.6. Labour Payments and Time Sheets

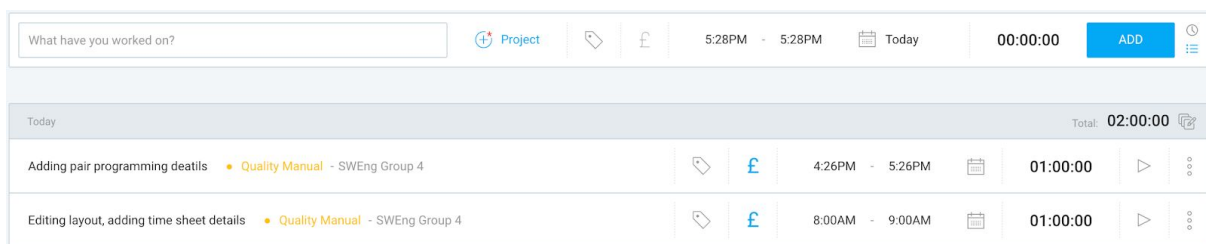
All members of the development team will be paid £12.50 per hour for their work. As all members are freelance they will need to deal with their own taxes at the end of the year. All work is to be paid according to their timesheet via Clockify. All hours must be in increments of 15 minutes and will be rounded down where they are not.

All estimations of how long tasks will take are agreed between the entire team with each team lead dictating how much time will be required for their role specific tasks and all other

group tasks voted on by the entire team for their budgeted duration. These estimations will be put forward with the financial business plan for project budgeting and all discrepancies where users go over time are to be brought up at meetings when assessing progress.

4.6.1. Clockify

Within Clockify there are agreed upon projects, that are then broken down into tasks where deemed necessary, for the team to bill their time to. All hours must be assigned to a project (and task if more focused work is taking place) on Clockify where the user has been working, with a note of work completed within this given time. An example of how to add to user time sheets including two time entries with progress notes, saved under the Quality Manual project, can be seen in Figure 4.



| What have you worked on? | | Project | £ | 5:28PM - 5:28PM | Today | 00:00:00 | ADD |
|--|----------------------------------|---------|-----------------|-----------------|-------|----------|-----|
| Today Total: 02:00:00 | | | | | | | |
| Adding pair programming details | • Quality Manual - SWEng Group 4 | £ | 4:26PM - 5:26PM | 01:00:00 | | | |
| Editing layout, adding time sheet details | • Quality Manual - SWEng Group 4 | £ | 8:00AM - 9:00AM | 01:00:00 | | | |

Figure 4 Example addition to the time sheet on Clockify. (Screen shot taken from <https://clockify.me/tracker> on 4/12/2019)

4.6.2. Paying the Development Team

Members of the development team will be paid at the end of every week according to their timesheet within Clockify. Hours used will be passed through the team lead to assess if the team is performing as expected and the progress is at the desired level for the team. Development team members are then paid and labour expenses are deducted from company capital.

Estimates for hours worked throughout the project are laid out in the Financial Business Plan. These estimates were calculated through a sum of the a group time estimate for each user story combined allowance for design and code review of each user story. Other allowances outside of these will be for specifics such as marketing, team-management and company finances. Further hour predictions have been made for specific group tasks. All extra hours outside of projections must be requested through the project and financial managers where not previously accounted for in team estimation sessions.

All team members will be in charge of their hour estimate for role specific hours where extra guidance has not explicitly been given by either the finance or project manager.

All hours estimated to a user story will be the responsibility of the individual leading the user story, for this individual to use or delegate where they see fit. User stories are estimated for pair programming so where a user is working alone they have double the estimated time. Where a user story goes over budget, it is the responsibility of the individual leading the user

story to bring this up at the team meeting with an explanation of why the story has gone over budget. This will enable the team to reassess finances where necessary and for the correct assistance to be allocated.

4.7. Team Programming Procedure

During the First Iteration the team will focus on pair programming. The workspace will be laid out to enable the development team to sit together and work in three sets of pair programmers. The seventh member of the team will be directing and collating between each of the pairs as the overall director, holding the key vision and promoting synergy, this will most commonly be the team lead. When all pairs are working and there is no overall driver required the seventh will remain in earshot to direct as required.

Each pair will consist of a driver who is doing the programming and a navigator who can think of solutions and where to go next while the driver is coding. This leaves the driver to concentrate on making clean and creative code without having to focus on the broad view. Pairing will also enforce good coding habits and ensure the code is understandable by any member of the team. Pairs will be changed regularly to ensure fresh perspectives and give the team an equal share in all aspects of the workload.

When timetabling issues are encountered and the team is unable to program in pairs, individual programming can take place if given certain tasks, while still working under other agile programming principles such as JUnit 4 testing.

The workspace will be laid out with three tables for pair programming and a close by desk for the seventh team member. There will be a larger desk available for work to be laid out and meetings to be held along with whiteboards covering all available walls for working on plans and ideas. Break out rooms will be available for company phone calls and an area for a particular problem to be discussed within a small group without disrupting the other pair programmers. While coding all team members will sit together in the same room at the stations. Each pair programming station will be large enough for two people to comfortably sit and observe two monitors. Where appropriate multiple mice and keyboards will be made available. An example of the office layout can be found in Figure 5.

As of the Second Iteration and onward, pair programming will be unrealistic thus the person who has taken a user story will be solely in charge of that story. Said person should liaise with either the Lead programmer or Design Manager to get an idea of how the story should connect with the wider application or present their own ideas and allow the group to sign off those ideas within the weekly meeting. As the member of the development team is solely in control of the user story, they are responsible for the time allocated on that user story. During the weekly team meetings, the developers discuss how the story is progressing and where they are in relation to the allocation in time compared to what has been delivered to that point. Any time over the original allocation must be addressed with reasons and what is happening going forward to minimise the financial impact.

Where stories have gone excessively over budget on the time estimated, a review into programmers work needs to be conducted.

User Stories will be ordered on Trello so that the most important are top working down. These stories are allocated in this order or at the Lead Programmer and Project Leaders discretion. Members of the Development team can also request starting a certain User Story and with good reasoning, they will be allocated that work.

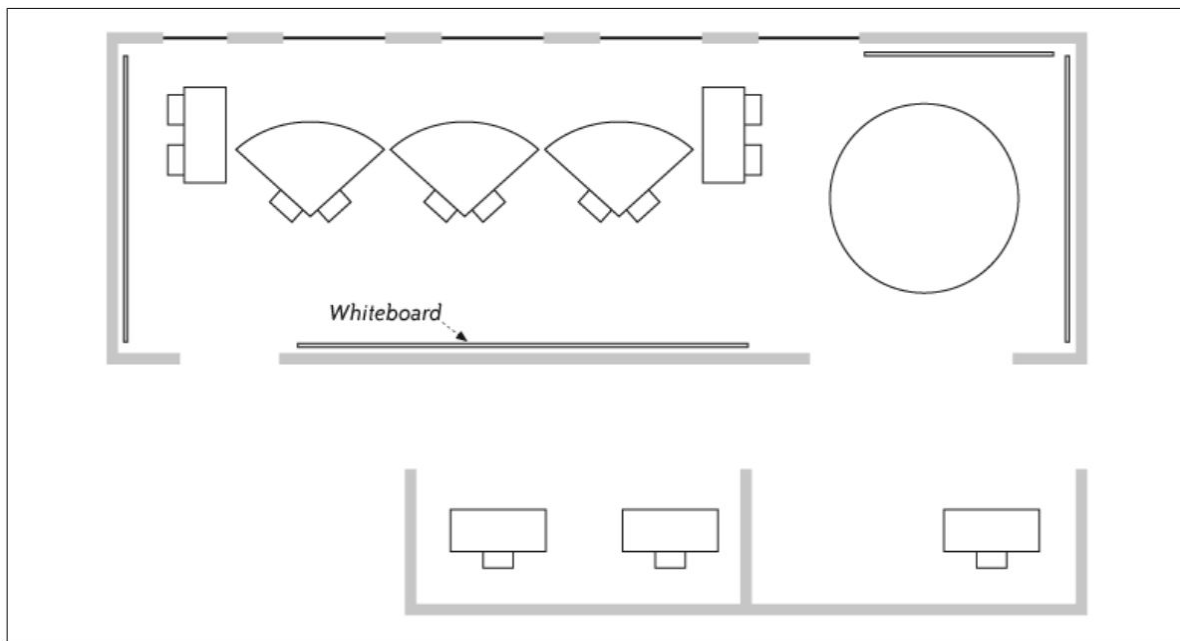


Figure 5 Example office layout for the seven development team members. This includes: three pair programming stations, desks close to the stations for the seventh member to work and direct within earshot, whiteboards all around, a large table for meetings and planning and break out rooms for discussions of particular company problems while other members are working. (Original picture: The Art Of Agile Development, O'Reilly, by James Shore and Shane Warden, 2007, Page 119)

4.8 Weekly Meetings

Weekly meetings will be held every monday morning to discuss the previous weeks work, any issues that have arisen and how we are to go forward in the coming week. This will allow the team to adjust the plan if needed and rectify any and all issues. As of the Second Iteration, all weekly meetings will be held online using skype or Google Hangouts.

5. Appendix A: Document Templates

Meeting Agenda

AGENDA

Meeting Title**Date****Start Time – End Time**

Meeting called by Facilitator Name

Attendees: Attendee Names**Please read:** Reading List**Please bring:** Supply List

| | | |
|------------------------------|--|-----------------|
| Start Time – End Time | Introduction | Location |
| | Continental Breakfast Topic Speaker | |
| Start Time – End Time | Item #1 | Location |
| | Topic Enter topic | |
| Start Time – End Time | Item #2 | Location |
| | Topic Enter topic | |
| Start Time – End Time | Item #3 | Location |
| | Topic Enter topic | |

Additional Instruction:

Use this section for additional instructions, comments, or directions.

Meeting Minutes

Team Meeting

Date
Time
Location

Meeting called by: Enter meeting organizer here **Type of meeting:** Enter meeting type here

Facilitator: Enter meeting facilitator here

Attendees: Enter attendees here

Please read: Enter reading list here

Please bring: Enter items to bring here

Minutes

Agenda item: Enter agenda item here **Presenter:** Enter presenter here

Discussion:

To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.

Conclusions:

Enter conclusions here.

Action items

- ✓ Enter action items here
- ✓ Enter action items here
- ✓ Enter action items here

Person responsible

Enter person responsible here
Enter person responsible here
Enter person responsible here

Deadline

Enter deadline here
Enter deadline here
Enter deadline here

Agenda item: Enter agenda item here **Presenter:** Enter presenter here

Discussion:

To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.

Conclusions:

Enter conclusions here.

Action items

- ✓ Enter action items here
- ✓ Enter action items here

Person responsible

Enter person responsible here
Enter person responsible here

Deadline

Enter deadline here
Enter deadline here

Other Information

Observers:

Enter observers here.

Resources:

Enter resources here.

Special notes:

Enter any special notes here.

Test Report

Group 4 Test Reports

Test Report

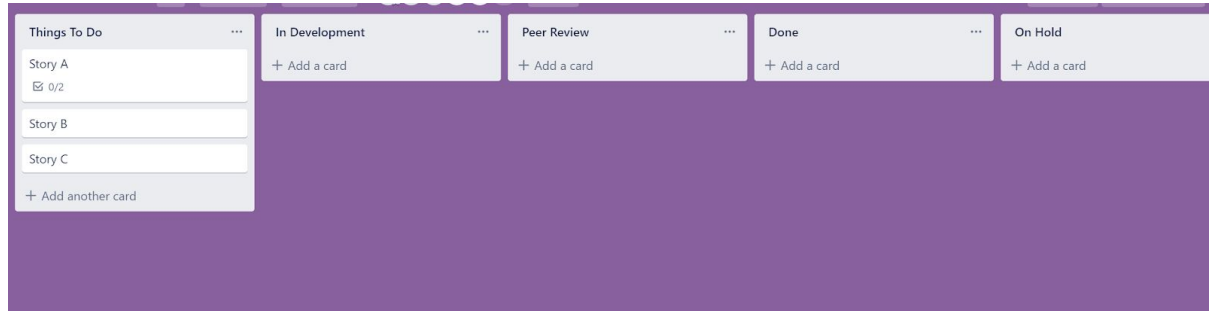
| | | | |
|---------------------|--|------|--|
| Test Report Title: | | Date | |
| User story: | | | |
| Unit Test Results: | | | |
| | | | |
| Other Testing done: | | | |
| | | | |
| Comments: | | | |
| | | | |
| Tester | | | |

Peer Review for stories

| | | | |
|---------------------------|--|-------|--|
| Story: | | Date: | |
| Story Checklist: | | | |
| | | | |
| Test Report Complete | | | |
| Travis Builds Sucessfully | | | |
| Unit Tests Run locally | | | |
| Comments/Issues/Notes: | | | |
| | | | |
| Reviewer | | | |

6. Appendix B: Design Methodologies

Kanban Board



Gantt Chart

