WEB APPLICATION

A PROJECT FOR QA CONSULTING

Authored

by

Shuaib Hussain

Supervised by Matt

for

QA Consulting

7th November 2019

TABLE OF CONTENTS

SECTION I: SUMMARY

In Section II the business case is presented. The case details the problems faced by Aurora Hotels and what they seek from LNBC & Company. The section then presents the *Lessons Log* and *BOSCARD*. Section III details the existing processes, has sample use cases and analysis the organisation structure and presents a RACI matrix. In Section III, assumptions based existing complaints are made to create the relevant processes.

SECTION II: PROJECT PLANNING

This section will detail and define Agile, its characteristics and principles, followed by detailing and defining scrum. The section will define a project management framework based upon Agile principles that will be utilised to complete this project.

Agile

Agile is a mindset and an umbrella term used for software development methodologies based upon agile principles. These software development methodologies seek to address the challenges of changing throughout and the need for adapting to constantly changing requirements.

The Agile Manifesto was developed by 17 software developers due to some software development methodologies which were considered overly regulated, planned and micromanaged. Based upon their combined experience of software development, they agreed that:

- **Individuals and interactions** take precedence over processes and tools. Scott Ambler further explains that processes and tools are important however having a competent, cohesive and collaborative team is more important
- **Working software** takes precedence over comprehensive documentation. Scott Ambler further elucidates that good documentation is useful in helping people understand how the software is built and how to use it, but the main point of development is to create software, not documentation
- **Customer collaboration takes** precedence over contract negotiation. Scott Ambler further clarifies that a contract is important but is not a substitute for working closely with customers to discover what they need
- **Responding to change** takes precedence over following a plan: Scott Ambler further delineates that a project plan is important but it should not be so rigid that it cannot accommodate changes in technology or the environment, stakeholders' priorities and people's understanding of the problem and solution.

Twelve principles of the Agile Manifesto

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace

9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Empirical Process Control

This process seeks to prioritise empirical evidence over planning and assumptions. The three pillars for empirical process control are as follows:

Transparency: with information to all stakeholders

Inspection: build a culture where inspection of people, processes, technology and the product is easy

Adaptation: adapt and continuously improve based on findings from inspection

Scrum

Scrum is software development framework that seeks to increase effectiveness and collaboration to develop complex products. The Scrum Guide explains Scrum clearly and contains all the definitions of Scrum and the roles, events, artefacts and the rules that bind them together.

Scrum Roles

The Scrum Team consists of a Product Owner, the Development Team and Scrum Master. Scrum Teams are self-organising and cross-functional. Self-organising team decide how to achieve their teams without being directed by employees outside the team. These cross-functional teams possess all the competencies needed to accomplish the work without depending on others.

Scrum Master

The Scrum Master is responsible for helping everyone understand Scrum theory, practises, rules and values. This enables the Scrum Master to effectively promote and support the implementation of Scrum. The Scrum Master is responsible for managing interactions between the Scrum Team and other professionals to maximise the value created by the Scrum Team.

Service to the Product Owner:

i) Ensuring that goals, scope and product domain are understood by all stakeholders

ii) Developing techniques for effective Product Backlog management

iii) Emphasising the need for clear and concise Product Backlog items

iv) Assisting the PO in Product Backlog arrangement to maximise value

v) Understanding and practicing agile principles

vi) Facilitating Scrum events as requested or needed

Service to the Development Team:

i) Coaching the Development Team (DT) in self-organisation, cross-functionality and Scrum

ii) Helping the DT create high value products

iii) Removing impediments and red tape to the Development Team’s progress

iv) Facilitating Scrum events as requested or needed

Service to the Organisation:

i) Leading and coaching the organisation in Scrum adoption

ii) Planning Scrum implementations

iii) Helping employees and stakeholders understand and enact Scrum and empirical development

iv) Driving change that increases productivity of the Scrum Team

Product Owner

A Scrum Product Owner is responsible for maximising the value of the product created by the Development Team. The Product Owner can only be one person. The Product Owner is solely responsible for managing the Product Backlog. Managing the Product Backlog includes:

i) Clearly expressing Product Backlog items

ii) Ordering the items to best achieve goals and missions

iii) Ensuring the Product Backlog is visible, transparent and clear to all and shows what the Scrum Team will work on next

iv) Ensuring the Development Team understand items in the Product Backlog

v) Optimising the value of the Development Team’s work

The Product Owner may or may not partake in development but remains accountable for development of the product. The Product wonder decides the Product Backlog item’s priority ordering. While the Scrum team and other organisational may influence the Product Owner, the Product Owner has full ownership of the Product Backlog.

Scrum Development Team

The Development Team consists of a professional or professionals who work to create the releasable Increment of Done at the end of each Sprint. A "Done" increment is required at the Sprint Review. Development Teams are structured and empowered by the organisation to organise and manage their own work. This synergy optimises the Development Team’s overall efficiency and effectiveness. Development teams have the following characteristics:

- They are self-organising, no one tells the team how to create the software
- Development teams are cross-functional with all the skills necessary to create a Product Increment
- There are no titles for anyone in the Development Team

- There are no sub-teams in the Development Team
- The Development Team is accountable as a whole regardless of individual expertise

Development Team sizes can range from small to large. Fewer than three members may reduce productivity and interactions. Smaller teams may also encounter skill constraints during the Sprint, causing the Development Team to be unable to deliver a releasable Increment. Having a team with more than nine member may require too much coordination. Large teams may generate too much complexity for an empirical process to be fruitful.

Scrum Events

Scrum Events intend to create regularity, minimise the need for undefined meetings. Have defined periods for set goals. Once a Sprint has begun its duration cannot be changed. The Scrum events are:

Sprints

A Sprint is a set time period with a maximum duration of a month. Each Sprint aims to produce a potentially releasable product is created. Sprints have consistent time periods throughout product development. During the Sprint:

i) Changes that endanger the Sprint Goal cannot be made

ii) Changes do not compromise quality

iii) Alterations to the scope can be made as the team learns

Each Sprint has to have a goal of what has to be built, a design and a flexible plan that will guide its development, the work that needs to be done and the resultant product increment.

Sprint Planning

The Sprint Plan details the work that needs to be completed in the Sprint and the Sprint Plan is created by the entire Scrum Team. The time period allowed for a Sprint Plan is eight hours for a one-month Sprint and the time allocated can be adjusted in that ratio for shorter Sprints. It is the Scrum Master's responsibility to ensure the event takes place, the attendants understand its purpose and ensures that the planning is completed within the allotted time period.

Sprint Planning answers:

i) What cans be delivered in the Increment

ii) How the work needed to devilled the increment will be achieved

During Sprint Planning work from the Product Backlog selected and placed into the Sprint Backlog.

Sprint Goal: is an objective set for that Sprint that is created during the Sprint Planning meeting. The Sprint Goal seeks to provide purpose and guidance to the Development Team along with flexibility regarding functionality. Product development is commenced with objective of achieving the Sprint Goal through the implementation of the Product Backlog.

Daily Scrum

The time period allocated for the Daily Scrum is 15 minutes for 24 hours. The Daily Scrum is held everyday at the same time and place and is only for Development Team. The Development Team synchronises activities and creates a plan for the next 24 hours. The Daily Scrum does not have a set structure, it may be based on questions or discussions. Sprint and the time allocated can be adjusted in that ratio for shorter Sprints. This allows effective collaboration and forecasting during the Sprint.

The Development Team utilises the Daily Scrum to assess and inspect how the progress towards completing the items in the Sprint Backlog. The Daily Scrum intends to optimise the probability that the Sprint Goal is met. The Development Team or other team members often meet after the Daily Scrum for detailed discussions to adapt, replan the rest of the Sprint's work.

It is the Development Team's responsibility to conduct the Daily Scrum. The Scrum Master ensures that the planning is completed within the allotted time period and coworkers from other teams do not disrupt the meeting.

Sprint Review

A Sprint Review is an informal meeting held at the end of a Sprint to inspect the Increment and adapt the Product Backlog if required. It is not a status meeting because it is intended to elicit feedback and foster collaboration, hence it is not Status meeting. A single or multiple deployments can lead to the inspection of an Increment. Key stakeholders discuss what was done in the Sprint and made changes to the Product Backlog accordingly. The stakeholders then discuss on what needs to be prioritised and done next.

The time period allowed for a Sprint Review is four hours for a one-month Sprint and the time allocated can be adjusted in that ratio for shorter Sprints. Scrum Master's responsibility to ensure the event takes place, the attendants understand its purpose and ensures that the planning is completed within the allotted time period.

The Sprint Review includes:

i) The stakeholders include: the Scrum team and key stakeholders invited by the Product Owner

ii) The Product Owner presents what Product Backlog items have and have not been Done

iii) The Development Team discusses what went well, the problems it faced and how those problems were solved

iv) The Development Team demonstrated the work it has Done and discusses the Increment

v) The Product Owner discusses the Product Backlog and may project target and delivery dates if essential

vi) The entire group collaborates on what needs to be done next; this provides valuable input to the next Sprint Planning

vii) Market changes are assessed and the potential use of the product and change in requirements and customer needs, exceptions and wants are assessed

viii) The timeline, budget, potential capabilities and market changes are reviewed to adjust capability and functionality of the product

The result of the Sprint Review is a revised Product Backlog, that may also be adjusted to meet new opportunities.

Sprint Retrospective

The Sprint Retrospective is an opportunity for the inspect its performance and create plans for improvement to be enacted during the next Sprint. The Sprint Retrospective is held after the Sprint Review and before the next Sprint Planning. The time period allowed for a Sprint Review is three hours for a one-month Sprint and the time allocated can be adjusted in that ratio for shorter Sprints. It is the Scrum Master's responsibility to ensure the event takes place, the attendants understand its purpose and ensures that the planning is completed within the allotted time period. The Sprint Retrospective includes a discussion on:

i) What went well in the Sprint

ii) What could be improved

iii) What the team will commit to improve in the next Sprint

The Scrum Team plans ways to increase product quality by improving work processes or by adapting the definition of Done without going against the company mission, set product or organisational standards.

The purpose of the Sprint Retrospective is to formalise the process by which the improvements needed are identified and how the Scrum team implement these changes in the next Sprint.

Scrum Artefacts

Scrum artefacts were created to maximise transparency of information to effectively provide opportunities for inspection and adaptability. The artefacts are:

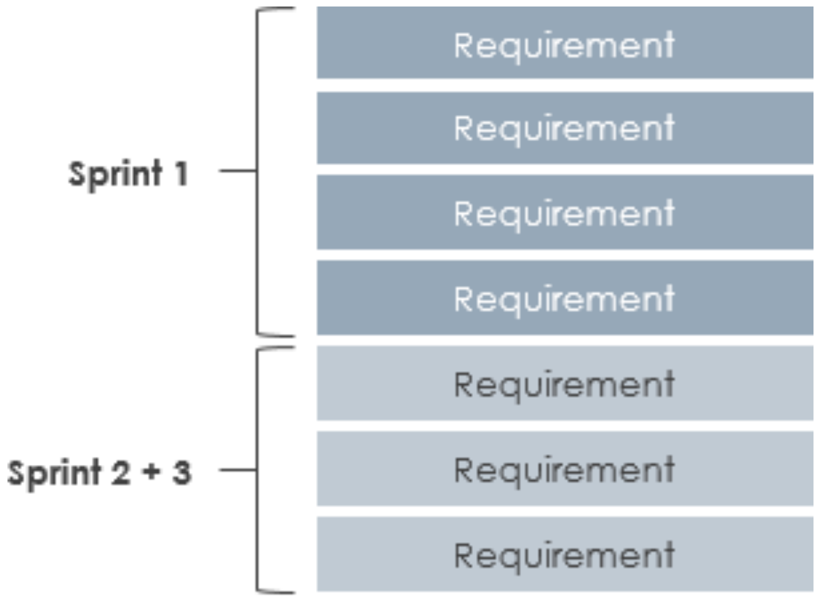
Product Backlog

The Product Backlog is an ordered list of everything that is needed in the product. It is a single source of requirements for any changes the product needs. The Product Owner is responsible for the Product Backlog, including its content, availability and ordering. A Product Backlog can never be complete. The Product Backlog with progression of the project and it dynamically changes to identify what the product needs to be competitive.

A Product Backlog can be refined by adding detail, making estimates or ordering the items in the Product Backlog. This is a continuous process that involves the collaboration of the *Product Owner* and *Development Team* to review and revise backlog items. Multiple Scrum Teams may work on the same product with a single Product Backlog.

Sprint Backlog

The Sprint Backlog is a set of the Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realising the Sprint Goal. The Development Team makes forecasts about the required functionality in the next Increment and the work needed to deliver that functionality to Done, thereby creating the Sprint Backlog in the process. The Development Team owns the Sprint Backlog.



The Sprint Backlog makes identifies the work that the Development Team prioritises. The Sprint Backlog must include at least one item from the Sprint Retrospective meeting. the Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. Often the requirement are deducted from User Stories.

Increment

An Increment is the sum of all the items of a Product Backlog completed during a Sprint and the all the previous Sprint combined. At the end of a Sprint, the new Increment must be Done. An Increment should be measurable, a step towards a vision or goal and usable regardless of whether it is released or deployed. It is agreed upon by the Scrum Team and can evolve throughout the process.

```
                                                  Daily Scrum
Product Backlog ------------> Sprint Backlog -|<- ---------> Sprint Review -----> Sprint Retro
                  Planning                               Sprint
```

Agile Methodology

Elements of Agile and Scrum will be utilised in this project. Since this is an individual project, the Agile role, events and artefacts will be defined as follows:

There will be a *Project Owner* who will be responsible for planning, creating, managing and deploying the entire software project. The duties of the Scrum Master will not be conducted. The Product Owner's responsibility of creating and defining the Product Backlog will be conducted by the *Project Owner*. The Development of the software and the Sprint Backlog will be undertaken by the *Project Owner*. The Project Owner is also responsible for changing the Sprint Backlog as the Sprint progresses.

The *Agile Events* will consist of a single Sprint, in which the entire project will be completed. This project will have a Sprint Planning phase, which may involve a simulation with QA as the client. In this meeting the client requirements will be gathered and the user stories will be mutually reviewed and assessed. *Artefacts* such as the Product Backlog and Sprint Backlog will be created from the client requirements and user stories based on MoSCoW principles. After which the Sprint will be conducted. After the completion and demonstration of the project, a Project Retrospective will be conducted with or without client interaction. This Project Retrospective will detail what could have done better, room for improvements in the software and how the process could be improved. Daily Scrums will not be conducted and there will be no team meetings.

Program Overview

In the first week Software principles such as Agile and Scrum were introduced. To effectively manage time and create user stories and complete software projects, a technology called Trello was introduced.

In the second week, SQL, Databases, Networks and Network Security were taught. Cloud Computing and its fundamentals were taught, different theoretical benefits of the cloud such as scalability, availability and low latency were introduced. Different cloud models, such as Public, Private, Community and Hybrid Clouds were introduced. Different cloud platforms and their benefits and constraints were discussed. This was followed by exercises involving setting up different virtual machines, how to set firewall configurations and connect a SQL database to a VM.

In the third week python was introduced along with how to push to GitHub and Git. Other aspects of python coding were also introduced such as running tests, setting up virtual environments, using PyMySQL. Python was then used to create a basic banking application. This was followed by an introduced to DevOps, Continuous Integration, Continuous Delivery and Continuous Deployment. A complete Continuous Integration Pipeline was introduced along with the concept of DRY (do not repeat yourself) and the importance of automation. Different features of Git and Jenkins were introduced such as branching, polling and webhooks. This was followed by an introduction to Test Driven Development and basic response codes for the HTTP(S) protocols.

In the fourth week, the concept of using secure socket shells to use other VMs was demonstrated. Libraries such as Flask andSQL Alchemy were introduced, through independent research

Bootstrap and WTForms were also discovered. This the fundamentals of how to utilised environment variables was presented. Jinja2 and form validators were taught.

In the fifth week, how to automate the deployment of flask through Systemd and Jenkins was taught. Then unicorn and unit testing in flask was discussed and taught.

Project Overview

The requirements have been listed below:

Presentation Requirements	
#	Requirements
1	Agile Methodology
2	Test Driven Development
3	Trello
4	ERD Diagrams
5	User Stories
6	Product Backlog
7	Sprint Backlog
8	Processes
9	Use Case Scenarios
10	CI Pipeline
11	Risk Assessment
12	Tests Log

The MoSCoW has been listed below:

#	MoSCoW	
1	MUST	Have Unit tests for all web pages
2	MUST	Use Test Driven Development methodology
3	MUST	Have automated deployment with a Jenkins CI Server
4	MUST	Have Version Control System (VCS) and a GitHub repository
5	MUST	Be developed on a feature/development branch
6	MUST	Use python
7	MUST	Use Flask
8	MUST	Use a could hosted database and GCP Compute Engine
9	MUST	Have a Trello board(s) with User Stories, User Requirements, Sprint Backlog
10	SHOULD	CSS template

11	SHOULD	Have unique book identifies like ISBN
12	SHOULD	Basic instance protections
13	SHOULD	Have Login features with hashed/encrypted passwords
14	COULD	Search
15	COULD	Have images
16	WOULD	Defend against security attacks

Introduction

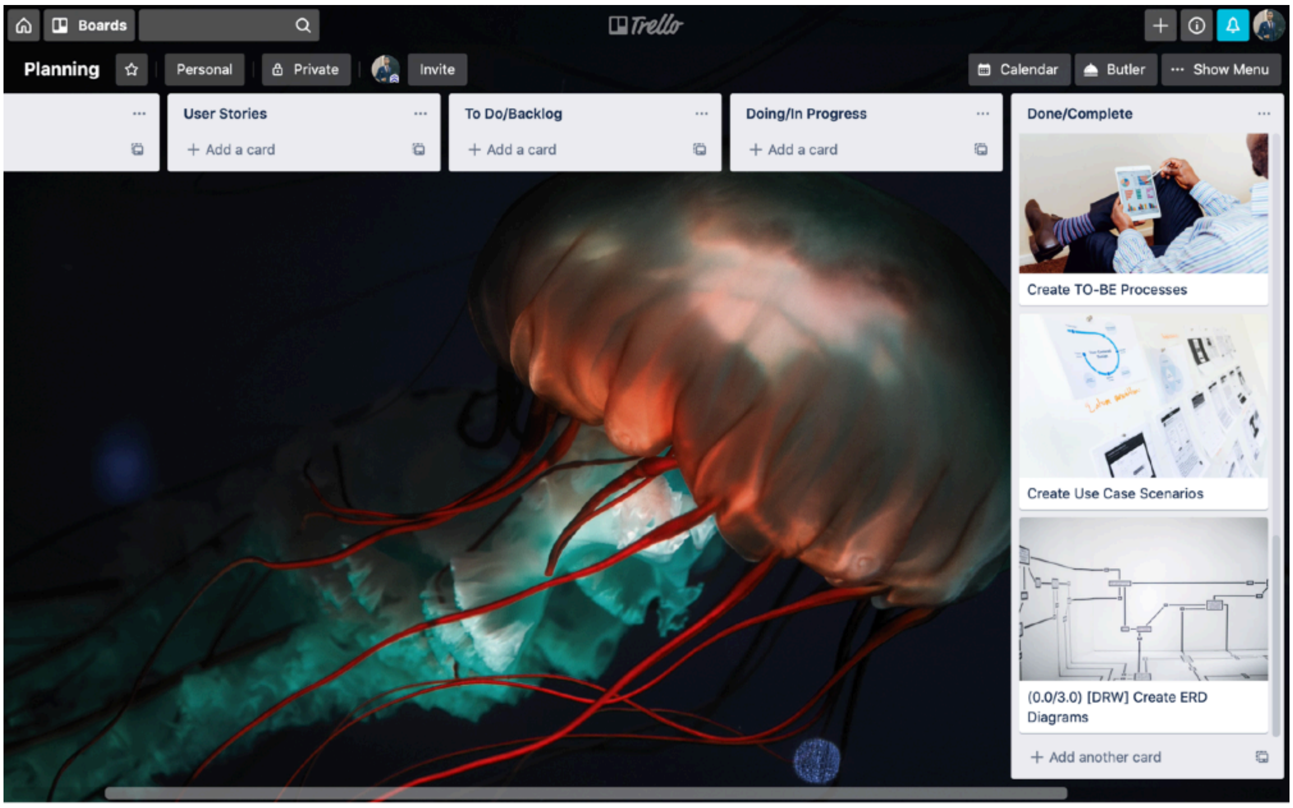
The project aims to create a book review application, that allows users to add, delete and update books and reviews to their respective pages.

Personal Goals

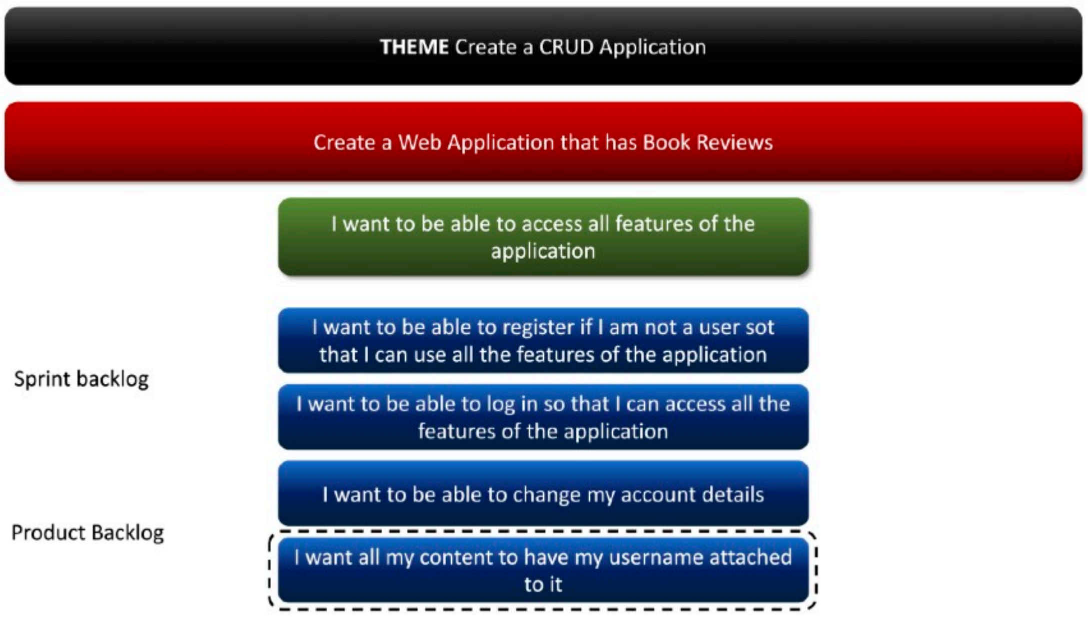
This included following the Keep it Simple Stupid (KISS) method. Hence, the entire project was designed to keep the GUI and user experience as simple as possible. It primarily aims to demonstrate that besides CRUD functionality, that all the other CI features had been included.

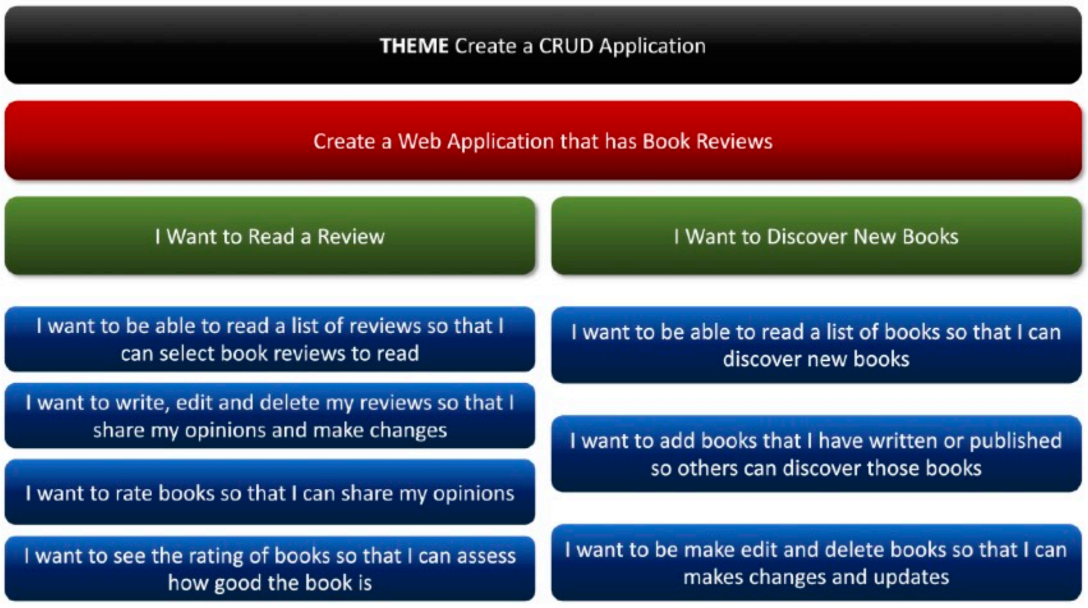
Project

Trello was utilised to create a plan. During this phase research was conducted on Agile and Kanban Trello boards and how they have been utilised by different companies such as MicroSoft. After which Trello was used to create User Stories, Sprint Backlog Items and the Backlog items were moved from To Do, Doing to Done as the project progressed. The Trello Board is shown below:

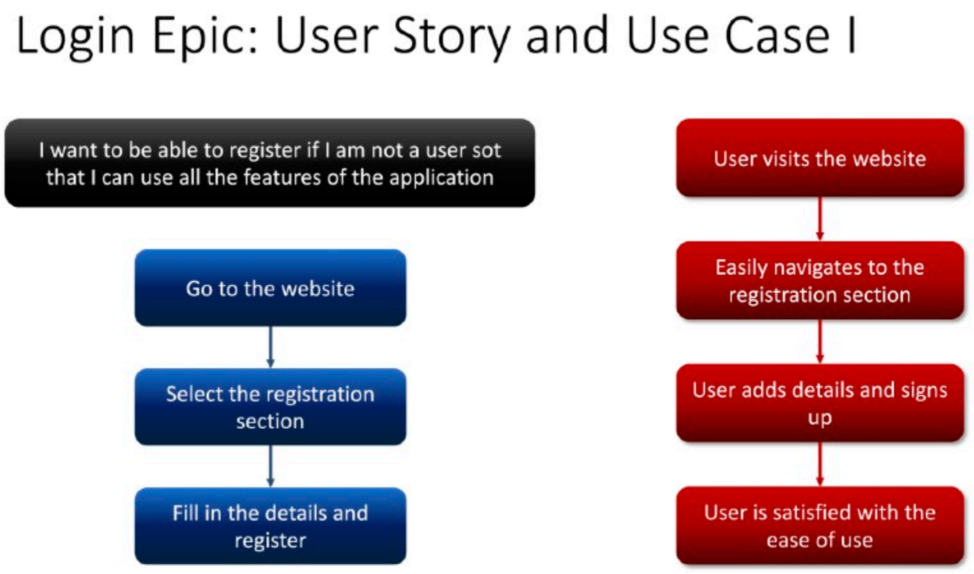


After completing this Initiatives, Themes Epics were researched which resulted in the following epics and user stories:

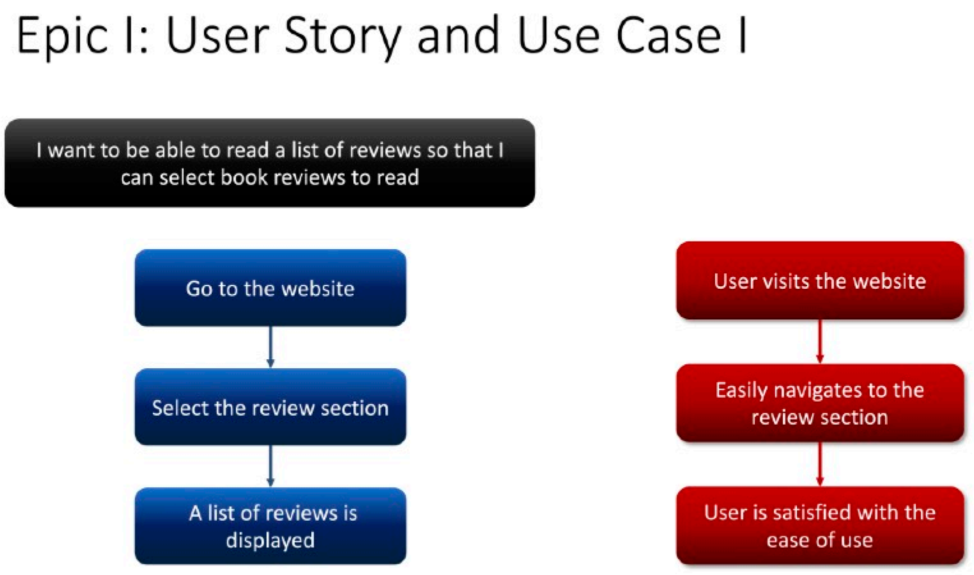

THEME Create a CRUD Application
Create a Web Application that has Book Reviews
I want to be able to access all features of the application
Sprint backlog
I want to be able to register if I am not a user sot that I can use all the features of the application
I want to be able to log in so that I can access all the features of the application
Product Backlog
I want to be able to change my account details
I want all my content to have my username attached to it


THEME Create a CRUD Application
Create a Web Application that has Book Reviews
I Want to Read a Review
I Want to Discover New Books
I want to be able to read a list of reviews so that I can select book reviews to read
I want to write, edit and delete my reviews so that I share my opinions and make changes
I want to rate books so that I can share my opinions
I want to see the rating of books so that I can assess how good the book is
I want to be able to read a list of books so that I can discover new books
I want to add books that I have written or published so others can discover those books
I want to be make edit and delete books so that I can makes changes and updates

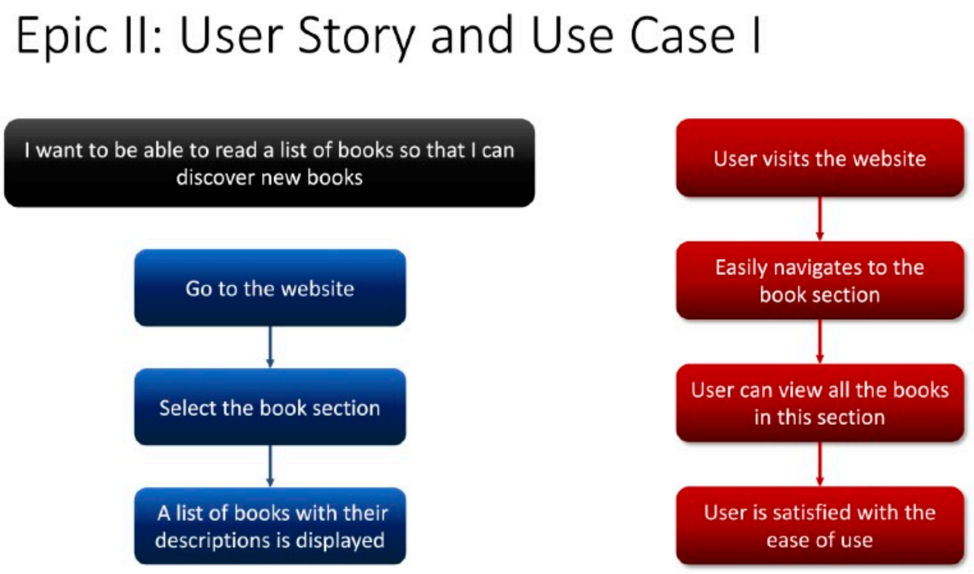
A user story, its process and use case from the Login Epic has been shown below:


Login Epic: User Story and Use Case I
I want to be able to register if I am not a user sot that I can use all the features of the application
Go to the website
Select the registration section
Fill in the details and register
User visits the website
Easily navigates to the registration section
User adds details and signs up
User is satisfied with the ease of use

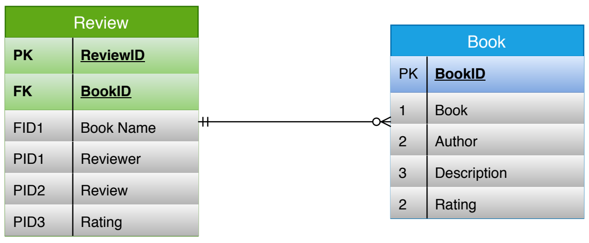
A user story, its process and use case from the Reading Reviews epic has been shown below:


Epic I: User Story and Use Case I
I want to be able to read a list of reviews so that I can select book reviews to read
Go to the website
Select the review section
A list of reviews is displayed
User visits the website
Easily navigates to the review section
User is satisfied with the ease of use

A user story, its process and use case from the Books epic has been shown below:

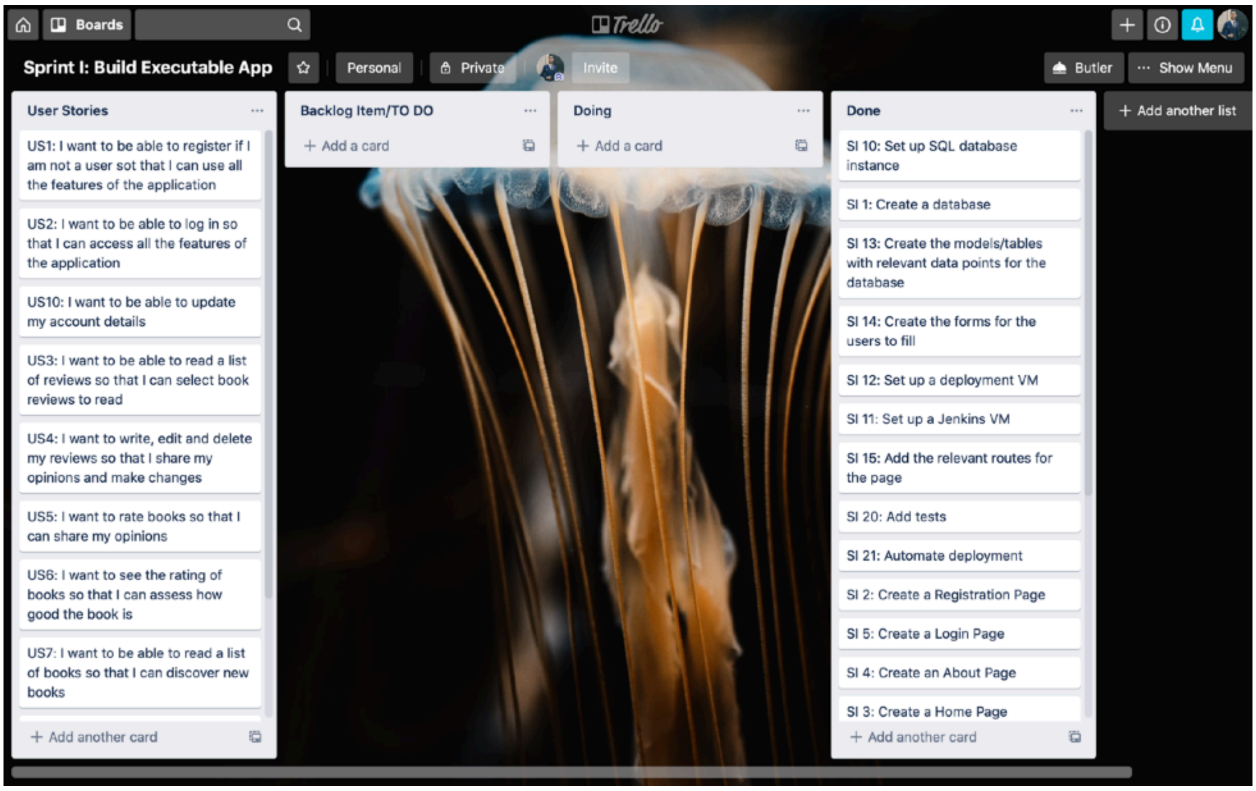

Epic II: User Story and Use Case I
I want to be able to read a list of books so that I can discover new books
Go to the website
Select the book section
A list of books with their descriptions is displayed
User visits the website
Easily navigates to the book section
User can view all the books in this section
User is satisfied with the ease of use

An initial Entity Relationship Diagram was created as demonstrated below:



The source code was connected to GitHub, which allowed version control and the ability to switch between different versions. It also allowed the project to be pulled onto different machines; and the addition and testing of new features from different machines.

The requirements for the project were traced using Trello. The Trello Sprint Board is shown below:


Sprint I: Build Executable App

User Stories
- US1: I want to be able to register if I am not a user sot that I can use all the features of the application
- US2: I want to be able to log in so that I can access all the features of the application
- US10: I want to be able to update my account details
- US3: I want to be able to read a list of reviews so that I can select book reviews to read
- US4: I want to write, edit and delete my reviews so that I share my opinions and make changes
- US5: I want to rate books so that I can share my opinions
- US6: I want to see the rating of books so that I can assess how good the book is
- US7: I want to be able to read a list of books so that I can discover new books

Backlog Item/TO DO

Doing

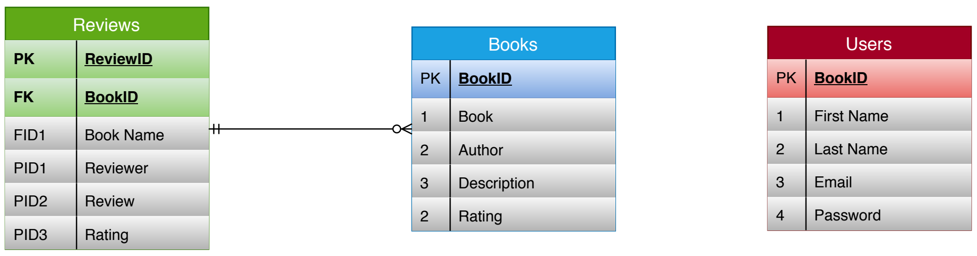
Done
- SI 10: Set up SQL database instance
- SI 1: Create a database
- SI 13: Create the models/tables with relevant data points for the database
- SI 14: Create the forms for the users to fill
- SI 12: Set up a deployment VM
- SI 11: Set up a Jenkins VM
- SI 15: Add the relevant routes for the page
- SI 20: Add tests
- SI 21: Automate deployment
- SI 2: Create a Registration Page
- SI 5: Create a Login Page
- SI 4: Create an About Page
- SI 3: Create a Home Page

A SQL database in GCP was spun up along with a VM for Jenkins and another for deployment. An Agile and Test Driven Approach was utilised, hence how to conduct tests was researched. Consideration of HTTP responses and the string responses within the HTML were made. A risk assessment was conducted and is shown below:

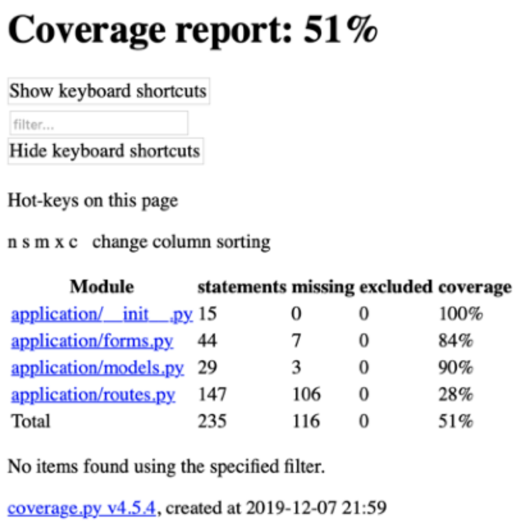
Risk Assessment				
Risk	Response Strategy	Likelihood	Impact	Risk Level
		After Prevention		
Hacking Scan	Terminate: Use environment variables and configuration files to prevent sensitive data being	8	0	0
Inadequate Functionality	Treat: Conduct static, dynamic testing and manual checking of software and whether it meets requirements	6	8	4.8
Inadequate Software Architecture	Treat: Testing of links between different software such as GitHub, Jenkins and Flask. Checks of integration between different instances	7	8	5.6
No Login Requirements	Terminate: Introduction of login requirements to make changes	0	8	0
Poor IP access rights to instances	Treat: Ensure that access rights have specific IP addresses or IP address ranges	2	8	1.6

Distributed Denial of Services	Treat: For a small CRUD application this threat should be tolerated and	2	8	1.6
Dictionary attack	Terminate: Use hashing; using dummy additions to each password can explored in the future	6	0	0
Password Attack	Treat: Setting a limit on the number of times a user can attempt to log in	3	2	0.6
IP Spoofing	Treat: Switching to IPv6, deploy packet filtering	2	8	1.6

A virtual environment was created, the program was written in a modular form and uploaded to branch on GitHub. This modular form enabled quick troubleshooting throughout the processes. As the software development progressed, the ERD was updated to the following:



Minor changes were intentionally made to follow the KISS principle (Keep it Simple Stupid). Furthermore, CRUD functionality tests were also taken into account. such as CRUD functionality, login and registration. Front-end development was also considered and was added to the Sprint. During the Sprint phase more more test consideration were taken into account, In total over 30 tests were written and tested:



Coverage report: 51%

Show keyboard shortcuts

filter...

Hide keyboard shortcuts

Hot-keys on this page

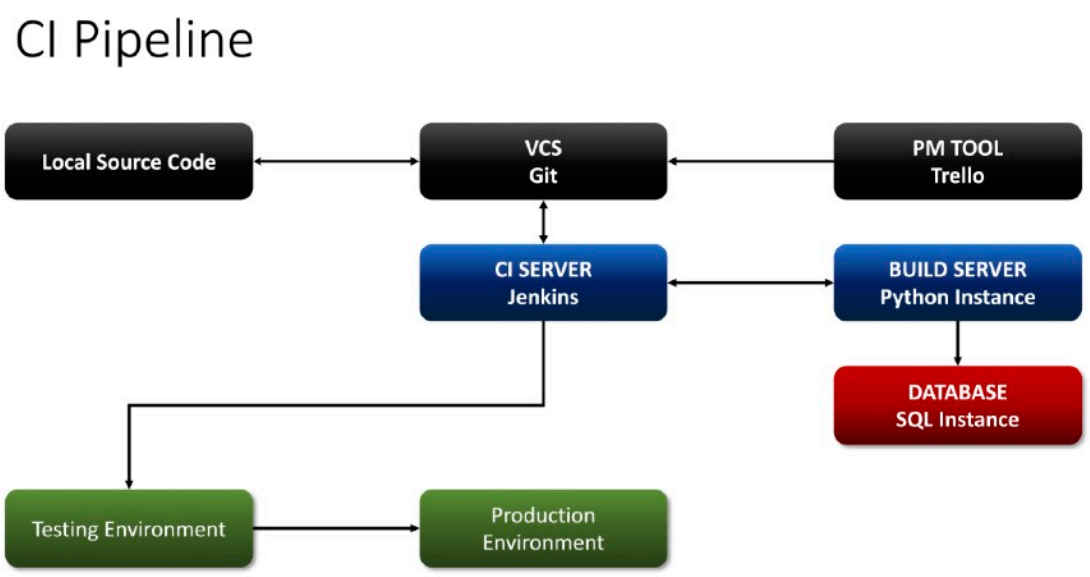
n s m x c   change column sorting

Module	statements	missing	excluded	coverage
application/__init__.py	15	0	0	100%
application/forms.py	44	7	0	84%
application/models.py	29	3	0	90%
application/routes.py	147	106	0	28%
Total	235	116	0	51%

No items found using the specified filter.

coverage.py v4.5.4, created at 2019-12-07 21:59

This was followed by integration with between the deployment VMs and Jenkins resulting in the following pipeline:

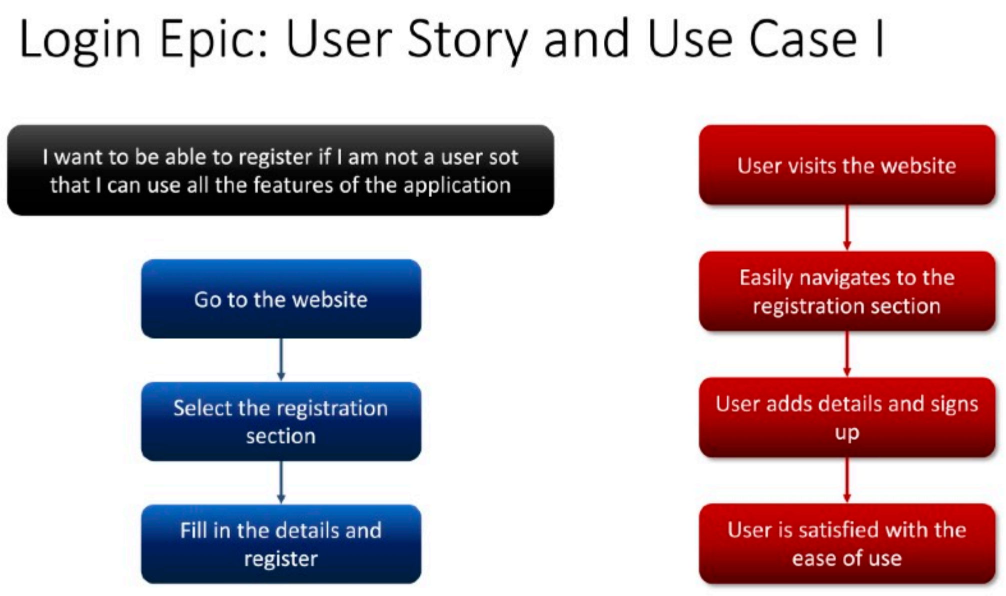


Further Improvements and Future

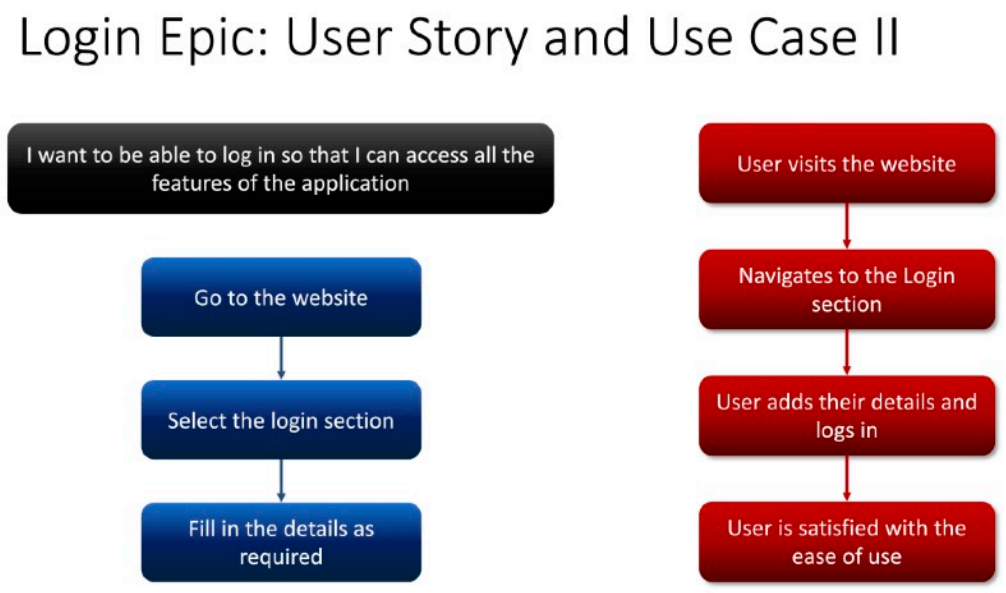
There are many improvements that can be made to this application. For example, there could be more stringent policies for who could add a book and less stringent requirements for who can add a reviews. Other features such as search could be added as well. More tests including tests of how the application behaves after a user has logged in could be added. Security could also be enhanced through the addition of dummy data to passwords before hashing and through the randomisation of the dummy data to ensure there are no patterns for the dummy data. The user could also be displayed an image along side each book. Usernames could be attached to each review and improvements to the front end can be made by changing the colours and fonts to fit book genre.

APPENDIX I

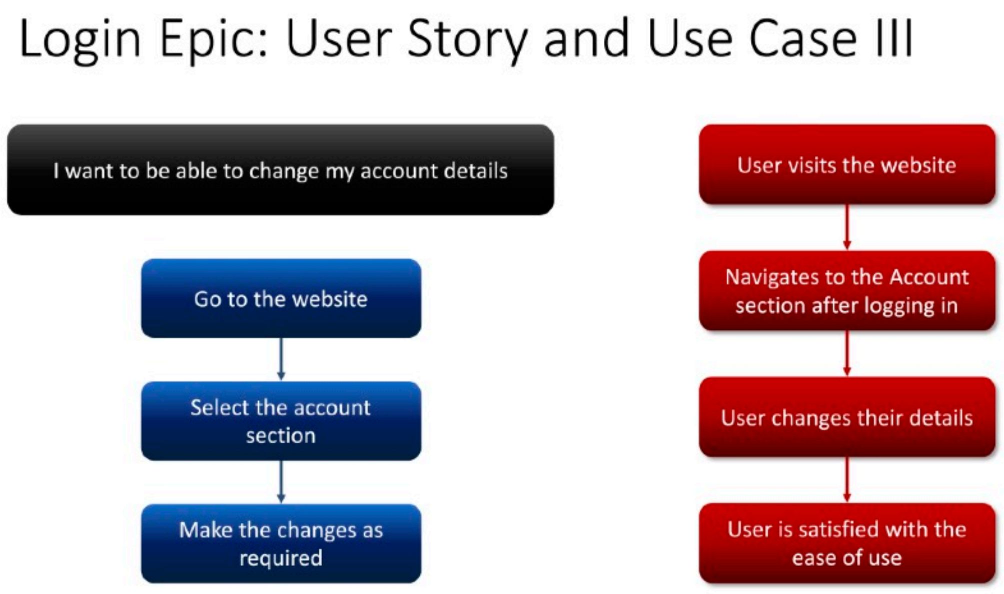
LOGIN EPIC: USER STORY I


Login Epic: User Story and Use Case I
I want to be able to register if I am not a user sot that I can use all the features of the application
Go to the website
Select the registration section
Fill in the details and register
User visits the website
Easily navigates to the registration section
User adds details and signs up
User is satisfied with the ease of use

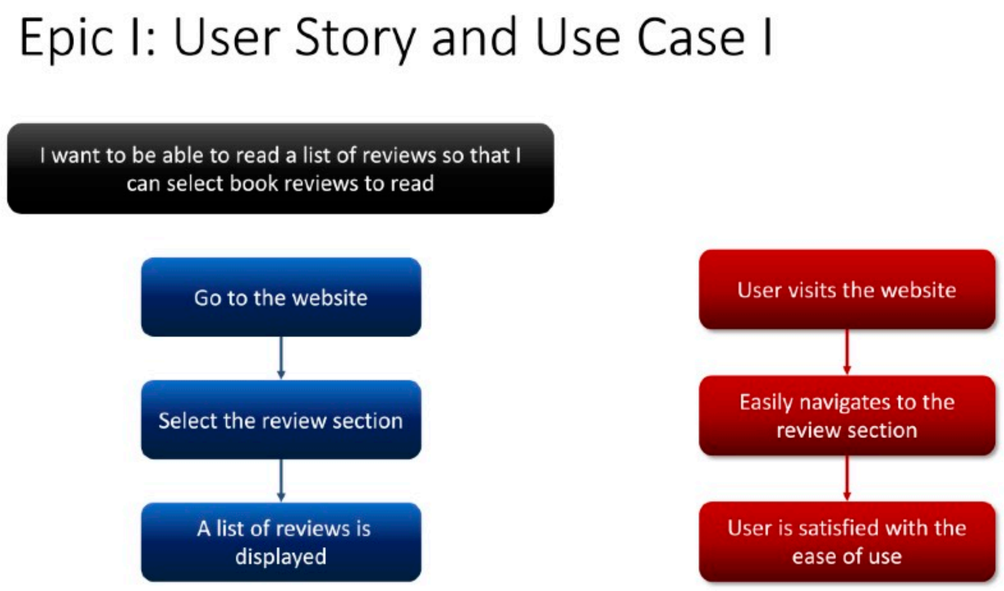
LOGIN EPIC: USER STORY II


Login Epic: User Story and Use Case II
I want to be able to log in so that I can access all the features of the application
Go to the website
Select the login section
Fill in the details as required
User visits the website
Navigates to the Login section
User adds their details and logs in
User is satisfied with the ease of use

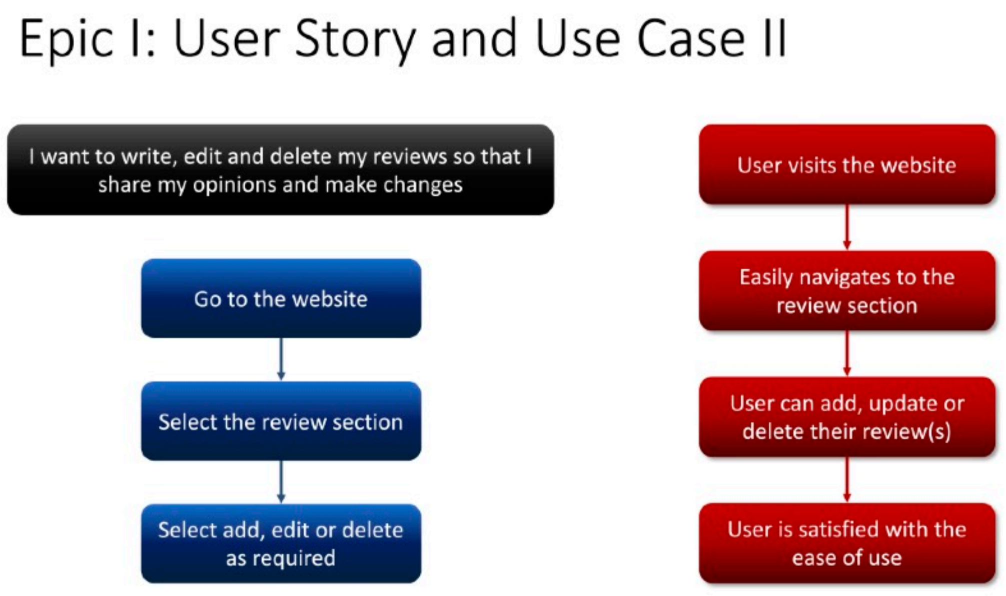
LOGIN EPIC: USER STORY III:


Login Epic: User Story and Use Case III
I want to be able to change my account details
Go to the website
Select the account section
Make the changes as required
User visits the website
Navigates to the Account section after logging in
User changes their details
User is satisfied with the ease of use

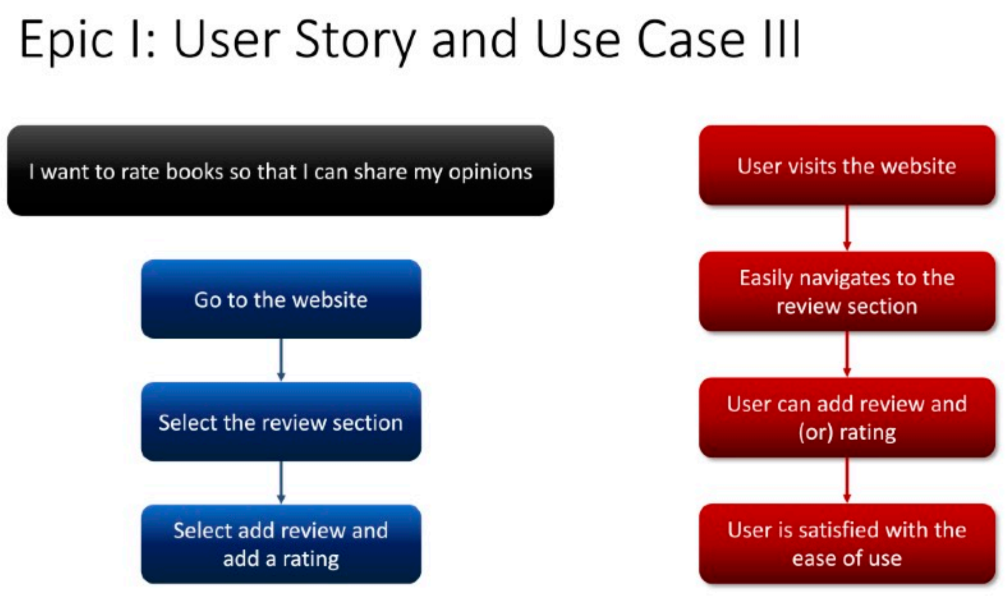
EPIC I: USER STORY I:


Epic I: User Story and Use Case I
I want to be able to read a list of reviews so that I can select book reviews to read
Go to the website
Select the review section
A list of reviews is displayed
User visits the website
Easily navigates to the review section
User is satisfied with the ease of use

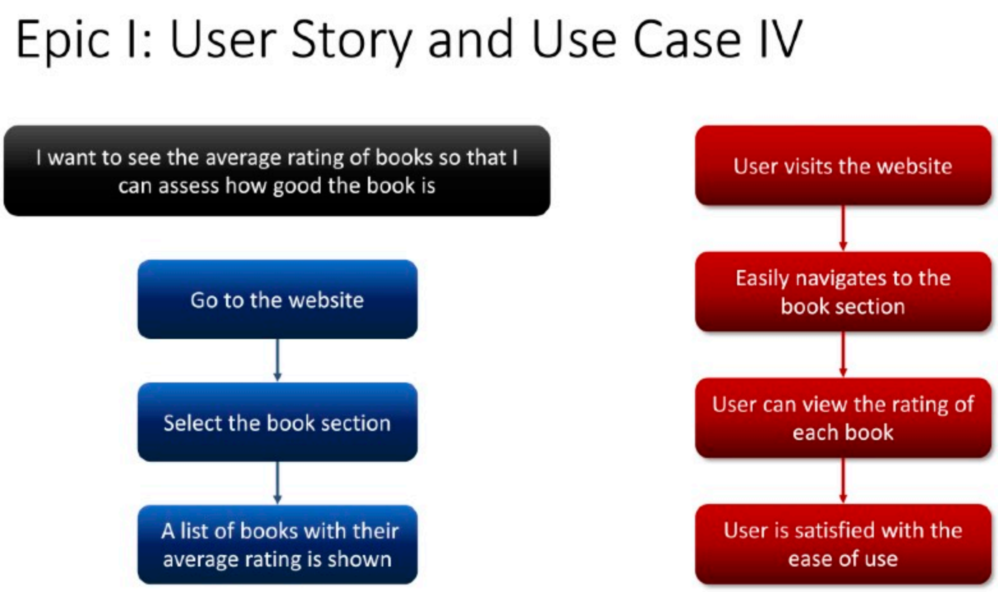
EPIC I: USER STORY II:


Epic I: User Story and Use Case II
I want to write, edit and delete my reviews so that I share my opinions and make changes
Go to the website
Select the review section
Select add, edit or delete as required
User visits the website
Easily navigates to the review section
User can add, update or delete their review(s)
User is satisfied with the ease of use

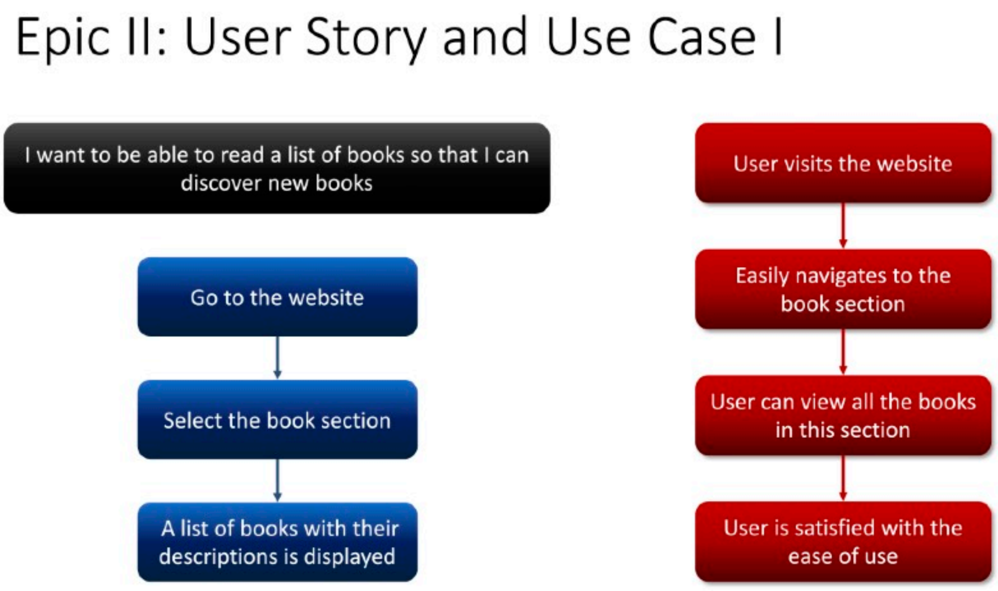
EPIC I: USER STORY III:


Epic I: User Story and Use Case III
I want to rate books so that I can share my opinions
Go to the website
Select the review section
Select add review and add a rating
User visits the website
Easily navigates to the review section
User can add review and (or) rating
User is satisfied with the ease of use

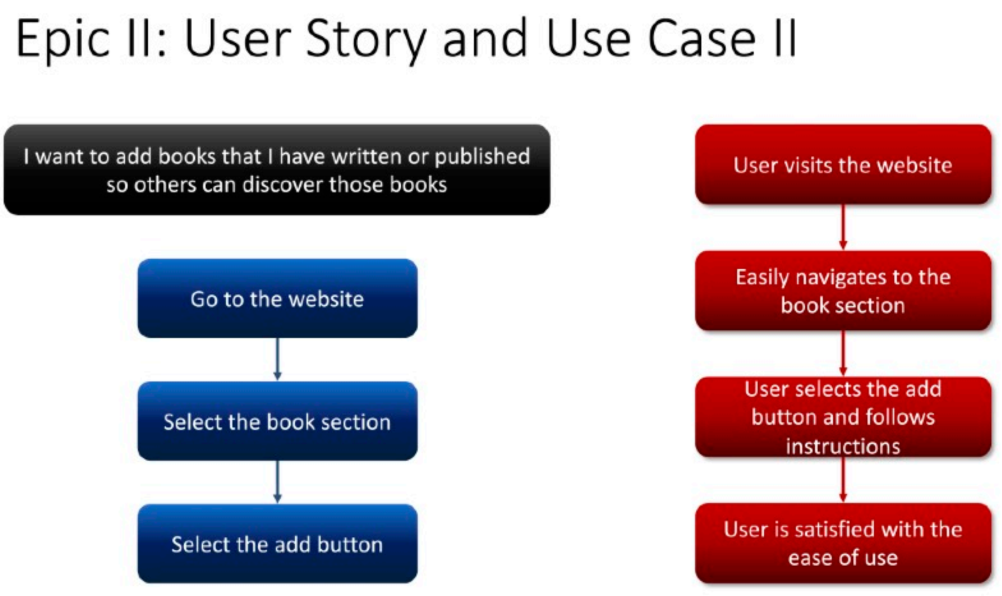
EPIC I: USER STORY IV:


Epic I: User Story and Use Case IV
I want to see the average rating of books so that I can assess how good the book is
Go to the website
Select the book section
A list of books with their average rating is shown
User visits the website
Easily navigates to the book section
User can view the rating of each book
User is satisfied with the ease of use

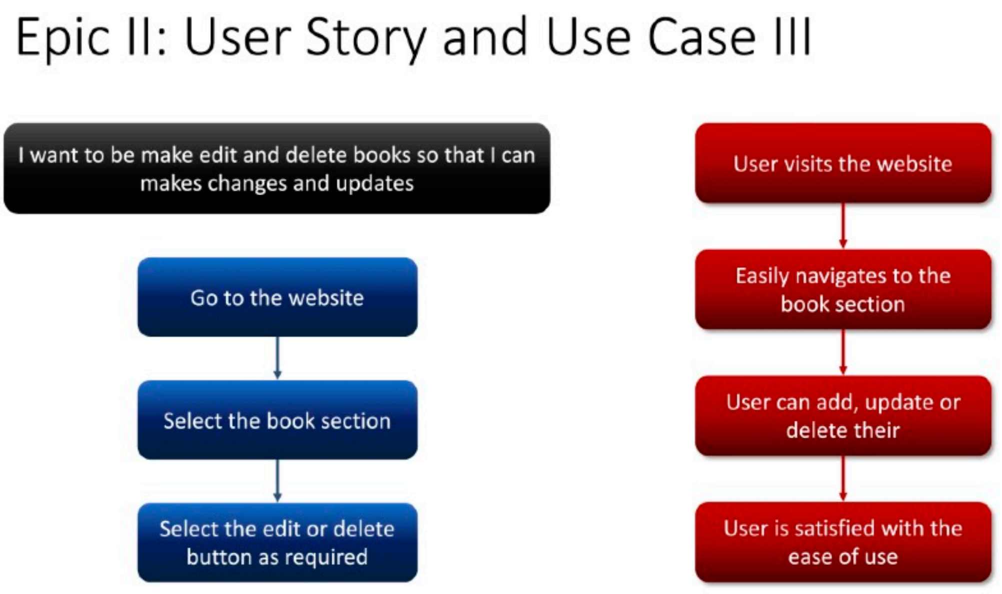
EPIC II: USER STORY I:


Epic II: User Story and Use Case I
I want to be able to read a list of books so that I can discover new books
Go to the website
Select the book section
A list of books with their descriptions is displayed
User visits the website
Easily navigates to the book section
User can view all the books in this section
User is satisfied with the ease of use

EPIC II: USER STORY II:


Epic II: User Story and Use Case II
I want to add books that I have written or published so others can discover those books
Go to the website
Select the book section
Select the add button
User visits the website
Easily navigates to the book section
User selects the add button and follows instructions
User is satisfied with the ease of use

EPIC III: USER STORY III:


Epic II: User Story and Use Case III
I want to be make edit and delete books so that I can makes changes and updates
Go to the website
Select the book section
Select the edit or delete button as required
User visits the website
Easily navigates to the book section
User can add, update or delete their
User is satisfied with the ease of use

APPENDIX II: EXAMPLE

#	Product Backlog		
	SPRINT	USER STORY	Requirements
1	SPRINT I	US1: I want to be able to register if I am not a user sot that I can use all the features of the application	Set up SQL database instance
2	SPRINT I	US2: I want to be able to log in so that I can access all the features of the application	Create a database
3	SPRINT I	US10: I want to be able to update my account details	Create the models/tables with relevant data points for the database
4	SPRINT I	US3: I want to be able to read a list of reviews so that I can select book reviews to read	Create the forms for the users to fill
5	SPRINT I	US4: I want to write, edit and delete my reviews so that I share my opinions and make changes	Set up a deployment VM
6	SPRINT I	US5: I want to rate books so that I can share my opinions	Set up a Jenkins VM
7	SPRINT I	US6: I want to see the rating of books so that I can assess how good the book is	Add the relevant routes for the page
8	SPRINT I	US7: I want to be able to read a list of books so that I can discover new books	Add tests
9	SPRINT I	US8: I want to add books that I have written or published so others can discover those books	Automate deployment
10	SPRINT I	US9: I want to be make edit and delete books so that I can makes changes and updates	Create a Registration Page
11	SPRINT II	I want people to know which reviews I write	Update ERD and models
12	SPRINT II	I want to be able to see images of the books	Research and update relevant files
13	SPRINT II	I want to be able to search for books and reviews so that I know who the authors are	Add search feature