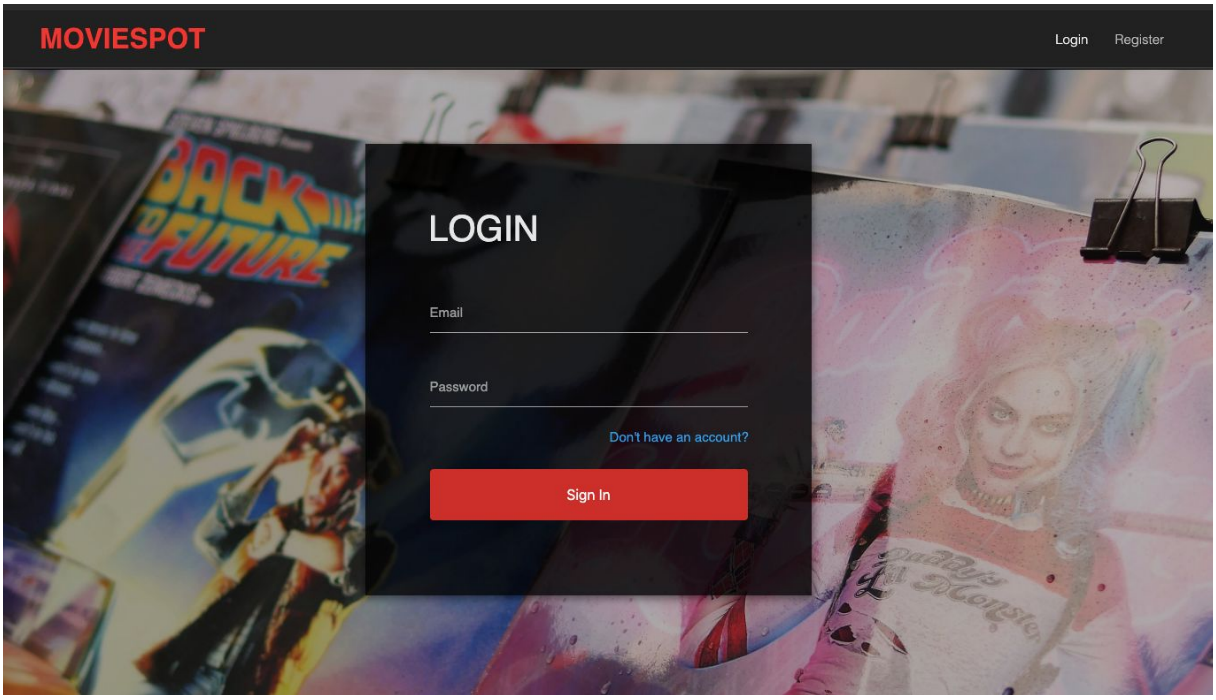
## Assignment 1: Analysis, Planning, Architecture & Design

# Online Movie System Project



## Date 19/04/2019

### Team 25 Group Roster

| Name | Student ID | Role |
|---|---|---|
| Calvin Thai | 12576260 | Scrum Master |
| Daniel Ebrahimian | 12961660 | Team Leader |
| Sami Khalil | 13044615 | Developer |
| Bowen Deng | 12486421 | Scrum Team member |

| | | |
|---|---|---|
| Balraj Brar | 12672085 | Scrum Team Member |

# Table of Contents

# 1. Overview

The objective of this project is to deliver a Online Movie Store (OMS) application for Movie Store (MS). The application will allow users (registered and anonymous) to login browse and purchase available movies. Once logged in, the application will allow the authorized MS staff users to manage the movie collection online and track customer orders, and allow customer users to browser, search, order movies online and track their orders. The authorised MS staff should be able to manage users (e.g. create, list, view, update, activate and deactivate records) and application access logs.

Overall, the OMS project will be using agile practices and following and MVC architecture designed and planned in the previous report. The current development environment includes React with redux as the front end and Django as the backend and controller.  Each of our 5 group members will be responsible for the following parts. The first release 1 comprising working software application must be delivered and released in production by no later than 31/05/2019.

| Module | Student ID | Student Name | Description |
|---|---|---|---|
| 1. Online User Registration | 12961660 | Daniel Ebrahiman | Allows user to register a new account. Allow existing users to login with their account. |
| 2. User Access Management | 12486421 | Bowen Deng | Login & Logout process, User Access Log Management (create and delete, search by date) |
| 3. Movie Catalogue Management | 12576260 | Calvin Thai | Browse movies and Movie management (Create, view, search,  update, delete). |
| 4. Order Management | 12672085 | Balraj Brar | Order Management<br>(Create, view, search,  update, delete) |
| 5. User Management | 13044615 | Sami Khalil | Allow admins to add a user, view all users, search by user name or phone number, edit a user, delete a user. The interface also shows a counter of how many users there are. |

# 1.2. Scope

## In-scope

Using the CRUD framework, the following features will be provided as in-scope items:

2.1. Online User Registration

- Validation
- Registration of a user
- View/Update details
- Cancel Registration

2.2. User Access Management

- Integrate user Access Logs
- View Access Logs
- Delete logs
- Search logs

2.3. Movie Catalogue Management

- Browse movies by genre
- View movie description/trailer
- Add Movie
- Update Movie
- Delete Movie

2.4 Order Management

- Order movie
- View order list
- Update order list
- Delete/Cancel orders

2.5 User Management

- Add User
- View existing users
- Update existing users
- Delete existing users

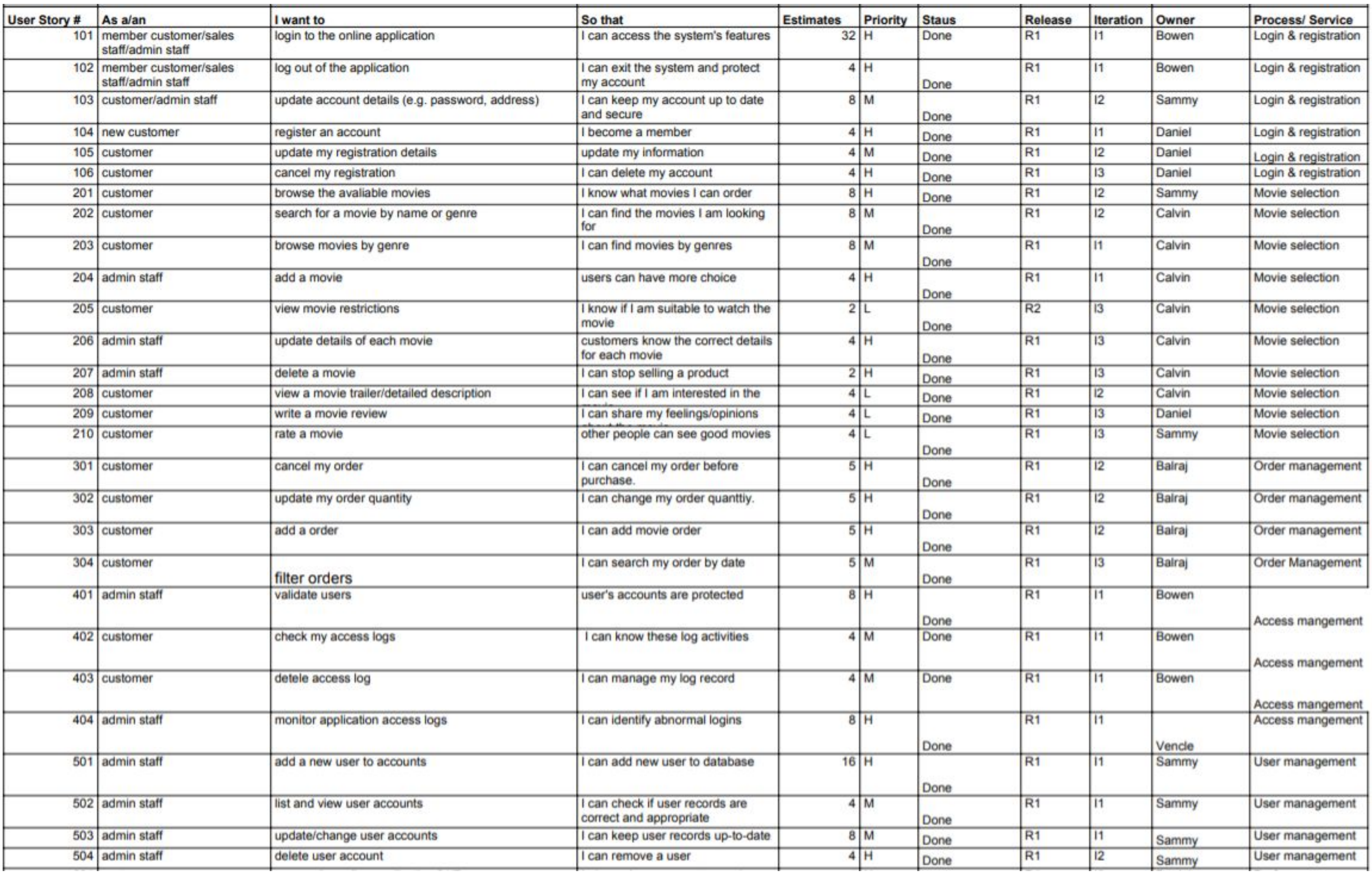
## Out-scope

- Payment Process
- Validation for movie inputs
-

# 1.3. Requirements

## 1.3.1 Functional Requirements

### User Stories



| User Story # | As a/an | I want to | So that | Estimates | Priority | Staus | Release | Iteration | Owner | Process/ Service |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | member customer/sales staff/admin staff | login to the online application | I can access the system's features | 32 | H | Done | R1 | I1 | Bowen | Login & registration |
| 102 | member customer/sales staff/admin staff | log out of the application | I can exit the system and protect my account | 4 | H | Done | R1 | I1 | Bowen | Login & registration |
| 103 | customer/admin staff | update account details (e.g. password, address) | I can keep my account up to date and secure | 8 | M | Done | R1 | I2 | Sammy | Login & registration |
| 104 | new customer | register an account | I become a member | 4 | H | Done | R1 | I1 | Daniel | Login & registration |
| 105 | customer | update my registration details | update my information | 4 | M | Done | R1 | I2 | Daniel | Login & registration |
| 106 | customer | cancel my registration | I can delete my account | 4 | H | Done | R1 | I3 | Daniel | Login & registration |
| 201 | customer | browse the avaliable movies | I know what movies I can order | 8 | H | Done | R1 | I2 | Sammy | Movie selection |
| 202 | customer | search for a movie by name or genre | I can find the movies I am looking for | 8 | M | Done | R1 | I2 | Calvin | Movie selection |
| 203 | customer | browse movies by genre | I can find movies by genres | 8 | M | Done | R1 | I1 | Calvin | Movie selection |
| 204 | admin staff | add a movie | users can have more choice | 4 | H | Done | R1 | I1 | Calvin | Movie selection |
| 205 | customer | view movie restrictions | I know if I am suitable to watch the movie | 2 | L | Done | R2 | I3 | Calvin | Movie selection |
| 206 | admin staff | update details of each movie | customers know the correct details for each movie | 4 | H | Done | R1 | I3 | Calvin | Movie selection |
| 207 | admin staff | delete a movie | I can stop selling a product | 2 | H | Done | R1 | I3 | Calvin | Movie selection |
| 208 | customer | view a movie trailer/detailed description | I can see if I am interested in the | 4 | L | Done | R1 | I2 | Calvin | Movie selection |
| 209 | customer | write a movie review | I can share my feelings/opinions | 4 | L | Done | R1 | I3 | Daniel | Movie selection |
| 210 | customer | rate a movie | other people can see good movies | 4 | L | Done | R1 | I3 | Sammy | Movie selection |
| 301 | customer | cancel my order | I can cancel my order before purchase. | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 302 | customer | update my order quantity | I can change my order quanttiy. | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 303 | customer | add a order | I can add movie order | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 304 | customer | filter orders | I can search my order by date | 5 | M | Done | R1 | I3 | Balraj | Order Management |
| 401 | admin staff | validate users | user's accounts are protected | 8 | H | Done | R1 | I1 | Bowen | Access mangement |
| 402 | customer | check my access logs | I can know these log activities | 4 | M | Done | R1 | I1 | Bowen | Access mangement |
| 403 | customer | detele access log | I can manage my log record | 4 | M | Done | R1 | I1 | Bowen | Access mangement |
| 404 | admin staff | monitor application access logs | I can identify abnormal logins | 8 | H | Done | R1 | I1 | Vencle | Access mangement |
| 501 | admin staff | add a new user to accounts | I can add new user to database | 16 | H | Done | R1 | I1 | Sammy | User management |
| 502 | admin staff | list and view user accounts | I can check if user records are correct and appropriate | 4 | M | Done | R1 | I1 | Sammy | User management |
| 503 | admin staff | update/change user accounts | I can keep user records up-to-date | 8 | M | Done | R1 | I1 | Sammy | User management |
| 504 | admin staff | delete user account | I can remove a user | 4 | H | Done | R1 | I2 | Sammy | User management |

## 1.3.2 User Narratives

### 1.3.2.1 Logging In (Daniel Ebrahimian)

| | |
|---|---|
| **Use Case ID** | UC101: Log into online application |
| **User Story** | As a member customer/sales staff/admin staff I want to login to the online application so that I can enter the system. |
| **Goal** | Successfully login to the system with email and password. |
| **Priority** | H |

| | |
|---|---|
| **Actors** | Primary Actor – Member customer/sales staff/admin staff<br>Secondary Actor – Login system/database |
| **Pre-conditions** | The user has an account. The application can communicate with the database. |
| **Post-conditions** | The user is now logged into the system. |
| **Trigger** | The user navigates to the login page of the application. |
| **Main Flow** | 1. The user navigates to the login page of the application<br>2. The user populates the email field<br>3. The user populates the password field<br>4. The user clicks the 'Login' button<br>5. The login system checks that the account exists. If an incorrect email is provided, then refer to **Alternate Flow 2 "Invalid Account"**<br>6. The login system checks that the correct password is supplied. If an incorrect email is provided, then refer to **Alternate Flow 2 "Invalid Account"**<br>7. The login system authenticates the user<br>8. The online application directs the user to the full application<br>9. The use case ends. |
| **Exceptions** | E1. Steps 5 & 6 - if the account doesn't exist or the credentials provided are incorrect, the system will reject the login attempt. The system will not direct the user to the full application. An error message will be provided instead. |
| **Includes** | |
| **Supporting Information** | Application needs to be able to communicate to the backend server and have access to the database. |
| **Non-functional Requirements** | Performance: Page load time<br>Security: Secure transfer of password over internet protocols such as HTTPS. |

| **Alternate Flow 2** | "Invalid Account" |
|---|---|
| **Trigger** | Incorrect email or password |
| **Step** | 1. The login system presents an error message to the user, "Invalid Login Credentials"<br>2. Re-join at Step 9 in the Main Flow. |
| **Alt Flow** | N/A |
| **Post conditions** | The login system successfully displayed an error message. |
| **Exceptions** | |

### 1.1.3.2 Searching a movie (Sami Khalil)

| **Use Case ID** | UC202: Search a movie |
|---|---|

| | |
|---|---|
| **User Story** | As customer I want to search for a movie so that find the movie that I am looking for. |
| **Goal** | Successfully find the movie. |
| **Priority** | M |
| **Actors** | Primary Actor – Customer<br>Secondary Actor – Database |
| **Pre-conditions** | The user has an account and is logged in. The application can communicate with the database. |
| **Post-conditions** | The application displayed the movie to the user. |
| **Trigger** | The user navigates to the home page of the application. |
| **Main Flow** | 1. The user clicks on the search icon at the top right corner<br>2. The user types in their movie query to search in the database<br>3. The applications returns movies or a movie that matches the query<br>4. The user can click on the movie to view trailer/details and buy it<br>5. The use case ends. |
| **Exceptions** | E1. Steps 3 - if there is not matching movie then the application will display “No movies found, search for something else.” |
| **Includes** | |
| **Supporting Information** | Application needs to be able to communicate to the backend server and have access to the database. |
| **Non-functional Requirements** | Performance: Page load time<br>Query response time |

### 1.1.3.3 Make an order for the selected movie (Bowen Deng)

| | |
|---|---|
| **Use Case ID** | UC301: Make an order for the movies I selected |
| **User Story** | As a customer, I want to make an order for the movies I selected so that I can purchase the movies |
| **Goal** | Successfully place the order to the system with correct information |
| **Priority** | High |
| **Actors** | Primary Actor – Customer<br>Secondary Actor – The online application |
| **Pre-conditions** | The customer has browsed available movies and know what movies to buy<br>The customer knows the contact details and shipping address. |

| | |
|---|---|
| **Post-conditions** | The customer has successfully placed an order via the Online System. |
| **Trigger** | The customer navigates to the movie detail page of the application |
| **Main Flow** | 1. Customer click the 'Add to Cart' button below the movie description<br>2. After repeating flow 1, customer click the 'shopping cart' icon on the top right of the interface<br>3. The system navigates to shopping cart page<br>4. Customer select the items that he wants to place in an order<br>5. Customer click 'Place an order' button at the bottom<br>6. System navigates customer to the order placement page.<br>7. The customer fills in the personal details (consignee name, contact detail, shipping address, etc).<br>8. Customer click the 'confirm' button.<br>9. The system navigates customer to the order detail confirmation page<br>10. Customer click 'Pay Now' button at the bottom. If the customer wants to go back to previous page, then refer to **Alternate Flow 2 "Go Back to previous page"** If the customer doesn't want to pay for the order now, then refer to **Alternate Flow 3 "Pay later & alternative payment"**<br>11. The system navigates customer to payment page.<br>12. The use case ends. |
| **Exceptions** | E1. Steps 1-12 – Customer closes the browser window at anytime, and then the system cancels the order placement |
| **Includes** | UC201: Browse available movies |
| **Supporting Information** | Customer, Purchased items, consignee details, order details |
| **Non-functional Requirements** | Performance: Page load time<br>Security: Secure order placement information |

| | |
|---|---|
| **Alternate Flow 2** | "Go Back to previous page" |
| **Trigger** | Customer clicks the 'Go Back' button |

| | |
|---|---|
| **Step** | 1. Re-join at Step 6 in the Main Flow |
| **Alt Flow** | N/A |
| **Post conditions** | The system successfully navigates customer to order placement page |
| **Exceptions** | E1. Steps 1-12 – Customer closes the browser window at anytime, and then the system cancels the order placement |
| **Alternate Flow 3** | “Pay later & alternative payment” |
| **Trigger** | Customer click the ‘Pay Later’ button |
| **Step** | 1. The system navigate customer back to current movie description page. |
| **Alt Flow** | N/A |
| **Post conditions** | The system successfully go back to current movie description page |
| **Exceptions** | E1. Steps 1-12 – Customer closes the browser window at anytime, and then the system cancels the order placement |

### 1.1.3.4 Register Account User Narrative (Balraj Brar)

**Register Account User Narrative**

| | |
|---|---|
| **Use Case ID** | UC 104: Register an Account |
| **User Story** | As a new customer I want to register to the online application so that I can enter the system. |
| **Goal** | Successful registration to the system with details such as email and password. |
| **Priority** | High |
| **Actors** | Primary Actor – New Member/Customer |
| **Pre-conditions** | The user does not have an existing account. |
| **Post-conditions** | A new account created in the system. |
| **Trigger** | The user navigates to the login page of the application. |

| | |
|---|---|
| **Main Flow** | 1. The user navigates to the Register page of the application<br>2. The user enters the required details such as name, address, mobile, credit card details, login ID and Password.<br>3. The user clicks the ‘Register’ button.<br>4. The register system checks that the account already exists with the email provided. If an existing email is provided, then refer to **Alternate Flow 2 “Existing Account”.**<br>5. The Registering system validates the new account details. If an incorrect detail is provided, then highlight the incorrect details with a red ‘*’.<br>6. The Register system checks that the correct details are provided<br>7. The Register system authenticates the new customer.<br>8. The online application directs the user to the login page of the application<br>9. The use case ends. |
| **Exceptions** | Steps 3 & 4 - if the account already exist, the system will reject the registration attempt. The system will not direct the user to the application. An error message will be provided instead. |
| **Includes** | |
| **Supporting Information** | Application needs to be able to communicate to the backend server and have access to the database. |
| **Non-functional Requirements** | Performance: Page load time<br>Security: Secure credit card details and customer’s personal information |

| | |
|---|---|
| **Alternate Flow 2** | “There is an existing account with the mail ” |
| **Trigger** | User already has an Account. |
| **Step** | 1. The login system presents an error message to the user, “There is an existing account with the email”<br>2. User uses existing account or creates a new account with another email address. |
| **Alt Flow** | |
| **Post conditions** | The registration window successfully displayed an error message. |
| **Exceptions** | |

### 1.1.3.5 View a movie trailer/ detailed description (Calvin Thai)

| | |
|---|---|
| **Use Case ID** | UC303: View a movie trailer/detailed description |
| **User Story** | As a customer I want to view a movie trailer/detailed description so that I can see if I am interested in the movie. |
| **Goal** | Customer can decide if they want to watch the movie. |
| **Priority** | Medium |
| **Actors** | Primary Actor – Customer<br>Secondary Actor – Youtube, Movie database |

| | |
|---|---|
| **Pre-conditions** | The user has successfully logged into the system.<br>The user can access Youtube.<br>The application can communicate with the database.<br>Internet connection |
| **Post-conditions** | The user can now view the movie trailer and/or detailed description. |
| **Trigger** | The user selects a movie and clicks on the movie trailer link. |
| **Main Flow** | 1. The user clicks on a movie from the catalogue.<br>2. The movie description pops up and displays the movie's detailed information and tabs to other information.<br>3. The user clicks on the trailer tab.<br>4. The tab displays an embedded Youtube trailer for the movie.<br>5. The user clicks the embedded Youtube trailer.<br>6. The youtube video plays the trailer.<br>7. The use case ends. |
| **Exceptions** | If the trailer link is unavailable then the contents of the Youtube video will be replaced with a "not available" screen. |
| **Includes** | |
| **Supporting Information** | Application needs to be able to communicate to the backend server and grab information from the database. The trailer link must direct the user to the appropriate movie trailer. |
| **Non-functional Requirements** | Visibility: Link needs to be distinguishable.<br>Performance: Response time from movie/link click.<br>Performance: Quick response from database query |

# 1.4 Data Requirements

## 1.4.1 Data Dictionary

| Ref. | Data Element | Format | Length | Description |
|---|---|---|---|---|
| 01 User | | | | |
| DE-1001 | ID | id | 10 | Primary key |
| DE-1002 | First Name | Varchar | 10 | Required, must be alpha,<br>e.g. John, |
| DE-1003 | Last Name | varchar | 30 | Required, last name |
| DE-1004 | Email | Varchar | 10 | Email, required, must be valid |
| DE-1005 | Password | Varchar | 20 | Password, required, must be 6 chars or more |
| DE-1006 | Mobile Number | varchar | 10 | Mobile phone contact number |
| DE-1007 | Shipping Address | varchar | 100 | Address for delivering goods |
| DE-1008 | Date of Birth | date | 10 | Know if we can legally sell them R rated films. |
| DE-1009 | Is Admin | Bool | 2 | Is the user an administrator or not |

| | | | | |
|---|---|---|---|---|
| 02 Movie | | | | |
| DE-2001 | ID | id | 10 | Primary key |
| DE-2002 | Title | Varchar | 20 | Required |
| DE-2003 | Year | Int | 4 | Required |

| | | | | |
|---|---|---|---|---|
| DE-2005 | Description | Varchar | 255 | Required |
| DE-2006 | Trailer | TextField | 200 | Required, link to youtube movie trailer |
| DE-2009 | Price | float | 100 | Required, e.g. $28.99 |
| DE-2010 | Maturity Rating | Varchar | 5 | Required, e.g. G, R, M, PG |
| DE-2011 | Purchase count | Int | 1000 | Required, increases when a customer purchases (to keep track of most popular movie) |
| DE-2012 | Stock | Int | 1000 | Required, number of movies available |

| 03 Order | | | | |
|---|---|---|---|---|
| DE-3001 | ID | id | 10 | Primary key |
| DE-3002 | Movie | id | 10 | Foreign key |
| DE-3003 | User | id | 10 | Foreign key |
| DE-3004 | Quantity | int | 100 | Default 1 |
| DE-3005 | Date | varchar | 10 | Current date |
| DE-3006 | Order type | varchar | 10 | Required e.g. Delivery or store pick-up |
| DE-3007 | Discount Modifier | float | 5 | Used for discounts. Eg 30% off. |
| DE-3008 | Total Cost | float | 5 | Total cost of the order |

| 04 Review | | | | |
|---|---|---|---|---|
| DE-4001 | ID | id | 10 | Primary key |
| DE-4002 | name | varchar | 20 | Author name |
| DE-4003 | rating | float | 10 | Required, Rating between 0.0 and 5.0 |
| DE-4004 | text | varchar | 255 | Review text |
| DE-4005 | Date | varchar | 10 | Current date |

| 05 Genre | | | | |
|---|---|---|---|---|
| DE-5001 | ID | id | 10 | Primary key |
| DE-5002 | name | varchar | 20 | Genre name |

| 06 Log | | | | |
|---|---|---|---|---|
| DE-6001 | ID | id | 10 | Primary key |
| DE-6002 | User | id | 10 | Foreign key |
| DE-6003 | Status | TextField | 50 | Required, Rating between 0.0 and 5.0 |
| DE-6004 | Time Stamp | DateTimeField | 10 | Time that the log was created |

| 07 MovieReview | | | | |
|---|---|---|---|---|
| DE-7001 | ID | id | 10 | Primary key |
| DE-7002 | Movie | id | 10 | Foreign key |
| DE-7003 | Review | id | 10 | Foreign key |

| 08 MovieOrder | | | | |
|---|---|---|---|---|
| DE-8001 | ID | id | 10 | Primary key |
| DE-8002 | Movie | id | 10 | Foreign key |
| DE-8003 | Order | id | 10 | Foreign key |

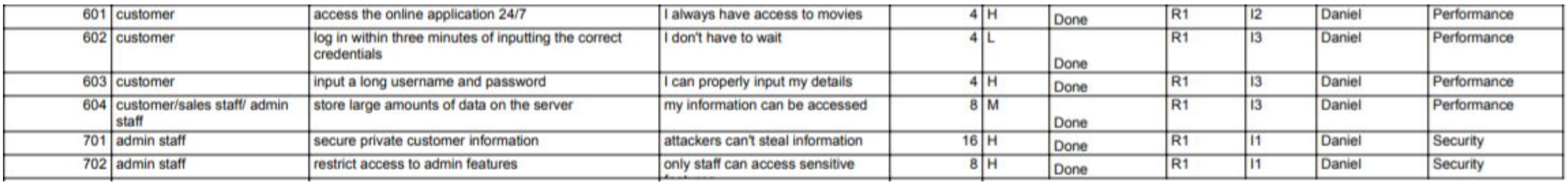
| 09 MovieGenre | | | | |
|---|---|---|---|---|
| DE-9001 | ID | id | 10 | Primary key |
| DE-9002 | Movie | id | 10 | Foreign key |

| | | | | |
|---|---|---|---|---|
| DE-9003 | Genre | id | 10 | Foreign key |

# 1.5 Non-functional Requirements

The non-functional requirements seen in Figure 1.3.1 outlines the criteria to judge the operability of the system. The criteria focuses on realistic expectations that are derived from current performance and security standards on the web.

## 1.5.1 Non-Functional Requirements Table



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 601 | customer | access the online application 24/7 | I always have access to movies | 4 | H | Done | R1 | I2 | Daniel | Performance |
| 602 | customer | log in within three minutes of inputting the correct credentials | I don't have to wait | 4 | L | Done | R1 | I3 | Daniel | Performance |
| 603 | customer | input a long username and password | I can properly input my details | 4 | H | Done | R1 | I3 | Daniel | Performance |
| 604 | customer/sales staff/ admin staff | store large amounts of data on the server | my information can be accessed | 8 | M | Done | R1 | I3 | Daniel | Performance |
| 701 | admin staff | secure private customer information | attackers can't steal information | 16 | H | Done | R1 | I1 | Daniel | Security |
| 702 | admin staff | restrict access to admin features | only staff can access sensitive | 8 | H | Done | R1 | I1 | Daniel | Security |

# 1.5 Constraints & risks

## 1.5.1 Constraints

The three main constraints that could limit a project success are:

**Time and Cost** it will take for completion of project? It is very difficult in a project to balance time and cost, as due to shift in any of the aspect it will imbalance other constrain.
For example delays in project delivery result in additional cost such as staff salaries or vice versa.

**Quality** whether the project build is in accordance to the client.
Some of the quality constraints are mentioned below.

1. If the website crashes all the current user (Customer) data will be lost.
2. Movie purchase per person is limited to 10 per movie.
3. The system stall run on Google Chrome, Firefox, Safari and Internet Explorer.
4. Once the movie is bought customer cannot cancel the order.

**Scope** What is the expected outcome of the project? Scope are the deliverables which have been agreed by project's owner, it is important to understand the client needs while making system and give regular updates of the project milestones to ensure it is made according to the customer needs.

## 1.5.2 Risks

### 1.5.2.1 Technical Risks

When developing software, it is important to understand that there might be some technical boundaries. These could be issues such as insufficient knowledge with tools and technologies available that could be used to create the software solution eg frameworks and programming languages. Furthermore, technical issues such as the following can occur:

- Website becoming inaccessible
- Website becoming inaccessible during payment processing. What will happen to the movie and customer's money?
- Movie out of stock
- Outdated movie list
- Stock levels not correctly represented between warehouse and online stock levels
- Customer cannot find the movie they are looking for
- Project is not on schedule and tasks are delayed, which could lead in delayed release
- Database is not secure which could lead to attackers inserting malicious SQL queries into the database to gain unauthorised access to sensitive customers data or deleting records.
- Poor design/UI, customer not able to navigate where they want to.

### 1.5.2.2 Requirement Risks

Incorrect understanding of the project requirements can have many unwanted side effects. Such as wasting time, creating an incorrect product or incorrectly implementing a desired core feature. It is important to have good team cohesion and teamwork and to carefully assign work to team members after identifying the most suitable candidate for the task.

### 1.5.2.3 Quality Risks

The lack of quality standards in the implementation of any software project or neglect of software quality supervision will pose risks to the development of the software. This can be quality issues such as:

- The software does not complete all the user requirements
- The software is not working in a stable manner

### 1.5.2.4 Team Risk

Creating a large project requires multiple members, therefore we utilise a team based approach. There are risks associated with working in a team of people. We must be able to handle a team member leaving for reasons such as illness or withdrawal of the subject. Potential issues such as poor communication amongst team members or certain team members not being able to get along with each other. These collaboration issues can hinder the production of this product and should be identified and resolved as early as possible if they emerge.

# 2. Plan

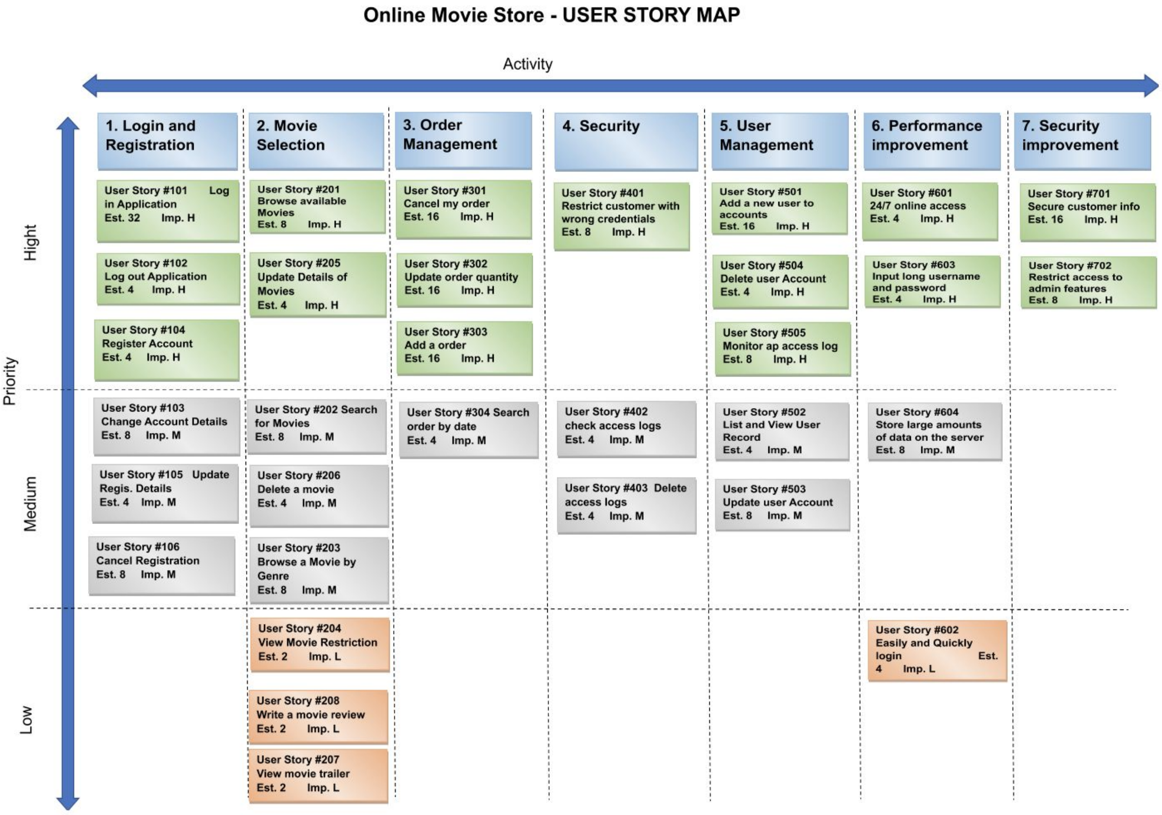
## 2.1 Project Schedule

### 2.1.1 Iteration Schedule

| Release | Iteration | Starting date<br>Ending data | Goals |
|---|---|---|---|
| Release 0 | Iteration 0 | 12/03/2019<br>17/03/2019 | ● Project analysis<br>● Project planning<br>● Solution design |
| | Iteration 1 | 18/03/2019<br>26/03/2019 | ● Build database<br>- Create data template for user record |
| | Iteration 2 | 27/03/2019<br>08/04/2019 | ● Achieve basic functionality (e.g. browse movies)<br>- Browse and search for movies<br>- View movie trailer<br>● Achieve basic user data management<br>- Activate user account<br>- Deactivate user account |
| | Iteration 3 | 08/04/2019<br>19/04/2019 | ● Achieve core functionality (e.g. purchase movies, track orders)<br>- View movie restrictions<br>- Update available stock of each movie<br>- Make an order<br>- Pay for the order<br>- Track orders |
| Release 1 | Iteration 0 | 20/04/2019<br>26/04/2019 | ● Deployment planning<br>● Testing planning |

| | | | |
|---|---|---|---|
| | Iteration 1 | 27/04/2019<br>12/04/2019 | ● Achieve complementary and advanced functionality (e.g. browse movies by genre)<br>- Browse movies by genre<br>● Achieve basic security<br>- Monitor application access logs<br>- Secure private customer information<br>- Restrict access to admin features<br>- Restrict customers when the wrong credentials are |
| | Iteration 2 | 13/04/2019<br>20/04/2019 | ● Achieve after-sell service functionality (e.g. cancel order) – in progress<br>- Send a order cancellation request<br>- Approve order cancellation<br>● Performance improvement – in progress<br>- Access the online application 24/7 |
| | Iteration 3 | 21/04/2019<br>31/05/2019 | ● Achieve after-sell service functionality (e.g. cancel order) – completed<br>- Rate a movie<br>- Write a movie review<br>- Reorder a movie<br>● Performance improvement – completed<br>- Login to the application with three minutes<br>- Store large amounts of data on the server |

## 2.1.2 User Story Map



Online Movie Store - USER STORY MAP

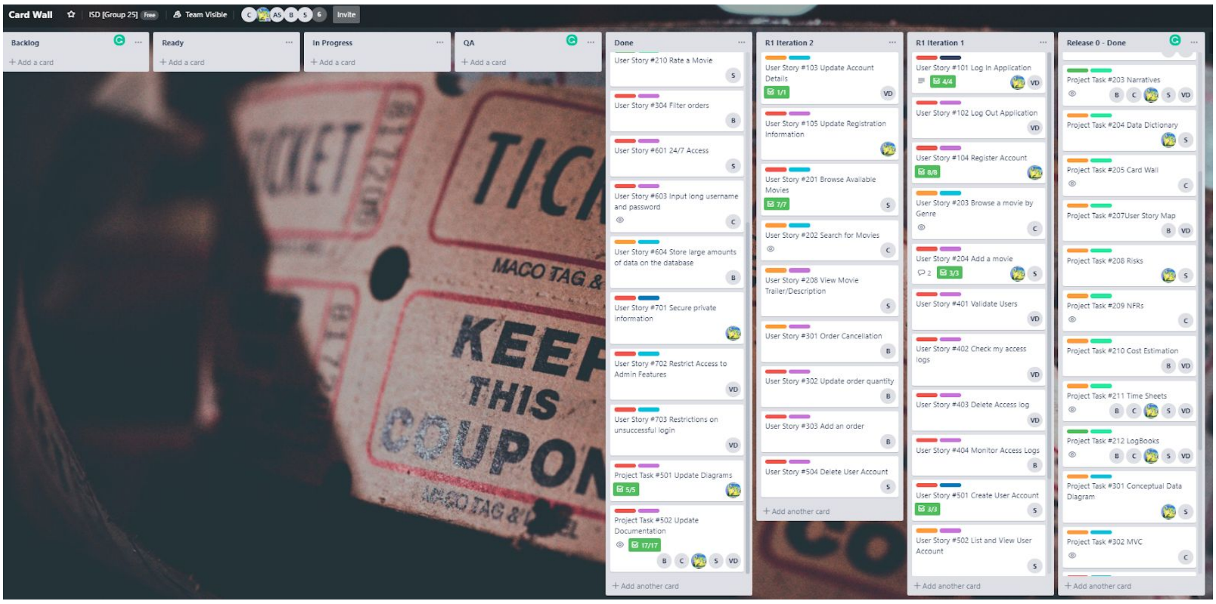
Activity

| Priority | 1. Login and Registration | 2. Movie Selection | 3. Order Management | 4. Security | 5. User Management | 6. Performance improvement | 7. Security improvement |
|---|---|---|---|---|---|---|---|
| Hight | User Story #101 Log in Application Est. 32 Imp. H<br>User Story #102 Log out Application Est. 4 Imp. H<br>User Story #104 Register Account Est. 4 Imp. H | User Story #201 Browse available Movies Est. 8 Imp. H<br>User Story #205 Update Details of Movies Est. 4 Imp. H | User Story #301 Cancel my order Est. 16 Imp. H<br>User Story #302 Update order quantity Est. 16 Imp. H<br>User Story #303 Add a order Est. 16 Imp. H | User Story #401 Restrict customer with wrong credentials Est. 8 Imp. H | User Story #501 Add a new user to accounts Est. 16 Imp. H<br>User Story #504 Delete user Account Est. 4 Imp. H<br>User Story #505 Monitor ap access log Est. 8 Imp. H | User Story #601 24/7 online access Est. 4 Imp. H<br>User Story #603 Input long username and password Est. 4 Imp. H | User Story #701 Secure customer info Est. 16 Imp. H<br>User Story #702 Restrict access to admin features Est. 8 Imp. H |
| Medium | User Story #103 Change Account Details Est. 8 Imp. M<br>User Story #105 Update Regis. Details Est. 4 Imp. M<br>User Story #106 Cancel Registration Est. 8 Imp. M | User Story #202 Search for Movies Est. 8 Imp. M<br>User Story #206 Delete a movie Est. 4 Imp. M<br>User Story #203 Browse a Movie by Genre Est. 8 Imp. M | User Story #304 Search order by date Est. 4 Imp. M | User Story #402 check access logs Est. 4 Imp. M<br>User Story #403 Delete access logs Est. 4 Imp. M | User Story #502 List and View User Record Est. 4 Imp. M<br>User Story #503 Update user Account Est. 8 Imp. M | User Story #604 Store large amounts of data on the server Est. 8 Imp. M | |
| Low | | User Story #204 View Movie Restriction Est. 2 Imp. L<br>User Story #208 Write a movie review Est. 2 Imp. L<br>User Story #207 View movie trailer Est. 2 Imp. L | | | | User Story #602 Easily and Quickly login Est. 4 Imp. L | |

## 2.1.3 Card Wall

Figure 2.1.3.1 is a screenshot taken during iteration one. Members of the team track progress by moving their task cards towards the done stage. A full image of the card wall is provided in Appendix 7.2.

### 2.1.3.1 R1 Iteration 3 Card Wall Screenshot



Backlog
+ Add a card

Ready
+ Add a card

In Progress
+ Add a card

QA
+ Add a card

Done
- User Story #210 Rate a Movie S
- User Story #304 Filter orders B
- User Story #601 24/7 Access S
- User Story #603 Input long username and password C
- User Story #604 Store large amounts of data on the database B
- User Story #701 Secure private information
- User Story #702 Restrict Access to Admin Features VD
- User Story #703 Restrictions on unsuccessful login VD
- Project Task #501 Update Diagrams 5/5
- Project Task #502 Update Documentation 17/17 B C S VD

+ Add another card

R1 Iteration 2
- User Story #103 Update Account Details 1/1 VD
- User Story #105 Update Registration Information
- User Story #201 Browse Available Movies 7/7 S
- User Story #202 Search for Movies C
- User Story #208 View Movie Trailer/Description S
- User Story #301 Order Cancellation B
- User Story #302 Update order quantity B
- User Story #303 Add an order B
- User Story #504 Delete User Account S

+ Add another card

R1 Iteration 1
- User Story #101 Log In Application 4/4 VD
- User Story #102 Log Out Application VD
- User Story #104 Register Account 8/8
- User Story #203 Browse a movie by Genre C
- User Story #204 Add a movie 2 3/3 S
- User Story #401 Validate Users VD
- User Story #402 Check my access logs VD
- User Story #403 Delete Access log VD
- User Story #404 Monitor Access Logs B
- User Story #501 Create User Account 3/3 S
- User Story #502 List and View User Account S

+ Add another card

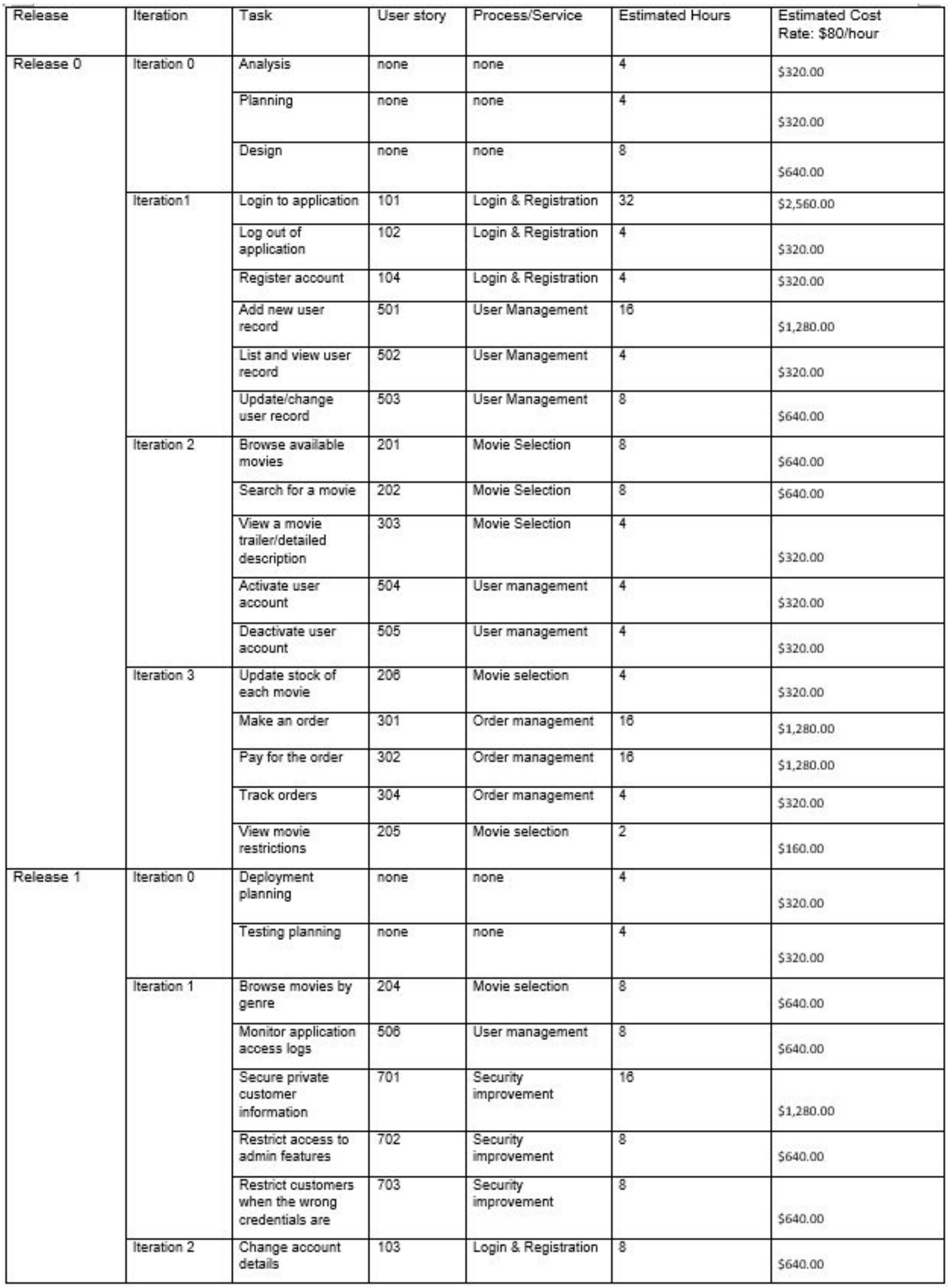
Release 0 - Done
- Project Task #203 Narratives B C S VD
- Project Task #204 Data Dictionary S
- Project Task #205 Card Wall C
- Project Task #207User Story Map B VD
- Project Task #208 Risks S
- Project Task #209 NFRs C
- Project Task #210 Cost Estimation B VD
- Project Task #211 Time Sheets B C S VD
- Project Task #212 LogBooks B C S VD
- Project Task #301 Conceptual Data Diagram S
- Project Task #302 MVC C

+ Add another card

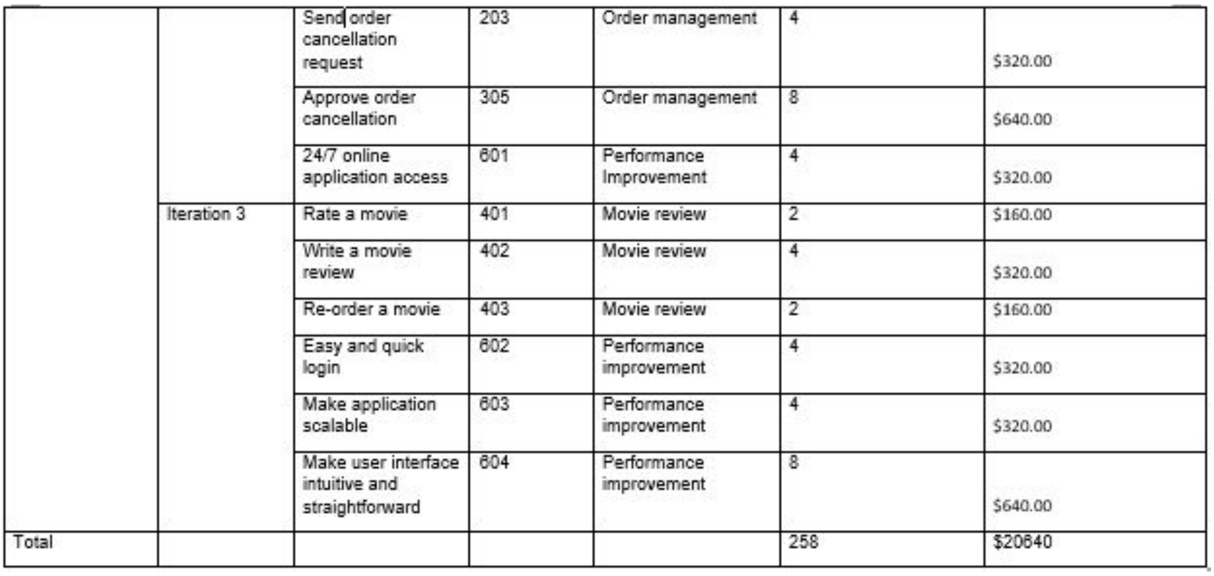
## 2.2 Project Cost Estimates

According to the project requirements and specifications, we assume each of our team member work 8 hours per week. Since we have 5 people in our team, we estimate 40 working hours per week. The project deadline is on 31st May 2019, as we start the project on 12 March 2019, project duration is estimated to be 12 Week at maximum, in other word, 480 hours. Thus, the initial budget for the project is approximately up to 40,000$ with a suggested rate of 80$/hour.

Besides, to bear any uncertainties which might occur during the project life cycle such as system crash, employee on leave, customer wish to change certain attribute in the system, we are planning a backup budget. By referencing historical similar project, it is estimated to be 10000$. Thus, the total estimated budget of the given system 50,000$(approx.) which would include the initial budget of 40,000$ and backup of 10,000$.



| Release | Iteration | Task | User story | Process/Service | Estimated Hours | Estimated Cost<br>Rate: $80/hour |
|---|---|---|---|---|---|---|
| Release 0 | Iteration 0 | Analysis | none | none | 4 | $320.00 |
| | | Planning | none | none | 4 | $320.00 |
| | | Design | none | none | 8 | $640.00 |
| | Iteration1 | Login to application | 101 | Login & Registration | 32 | $2,560.00 |
| | | Log out of application | 102 | Login & Registration | 4 | $320.00 |
| | | Register account | 104 | Login & Registration | 4 | $320.00 |
| | | Add new user record | 501 | User Management | 16 | $1,280.00 |
| | | List and view user record | 502 | User Management | 4 | $320.00 |
| | | Update/change user record | 503 | User Management | 8 | $640.00 |
| | Iteration 2 | Browse available movies | 201 | Movie Selection | 8 | $640.00 |
| | | Search for a movie | 202 | Movie Selection | 8 | $640.00 |
| | | View a movie trailer/detailed description | 303 | Movie Selection | 4 | $320.00 |
| | | Activate user account | 504 | User management | 4 | $320.00 |
| | | Deactivate user account | 505 | User management | 4 | $320.00 |
| | Iteration 3 | Update stock of each movie | 206 | Movie selection | 4 | $320.00 |
| | | Make an order | 301 | Order management | 16 | $1,280.00 |
| | | Pay for the order | 302 | Order management | 16 | $1,280.00 |
| | | Track orders | 304 | Order management | 4 | $320.00 |
| | | View movie restrictions | 205 | Movie selection | 2 | $160.00 |
| Release 1 | Iteration 0 | Deployment planning | none | none | 4 | $320.00 |
| | | Testing planning | none | none | 4 | $320.00 |
| | Iteration 1 | Browse movies by genre | 204 | Movie selection | 8 | $640.00 |
| | | Monitor application access logs | 506 | User management | 8 | $640.00 |
| | | Secure private customer information | 701 | Security improvement | 16 | $1,280.00 |
| | | Restrict access to admin features | 702 | Security improvement | 8 | $640.00 |
| | | Restrict customers when the wrong credentials are | 703 | Security improvement | 8 | $640.00 |
| | Iteration 2 | Change account details | 103 | Login & Registration | 8 | $640.00 |



| | | | | | | |
|---|---|---|---|---|---|---|
| | | Send order cancellation request | 203 | Order management | 4 | $320.00 |
| | | Approve order cancellation | 305 | Order management | 8 | $640.00 |
| | | 24/7 online application access | 601 | Performance Improvement | 4 | $320.00 |
| | Iteration 3 | Rate a movie | 401 | Movie review | 2 | $160.00 |
| | | Write a movie review | 402 | Movie review | 4 | $320.00 |
| | | Re-order a movie | 403 | Movie review | 2 | $160.00 |
| | | Easy and quick login | 602 | Performance improvement | 4 | $320.00 |
| | | Make application scalable | 603 | Performance improvement | 4 | $320.00 |
| | | Make user interface intuitive and straightforward | 604 | Performance improvement | 8 | $640.00 |
| Total | | | | | 258 | $20640 |

## 2.3.0 Project Timesheet

### 2.3.1 Daniel Ebrahimian

| Day | Week 1 (Date) | Week 2 (Date) | Week 3 | Week 4 | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1.00 |
| Tuesday | 2 | 2 | 2 | 9 | 2 | 8 | 3 | 4 | 5 | 2 | 4 | 43.00 |
| Wednesday | | | 2 | | 2 | 4 | | | | | 1 | 9.00 |
| Thursday | 1 | 2 | | | | 5 | 4 | 4 | 2 | 3 | 1 | 22.00 |
| Friday | | | | | | 1 | | | | 1 | 5 | 7.00 |
| Saturday | 2 | 3 | | 1 | 3 | | 3 | 2 | 5 | 1 | 0 | 20.00 |
| Sunday | 1 | 2 | | | | | | 1 | 3 | 1 | 0 | 8.00 |
| Total | 6.00 | 9.00 | 4.00 | 10.00 | 7.00 | 18.00 | 10.00 | 11.00 | 15.00 | 8.00 | 12.00 | **110.00** |
| Rate | $80 | $ 80.00 | $80 | $ 80.00 | $80 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | |
| Total | $ 480.00 | $ 720.00 | $ 320.00 | $ 800.00 | $ 560.00 | $ 1,440.00 | $ 800.00 | $ 880.00 | $ 1,200.00 | $ 640.00 | $ 960.00 | **$ 8,800.00** |

## 2.3.2 Sami Khalil

| Day | Week 1 (Date) | Week 2 (Date) | Week 3 | Week 4 | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | 2 | 1 |  | 3 | 2 | 2 |  |  | 2 | 1 | 5 | 15.00 |
| Tuesday | 2 | 2 | 2 | 4 | 4 |  | 5 | 6 |  | 2 |  | 21.00 |
| Wednesday |  |  | 2 |  | 3 | 3 |  |  | 3 |  | 4 | 13.00 |
| Thursday | 1 | 2 |  |  | 2 | 3 | 5 |  | 3 |  |  | 13.00 |
| Friday |  | 1 | 4 |  |  |  |  | 4 |  | 3 | 5 | 12.00 |
| Saturday |  | 1 | 6 | 4 |  |  |  |  | 5 | 5 | 1 | 15.00 |
| Sunday | 1 |  | 6 |  |  |  |  | 2 |  |  |  | 2.00 |
| Total | 6.00 | 7.00 | 20.00 | 11.00 | 11.00 | 8.00 | 10.00 | 12.00 | 13.00 | 11.00 | 15.00 | **91.00** |
| Rate | $80 | $ 80.00 | $80 | $ 80.00 | $80 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 |
| Total | $ 480.00 | $ 560.00 | $ 1,600.00 | $ 880.00 | $ 880.00 | $ 640.00 | $ 800.00 | $ 960.00 | $ 1,040.00 | $ 880.00 | $ 1,200.00 | **$ 7,280.00** |

## 2.3.3 Calvin Thai

| Day | Week 1 (18/03) | Week 2 (25/03) | Week 3 (01/04) | Week 4 (08/04) | Week 5 (15/04) | Week 6 (22/04) | Week 7(29/04) | Week 8 (06/05) | Week 9 (13/05) | Week 10 (20/05) | Week 11 (27/05) | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 7.00 |
| Tuesday | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 3 | 3 | 3 | 14.00 |
| Wednesday | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 6 | 5 | 19.00 |
| Thursday | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 5 | 13.00 |
| Friday | 2 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5.00 |
| Saturday | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 10.00 |
| Sunday | 0 | 4 | 1 | 3 | 0 | 0 | 0 | 2 | 4 | 1 | 0 | 15.00 |
| Total | 6.00 | 7.00 | 8.00 | 5.00 | 2.00 |  |  | 6.00 | 15.00 | 16.00 | 18.00 | 83.00 |
| Rate | $80 | $ 80.00 | $80 | $ 80.00 | $80 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Total | $ 480.00 | $ 560.00 | $ 640.00 | $ 400.00 | $ 160.00 | | | $ 480.00 | $ 1,200.00 | $ 1,280.00 | $ 1,440.00 | $ 6,640.00 |

## 2.3.4 Bowen Deng

| Day | Week 1 (Date) | Week 2 (Date) | Week 3 | Week 4 | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | | | | | | | | | | 1 | | 1.00 |
| Tuesday | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 4 | 2 | 26.00 |
| Wednesday | 1 | 3 | 2 | | 1 | | | 2 | 1 | 1 | | 11.00 |
| Thursday | | | | | | | | | | | | |
| Friday | 2 | | 1 | | 3 | | | | 1 | | | 7.00 |
| Saturday | | | | 2 | | | | 3 | 2 | | | 7.00 |
| Sunday | 2 | | | | | | | | | 3 | | 5.00 |
| Total | 7.00 | 5.00 | 5.00 | 5.00 | 6.00 | 2.00 | 2.00 | 7.00 | 7.00 | 9.00 | 2.00 | **57.00** |
| Rate | $80 | $ 80.00 | $80 | $ 80.00 | $80 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | $ 80.00 | |
| Total | $ 560.00 | $ 400.00 | $ 400.00 | $ 400.00 | $ 480.00 | $ 160.00 | $ 160.00 | $ 560.00 | $ 560.00 | $ 720.00 | $ 160.00 | **$ 4,560.00** |

## 2.3.5 Balraj

| Day | Week 1 (Date) | Week 2 (Date) | Week 3 | Week 4 | Week5 | Week6 | Week7 | Week8 | Week9 | Week 10 | Week 11 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | | | 0 | 0 | 0 | | | | | | | |
| Tuesday | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2 | 3 | 2 | 2 | 23.00 |
| Wednesday | | | | | 2 | | | | | | | 2.00 |
| Thursday | | | | | 2 | | | | 5 | | | 7.00 |
| Friday | 2 | 2 | 2 | 2 | | | | | | 5 | 6 | 19.00 |
| Saturday | 1 | | | 1 | | 1 | | 3 | | 8 | | 14.00 |
| Sunday | 2 | 1 | 2 | | | | | 5 | 5 | 10 | | 25.00 |
| Total | 7.00 | 5.00 | 6.00 | 5.00 | 7.00 | 2.00 | 2.00 | 10.00 | 13.00 | 25.00 | 8.00 | **90.00** |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rate | $80 | $ 80.00 | $80 | $ 80.00 | $80 | $ 80.00 | $80.00 | $80.00 | $80.00 | $80.00 | $80.00 | |
| Total | $ 560.00 | $ 400.00 | $ 480.00 | $ 400.00 | $ 560.00 | $ 160.00 | $160.00 | $800.00 | $1040.00 | $2000.00 | $640.00 | **$7200** |

# 3. Solution Architecture

## 3.1 Data Architecture Model

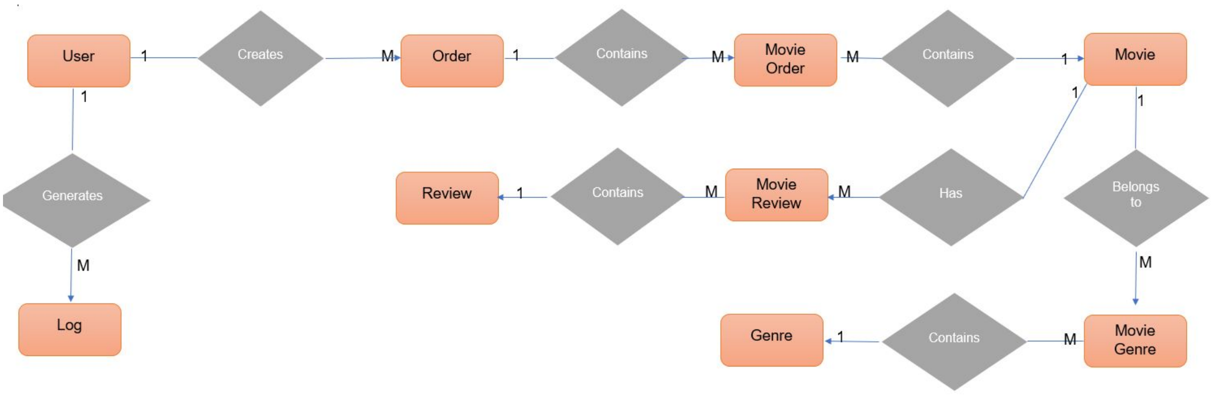
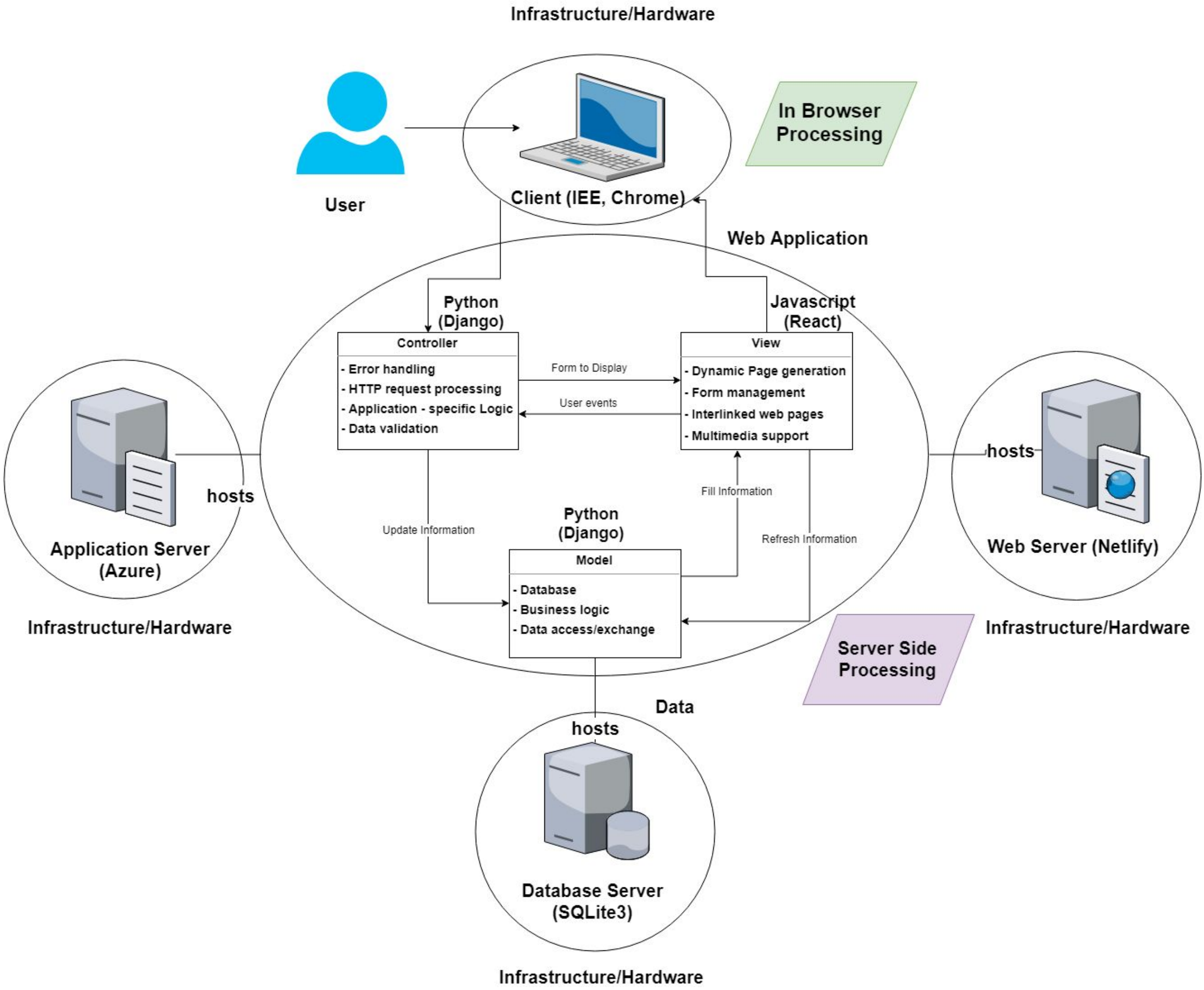


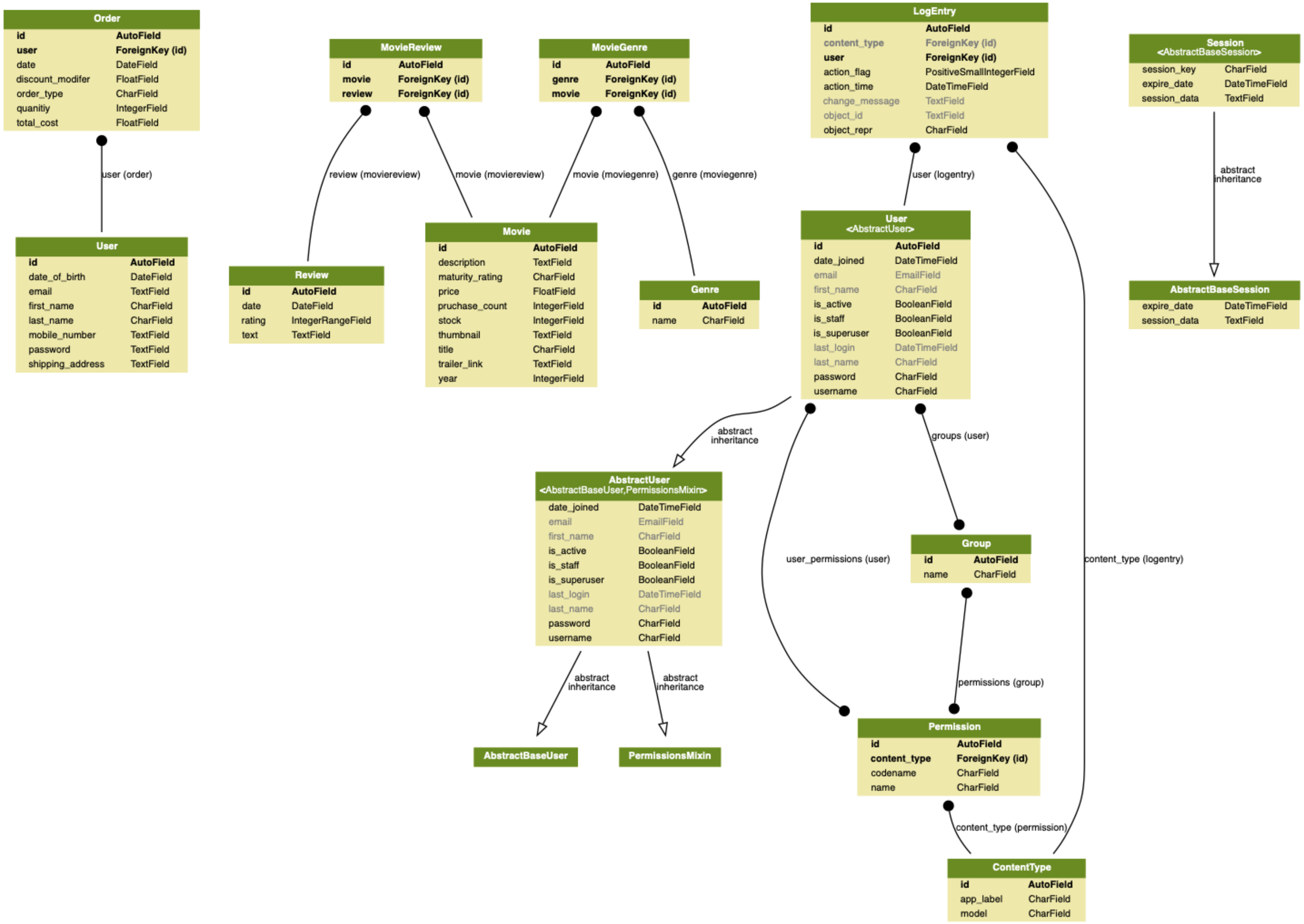
Fig: Above figure is an Data Architecture Model explaining relationship between different entities.
Relationships:
Admin can manage multiple orders.
User can create multiple orders.
Order can have multiple movies.
User can give multiple reviews.
A movie can have multiple reviews.

## 3.2 Application Architecture Model

A application architecture model is has been included to illustrate the overall architecture of the web application. The figure below shows the interdependence of the hardware/software components used to host the web application. The core system will operate under the MVC model to split the major components into three connected components. The main drivers of this decision were to allow parallel development, code flexibility and manageability. Additionally, the database hosts were chosen based on various criteria such as flexibility, usability and functionality.

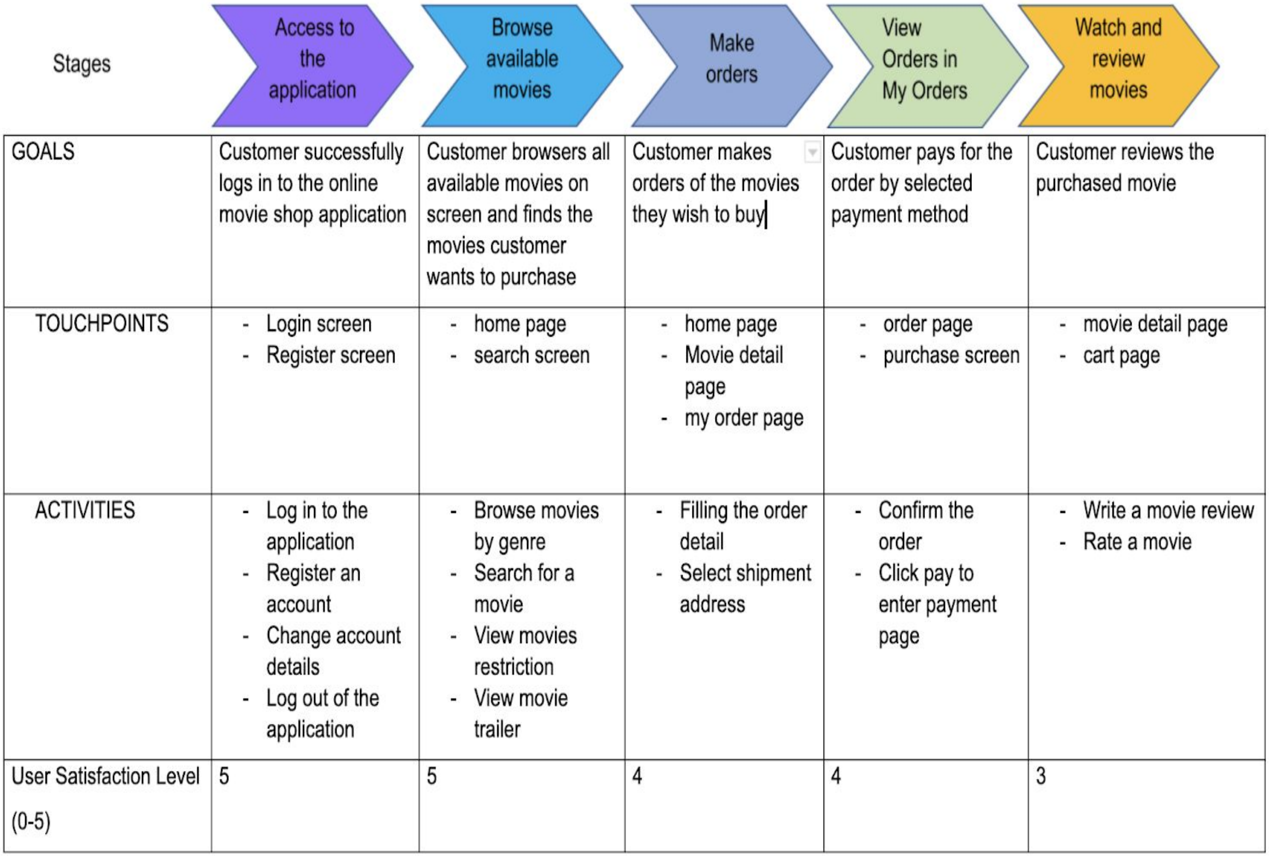

Infrastructure/Hardware
User
Client (IEE, Chrome)
In Browser Processing
Web Application
Python (Django)
Javascript (React)
Controller
- Error handling
- HTTP request processing
- Application - specific Logic
- Data validation
Form to Display
User events
View
- Dynamic Page generation
- Form management
- Interlinked web pages
- Multimedia support
hosts
Application Server (Azure)
Infrastructure/Hardware
Update Information
Python (Django)
Fill Information
Refresh Information
Model
- Database
- Business logic
- Data access/exchange
hosts
Web Server (Netlify)
Infrastructure/Hardware
Server Side Processing
Data
hosts
Database Server (SQLite3)
Infrastructure/Hardware

## 3.3 Django Architecture



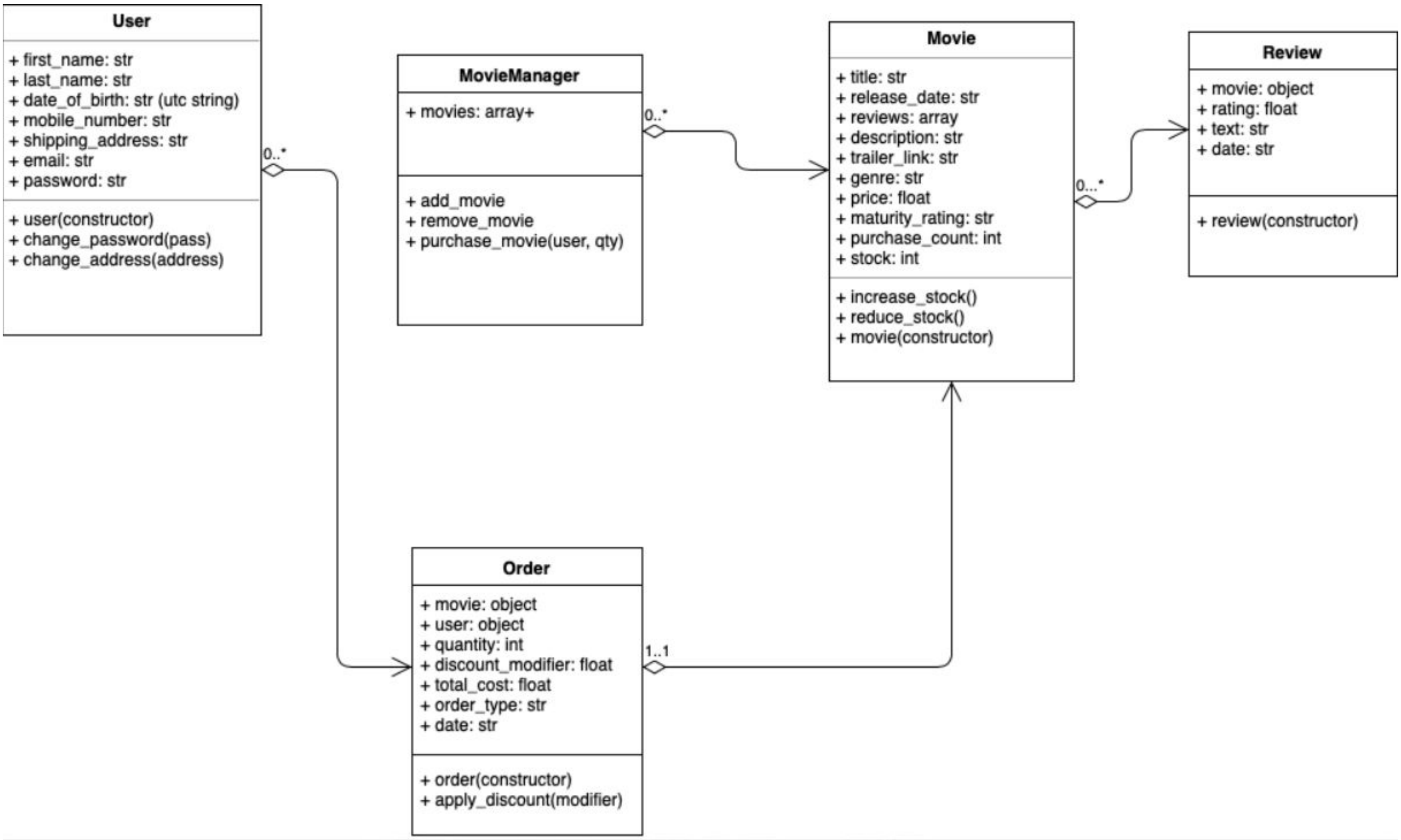
# 4. Solutions Design

## 4.1 Interaction Design Model

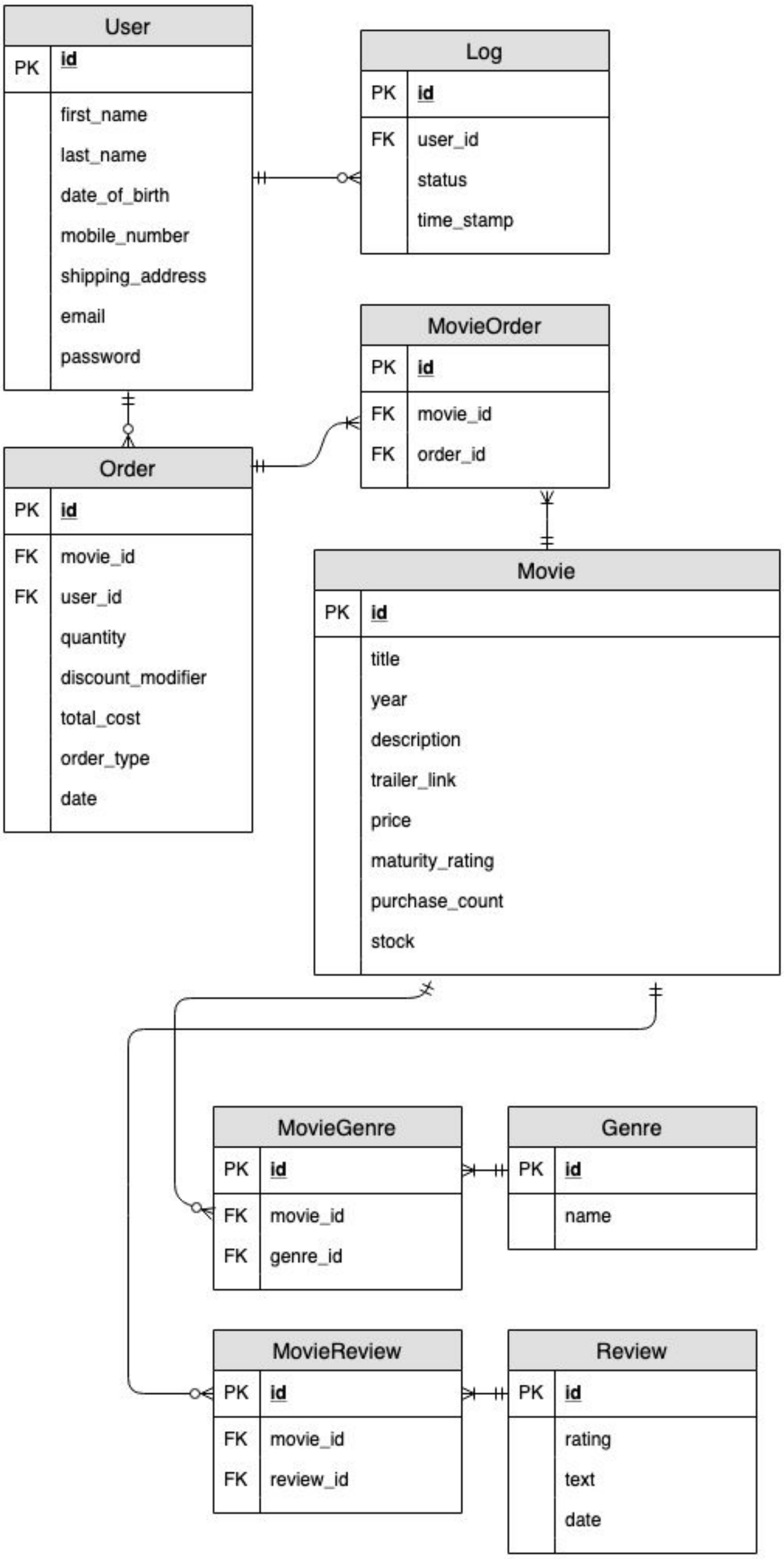


| Stages | Access to the application | Browse available movies | Make orders | View Orders in My Orders | Watch and review movies |
|---|---|---|---|---|---|
| GOALS | Customer successfully logs in to the online movie shop application | Customer browsers all available movies on screen and finds the movies customer wants to purchase | Customer makes orders of the movies they wish to buy | Customer pays for the order by selected payment method | Customer reviews the purchased movie |
| TOUCHPOINTS | - Login screen<br>- Register screen | - home page<br>- search screen | - home page<br>- Movie detail page<br>- my order page | - order page<br>- purchase screen | - movie detail page<br>- cart page |
| ACTIVITIES | - Log in to the application<br>- Register an account<br>- Change account details<br>- Log out of the application | - Browse movies by genre<br>- Search for a movie<br>- View movies restriction<br>- View movie trailer | - Filling the order detail<br>- Select shipment address | - Confirm the order<br>- Click pay to enter payment page | - Write a movie review<br>- Rate a movie |
| User Satisfaction Level (0-5) | 5 | 5 | 4 | 4 | 3 |

## 4.2 Object Design Model


User
+ first_name: str
+ last_name: str
+ date_of_birth: str (utc string)
+ mobile_number: str
+ shipping_address: str
+ email: str
+ password: str
+ user(constructor)
+ change_password(pass)
+ change_address(address)
0..*
MovieManager
+ movies: array+
+ add_movie
+ remove_movie
+ purchase_movie(user, qty)
0..*
Movie
+ title: str
+ release_date: str
+ reviews: array
+ description: str
+ trailer_link: str
+ genre: str
+ price: float
+ maturity_rating: str
+ purchase_count: int
+ stock: int
+ increase_stock()
+ reduce_stock()
+ movie(constructor)
0...*
Review
+ movie: object
+ rating: float
+ text: str
+ date: str
+ review(constructor)
Order
+ movie: object
+ user: object
+ quantity: int
+ discount_modifier: float
+ total_cost: float
+ order_type: str
+ date: str
+ order(constructor)
+ apply_discount(modifier)
1..1

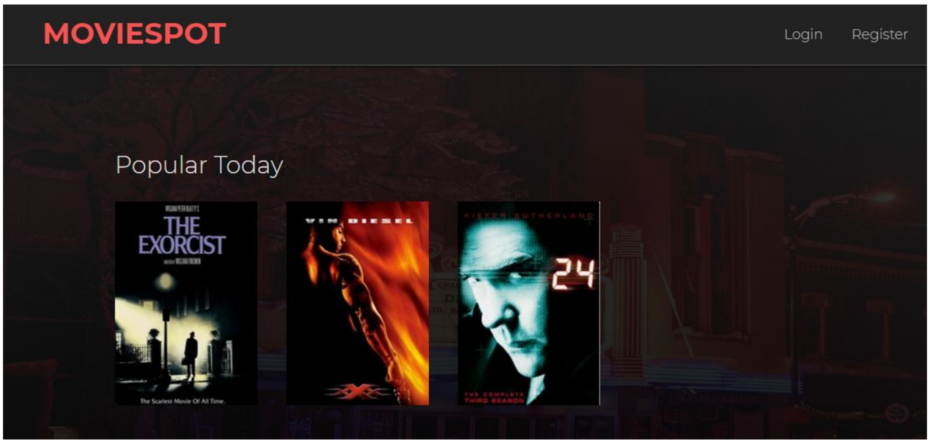
## 4.3 Backend Database Design Model



# 5. Architecture Spike (Software Prototype)

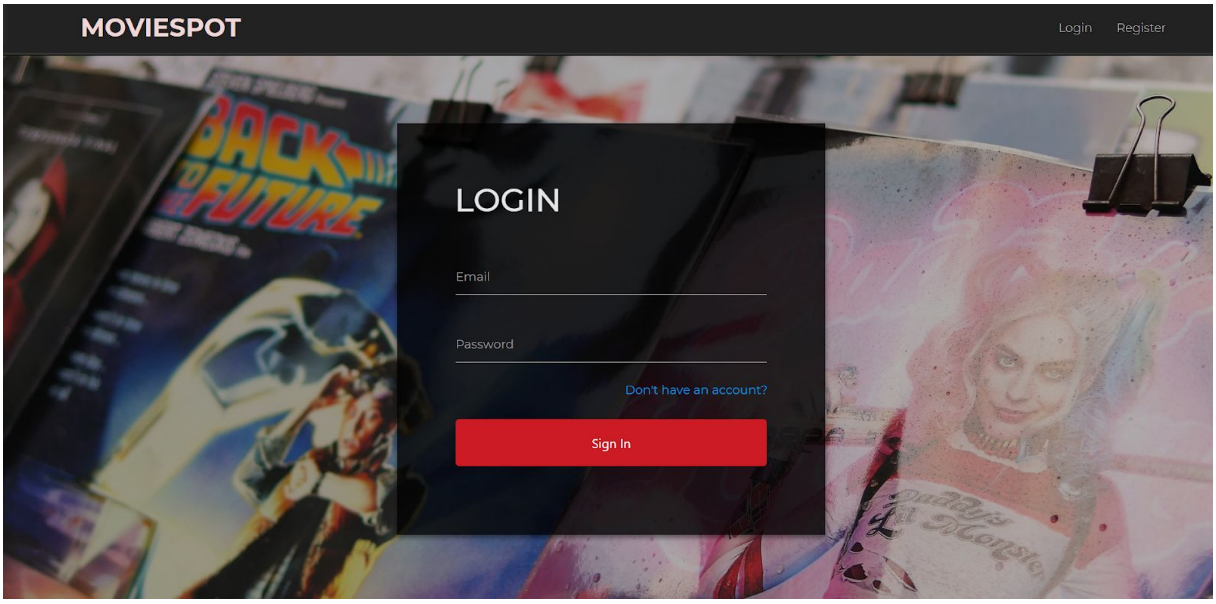
## 5.1 Interfaces

### 5.1.1 Home Page



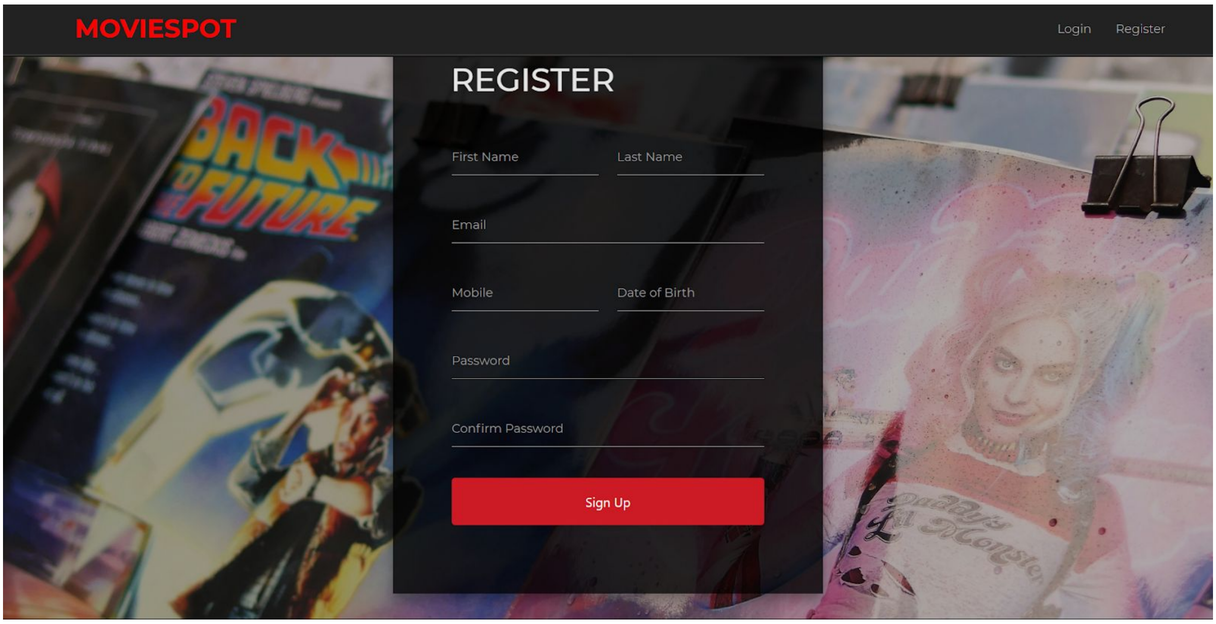
It is the starting point of the web page application. On the top right of the page Login for existing customers and Register button for existing customers is provided.

### 5.1.2 Login Page



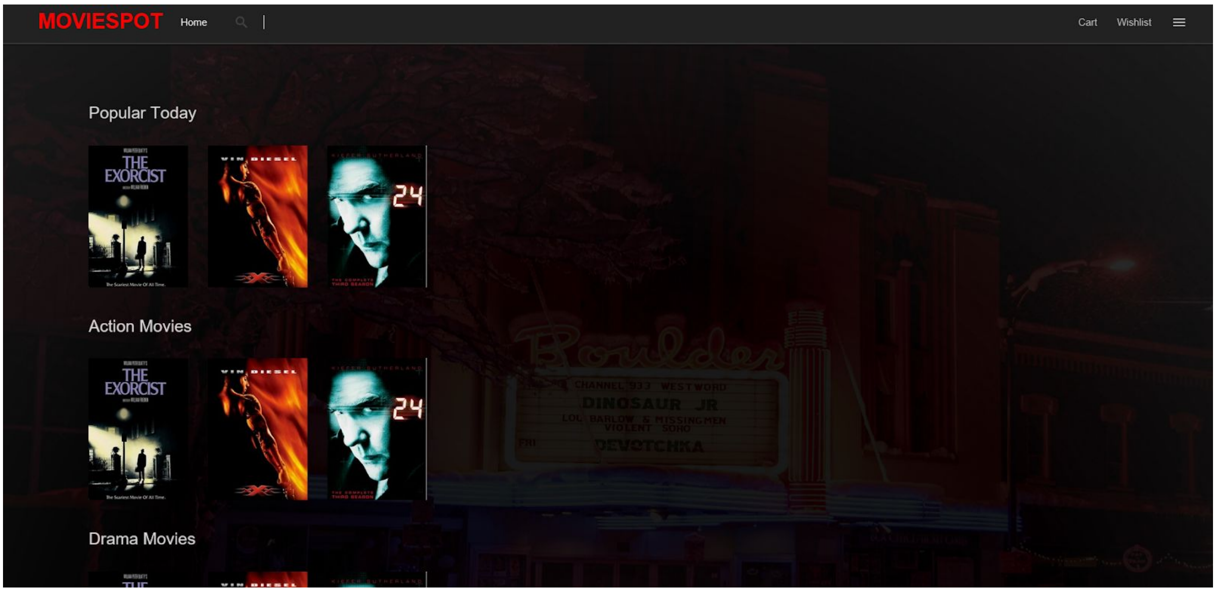
Once the existing customer clicks the login button, they will be redirected to login page were the user can login to access the movie page.

### 5.1.3 Register Page



Once the new customer clicks the register button, they will be redirected to register page were the new user can enter their details and create a password.
Once they enter sign up they will be redirected to login page to enter the movie store.

### 5.1.4 Movie Display Screen



After logging in the application the customer will be greeted by the movie screen.
On this screen customer and search movies.
On the top right they is also cart option to view existing movies in the cart for purchase and wish list where customer can store their prefered movies to order in future.

## 5.2. Presentation

A live deployment of the online movie store is available at:
https://online-movie-store.netlify.com/

## 5.3 GitHub Project

The source code can be found at: https://github.com/danielebra/online-movie-store

# 6. Current Features

## 6.1 User Access Management

**Software Application Feature or Service or Module**

Our project requires customer to access the app through 'Login' process and to safely exit through 'Logout' process. When customer click 'Login' button, authentication is done at backend server to ensure security. When customer click 'LogOut' button, the app will navigate customer back to Login page.

| Mapping User Stories | |
|---|---|
| #101 | As a customer, I want to log in to the online application, so that I can access the system's features |
| #102 | As a customer, I want to log out of the online application, so that I can exit the system securely |
| #402 | As a customer, I want to check my access logs, so that I can know my logs activities |
| #403 | As a customer, I want to delete a particular access log, so that I can manage my log records |

**Non-functional Aspects**

Access log management is considered as a non-functional feature to keep the system secure. Each login and logout activity will be recorded and stored at backend and these records can be viewed by customer on access log page. Customer can search for logs by date as well as deleting a particular log. However, for security concern, customer cannot update or edit logs. Additional, customer is able to access the application 24/7. Log in process is completed within 2 minutes of inputting the correct credentials. Customer is restricted to admin features such as deleting customer users.

**Software Testing Results**

Acceptance test criteria

| User Story | Criteria |
|---|---|
| #101 | 1. Login page displays all required data fields<br>2. Account and password are authenticated on backend server. If authentication is corrected, the app gets that user object as response. Otherwise, the app receives an error<br>3. If login succeed, app directs customer to home page. |
| #102 | 1. When log out of the app, user account details are removed from local storage.<br>2. After logging out of the app, customer is navigated to Login page |
| #402 & #403 | 1. Both login and logout activities are stored on backend.<br>2. Access logs page display all log activities.<br>3. A log is deleted once the 'delete' button is clicked, that log is removed on both front-end and back-end. |

Test case record

| | Test Case | Date | User Story | Test Criteria | Test Result |
|---|---|---|---|---|---|
| Test Case #1 | Fill in login page with a wrong account, click on 'Login' button | 7 May 2019 | #101 | 1. Login page displays all required data fields<br>2. Account and password are authenticated on backend server. If authentication is corrected, the app gets that user object as response. Otherwise, the app receives an error | Fail |

| | | | | | |
|---|---|---|---|---|---|
| Test Case #2 | Login to the app with an existing account, go to ‘Access Logs’ page, look at what is displayed | 20 May 2019 | #402 | 1. Login activity is stored on backend server<br>2. If login succeed, app directs customer to home page.<br>3. Access logs page display all log activities. | Success |
| Test Case #3 | Logout of the app and login with the account used in test case #2, go to ‘Access Logs’ page, look at what is displayed | 20 May 2019 | #402 | 1. Logout activity is stored on backend server<br>2. Access logs page display all log activities. | Success |
| Test Case #4 | Login with the account used in test case #2, go to ‘Access Logs’ page, delete the first 2 logs. | 22 May 2019 | #403 | 1. A log is deleted once the ‘delete’ button is clicked, that log is removed on both front-end and back-end. | Fail |

**Defect Log**

| Defect ID | Defect Description | Defect Date | Test Case ID | Tester Name | Responsible | Status |
|---|---|---|---|---|---|---|
| DI001 | When login with a wrong account, I can still access to the system | 7 May | Test Case #1 | Calvin Thai | Bowen Deng | resolved |

| | | | | | | |
|---|---|---|---|---|---|---|
| DI002 | After clicked on the 'delete' button, the logs are not removed from server | 22 May | Test Case #4 | Balraj Brar | Bowen Deng | resolved |
| Summary: total defected – 2, resolved - 2 | | | | | | |

## 6.2. Online User Registration

**Software Application Feature or Service or Module**

Our application allows a new customer to register their account, so that they can access features such as placing orders, adding movies to wish list, paying for the orders. The customer can view their registration details and update information such as email address and contact details. Customers are able to exit the registration process at any time before submitting the form.

| Mapping user stories | |
|---|---|
| #104 | As a new customer, I want to register an account, so that I become a member of the store |
| #103 | As a customer, I want to change account details, so that I can keep my account up to date and secure. |

**Non-functional Aspects**

The registration process is quick and efficient once customer fills in the registration form. Our app allows the customer to make long and complicated input (e.g. password). To ensure security, once the customer exits the registration process, all private information will be destroyed.

**!**

**Software Testing Results**

Acceptance test criteria

| User Story | Criteria |
|---|---|

| | |
|---|---|
| #104 | 1. Display comprehensive input field such as name, email, phone number, address etc. no information missing<br>2. Allow customer to exit the process<br>3. Password setting can be complex, consisting of number, letter and symbol.<br>4. Created new account is sent to backend API<br>5. Cancel the process will destroy all private information |
| #103 | 1. Registration details is changeable during sign-up process and after successful registration.<br>2. Some information is not changeable such as name, gender.<br>3. Updated detail will be post to backend API |

**Test case record**

| | Test Case | Date | User Story | Test Criteria | Test Result |
|---|---|---|---|---|---|
| Test Case #1 | Click on ‘Register’ tab to check if all input fields are display correctly | 5 May 2019 | #104 | Display comprehensive input field such as name, email, phone number, address etc. no information missing | Success |
| Test Case #2 | Click on ’Register’ tap to enter registration, and click on other tab to see if exit the process | 5 May 2019 | #104 | 1. Allow customer to exit the process<br>2. Cancel the process will also be destroyed all private information | Fail |

| | | | | | |
|---|---|---|---|---|---|
| Test Case #3 | Click on 'Register' tap and following the process, click 'sign up' button to see if sign-up is successful | 6 May 2019 | #104<br>#103 | 1. Password setting can be complex, consisting of number, letter and symbol.<br>2. Created new account is posted to backend API<br>3. Registration details is changeable during sign-up process | Success |
| Test Case #4 | Login with account created in Test Case #3, click 'account detail' tab, check if email address, name, gender can be changed | 6 May 2019 | #103 | 1. Registration details is changeable after registration<br>2. Some information is not changeable such as name, gender.<br>3. Updated detail will be post to backend API | Success |

**Defect Log**

| Defect ID | Defect Description | Defect Date | Test Case ID | Tester Name | Responsible | Status |
|---|---|---|---|---|---|---|
| DI001 | After exit the registration form, the fields did not be destroyed | 5 May 2019 | Test Case #2 | Sammy | Daniel | Resolved |
| Summary: total defected – 1, resolved - 1 | | | | | | |

## 6.3. Movie Catalogue Management

**Description**
The Movie Catalogue is the main screen of the application shown to logged in users. Users can interact with this main menu to browse through information and find more information about movies. The main functions of the Movie Catalogue is, search, view by genre and view movie details. Additionally, there are admin features that allow administrative users to add, delete and update movies from the database. The following user stories have been implemented with this module.

| User Story ID | User Story |
|---|---|
| **201** | **As a user I want to browse the available movies so that I know what movies I can order** |
| **202** | **As a user I want to search for a movie so that I can find the movies I am looking for** |
| **204** | **As a user I want to browse movies by genre so that I can find action/drama movies.** |
| **206** | **As a user I want to update available stock of each movie so that customers know which movies are available for purchase.** |
| **303** | **As a user I want to view a movie trailer/detailed description so that I can see if I am interested in the movie.** |

**Non Functional Aspects**

- Users can browse the application using a mobile phone
- The movie screen will display <loading> while no movies are found
- The web pages are designed using usability heuristics.
- Users can easily view the table and manage movies all in one location.

**Software Testing**

| ID | Test Case | Date | User Story | Test Criteria | Test Result |
|---|---|---|---|---|---|
| Test Case #301 | Scroll through main menu and select ‘FAST & FURIOUS 8’. | 16 May 2019 | #201 | A user can browse movies and select a movie. | Success |
| Test Case #302 | Click on the white search bar and type in ‘FAST’. Select ‘FAST & FURIOUS 8’ | 19 May 2019 | #202 | A user must be able to search for a movies/genres by typing into a interactive bar. | Success |

| | | | | | |
|---|---|---|---|---|---|
| Test Case #303 | Click on white search bar and type in ‘ACTION’. Select on ‘FAST & FURIOUS 8’ | 19 May 2019 | #204 | Users can search movies by genre. | Fail |
| Test Case #304 | Select the hamburger menu in the top right and select Update Movie from the dropdown menu. Find the ‘FAST & FURIOUS 8’ log and click on the small pencil. Select the value in the stock column and change the value to ‘30’. | 22 May 2019 | #206 | Change the amount of available stock for a movie to 30. | Success |
| Test Case #305 | Select ‘FAST & FURIOUS 8’ image. Once the popup appears view description and scroll down for trailer. Click on embedded youtube trailer to watch. | 18 May 2019 | #303 | View a movie description and trailer using the main menu. | Success |

**Defect Log**

| Defect ID | Defect Description | Defect Date | Test Case | Tester’s Name | Responsible | Status | Comments |
|---|---|---|---|---|---|---|---|
| DI 501 | No movies shown on the main screen, | 10/05/2019 | #301 | Calvin | Calvin | Fixed | Movie list not populated. |
| DI 502 | Movies were shown but cannot be selected. | 11/05/2019 | #301 | Calvin | Calvin | Fixed | Movie object not created. |
| DI 503 | Failed to Add Movie to Database | 14/05/2019 | #301 | Calvin | Calvin | Fixed | Failed to call right functions. |
| DI 504 | Search bar not affecting movie list | 14/05/2019 | #302 | Calvin | Calvin | Fixed | Search functions not built. |
| DI 505 | Changed values did not change when save was clicked. | 18/05/2019 | #304 | Calvin | Calvin | Fixed | Changes were not reflected in the backend. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| DI 506 | The movie trailer loads but cannot be played. | 18/05/2019 | #306 | Calvin | Calvin | Fixed | Embedded URL not provided |
| DI 507 | Movie Management Search bar is not filtering results correctly. | 18/05/2019 | #303 | Calvin | Calvin | Fixed | Search bar did not have required functions |
| DI 508 | Update genre button not working. | 29/05/2019 | #304 | Calvin | Calvin | Fixed | Genres called from state |
| Total | Total Defects: 8, Resolved 8 | | | | | | |

## 6.4. Order Management

**Software Application Feature or Service or Module**
Our online movie store project allows customer to place order(s) on selected movie, and customer is able to cancel the process during order placement. Then the order is stored in the order list as historical record, customer can go to the order to check his/her order. Additional, customer can use the search bar to search order by using order date and time (each order has a unique date and time). If the order has not been paid, customer can cancel and update the order. Customer can also update their order quantity on orders page.

| Mapping user stories | |
|---|---|
| #301 | As a customer, I want to make an order for the movie(s) I selected, so that I can purchase the movie(s). |
| #302 | As a customer, I want to update my order quantity. |
| #303 | As a customer, I want to send an order cancellation request, so that I can apply to delete my order. |
| #304 | As a customer, I want to search an order, so that I can easily find out and check a particular order. |

**Non-functional Aspects**

To smoothen order placement process, the UI is designed to be concise and easy to understand. Apart from basic order management functionality, to make our customer feel more comfortable and satisfied, we could add some extended features. For example, if customer want to order a movie later, he/she could add that movie to ‘wish list’. On the other hand, to improve security, once customer clicks on ‘Cancel’ during the process, all private data will be destroyed on both front end and back end.

**Software Testing Results**

**Acceptance test criteria**

| User Story | Criteria |
|---|---|
| #301 | 1. Display movie details (title, intro, price, stock) before customer place an order<br>2. Allow customer to choose how many copies to buy<br>3. Display total price on the screen<br>4. Created new order is stored on the order list and back-end database |
| #302 | 1. Order detail can be changed at any time before check-out<br>2. Edited order is updated on back-end database |
| #303 | 1. Allow customer to cancel unpaid order at any time<br>2. Cancelled order is removed from order list and back-end database<br>3. Any information about Cancelled order is destroyed |

| | |
|---|---|
| #304 | 1. Order can be search by entering order date. |

**Test case record**

| | Test Case | Date | User Story | Test Criteria | Test Result |
|---|---|---|---|---|---|
| Test Case #1 | Check movie page is displaying movie details or not | 8 May 2019 | #301 | Display movie details (title, intro, price, stock) before customer place an order | Success |
| Test Case #2 | Click on the ‘+’ and ‘-’, see if the quantity is changing and display correct total price | 8 May 2019 | #301 | 1. Allow customer to choose how many copies to buy<br>2. Display total price on the screen | Success |
| Test Case #3 | Create a new order on ‘FAST & FURIOUS 8’ 2 copy, check if new order is stored on order list and backend database | 9 May 2019 | #301 | Created new order is stored on the order list and back-end database | Fail |
| Test Case #4 | Change the quantity of the order created in Test Case #3, check if the order is updated on back-end database | 9 May 2019 | #302 | 1. Order detail can be changed at any time before check-out<br>2. Edited order is updated on back-end database | Fail |

| | | | | | |
|---|---|---|---|---|---|
| Test Case #5 | Cancel the order created on Test Case #3 by clicking on ‘Cancel’ button, check if order is removed from order list and back-end database | 12 May 2019 | #303 | 1. Allow customer to cancel unpaid order at any time<br>2. Cancelled order is removed from order list and back-end database<br>3. Any information about Cancelled order is destroyed | Success |
| Test Case #6 | Create 5 new orders, input one of the order date, check if proper result is shown | 15 May 2019 | #304 | 1. Order can be search by entering order date | Success |
| Test Case #7 | System prevents user from purchasing a movie that has 0 stock | 31 May 2019 | | 1. Place an order for an out of stock item | Fails |

**Defect Log**

| Defect ID | Defect Description | Defect Date | Test Case ID | Tester Name | Responsible | Status |
|---|---|---|---|---|---|---|
| DI001 | New order was unable to be posted to backend database | 9 May 2019 | Test Case#3 | Bowen Deng | Balraj | Resolved |
| DI002 | Edited order was unable to be updated on backend database | 9 May 2019 | Test Case#4 | Bowen Deng | Balraj | Resolved |

| | | | | | | |
|---|---|---|---|---|---|---|
| DI003 | Search bar was not giving the correct result order | 15 May 2019 | Test Case#6 | Bowen Deng | Bowen Deng | Resolved |
| DI004 | Can purchase out of stock movie | 31 May 2019 | Test Case #7 | Daniel | Balraj | Unresolve d |
| Summary: total defected – 4, resolved - 3 | | | | | | |

## 6.5 User Management

**Software Application Feature or Service or Module**

Our online movie store project allows user with an Admin login to access user management feature.

Under this feature the admin can perform following-

1. Amin can view, add, update and delete a new user
2. can activate and deactivate the user status.
3. Can search user based on their full name and phone number.

| Mapping user stories | |
|---|---|
| #501 | As an Admin, I want to create a new user. |
| #502 | As an Admin, I want to update existing user details. |
| #503 | As an Admin, I want to view user details. |
| #504 | As an Admin, I want to delete a user. |
| #505 | As an Admin, I want to activate and deactivate user details. |
| #506 | As an Admin, I want to search user based on their full name and phone number. |

**Non-functional Aspects**

To make user management efficient process, the UI design is kept simple and easy to navigate.

- Users can browse the application using a mobile phone
- The web pages are designed using usability heuristics.
- Users can easily view the table and manage user details all in one location.
- This feature is not available to users who do not hold ADMIN login.

**Software Testing Results**

Acceptance test criteria

| User Story | Criteria |
|---|---|
| #501 | 1. Display '+' icon so admin can access add user field.<br>2. Display all the necessary fields to add a new user and click sign up.<br>3. Display added user details to user log and user management api in backend. |
| #502 | 1. Display update icon and enable click.<br>2. Make user details editable.<br>3. When user details changed and saved, display updated user details. |
| #504 | 1. Display delete icon and enable click.<br>2. Display message to confirm delete.<br>3. Removed delete user details from user management page and backend database. |

**Test case record**

| | Test Case | Date | User Story | Test Criteria | Test Result |
|---|---|---|---|---|---|
| Test Case #1 | Click on 'pencil' icon to change user details, click update. Check if user details updated. | 6 May 2019 | #502 | 1. Click pencil icon to update.<br>2.Icon not responsive, hence user details cannot be updated. | Fail |

| | | | | | |
|---|---|---|---|---|---|
| Test Case #2 | Press delete icon to check if user is removed from database | 21 May 2019 | #504 | 1. Press delete icon<br>2. Click ok in the message box.<br>3.User deleted from database. | Success |
| Test Case #3 | Admin click on ‘+’ Icon to add new user and press signup to check if new user added. | 20 May 2019 | #501 | 1.Click + icon<br>2.Add the new user details are filled.<br>3. Press signup button<br>4.New user details added both in database and frontend. | Fail |

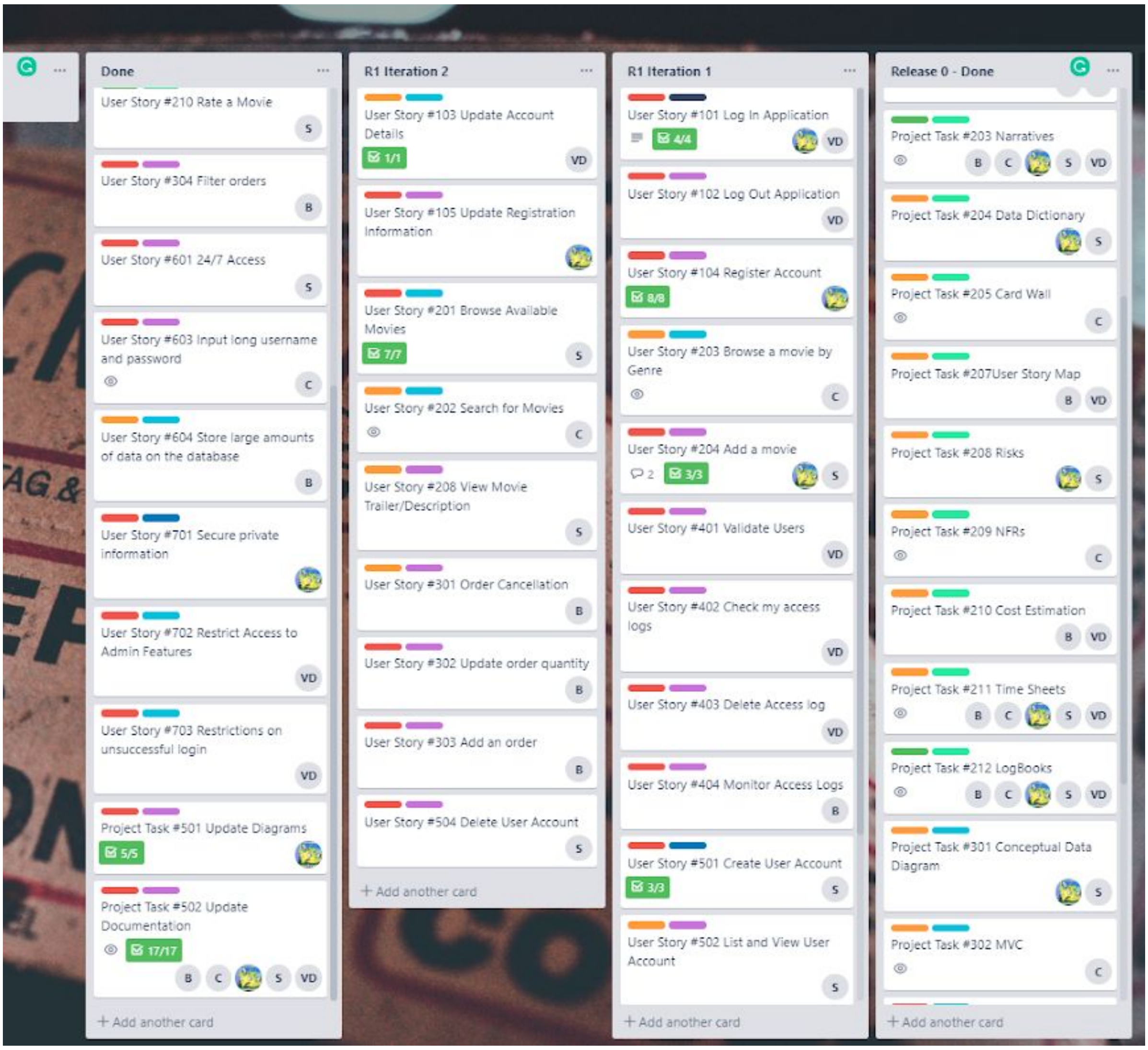
**Defect Log**

| Defect ID | Defect Description | Defect Date | Test Case ID | Tester Name | Responsible | Status |
|---|---|---|---|---|---|---|
| DI001 | Amin cannot update user details. | 15 May 2019 | Test Case #1 | Balraj | Sammy | Resolved |
| DI002 | If an admin trys to add a user and exits the pop up, the error messages are displayed on the main interface. | 15 May 2019 | Test Case #3 | Sammy | Sammy | Resolved |
| Summary: total defected – 1, resolved - 1 | | | | | | |

# 7. Appendices

## 7.1 Diagrams

### 7.1.1 - Appendix 2. Card Wall



**Done**

- User Story #210 Rate a Movie — S
- User Story #304 Filter orders — B
- User Story #601 24/7 Access — S
- User Story #603 Input long username and password — C
- User Story #604 Store large amounts of data on the database — B
- User Story #701 Secure private information
- User Story #702 Restrict Access to Admin Features — VD
- User Story #703 Restrictions on unsuccessful login — VD
- Project Task #501 Update Diagrams — 5/5
- Project Task #502 Update Documentation — 17/17 — B C S VD

+ Add another card

**R1 Iteration 2**

- User Story #103 Update Account Details — 1/1 — VD
- User Story #105 Update Registration Information
- User Story #201 Browse Available Movies — 7/7 — S
- User Story #202 Search for Movies — C
- User Story #208 View Movie Trailer/Description — S
- User Story #301 Order Cancellation — B
- User Story #302 Update order quantity — B
- User Story #303 Add an order — B
- User Story #504 Delete User Account — S

+ Add another card

**R1 Iteration 1**

- User Story #101 Log In Application — 4/4 — VD
- User Story #102 Log Out Application — VD
- User Story #104 Register Account — 8/8
- User Story #203 Browse a movie by Genre — C
- User Story #204 Add a movie — 2 — 3/3 — S
- User Story #401 Validate Users — VD
- User Story #402 Check my access logs — VD
- User Story #403 Delete Access log — VD
- User Story #404 Monitor Access Logs — B
- User Story #501 Create User Account — 3/3 — S
- User Story #502 List and View User Account — S

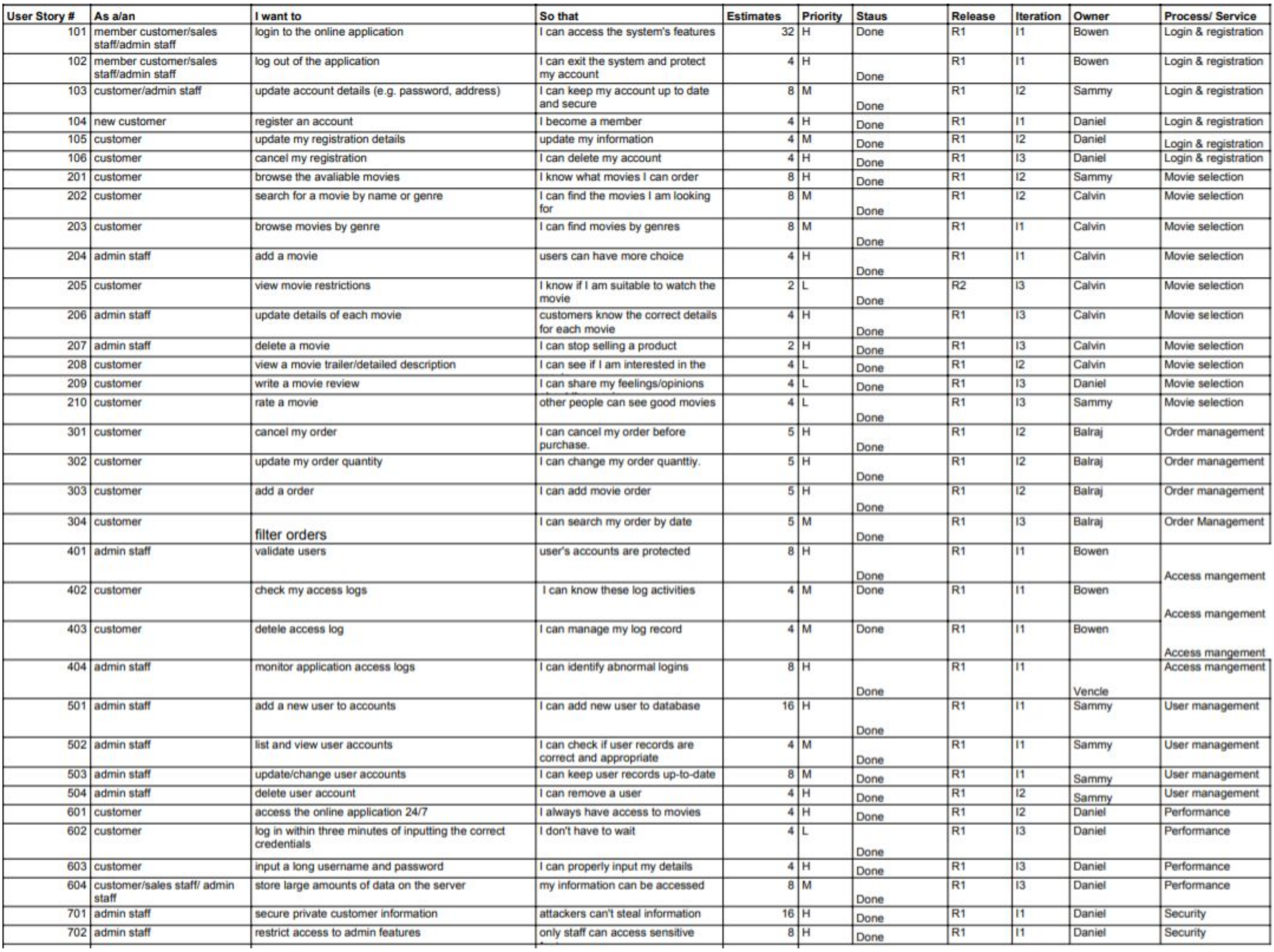
+ Add another card

**Release 0 - Done**

- Project Task #203 Narratives — B C S VD
- Project Task #204 Data Dictionary — S
- Project Task #205 Card Wall — C
- Project Task #207User Story Map — B VD
- Project Task #208 Risks — S
- Project Task #209 NFRs — C
- Project Task #210 Cost Estimation — B VD
- Project Task #211 Time Sheets — B C S VD
- Project Task #212 LogBooks — B C S VD
- Project Task #301 Conceptual Data Diagram — S
- Project Task #302 MVC — C

+ Add another card

### 7.1.1 - Appendix 1. Requirements Matrix



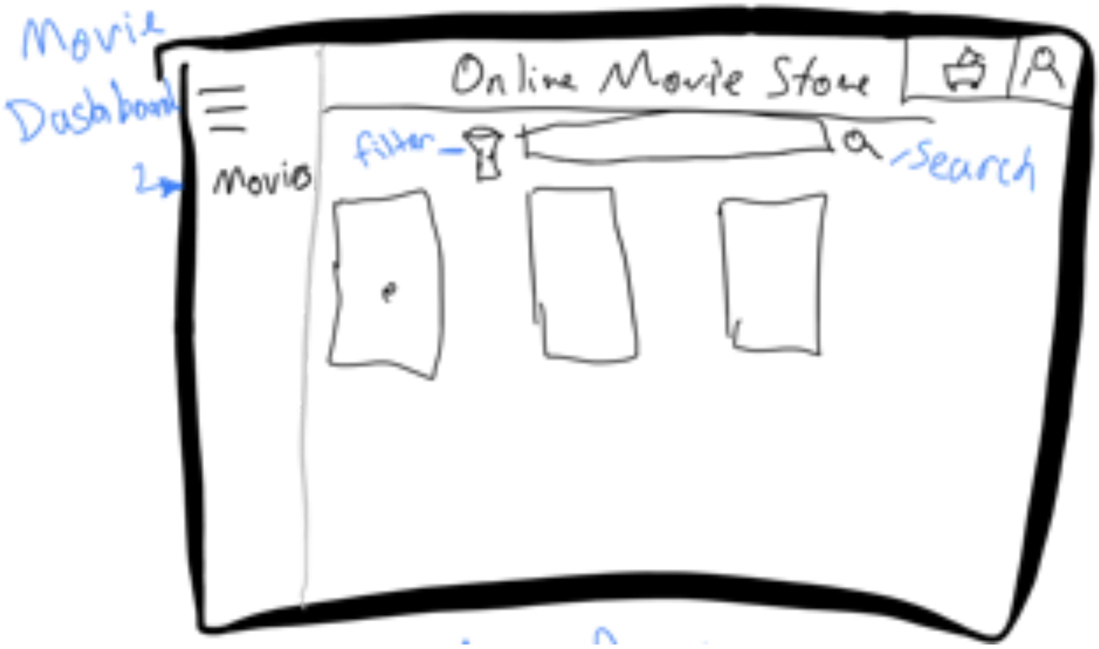
| User Story # | As a/an | I want to | So that | Estimates | Priority | Staus | Release | Iteration | Owner | Process/ Service |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | member customer/sales staff/admin staff | login to the online application | I can access the system's features | 32 | H | Done | R1 | I1 | Bowen | Login & registration |
| 102 | member customer/sales staff/admin staff | log out of the application | I can exit the system and protect my account | 4 | H | Done | R1 | I1 | Bowen | Login & registration |
| 103 | customer/admin staff | update account details (e.g. password, address) | I can keep my account up to date and secure | 8 | M | Done | R1 | I2 | Sammy | Login & registration |
| 104 | new customer | register an account | I become a member | 4 | H | Done | R1 | I1 | Daniel | Login & registration |
| 105 | customer | update my registration details | update my information | 4 | M | Done | R1 | I2 | Daniel | Login & registration |
| 106 | customer | cancel my registration | I can delete my account | 4 | H | Done | R1 | I3 | Daniel | Login & registration |
| 201 | customer | browse the avaliable movies | I know what movies I can order | 8 | H | Done | R1 | I2 | Sammy | Movie selection |
| 202 | customer | search for a movie by name or genre | I can find the movies I am looking for | 8 | M | Done | R1 | I2 | Calvin | Movie selection |
| 203 | customer | browse movies by genre | I can find movies by genres | 8 | M | Done | R1 | I1 | Calvin | Movie selection |
| 204 | admin staff | add a movie | users can have more choice | 4 | H | Done | R1 | I1 | Calvin | Movie selection |
| 205 | customer | view movie restrictions | I know if I am suitable to watch the movie | 2 | L | Done | R2 | I3 | Calvin | Movie selection |
| 206 | admin staff | update details of each movie | customers know the correct details for each movie | 4 | H | Done | R1 | I3 | Calvin | Movie selection |
| 207 | admin staff | delete a movie | I can stop selling a product | 2 | H | Done | R1 | I3 | Calvin | Movie selection |
| 208 | customer | view a movie trailer/detailed description | I can see if I am interested in the | 4 | L | Done | R1 | I2 | Calvin | Movie selection |
| 209 | customer | write a movie review | I can share my feelings/opinions | 4 | L | Done | R1 | I3 | Daniel | Movie selection |
| 210 | customer | rate a movie | other people can see good movies | 4 | L | Done | R1 | I3 | Sammy | Movie selection |
| 301 | customer | cancel my order | I can cancel my order before purchase. | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 302 | customer | update my order quantity | I can change my order quanttiy. | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 303 | customer | add a order | I can add movie order | 5 | H | Done | R1 | I2 | Balraj | Order management |
| 304 | customer | filter orders | I can search my order by date | 5 | M | Done | R1 | I3 | Balraj | Order Management |
| 401 | admin staff | validate users | user's accounts are protected | 8 | H | Done | R1 | I1 | Bowen | Access mangement |
| 402 | customer | check my access logs | I can know these log activities | 4 | M | Done | R1 | I1 | Bowen | Access mangement |
| 403 | customer | detele access log | I can manage my log record | 4 | M | Done | R1 | I1 | Bowen | Access mangement |
| 404 | admin staff | monitor application access logs | I can identify abnormal logins | 8 | H | Done | R1 | I1 | Vencle | Access mangement |
| 501 | admin staff | add a new user to accounts | I can add new user to database | 16 | H | Done | R1 | I1 | Sammy | User management |
| 502 | admin staff | list and view user accounts | I can check if user records are correct and appropriate | 4 | M | Done | R1 | I1 | Sammy | User management |
| 503 | admin staff | update/change user accounts | I can keep user records up-to-date | 8 | M | Done | R1 | I1 | Sammy | User management |
| 504 | admin staff | delete user account | I can remove a user | 4 | H | Done | R1 | I2 | Sammy | User management |
| 601 | customer | access the online application 24/7 | I always have access to movies | 4 | H | Done | R1 | I2 | Daniel | Performance |
| 602 | customer | log in within three minutes of inputting the correct credentials | I don't have to wait | 4 | L | Done | R1 | I3 | Daniel | Performance |
| 603 | customer | input a long username and password | I can properly input my details | 4 | H | Done | R1 | I3 | Daniel | Performance |
| 604 | customer/sales staff/ admin staff | store large amounts of data on the server | my information can be accessed | 8 | M | Done | R1 | I3 | Daniel | Performance |
| 701 | admin staff | secure private customer information | attackers can't steal information | 16 | H | Done | R1 | I1 | Daniel | Security |
| 702 | admin staff | restrict access to admin features | only staff can access sensitive | 8 | H | Done | R1 | I1 | Daniel | Security |

## 7.2 User Interface

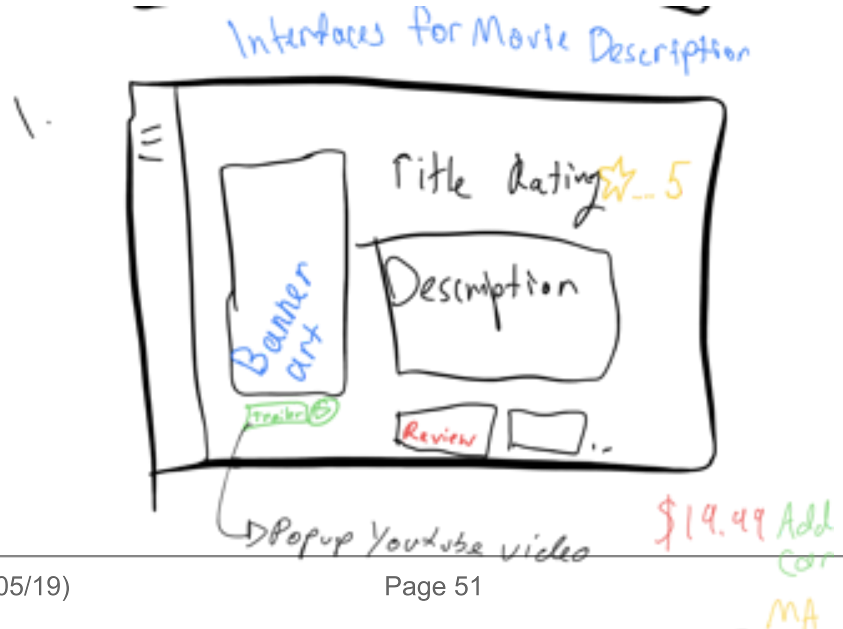
The following section are interface designs we developed during team meetings. The interface is broken down into components; particularly, the movie screen. The movie screen

(7.2.2) has embedded interfaces via a tab system of ‘Info’ (7.2.3), ‘Reviews’ (7.2.4) and ‘Trailer’ (7.2.5).

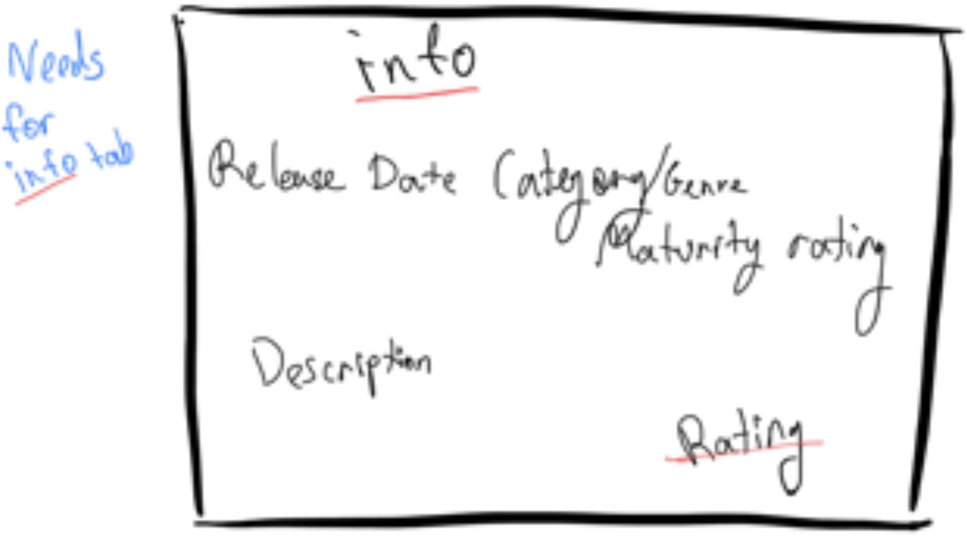
### 7.2.1 Home Screen

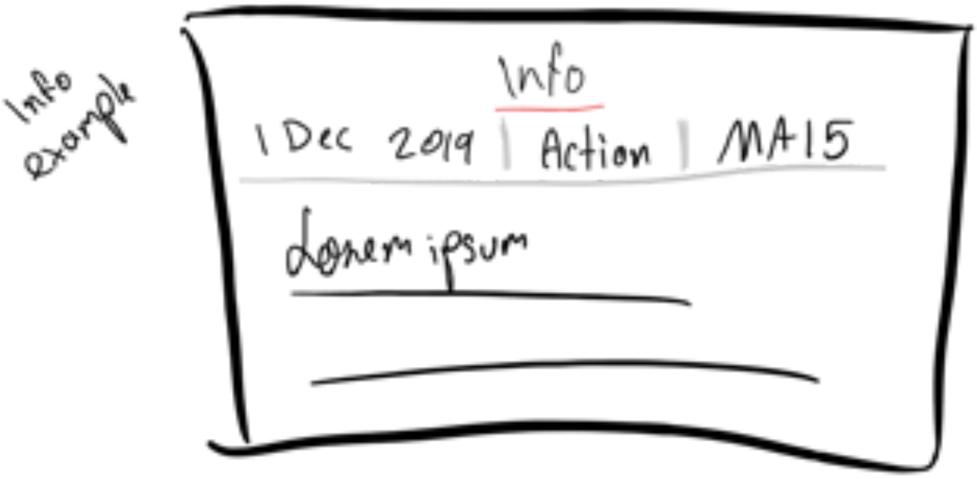


### 7.2.2 Movie Details

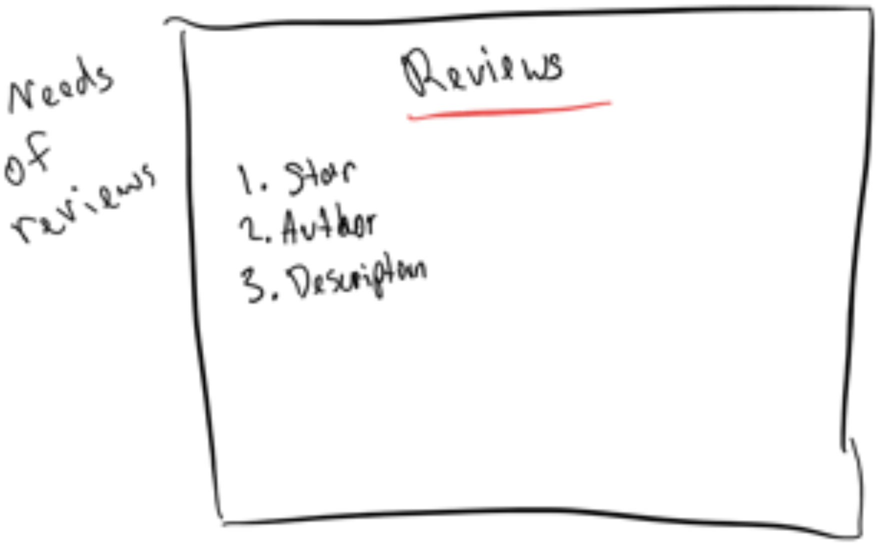


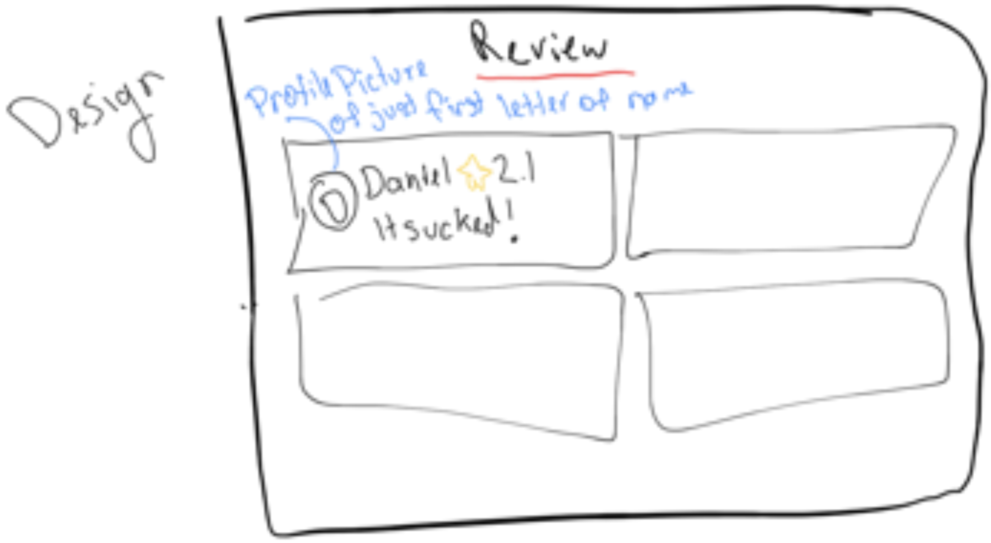
### 7.2.3 Movie Info Section



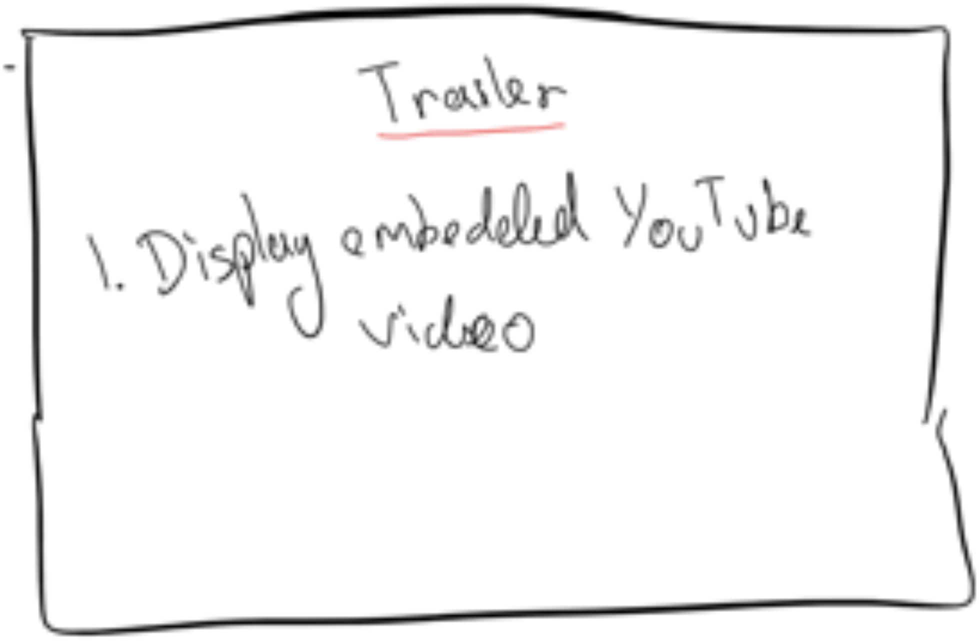


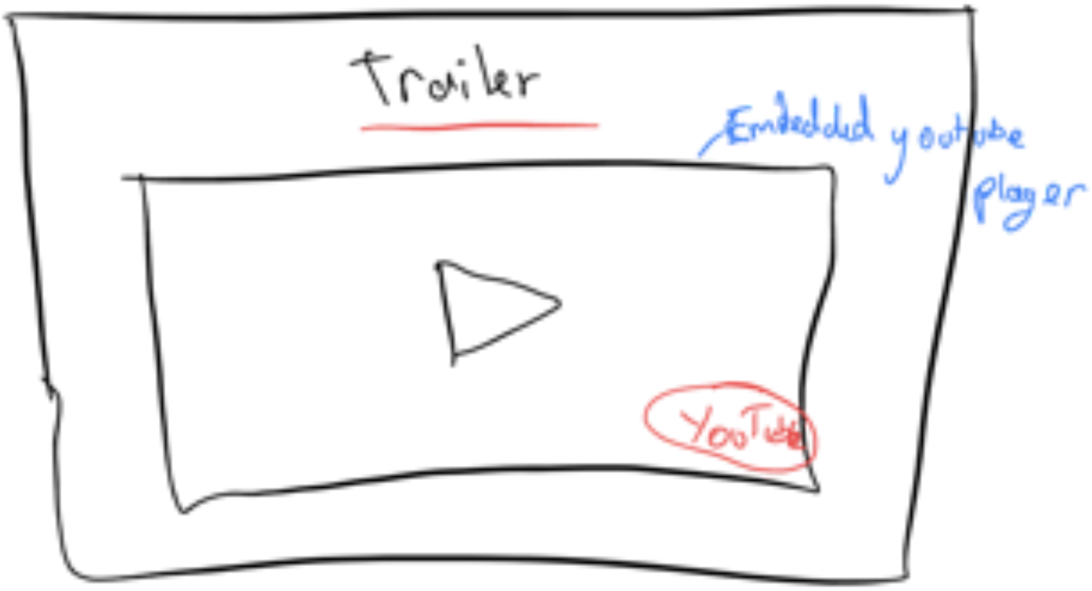
### 7.2.4 Review Section



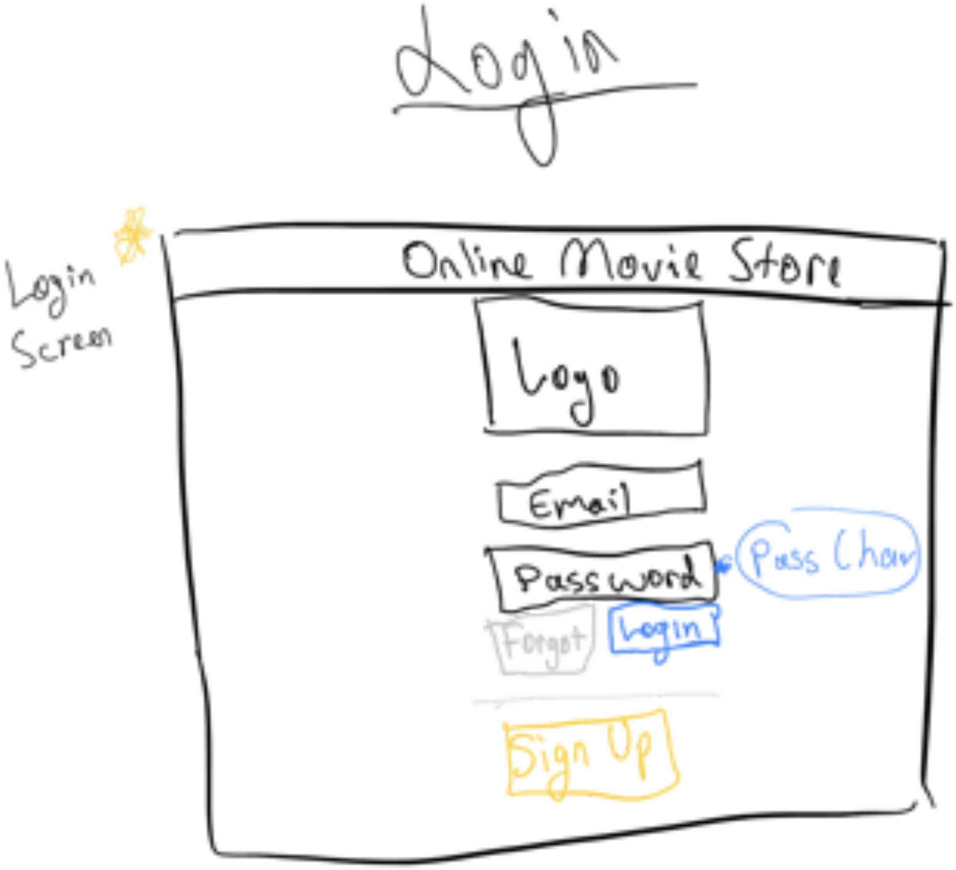


### 7.2.5 Trailer Section

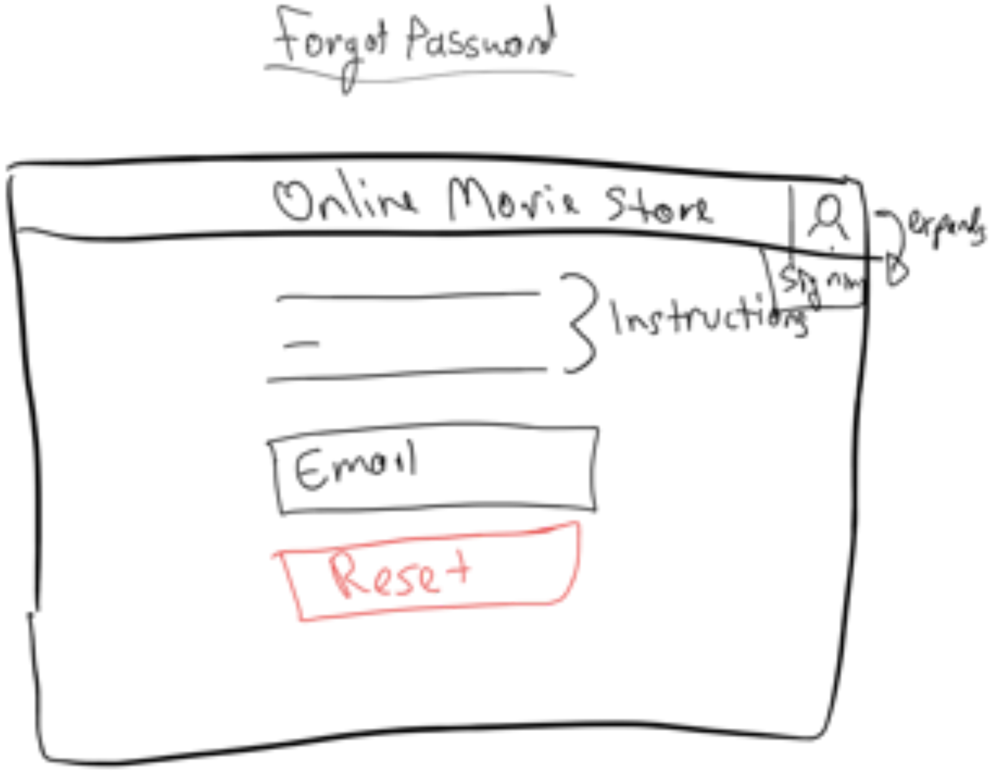




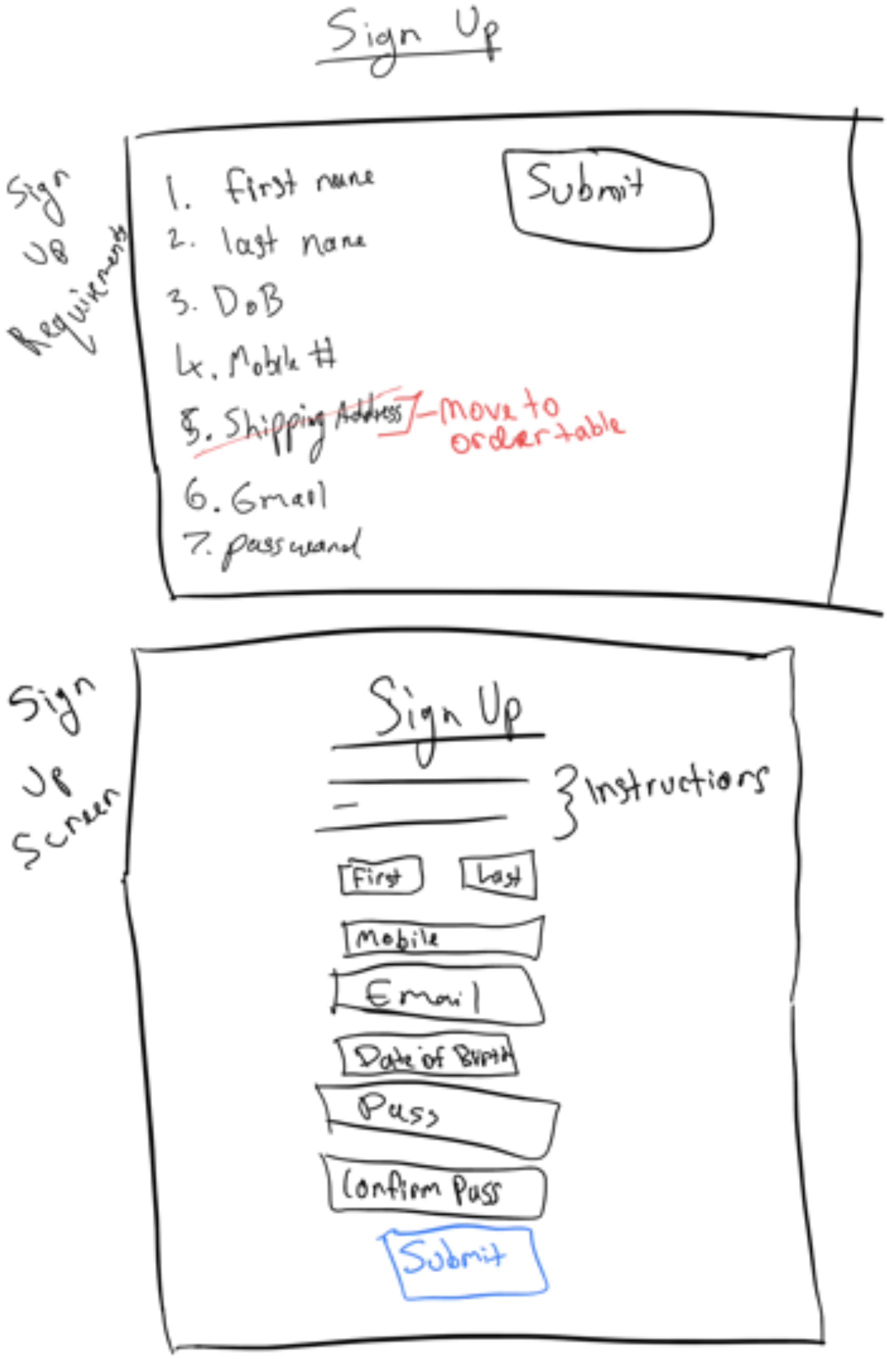
### 7.2.6 Login Screen



### 7.2.7 Forgot Password



### 7.2.8 Sign Up


Sign Up
Sign Up Requirements
1. first name
2. last name
3. DoB
4. Mobile #
~~5. Shipping Address~~
move to order table
6. Gmail
7. password
Submit
Sign Up Screen
Sign Up
Instructions
First
Last
Mobile
Email
Date of Birth
Pass
Confirm Pass
Submit

## 7.3 Log Books

### 7.3.1 Daniel Ebrahimian

- 14/3/19
    - Create GitHub project
    - Deploy GitHub project to netlify with CI
        - https://online-movie-store.netlify.com/
- 19/3/19
    - Discussed assignment with group members
    - Created Trello
    - Created Google Drive
    - Created Facebook chat
    - Created Slack chat
- 21/3/19
    - Worked on user stories
    - Created assignment 1 document
    - Performed code review of pull request
    - Modified and merged pull request
    - Researched more about django
- 23/3/19
    - Worked on data dictionary
    - Created entity-relationship-diagram
    - Update assignment 1 document to have more sections and add header numbering
    - Create logging in User Narrative
    - Review user stories and provide feedback
- 24/3/19
    - Write about different types of risks
- 26/3/19
    - Team meeting
    - Assign new tasks
    - Create interface drafts
    - Choose preferred interfaces with team
- 27/3/19
    - Work on Conceptual Data Model
    - Create Class Diagram
- 2/4/19
    - Code review of front end
    - Refactor card system
    - Merge changes
    - Team meeting
    - Discuss completed tasks
    - Fix deployment issue when redux dev tools is not available on users’ browser

- 6/4/19
    - Work on assignment documentation
- 9/4/19 - Tuesday
    - Team meeting
    - Discuss progress of tasks with members
    - Assign new tasks for next week
- 10/4/19 - Wednesday
    - Research Django
    - Work on Django backend
    - Work with sammy
    - Create database entities with sammy
- 13/4/19 - Saturday
    - Work on assignment documentation
- 16/4/19 - Tuesday
    - Work on assignment documentation
    - Work on backend
    - Host backend on azure
    - Work on front end
    - Modify database
- 17/4/19 - Wednesday
    - Work on diagrams
    - Work on API
    - Research more about Django
    - Work on assignment documentation
- 18/4/19 - Thursday
    - Create redirect proxy to pass HTTPS requests to HTTP backend
    - Deploy official backend to azure and use it in the live website
    - Stop using mock data in front end
    - Create genre api
    - Populate genre and review information of database junction tables via custom get request
    - Finalize release 0 and deploy to production
    - Clean up source control platform and create new workflow for release 1
- 14/5/19
    - Work on registration API
    - Work on change password API
    - Refactor database
- 15/5/19
    - Change database datatypes to be more type safe and efficent
    - Work on user details changer API
    - Provide meaningful status errors when API has a bad form request
    - Work on movie creation API
    - Work on genre API
    - Work on MovieGenre API
- 18/5/19

  - Work on review API
  - Refactor database migrations
  - Work on order API
  - Expand movie API return values to populate reviews and genres
  - Deploy expanded API data to production
  - Work on deleting user API
  - Work on deleting movie API
  - Work on change order API
  - Work on change user API
  - Deploy new backend to production
- 19/5/19
  - Work on returned error statuses when invalid API requests are made
- 20/5/19
  - Work on user logging
  - Make add movie screen mobile friendly with flex
  - Deploy user logging to production
- 22/5/19
  - Work on delete user logs API
- 23/5/19
  - Work on source control issues
- 25/5/19
  - Deploy current updates
  - Assist with movie interface design
- 26/5/19
  - Work on MovieGenre API
  - Work on api to view all genres related to a movie
  - Update production environment
- 27/5/19
  - Expose movie title in order api
  - Resolve repository issues related to dumping of .DS_Store files
  - Work on API to provide all orders that a user has made
- 28/5/19
  - Update production environment
  - Group meeting
  - Help group members
- 31/5/19
  - Identify production breaking bug
  - Lint all files
  - Assist group members

### 7.3.2 Sami Khalil

- 12/3/19
  - Formed group and disscused assignment plan
  - Watched and studied lecture materials

    - Discussing which tech stack to use

- 19/3/19
    - Worked on user stories
    - Collborate on Google drive, trello, slack and facebook
    - Researched on Django python backend
    - Created a data dictonary for the different objects of the application.

- 21/3/19
    - Watched tutorials on Django
    - Forked master repo
    - Created a basic django app setup and pushed to master

- 23/3/19
    - Added some risks

- 26/3/19
    - Team meeting
    - Discussed how the user interface should look
    - Sketch UI

- 27/3/19
    - Created Conceptual Data Model
    - Worked on Class Diagram

- 29/3/19
    - Started working on the UI of the application
    - Added Redux setup to the frontend

- 30/3/19
    - Completed Login and Register page
    - Decided to go with a dark theme layout
    - Took insprations from Netflix

- 31/3/19
    - Designed the home page with list of movies
    - Added static movies to give an idea of how the page might look like
    - Showed the team to get some feedback
    - Pushed to the master repo

- 2/4/19
    - Team meeting
    - Discussed completed tasks and incompleted tasks
    - Identified bugs on the live site
- 6/4/19

    - Tested out different materalise components
    - Drafted out the movie UI
- 7/4/19
    - Created the movie info, reviews and trailer layout
    - Added state within React to pass data in the components
- 9/4/19
    - Polished the UI, colours and animations
    - Showed the team
    - Pushed code to the live site
- 10/4/19
    - Work on creating the model for the database
    - Fix migration issues
- 11/4/19
    - Develop a simple REST API for users and movies
- 14/4/19
    - Work on backend and database
    - Modify tables
- 17/4/19
    - Research more about django and sqlite
    - Fix bugs
- 18/4/19
    - Continue to work on Movies API
    - Connect frontend to movies database
    - Created login and register interface
- 22/4/19
    - Populate movies interface with movies from API
- 27/4/19
    - Create a search function for movies
- 30/4/19
    - Create a wishlist/favourite list
- 2/5/19
    - Create redux state to store data from API locally and use it in different components
    - Move api calls to redux
    - Structure code
- 9/5/19
    - Create a custom header that will change depending in user is admin or not or is logged in
- 10/5/19
    - Work on change account details page
- 15/5/19
    - Develop user management interface
- 17/5/19
    - Dynaminally edit user details
    - Search user by phone or name

    - Add user via a pop up
- 20/5/19
    - Allow the site to be accessed via a user without registeration
    - Ask anonyoums user to provide details upon ordering
    - Allow user to register if they want to
- 25/5/19
    - Help team mates with their features
- 31/5/19
    - Refactor code
    - Work on report docs
    - doneee

## 7.3.3 Balraj Brar

- 19/03/2019

  Discussed assignment with group members<br>
  Worked on User Stories<br>
  Established communication through messenger, trello, google dox.
- 22/03/2019

  Worked on finalizing user stories and estimates and prioritization with Vencle<br>
  Created rough sketch of user story maps
- 23/03/2019

  Worked on User Story Map<br>
  Wrote Narrative<br>
  Researched on budget estimation

- 24/03/2019

  modify user story map<br>
  Uploading progress on team dox.

- 26/03/19

  Group Meeting<br>
  Worked on movie application model

- 29/03/2019

  planning UXD with Vencle<br>
  completing user stories

- 31/03/2019

  Discussed task to complete<br>
  Draft user experience design<br>
  completed user story map

- 02/04/2019
  Planned UXD with Vencle
  Worked on UXD design (homepage, login, register)

- 05/04/2019
  Worked on release 0 documentation

- 07/04/2019
  Project Documentation

- 09/04/2019
  Group Meet- Discussing and finalising online movie website interface

- 12/04/2019
  project constraint and customer journey map with Vencle

- 13/04/2019
  Changes in User Narrative
  Discussing Data Architecture Model diagram
  Adding content to Data Architecture Model

- 16/04/2019
  Group Interaction (Discussion about Document Finalisation)
  Working on Architecture Spike (Software Prototype)
  Appendices

- 17/04/2019
  Final edits on the document
- /04/2019
  Release 1 assignment discussion

- 23/04/2019
  Research work for project
  Github login

- 27/04/2019
  Research django framework

- 30/04/2019
  Research react, node.js

- 06/05/2019

System config.
Terminal setup

- 11/05/2019
  Project Initiation

- 12/05/2019
  Designing My orders page layout
- 14/05/2019
  Creating api -backend

- 16/05/2019
  Developing front end my order page

- 19/05/2019
  Creating interface for my order page
  Testing api in postman
- 21/05/2019
  Api order creating fields name, price etc
  Bug fixes

- 24/05/2019
  Front end- search movie

- 25/05/2019
  Front end- bug fixing - movie title display
  Add movie button

- 26/05/2019
  Front end - order update and delete feature
  Bug Fixes

- 28/05/2019
  Updating user stories and user story maps
  Creating customer journey map

- 31/05/2019
  Refractor code
  Final Update on assignment dox and Review

### 7.3.4 Calvin Thai

**Logbook for Calvin Thai**

12/03/19

- Studied the case study, lectures and assignment brief.
- Group discussions about roles, technology and assigning work.

13/03/19

- Researched and added user cases to the Requirements matrix and user narrative.

15/03/19

- Finalized View movie trailer/detailed description user narrative.
- Added user stories to the requirements matrix.

21/03/19

- Added non-functional requirements to the matrix table.

22/03/19

- Finalized Non-functional requirements to the matrix table.

24/03/19

- Research and drafted a card wall.

28/03/19

- Finalized card wall.

30/03/19

- Researched and drafted MVC diagram.

01/04/19

- Finalised MVC diagram.

04/04/19

- Filled owners column of the matrix table

06/04/19

- Fixed and improved the presentation of the report document.
- Filled missing sections of the report document.

12/04/19

- Added descriptions to diagrams, 3.2
- Fixed and improved the presentation of the report document.

14/04/19

- Updated card wall to include updated matrix fields.
- Modified MVC diagram.

15/04/19

- Updated the assignment document

16/04/19

- Finalize Assignment 1

07/05/19

- Research and install environments, IDE, Github

08/05/19

- Read Specifications and begin planning
- Learned JS React, redux framework and Django

13/05/19

- Studied existing code/progress

14/05/19

- Began front end programming

19/05/19

- Test and log errors

21/05/19

- Bugfixed movie front end component
- Added addmovie front end component

22/05/19

- Connected addmovie front end component with back end via actions
- Added genre support to addMovie

25/05/19

- Bugfixed addmovie
- Tested and logged errors
- Finalized addmovie
- Added updateMovie component

26/05/19

- Bugfix
- Tested and logged errors

27/05/19

- Connected updateMovie with backend via actions

28/05/19

- Bugfixed updateMovie

- Added genre support for updateMovie
- Fixed Styling

29/05/19

- Added delete movie function to updateMovie

30/05/19

- Bugfixed delete movie
- Tested and log errors
- Finalized updateMovie

31/05/19

- Updated and Finalize report

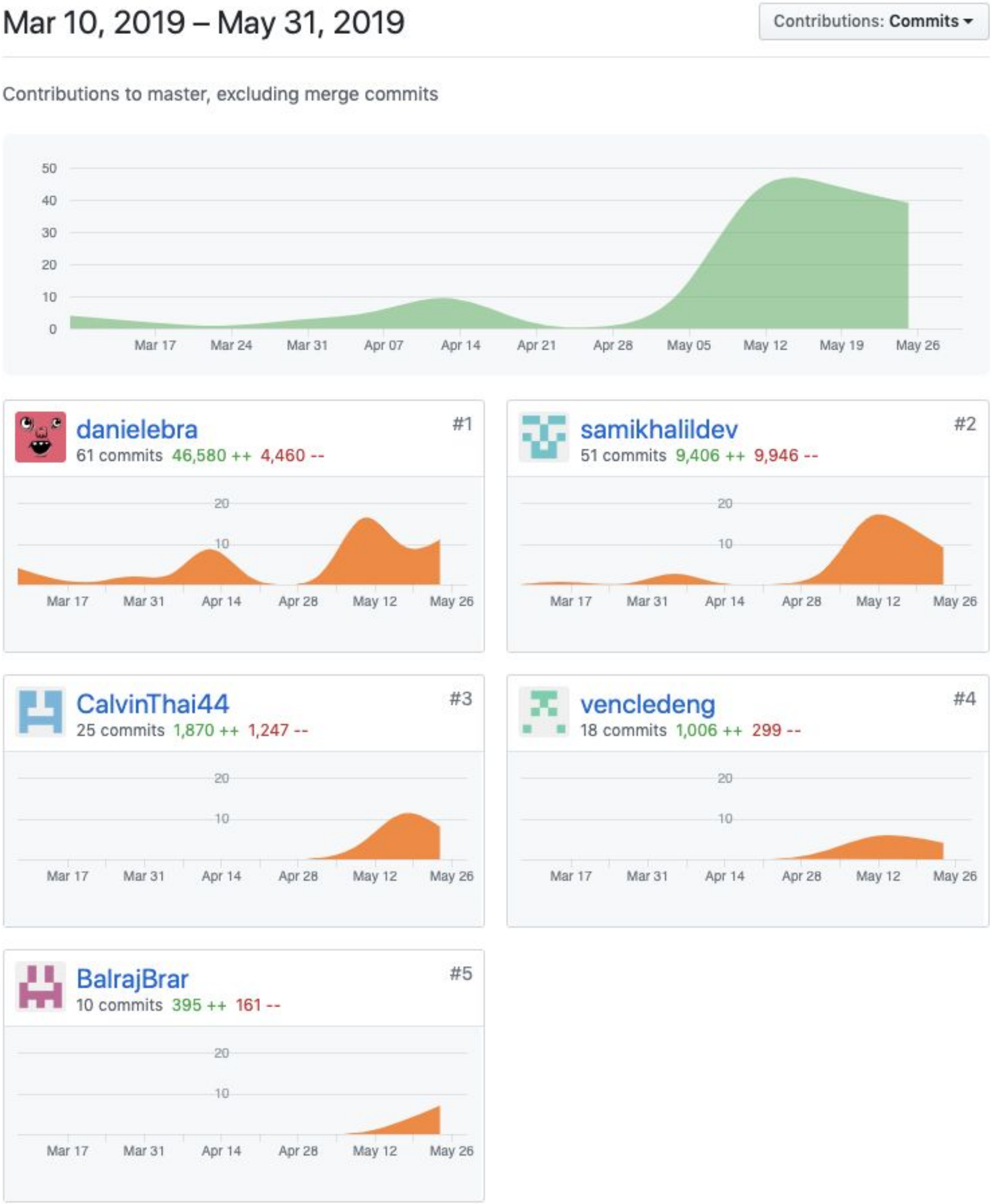
### 7.3.5 Bowen Deng

- 19/03/2019
    - Worked with group on requirement matrix
- 20/03/2019
    - Worked on user stories
    - Researched Django
- 22/03/2019
    - Worked on user stories estimate and prioritization with Balraj
    - Draw the user story map (not finished)
- 24/03/2019
    - Completed user stories map with Balraj
- 26/03/2019
    - Group meeting, got feedback from tutor
- 27/03/2019
    - Updated requirement matrix
    - Updated user story map
    - Completed project schedule with group
    - Completed cost estimate with Balraj
- 29/03/2019
    - Discussed the detail of User Experience Design with Balraj

- 02/04/2019
  - Planned User Experience Design with Balraj
  - Worked on User Experience Design (homepage, login page and register page) with Balraj
- 05/04/2019
  - Worked on release 0 documentation
- 09/04/2019
  - Met with group to management remaining work for release 0
- 10/04/2019
  - Planed customer journey design
- 12/04/2019
  - Worked on project constraint and customer journey map with Balraj
- 16/04/2019
  - Met with group to discuss about final editing of documentation and submission
- 30/04/2019
  - Project release 0 showcase
- 07/05/2019
  - Met with team to discuss about back-end implementation and distributed system features
- 08/05/2019
  - Researched authentication in Django platform
- 11/05/2019
  - Implemented account authentication on backend server
  - Tested back end authentication functionality
- 14/05/2019
  - Met with team and reported progress
  - Discussed the solution of Access Log management
- 15/05/2019
  - Worked on Access Logs page front-end UI

- 17/05/2019
    - Worked on Access Logs page front-end UI
- 18/05/2019
    - Completed Access Logs page front-end UI
- 20/05/2019
    - Connected front-end to back-end so that data is processed
    - Tested connection: user login and logout should generate log records and stored on backend server
- 21/05/2019
    - Deployed access log page to production
    - Met with team and discussed the progress
    - Discussed about some technique issues
- 22/05/2019
    - Implemented delete access log function
    - Tested for delete function
- 26/05/2019
    - Fixed bugs on access log management

## 7.5 GitHub Metrics



Mar 10, 2019 – May 31, 2019

Contributions: **Commits**

Contributions to master, excluding merge commits

danielebra #1
61 commits 46,580 ++ 4,460 --

samikhalildev #2
51 commits 9,406 ++ 9,946 --

CalvinThai44 #3
25 commits 1,870 ++ 1,247 --

vencledeng #4
18 commits 1,006 ++ 299 --

BalrajBrar #5
10 commits 395 ++ 161 --