



|  |
| --- |
| **CLEVO - a budding sharemarket investor application**  **Technical Solution Design** |



Version: V1.0

Date: 18/09/2016

Sponsor: RMIT

Author: Carlo R Beasley, Lucas Brook, Evan Le Clercq, Vio Marcu, Ocal Ogten

Document Control

**Distribution**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Issued** | **Recipient** | **Position** |
| V 1.0 | 19/11/2016 | Amir Homayoon Ashrafzadeh | Supervisor |
|  |  |  |  |

**Amendment History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Section** | **Page** | **Version** | **Comment** |
| <Enter Doc. Section No.> | <Enter Page No.> | <Enter Version No.> | <Enter Comments to explain the reason for the document text or other changes,    e.g., Updated text after walkthrough with the stakeholders, or    e.g., Updated section after technical consultation> |
|  |  |  |  |

.

**Staff or Entities Consulted**

|  |  |
| --- | --- |
| **Name** | **Position / Organization** |
| Carlo R Beasley  Lucas Brook  Evan Le Clercq  Vio Marcu  Ocal Ogten | Development Team  Team Leader  Development Team  Project Manager  Development Team |

**Related Documents**

|  |  |  |
| --- | --- | --- |
| **Name** | **Author** | **Description** |
| <Enter Document Name> | <Enter Author> | <Enter Document Description> |
|  |  |  |

***Preface***

The purpose of this document is to outline the Technical Solution Design for the Budding Sharemarket Investor application project.This document describes the technical environment, overall project architecture, system architecture, application functionalities and features, database architecture, implementation instructions, non-functional specifications, summary of test results, known issues and risks.

[**1. Introduction**](#_xvbmwprndg7x)[**6**](#_xvbmwprndg7x)

[**2. Technical Environment**](#_158op5pdgz9)[**7**](#_158op5pdgz9)

[**3. Overall Architecture**](#_2kwthinhb037)[**9**](#_2kwthinhb037)

[**4. System Architecture**](#_25jwuhbqfk1i)[**10**](#_25jwuhbqfk1i)

[4.1 Functionalities/features](#_nm2cgovgyg17) [10](#_nm2cgovgyg17)

[4.1.1 Register](#_163108fj05e7) [10](#_163108fj05e7)

[4.1.2 Login/Logout](#_1x9dm0qs5q05) [11](#_1x9dm0qs5q05)

[4.1.3 Buy Shares](#_cy71w5al9cch) [12](#_cy71w5al9cch)

[4.1.4 Sell Shares](#_3h04q6fvai6u) [14](#_3h04q6fvai6u)

[4.1.5 Application provides live updates of share pricing and fluctuations](#_p1p48l869zkq) [15](#_p1p48l869zkq)

[4.1.6 Application provides player’s current balance](#_uzrs13j8mzcm) [15](#_uzrs13j8mzcm)

[4.1.7 Application maintains an updated leader board](#_t2gdruqk0jff) [16](#_t2gdruqk0jff)

[4.1.8 Application offers appropriate admin functionality](#_jdwp58c0a02o) [16](#_jdwp58c0a02o)

[**5. Database Architecture**](#_g25rt76gdc92)[**17**](#_g25rt76gdc92)

[**6. Implementation Instructions**](#_3azz36iqykn)[**18**](#_3azz36iqykn)

[**7. Non-functional specifications**](#_y0devlusz1ak)[**19**](#_y0devlusz1ak)

[**8. Summary of test results**](#_sqr022plndrp)[**19**](#_sqr022plndrp)

[**9. Known Issues & Risks**](#_bxifl0fmaniz)[**19**](#_bxifl0fmaniz)

[**10. Other Considerations - Ask Homy**](#_zc89go42ouwo)[**19**](#_zc89go42ouwo)

[**11. Appendix**](#_299l28uhgi6d)[**19**](#_299l28uhgi6d)

# 

# 

# 

# 1. Introduction

Summary of the technical solution that was completed including:

· Brief project description

· Brief description of technical environment

· Estimated level of complexity

· **Estimated benefits/Problems it solved (quantitative & qualitative)***.*

The Budding Sharemarket Investor game will allow players to learn how the sharemarket works using a virtual share market platform. This will give players the experience required to make informed decisions regarding buying and selling shares in a controlled environment with bogus money before they enter the workforce and begin trading shares using actual money of their clients. The Budding Sharemarket Investor game will allow players to: create an account, log into the web application and use bogus money to simulate buying and selling of shares based on actual ASX data. Data will be retrieved via web service calls to a finance API, and will be used in the buying, selling and tracking of shares in the market.

The budding investor game will be hosted online on an AWS environment and built with a combination of PHP, mySQL, HTML and CSS. Users will be able to navigate to the site, login or register, and begin dealing in the faux marketplace. There will also be admin functionality built into the site. GitHub will be used for a version control tool, and some testing tools such as Unit Testing and PHP Testing.

The application will initially display a lower level of complexity and as the project development progresses, it will transition into a moderate level of complexity while remaining objective to the project requirements.

**Estimated benefits/Problems it solved (quantitative & qualitative)***.*

# 

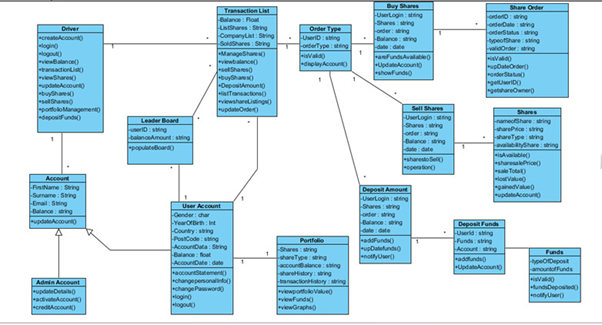
# 

# 2. Technical Environment

|  |  |
| --- | --- |
| **Component** | **Description** |
| **PHP** | PHP is used for all server-side scripting to ensure secure access and modification of data. It also has the ability to do database queries and has a more secure login. |
| **HTML5** | Used to create the web pages of the project |
| **CSS3** | Used to style the web pages |
| **AJAX** | Used to communicate with server-side scripts including JSON |
| **Laravel** | The main framework used to build the PHP project including structuring the location of pages, styling, scripts and databases. |
| **GitHub** | Used for hosting our source control. GitHub integrates with other software used such as Source Tree and can also have more than 5 users connected to it at no charge unlike BitBucket which is limited to 5 users. |
| **SourceTree** | Local Git client used to pull, commit, push, merge and create other branches, whilst connected to the GitHub source control. |
| **Sublime Text, Notepad++** | Tools of choice, used individually by group members to perform coding tasks. |
| **Composer** | A tool for dependency management in PHP. |
| **AWS** | AWS will be used to host the project. It will allow tracking of different versions of our project and where the client can interact with the product: http://clevo-rmit.space/public |
| **Lean Testing** | Used to plan, track and record tests and bugs. |
| **Trello** | Online tool used to manage the project sprints - current in progress, coming up next week, completed, and ongoing. |
| **Slack** | Tool used for group communication, structured into multiple channels such as API, UI, General etc. |
| **XAMPP** | used locally to run the project individually before pushing to GitHub |
| **Google Drive, Docs, Sheets** | Used to storage and collaboration of all documentation related to the project. |
| **PHPUnit** | Used to create unit tests for the project |
| **SQLite** | A lightweight and portable SQL database engine used for storing all user and company data |

# 3. Overall Architecture

<Outline the overall architecture of the solution which details how the system will interact with the world or other systems etc. Explain it using a diagram.>



# 

# 

# 

# 4. System Architecture

<Detail the system that was built/completed. Explain each component thoroughly. A architecture diagram is essential. >

## 4.1 Functionalities/features

1. User Account - Register
2. User Login / Logout
3. User buying of shares
4. User selling of shares
5. Applications provides live updates of share price and fluctuations
6. Application provides a history of purchased shares
7. Application provides player’s current balance
8. Application maintains an updated leader board
9. Application offers appropriate admin functionality

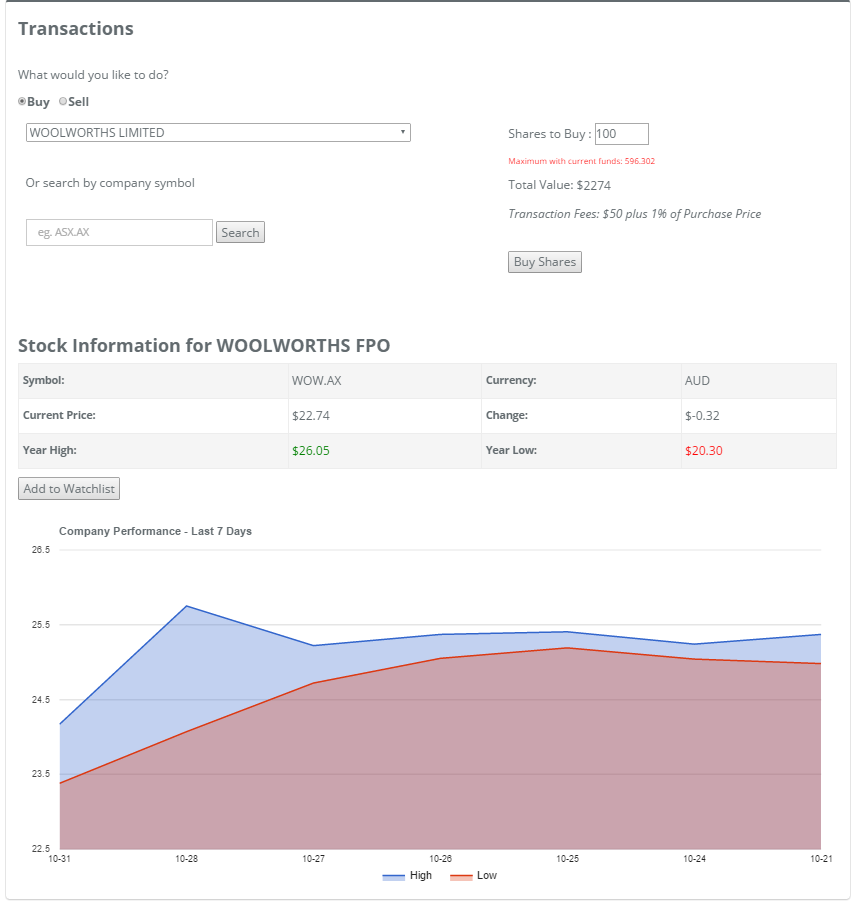
### **4.1.1 Register**

1. User navigates to the website: <http://clevo-rmit.space/public>
2. User clicks on the register button
3. User fills out the Name field
4. User fills out the Email Address field
5. User fills out the City field
6. User fills out the Age field
7. User chooses a password and inputs it into the Password field
8. User confirms password chosen by typing it again into the Confirm Password field
9. User registers the account by pressing the Register button
10. If successful, will be redirected to the Dashboard page
11. If validation fails, user will be notified on the form requirements such as: minimum age must be 18, password must be at least 8 characters long and must contain at least a letter, a number and special character.

### **4.1.2 Login/Logout**

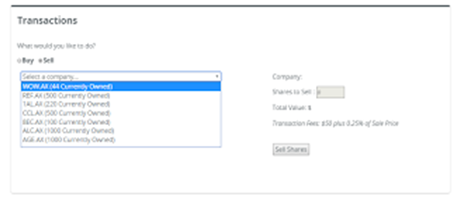
1. User navigates to the homepage: <http://clevo-rmit.space/public>
2. User fills out the Email Address field
3. User fills out the Password field
4. User has the option to tick the “Remember Me” radio box for fast access next time user logs in
5. User clicks on Login and will be redirected to the Dashboard page
6. While user is logged in, throughout the website has the option to logout by clicking/pressing the Logout button located on top right of the page, which will redirect user back to the Login page
7. User is presented with a “Forgot Your Password” option to reset the forgotten password via link sent by email.

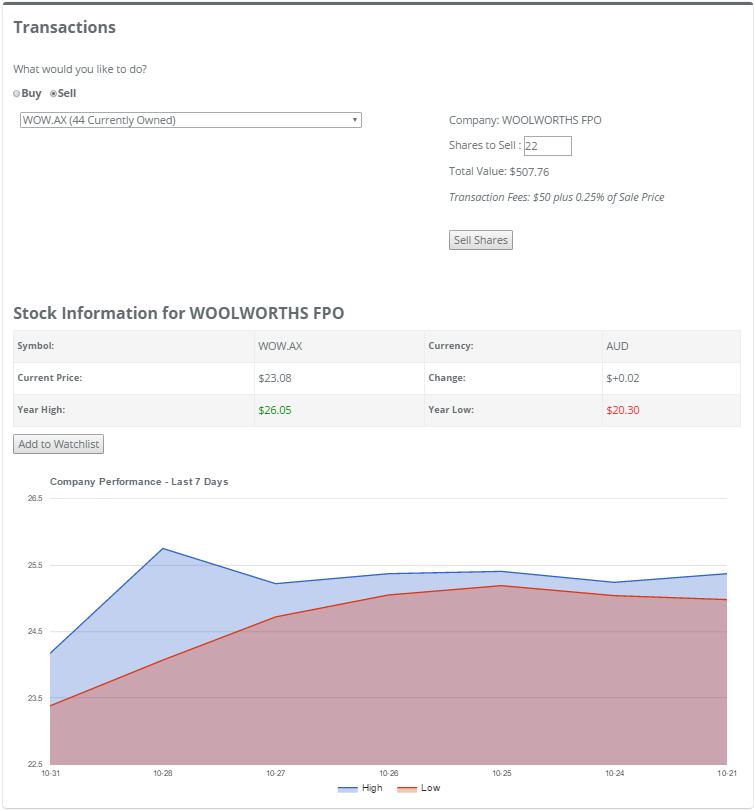
### **4.1.3 Buy Shares**



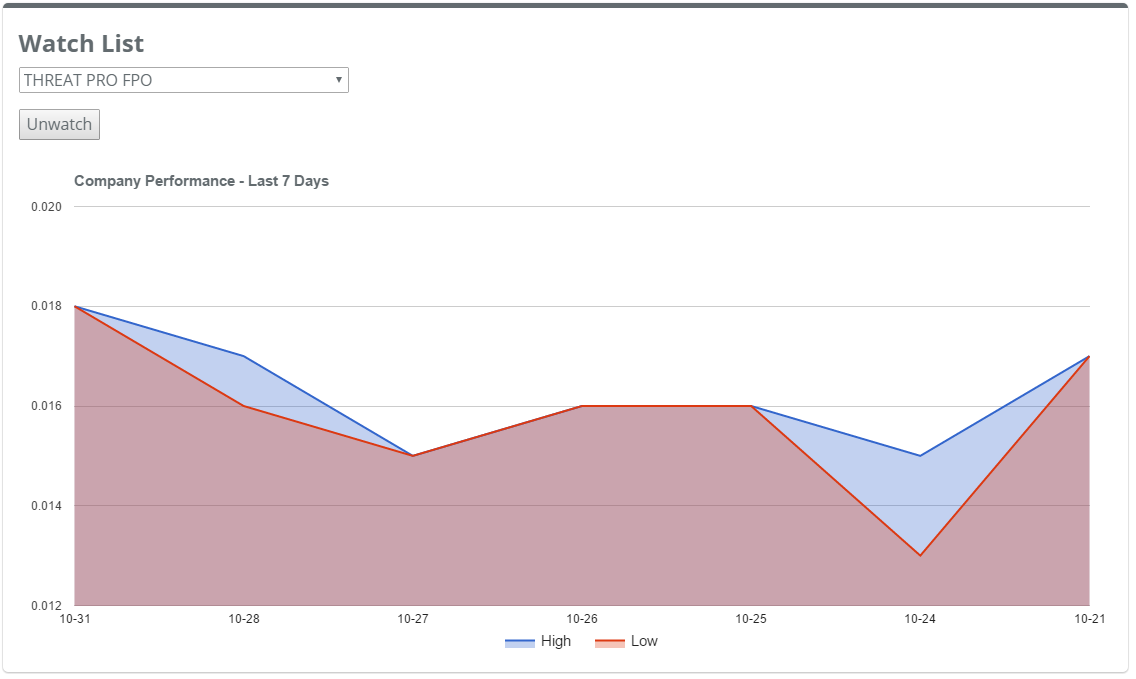
1. User logs in
2. User navigates to Transactions page
3. User will select the Buy radio button
   1. This will present user with an option to select a company from a complete list in a dropdown form and also an option to search for a company by its unique sharemarket code
   2. The buy box will present user with the current share value and latest price changes and also gives user the option to add the company to the watch list
   3. User is also presented with a history chart of previous week share values
   4. The buy box will update with the maximum number of shares that can be purchased based on user’s current balance.
4. User will select number of the shares to be purchased
5. Total value of shares will update accordingly
6. Transaction fees are being displayed for user’s reference
7. User will press the ‘Buy Shares’ button
8. Application to check if the user has sufficient funds to make transaction.
   1. If insufficient funds error message will show.
   2. If sufficient funds will proceed to step 9.
9. Application calculates the total cost of the transaction and updates the ‘current\_balance’ field in the database.
10. Application will update the current\_holdings field in the database with the new stock This will be hashed to increase security of the information.
11. User will be notified of the success of purchase and the amount deducted from their balance.

### **4.1.4 Sell Shares**

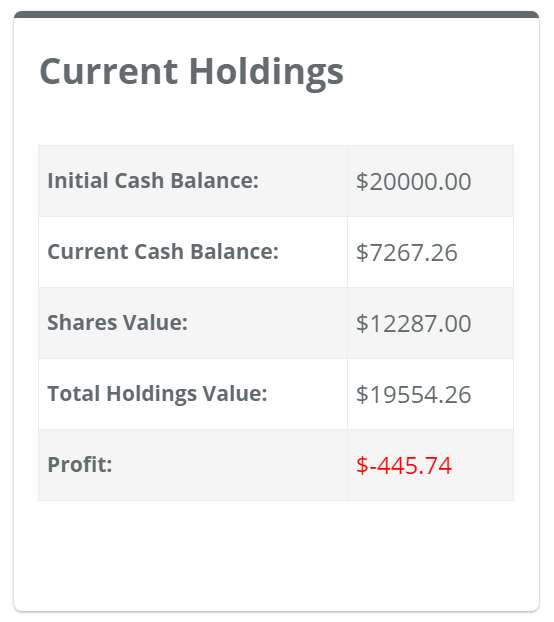


1. User logs in
2. User navigates to Transaction page
3. User selects the Sell radio button
4. User will select desired stock from the Currently Owned shares’ dropdown menu, this will update the sell box with company information
5. User will enter the number of shares to be sold
6. Total value of shares to be sold updated
7. User will push the ‘Sell Shares’ button
8. Application confirms the number of shares entered is equal to or less than the number currently owned.
   1. If too many shares entered error message will be displayed
   2. If number of shares are being sold is valid will proceed to step 9
9. Application calculates the amount to be added to the user balance and update the ‘current\_balance’ field in the database.
10. Application updates the ‘current\_holdings’ field in the database with the new stock information.
11. User is notified of the success of the sale, with a complete transaction summary and option to return to the Dashboard or Transactions page

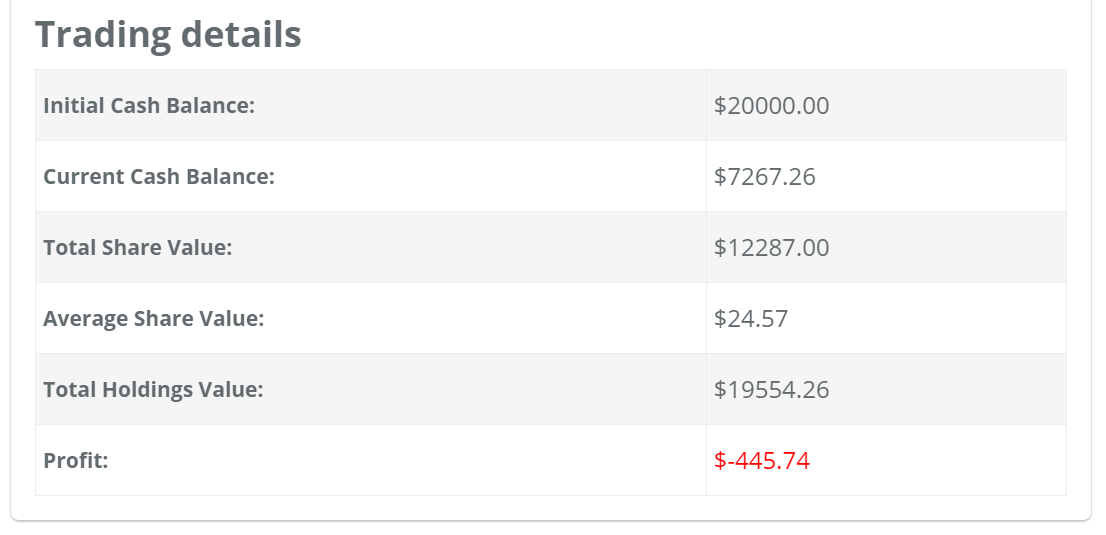
### **4.1.5 Application provides live updates of share pricing and fluctuations**

1. Application presents user with a live share prices on the dashboard page when a company is selected from the Watch List drop down menu
2. Application presents user with a live share price when the buy or sale option is selected and a company chosen from the drop down under the transactions page

### **4.1.6 Application provides player’s current balance**

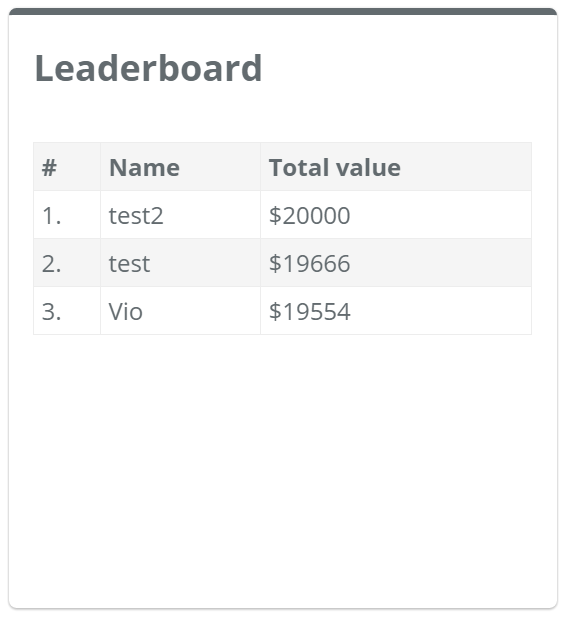


1. User logs in
2. Views current holdings section in the middle of the dashboard which includes the cash balance, the shares values and the total holdings value.



1. User logs in
2. User navigates to profile page
3. User views trading details in the bottom section of the page which includes the current cash balance, total share value the average share value and the total holdings value.

### **4.1.7 Application maintains an updated leader board**



1. User logs in
2. User views the leaderboard section on the righthand side of the dashboard. The leaderboard is populated with all registered users total holdings including cash and share value.

### **4.1.8 Application offers appropriate admin functionality**

# 

# 

# 

# 5. Database Architecture

<Explain the database architecture/schema and why it is built in the way it is and how scalable it will be.>

SQLite has been used as the database solution for a number of reasons. The simplicity and portability of SQLite saves time during development as it allows the database to be embedded in the project files allowing easy access for all developers.The traffic of the final product is expected to be classified as low which also makes SQLite a suitable solution. Laravel, the PHP framework used to develop the application has built-in support for SQLite including the ability to have multiple databases being used concurrently. Due to the nature of the project it is unlikely that the application will be expanded to the point where a more powerful database solution would be needed.

The project features two separate databases:

|  |  |
| --- | --- |
| **Main database** | Contains user account information, user owned shares, balance, transaction history |
| **ASX database** | Contains a complete list of all companies trading on the Australian stockmarket. Includes company name and symbol to be used in API requests. |

**Main database**

The main database is where all user data is stored. This includes registration information, balance, transactions history and currently owned shares.

<INSERT SCHEMA HERE>

**ASX database**

This database contains the name and symbol of all companies trading on the Australian sharemarket. This data has been kept separate because it does not need to be modified and can safely be set to read-only. Instead of being modified the whole database file can be replaced when the ASX list needs updating. Another reason this data was isolated from the main database was to facilitate the addition of data from different markets. The entire database can also be replaced allowing the project to be switched to a different market with zero modification to the rest of the project.

<INSERT SCHEMA HERE>

# 6. Implementation Instructions

*< List the implementation instructions and the basic specs of the server needed. Also provide details regarding data migration or if it’s replacing or enhancing an existing system what are the steps needs to take to ensure there is no data loss.>*

The application requires basic hardware and either a Windows or MAC environment. It can be deployed on a cloud environment such as AWS free of charge (or a very minimum fee of $0.75 - $1.5/week for additional features such as adding a custom domain name).

Migrating data from a local environment to a cloud environment consists of the simple task of uploading a zipped folder, which contains the entire project, to a cloud environment such as AWS, followed by deployment which consists of selecting the most current version of the app and choosing the deploy option. The AWS systems will enable the most recent version f the website when the user navigates to the homepage.

To ensure there was no data loss, we kept the source code in a cloud environment called GitHub and also, each member of the group kept a local copy of the entire project.

Within GitHub and SourceTree, we have a number of commits since the beginning of the project and at any given time we can reverse changes if proven undesirable.

Within AWS we have a number of versions deployed with a short description against each version. This gives us an idea of what each version represents, and again we can reverse the online version to any previous version at any time in a matter of minutes. This is another way of backing up our work environment.

# 

# 

# 7. Non-functional specifications

<Detail all the non-functional specification of the system.>

Admin functionality - To be completed on Saturday/Sunday 19th/20th November

# 8. Summary of test results

<Provide the summary of the test cases and results in tabular format.>

The Lean Testing tool was used to conduct tests and identify bugs. We also used the Testing Register form to document the tests performed. Here is a summary of the test results:

To be completed as soon as we have testing register document completed by Sunday 20th November

# 9. Known Issues & Risks

<Outline any known issues and/or risks that are likely to impact or be caused by this initiative in any way.>

The Risk Register form is used to record risks related to the project. The Issue Register form is used to record issues related to the project and are attached in the appendices.

# 10. Other Considerations - Ask Homy

<Discuss any other considerations for this project proposal’s acceptance and delivery.>

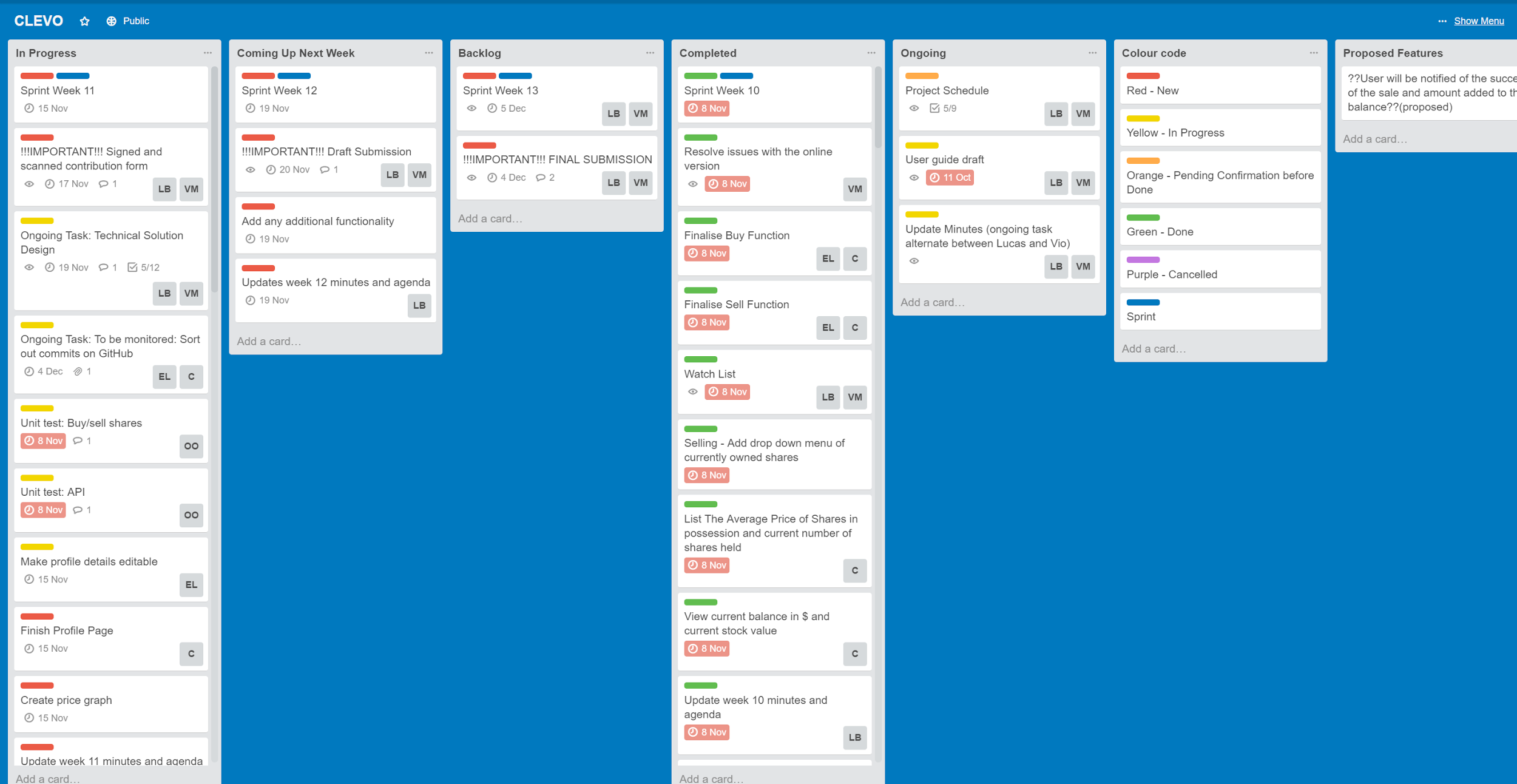
# 11. Appendix

<Refer the tool that is used to capture the functional requirement and if possible provide the references to the tool and also a summary of the functional requirement. It can be simply compilation or copy of the user stories from JIRA/Rally.>

<NOTE: These headings are guidelines only. Based on your project, you might require additional headings, so feel free to add headings as required.>

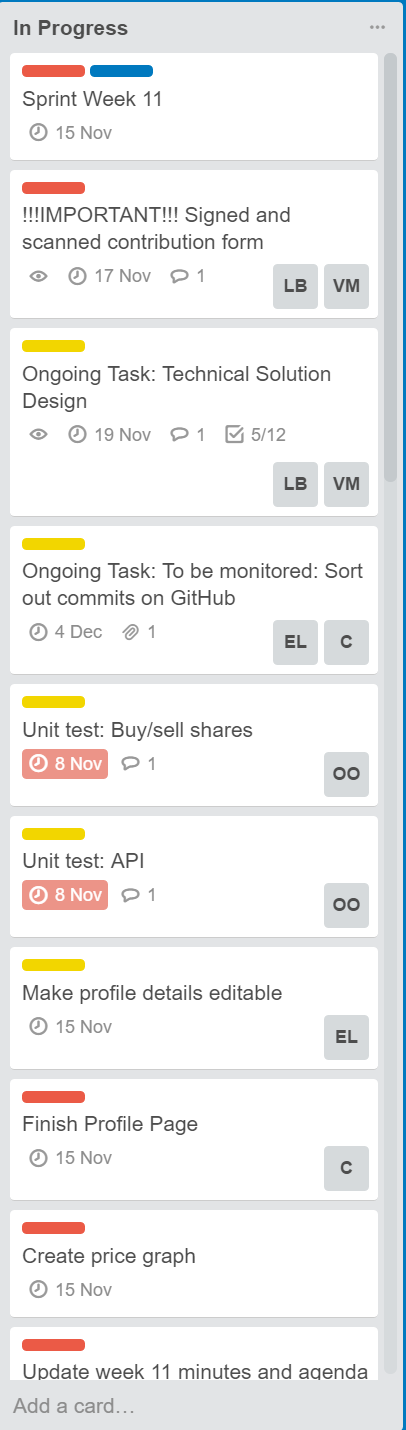
See below

Trello

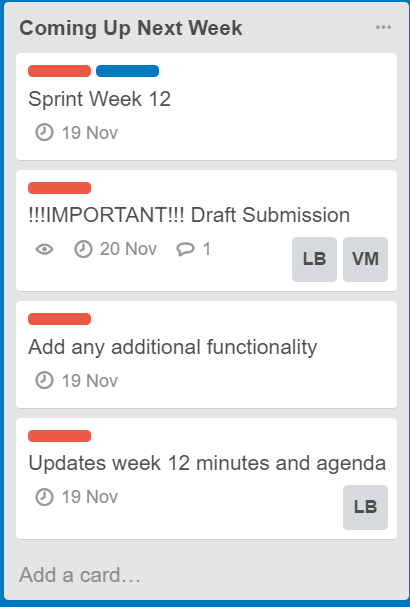


We used Trello as a tool for managing the project.

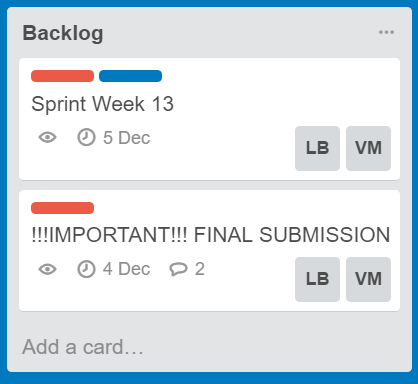
We have structured Trello in Sprints according to the Scrum methodology, and grouped the sprints into columns named in a way to make it easier to track the progress of each sprint and see upcoming sprints and the completed ones.



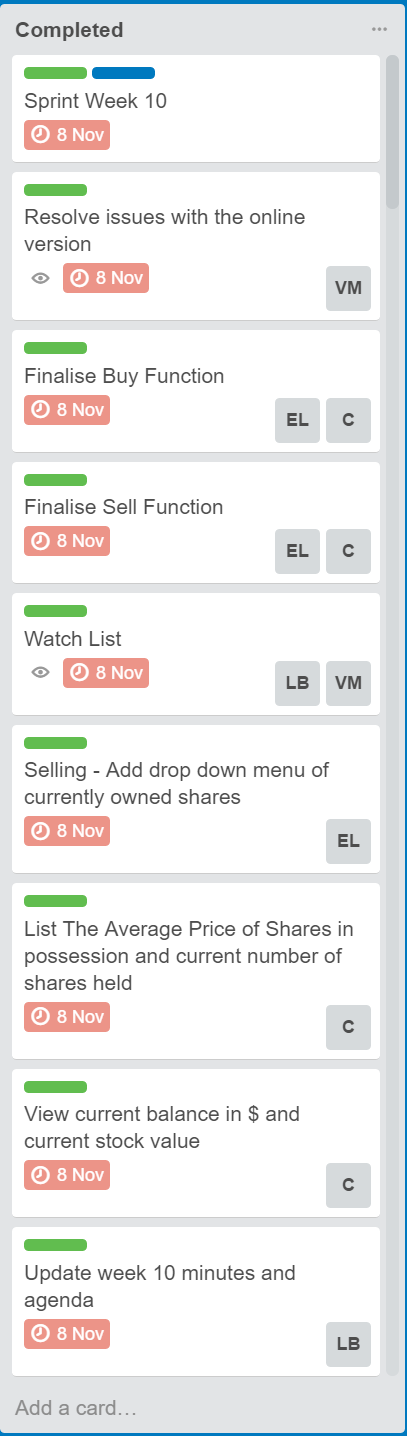
We have setup a “In Progress” column which contains the weekly sprint that we work on and all tasks the sprint contains. Each task is assigned to one or multiple group members, contains a due date and a colour status. The colour legend is described in its own column.



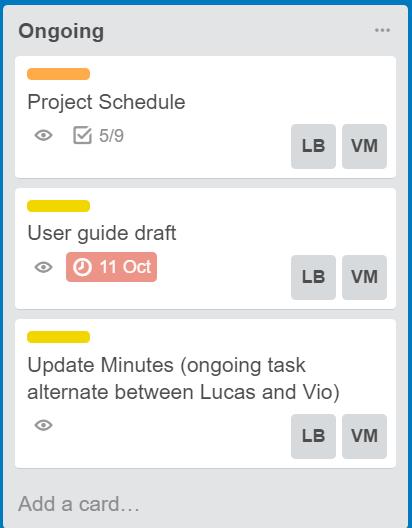
We have setup a “Coming Up Next Week” column which contains the following week’s sprint with its tasks. The tasks under this sprint as well contain a due date, is assigned to a group member or might still be



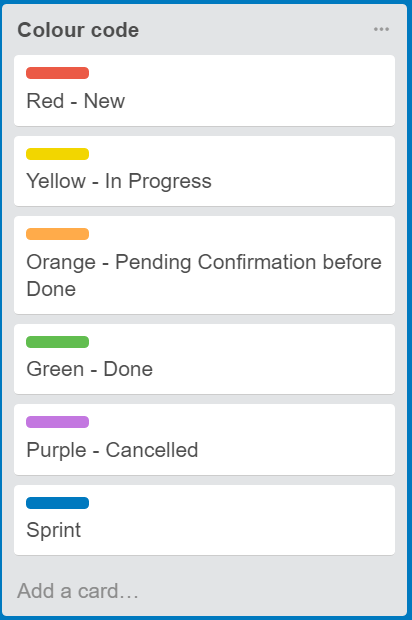
We have setup a “Backlog” column which contains all known future sprints and tasks till the end of the project, also assigned to a group member and a due date against each task



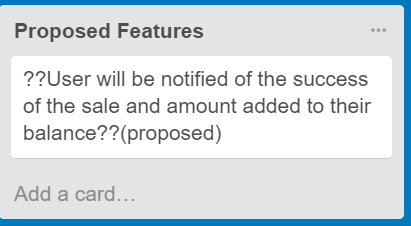
We have also setup a completed column where all completed sprints move to from the “In Progress” column structured in a Sprint due date order so we can refer back to task that was completed as part of a sprint in the past



We have setup an “Ongoing” column which contains the ongoing tasks such as Project schedule which require weekly even daily updates, or the minute taking task from which an instance is being created on weekly sprints.



We have setup a “Colour code” column which is the colour legend which describes the colour codes that each sprint and tasks are tagged with.



We have setup a “Proposed Features” column which contains additional features proposed by our group.