*James Rehak*

*COSC 880 | Dr. Josh Dehlinger*

Department of Computer and Information Sciences

Towson University

Fall 2016

*My Shave of the Day*

Final Report

Table of Contents

[Introduction 3](#_Toc468724999)

[Objective / Scope 3](#_Toc468725000)

[Technology Used / Literature review 3](#_Toc468725001)

[Web Application 4](#_Toc468725002)

[Database 4](#_Toc468725003)

[Android 5](#_Toc468725004)

[Version Control 5](#_Toc468725005)

[IDE 5](#_Toc468725006)

[User Requirements 6](#_Toc468725007)

[User Characteristics 6](#_Toc468725008)

[Functional Requirements 6](#_Toc468725009)

[Non-functional 6](#_Toc468725010)

[Constraints 7](#_Toc468725011)

[System Design 7](#_Toc468725012)

[Architectural 7](#_Toc468725013)

[Data Design 8](#_Toc468725014)

[User Interface Design 10](#_Toc468725015)

[Implementation 10](#_Toc468725016)

[Use Cases 11](#_Toc468725017)

[1 - The user shall be able to create a shave record. 12](#_Toc468725018)

[2 - The user shall be able to track their shaving expenses 14](#_Toc468725019)

[3 - The user shall be able to browse their shaving history and can edit or delete it. 16](#_Toc468725020)

[4 - The user shall be able to share data, via social media platform 18](#_Toc468725021)

[5 - The user shall be able to create an item category. 20](#_Toc468725022)

[6 - The user shall be able to categorize and store shaving items in the ‘shave den’ database. 21](#_Toc468725023)

[7 - The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item. 24](#_Toc468725024)

[8 - The user shall be able to view and alter shaving categories in the database. 26](#_Toc468725025)

[9 - The user shall be able to track his frequency of usage of a selected item. 28](#_Toc468725026)

[10 - The user shall be able to register an account. 30](#_Toc468725027)

[11 - The user shall be able to login with a registered account. 32](#_Toc468725028)

[Screen Shots 33](#_Toc468725029)

[Test Cases 39](#_Toc468725030)

[Conclusion 40](#_Toc468725031)

[Challenges Faced / Lessons Learned 40](#_Toc468725032)

[Experience Retrospective 41](#_Toc468725033)

[Future work 42](#_Toc468725034)

[Proposal 42](#_Toc468725035)

[Project Description 45](#_Toc468725036)

[Works Cited 47](#_Toc468725037)

# Introduction

For many people, shaving is a fact of life. Some people see shaving as chore, whereas others see it as a ritual. Some shaves are quite simple, only involving an electric trimmer, other shaves involve a wide range of tools, soaps, and chemicals. Some people like to share the product of their shave, while others prefer to simply track their expenses. The ability to track one’s shaves and all the items required for shaving will help all shaving people get the most out of their experiences. Keeping an ongoing record allows shavers to see how much they are spending, what equipment works the best, and to see what shaves they liked the best. In this age of social media, the ability to share and show off what you are doing is ever important. Sharing one’s latest shave will be of great convenience to shaving enthusiasts and people that just want to show off their latest facial hair style.

## Objective / Scope

The primary objective of this project is to is to create an easily accessible application, called “My Shave of the Day”, that allows its users to track their shaving habits. Users will be able to store their shaves as records with photos, text descriptions, and ratings. By recording their shaving methods, the user can improve their future shaves though the analytics of their own methods. In addition to tracking shaves, the application will also serve as an inventory system for shaving equipment and accessories. Users will be able to keep track of items such as razors and shaving cream to see how much they are spending on shaving and to never be caught in a situation where they must shave with a dull razor. Any increase in efficiency, whether it be through managing expenses, deceasing shave time, or improving technique, improves the quality of life for the ‘my shave of the day’ user.

# Technology Used / Literature review

There are many different methods one could use to create “My Shave of the Day”, involving numerous different technologies. For my methodology, I wanted to use technology and techniques that were well documented and practiced by a large community. I wanted the technology I used to be open source as much as possible. Keeping those two aspects in mind, would allow me to be more successful and surmount any road blocks I might hit during development. Ultimately I decided to build “My Shave of the Day” as a web application with an android wrapper. In the following paragraphs, I will explain my reasoning for selecting each technology.

## Web Application

A web application is an application that is built using the client-server model. The application server accepts incoming requests from a client, using a browser or phone, processes those requests, and then sends back a response [1]. I decided to build mySOTD using Ruby on Rails. The Rails framework strictly adheres to the Model- View- Controller design pattern. The MVC framework allows for separation of application state, interaction state, and data state [2]. The system is structured into three logical components. The model manages data and data manipulation. The view manages the visual interface, and the controller passes interactions between the view and model.

Building the application as a rails web application gave me several advantages. The first was the design framework. I knew mySOTD would heavily involve Create, Read, Update, and Delete actions. The MVC design pattern allows me to easily handle those actions. Using this framework also guides my application to be more structured and therefore more maintainable. Another advantage to using a web application, is portability. By having the application logic be stored on a server, I can rapidly deploy it to new platforms with little to no compatibility issues. This saves a great amount of development time and allows the service to reach a much wider audience. A web application is accessible to multiple devices simultaneously, giving its users more opportunity for interaction. In addition to repaid deployment, building as a web application makes adding new functionality, not in the original requirements, much simpler. All the necessary infrastructure is present, it is just a matter of adding additional models, controllers, and views. All of those mentioned advantages are a boon for software maintainability, which is the greatest cost during the applications life cycle [2]. Another advantage to using Rails is the pleather of gems available to aid development. Using community created gems allows for more rapid development by not having to program certain aspects, such as user login encryption, from scratch.

To deploy this application, I decided to use Heroku. Heroku is a cloud platform that functions as a server and database. Heroku allows developers to focus on developing apps rather than dealing with infrastructure issues of setting up a local server manually. Heroku streamlines the processes of deploying, configuring, and scaling applications [3].

## Database

The decision to use Heroku required me to use PostgreSQL as the Relational Database Management system, as it is the only RDBMS it supports. I had prior experience using MySQL, SQLite, and PostgreSQL from previous classes. The PostgreSQL requirement was not a deal breaker for using Heroku as I found PostgreSQL to be a very reliable and fast RDBMS. For mySOTD, data integrity was of the upmost importance, as many of the entities have relations to each other. PostgreSQL even supports more features than MySQL [4]. In addition to the Heroku requirement, many rails developers seem to favor PostgreSQL, which makes trouble shooting issues mush easier. The database used in mySOTD is referred to as the Shave Den database throughout this report.

## Android

To keep the majority of my code base more maintainable, I decided to deploy the Android Application as a web client. While not necessary to wrap the web application into a native android application, as all functions can be accessed by the mobile browser, there are some advantages to doing this. The first one is visibility. Having the web application running in an android application, allows it to be seen in the Android Play Store. This increases visibility and exposure to new users. The second is instant access to the phone’s camera, rather than having to exit the application to upload a photo. Having the android application act as a client also gave the advantage of data protection. In the event a user were to drop their phone into a sink full of water during a shave, all of their shaving data would be safe at the remote database. The user would simply have to login again on their new phone to access their data. The third advantage was the option to keep all User Interface development in the web application repository. This allowed me to stay within the MVC framework and modify views without having to adjust the android code. I could also use Google Chrome’s developer tools to simulate running the UI on various android screen sizes to check for spacing and format issues. When testing the actual android application, I found it more efficient to test it on my phone rather than going through an emulator.

## Version Control

I decided to employ version control despite developing the application on a single machine. Version control is a system that keeps a record of changes to a project, so that specific versions can be recalled later [5]. My Shave of the Day is hosted on GitHub and Heroku. Both repositories allow versioning and roll back to previous states. I found it easier to use two different Version control systems to communicate my local repository to the remote ones. I used GIT for Heroku, as that was a requirement of Heroku, and Subversion (SVN) to push my code to GitHub. I initially only used GIT, but ran into tracking issues when pushing code to each remote repo. I found myself mainly using version control to maintain a remote backup of my project, rather than using it to create separate branches for new features. I felt confident, should anything happen to my machine, I would be able to get up and running again wherever I left off.

## IDE

For this project, I used RubyMine to develop the web application part of the project and Android Studio to build the Android web client. Both Integrated Developer Environments are based on the IntelliJ platform. Because both IDE’s are based on the same platform, it was very easy to switch between the two, as menus and windows were placed in the same location. I choose to use the IDEs because I was previously familiar with RubyMine and Android Studio is the recommended IDE to build android applications with now. I also appreciated that both IDEs contained version control integration as well as quick access to a fully operational terminal shell.

# User Requirements

## User Characteristics

The ‘my shave of the day’ user is expected to be a person who has mild to extreme interest in shaving. The user is expected to have a personal computer or android smart phone connected to the internet. The user is expected to have a casual amount of experience using such devices. The users will most likely be men with facial hair.

## Functional Requirements

The following are a list of functional requirements the user must be able to perform for the application to be successful. See section implementation section for detailed use cases.

1. The user shall be able to create a shaving record for SOTD.
2. The user shall be able to track their shaving expenses.
3. The user shall be able to browse his shaving history and can edit or delete it.
4. The user shall be able to share data with other shavers, via social media platform.
5. The user shall be able to create an item category.
6. The user shall be able to categorize and store shaving items in the ‘shave den’ database.
7. The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item.
8. The user shall be able to view and alter shaving categories in the database.
9. The user shall be able to track his frequency of usage of a selected item.
10. The user shall be able to register an account.
11. The user shall be able to login with a registered account.

## Non-functional

1. The system shall be able to support multiple users, concomitantly.
2. The system shall respond to user input with in two seconds.
3. The system shall contain the correct data for the user.
4. The system shall not allow a user to see another user’s data.
5. The system shall not have a shave record limit.
6. The system shall prevent the user from storing incorrect or incomplete data.
7. The system shall have full functionality in mobile view.

## Constraints

1. The system shall communicate over the internet using HTTP.
2. The system database shall use a PostgreSQL.
3. The system shall support the following web browsers: Google Chrome, Fire fox, Safari.
4. The system shall run through an Android Web client.
5. The system shall run on a web server and be hosted over the internet.
6. The system shall be written as a web application.

# System Design

## Architectural

My Shave of the Day follows the Mode View Controller design pattern. After analyzing the requirements, I created models for each entity that required CRUD operations. There are 6 models used in mySOTD.

1. User – model associated with user creation and operation.
2. Shaving\_record –model associated with the user’s shaves.
3. Item – model responsible for the user’s shaving related items.
4. Shaving\_item – model which acts as the connection between shaving\_records and items
5. Category – model which controls categories
6. Identity – model used to connect social media accounts to user.

There are 9 controllers that accept user requests, manipulate data, and send responses.

1. Application\_controller – the overall controller of the application.
2. Categories\_controller – CRUD operations for categories.
3. Expense\_controller – Responsible for summing operations of users’ shaving items.
4. Home\_controller – there to call the landing page when starting the application.
5. Items\_conroller – CRUD operations for items.
6. Omniauth\_callbacks\_controller – for completing user registrations using social media accounts.
7. Shaving\_records\_controller – CRUD for shaves.
8. Tweets\_controller – used to generate tweets.
9. User\_controller – CRUD and for users.

Not all controllers need their own views. The following views were created to generate the UI.

1. Shaving\_records – shows index and CRUD views.
2. Items – shows index and CRUD views.
3. Categories – shows index and CRUD views.
4. Expenses – Shows expense views.
5. Users – Show Update and Delete views
6. Devise – gem generated view for account registration and login.
7. Home – contains application landing page.
8. Layouts – generates the overall layout of the application that each view renders within.

## Data Design

One of the advantages of using Rails is the ability to use Object Relational Mapping. This allowed me to connect each model to each other without resorting to writing SQL statements to access the shave den database directly. The data design of mySOTD required that all the models have some relation to each other, see figure 1 to see all relationships. To allow users to sign in with social media, their social media credentials had to be stored in the identities model, which belong to the users model. In order to prevent users from seeing each other’s shaves, items, and categories. The user model has a “has\_many” relationship with each of those models. Items belong to a category and categories has many items. Connecting shaving records to items present an interesting challenge because it is a many-to-many relationship. I decided to connect them using the has many though relationship in rails. Shaving\_records has many items through shaving\_items and vice versa. This allowed me to created nested associations. For instance, a shaving record may have multiple items used, this allows the shave\_record controller show all available items that user has. As well as allow the database to retain what those items were when they save it.

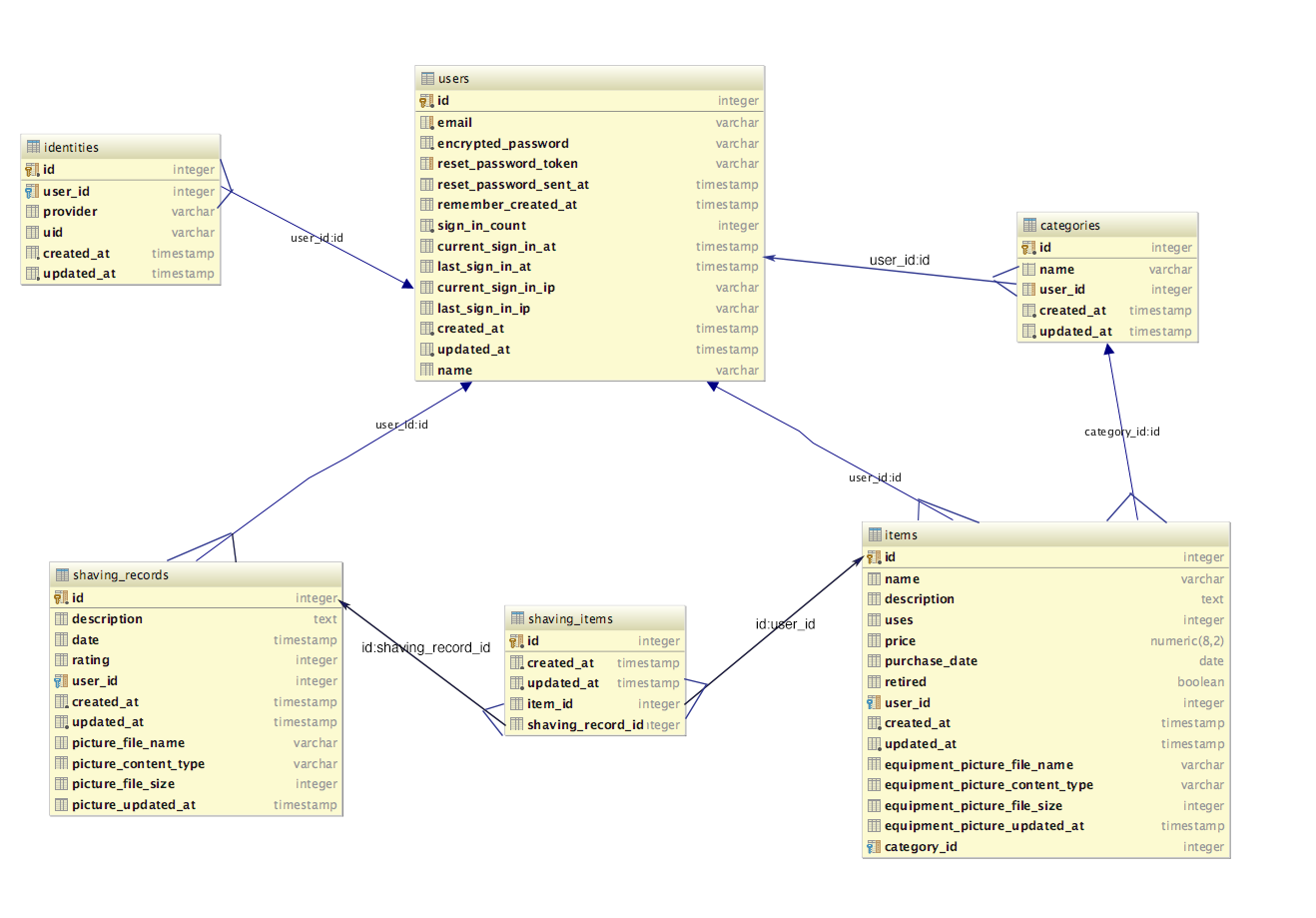


Figure Shave Den database

The following figure shows the textual PostgreSQL table structure of the Shave Den database. 

Figure Shave Den Database Schema

## User Interface Design

The best performing application will still be a failure if it has poor usability. To create the UI in an efficient and usable manner I replied on the Bootstrap framework. Bootstrap allows me to create a single front end code base that can scale to all sizes of screens easily. I initially used the twitter bootstrap gem, which allowed me to convert my views to bootstrap enable ones very seamlessly. To differentiate mySOTD from other twitter bootstrap using applications, I altered the bootstrap theme to the Paper theme found at <https://bootswatch.com/paper/bootstrap.css>. This allowed me to build an application that matched more closely with Google’s material design language, where all UI elements are supposed to be grounded in reality. Material design is inspired by print-based design. Surfaces and raised elements should cast shadows to show separation. The paper bootstrap theme allows me to create a design that has elements of print-based design. The biggest influence from material design was displaying all shaving and item records as cards rather than using a table. This gives the user a more tactile feel, like they are moving paper, when interacting with the application.

# Implementation

All of the Ruby source code used to create the mySOTD web application can be found at <https://github.com/jrehakTow/mysotd_sv.git>. The java source code to implement the android web app wrapper can be found at <https://github.com/jrehakTow/MyShaveAndroid.git>. The android web client implements android’s WebView extension of Android’s view class. Additional methods to handle activity triggers were required to give the web view client permission to access the phone’s camera and file system for the image upload requirement of mySOTD. This allows the android application to act as a mobile chrome web browser client for mySOTD.

The mySOTD application was deployed on a free Heroku Dyno and is accessible via all browsers. The following sections contain the implemented Use Cases as well as screen shots for My Shave of the Day.

## Use Cases

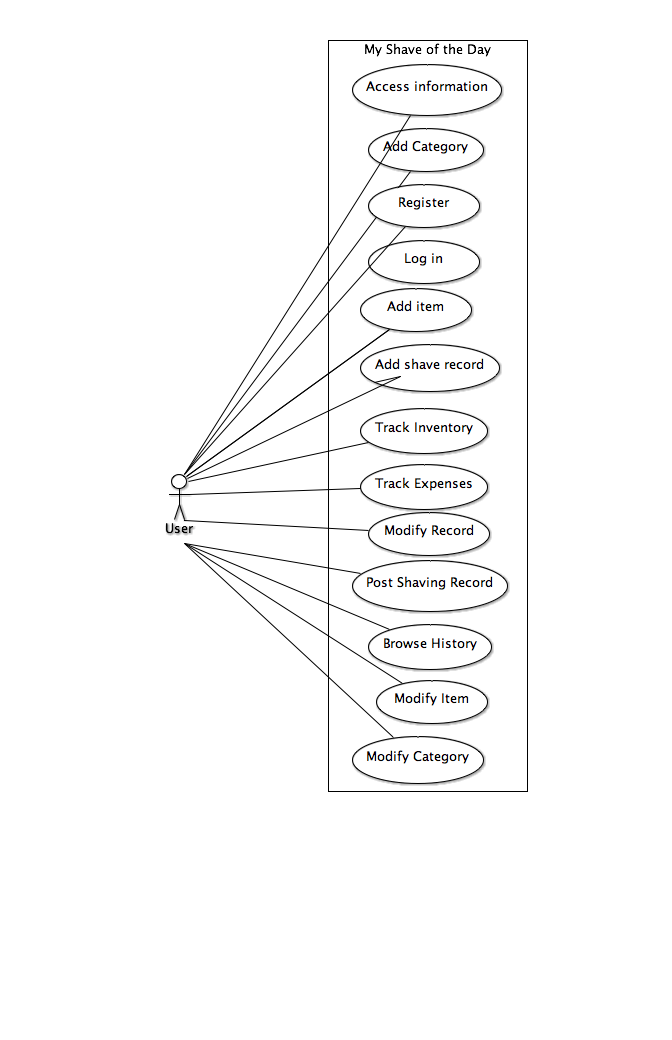
The following is a high-level use case for the “my shave of the day” application. 

Figure High-level use case for mySOTD

### 1 - The user shall be able to create a shave record.

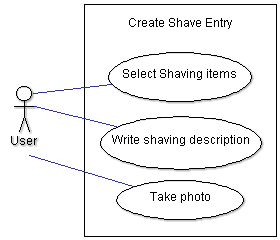


Figure Detailed Use Case Diagram, Create shave record

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Creation of shave of the day entry. |
| Trigger: | User presses “Create Shave” button |
| Preconditions: | 1. The user is registered to the shave den database |
| Postconditions: | 1. A shave record is created. |
| Normal Flow: | 1. The user presses the “Create Shave” button 2. The user is prompted to select shaving items from Shave den DB 3. The user selects items used in their shave    1. If no items exist in shave den user will be prompted to add items via shave den’s UI. 4. The user may write a text description of their shave in the provided text box 5. The user may press the add picture of shave button    1. The user will be prompted to take a photo or choose one from the phone’s default gallery    2. See alternative flow    3. User confirms selected picture 6. User presses “Finish” Button to complete and save shave record. |
| Alternative Flows: | (*The user does not have items in the shave den DB*)   1. The user is prompted to add an item to shave den 2. The user is changed to the “add item to shave den” interface 3. Items added by the user are added to the shave record 4. The user hits the “Done” button completing the Select items used in their shave step. 5. The normal flow is resumed   (*User chooses a picture from gallery*)   1. The user presses add picture of shave button 2. The user is prompted to take a photo or choose one from the phone’s default gallery 3. The user selects gallery 4. The phone’s native gallery photos are presented in a scrollable flow. 5. The user selects the photo of their shave. 6. The normal flow is resumed   (*User takes a picture*)   1. The user presses add picture of shave button 2. The user is prompted to take a photo or choose one from the phone’s default gallery 3. The user selects to take a photo 4. The phone’s default camera application opens. 5. The user takes a photo. 6. A confirmation or cancel choice is presented with the just taken phone, if the user selects cancel, revert to step 6. 7. The user confirms the photo 8. The photo is added to the shave record 9. The normal flow is resumed |
| Exceptions: | The user chooses not to select an item from the shave den |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As frequently as the user shaves |
| Business Rules: | Shave records shall not be shared to third party applications without permission from the user |
| Special Requirements: | None |
| Assumptions: | mySOTD has permission to use the phone’s camera  mySOTD has permission to access phone’s gallery pictures |
| Notes and Issues: | Selective permissions may prevent the photo functionality of creating a shave record |

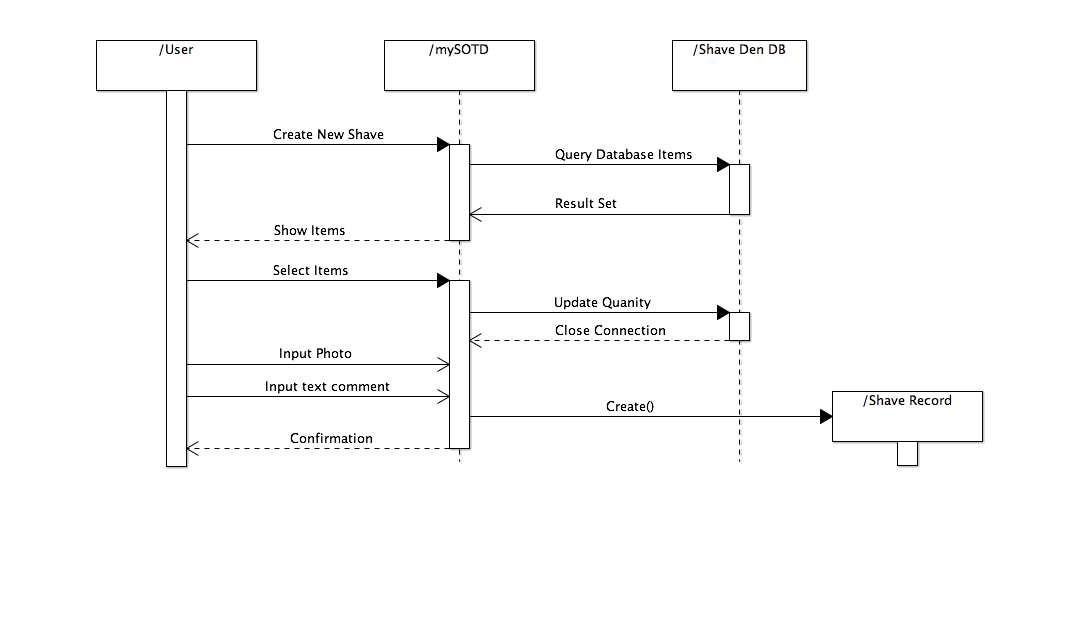


Figure Sequence Diagram, Create Shave Record

### 2 - The user shall be able to track their shaving expenses

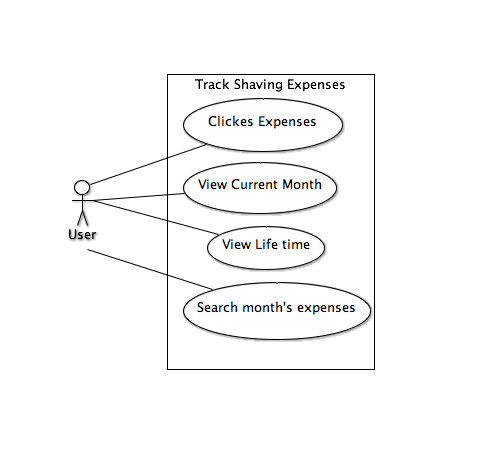


Figure Detailed Use Case, Track Shaving Expenses

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Tracking shaving expenses. |
| Trigger: | The user clicks on “Expenses” |
| Preconditions: | 1. The user is logged into his account. |
| Postconditions: | 1. The user monitors his expenses for shaving consumables. |
| Normal Flow: | 1. The user clicks on the “Expense” option on the interface. 2. The user see’s their current month’s expenses as well as lifetime expenses |
| Alternative Flows: | (the user views expenses of a particular month)   1. The user enters the month and year in the search field and clicks search. 2. The user sees their total expenses of that month. |
| Exceptions: | The user leaves a search field blank. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | Upon request of the user to track expenses. |
| Business Rules: | The user tracking data will be confidential and remain secure. |
| Special Requirements: | N/A |
| Assumptions: | * The user will be authenticated upon login. * mySOTD is will retain all of the user’s “Expense Tracking” data. |
| Notes and Issues: | N/A |

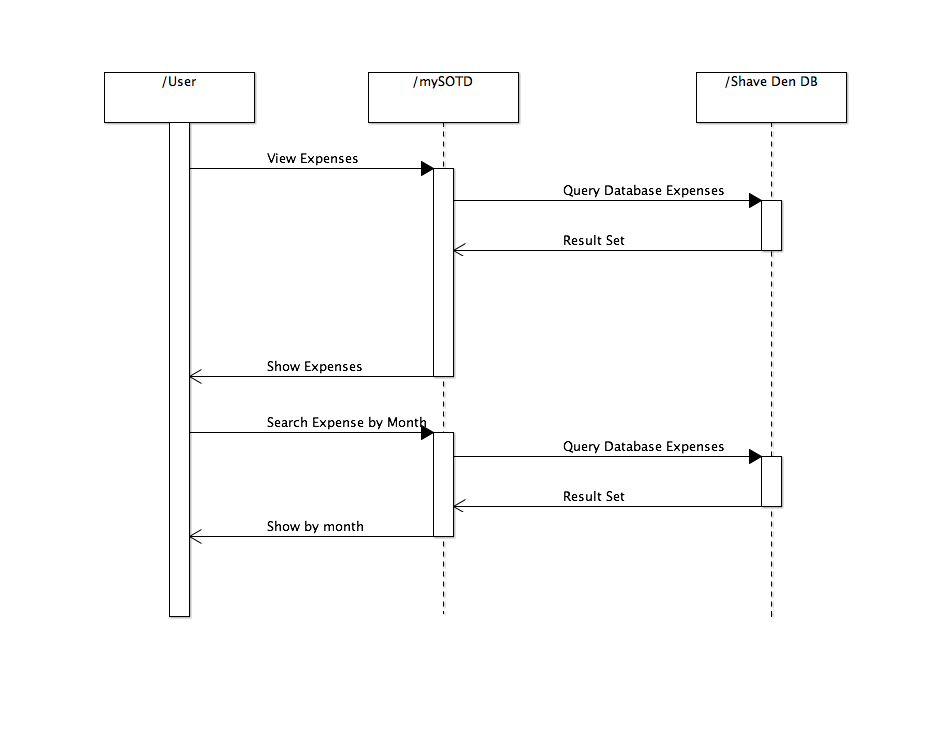


Figure Sequence Diagram, Track Expenses

### 3 - The user shall be able to browse their shaving history and can edit or delete it.

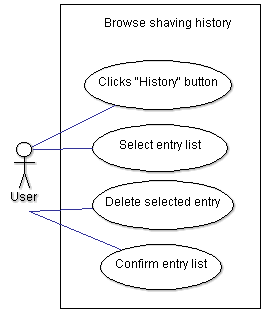


Figure Detailed Use Case, Browse and alter Shaving history

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Browse shaving history. |
| Trigger: | The user clicks ‘My Shaves’ tab. |
| Preconditions: | 1. The user is logged into his SOTD account. |
| Postconditions: | 1. The user knows his shaving history. |
| Normal Flow: | 1. The user clicks ‘My Shaves’ tab. 2. The user can browse his shaving history starting from his last entry. 3. The user will see a list of entries he made. Each entry consists of: a date, a name, and an image (if exist) of a product. 4. The user can select any entry to see more detailed information such as description of a product, comments about using a product, and a name of a social network (if any) where selected SOTD was shared. 5. The user can delete selected entry by clicking ‘Delete’ button. If deletion confirmed, the entry will be removed from the database. 6. The user can edit a selected entry by clicking ‘Edit’ button. The user can edit his old comments. The user saves changes by clicking ‘Save’ button. 7. The user can return to the home page by clicking ‘Home’ button. |
| Alternative Flows: | N/A |
| Exceptions: | My Shaves will be blank if no shave records exists. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As requested by user. |
| Business Rules: | The shaving history of the user shall not be shared without proper authorization. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall be registered. * The user shall be logged in. * The user should have shaving history in the Shave den database. |
| Notes and Issues: | N/A |

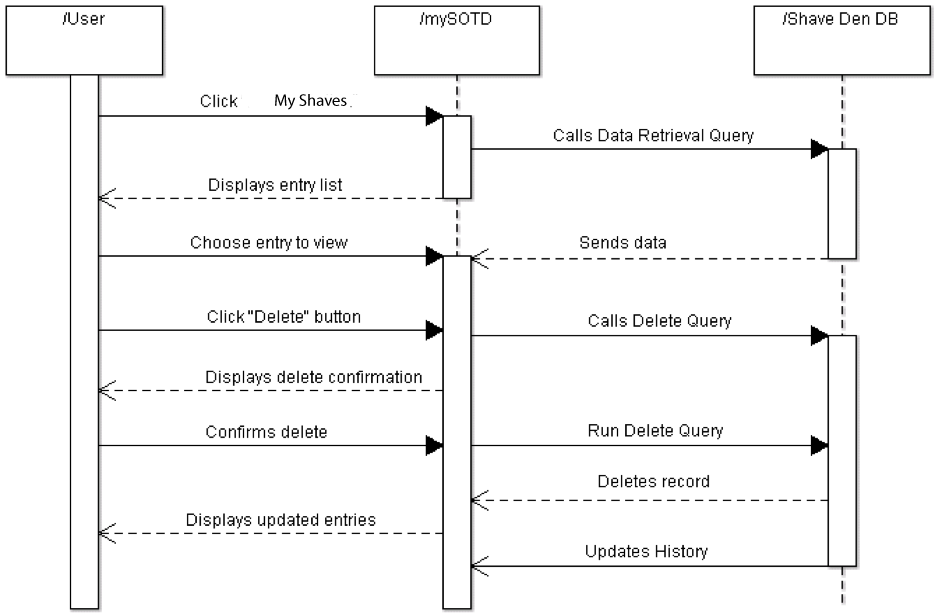


Figure Sequence Diagram, Browse and Alter shaving History

### 4 - The user shall be able to share data, via social media platform

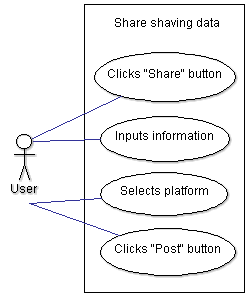


Figure Detailed Use Case, Share shaving data via social media

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Share data, via social media platform. (Twitter) |
| Trigger: | The user clicks on “Post”, under the “Share with Others” section. |
| Preconditions: | 1. The user has information to share with other shavers. |
| Postconditions: | 1. The user posts the shaving information on the platform. |
| Normal Flow: | 1. The user clicks on the shave record they want to share 2. The user clicks the “Post” option. 3. The user confirms “Post”, and clicks “OK”. 4. The information is shared on the social media platform. |
| Alternative Flows: | N/A |
| Exceptions: | The user is not logged in to twitter. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | Upon request of the user to share information with others, using a social media platform. |
| Business Rules: | The user’s information that is shared on a social media platform, or with other shavers, will be at the disclosure of the user. |
| Special Requirements: | N/A |
| Assumptions: | · The user will be authenticated upon login.  · The user will have not have sole access to shaving information that is shared with other shavers, via social media platform.  · mySOTD is will retain all of the user’s shared data, in the application history section. |
| Notes and Issues: | The user may delete a post or share, upon necessity. |

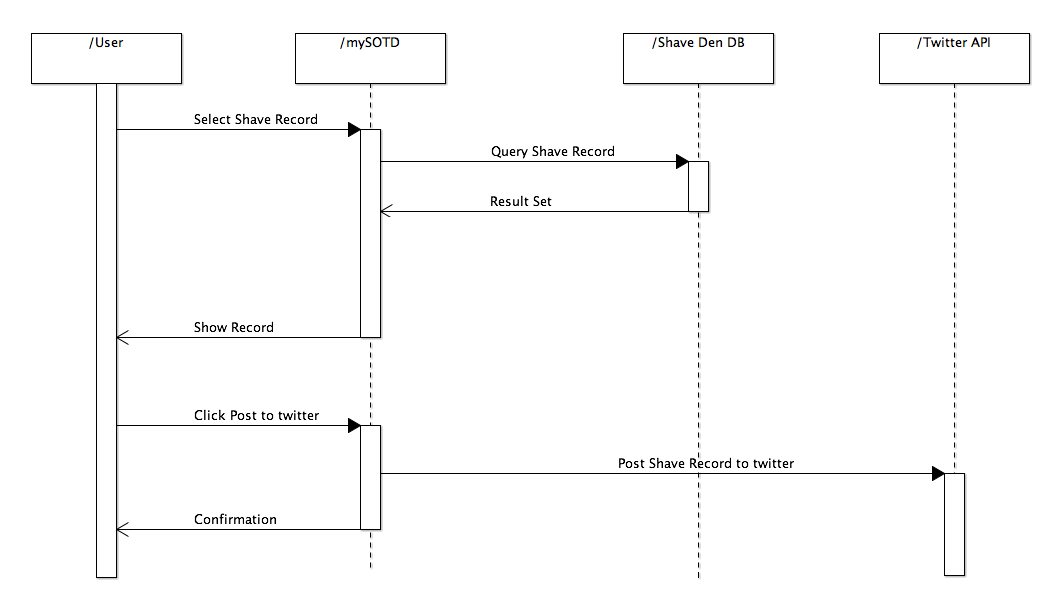


Figure Sequence Diagram of Share to social media

### 5 - The user shall be able to create an item category.

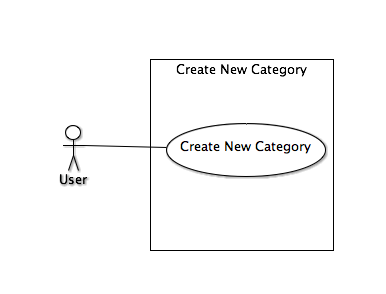
**

Figure Detailed Use Case, Create new category

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Creation of Category for items |
| Trigger: | User presses “New” button in category index view. |
| Preconditions: | 1. The user is registered to the shave den database and logged in. |
| Postconditions: | 1. A category is created |
| Normal Flow: | 1. The user selects “Categories” tab 2. The user clicks “new” button. 3. User inputs Category name and hits “Create Category” button |
| Alternative Flows: | N/A |
| Exceptions: | N/A |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As frequently as the user needs new categories. |
| Business Rules: | Categories shall not be shared with other users. |
| Special Requirements: | None |
| Assumptions: | User is logged in. |
| Notes and Issues: | N/A |

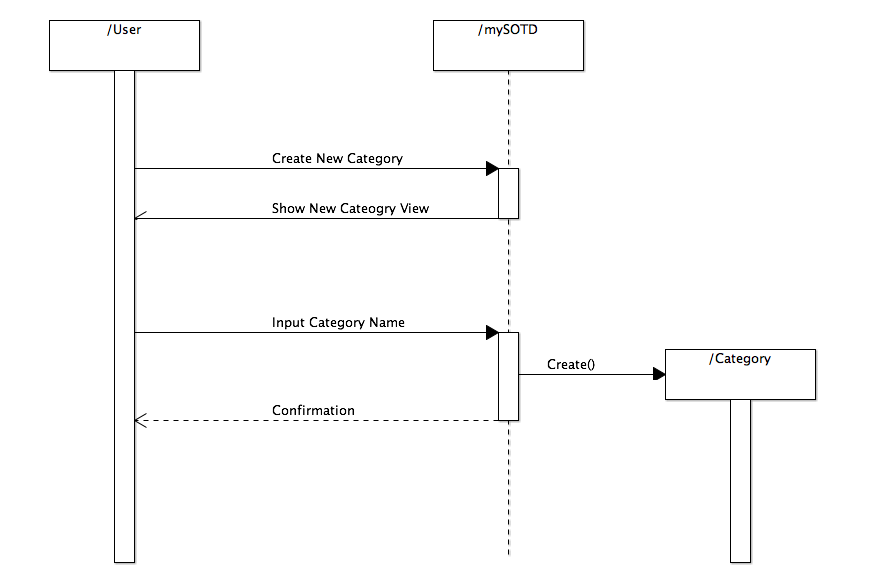


Figure Sequence Diagram, Create new category

### 6 - The user shall be able to categorize and store shaving items in the ‘shave den’ database.

**

Figure Detailed Use Case, Create item

|  |  |
| --- | --- |
| Actors: | User |
| Description: | Create New item |
| Trigger: | User presses “New” button in “My Items” index view. |
| Preconditions: | 1. The user is registered to the shave den database and in the “My Items” view |
| Postconditions: | 1. A shaving related item is created. |
| Normal Flow: | 1. User Selects “new” button 2. The user selects category their item belongs to. 3. The user may input a name, description, price, purchase date in provided fields. 4. The user may press the add picture of item button    1. The user will be prompted to take a photo or choose one from the phone’s default gallery    2. See alternative flow    3. User confirms selected picture 5. User presses “Create Item” Button to complete and save item. |
| Alternative Flows: | (*User chooses a picture from gallery*)   1. The user presses add picture of shave button 2. The user is prompted to take a photo or choose one from the phone’s default gallery 3. The user selects gallery 4. The phone’s native gallery photos are presented in a scrollable flow. 5. The user selects the photo of their shave. 6. The normal flow is resumed   (*User takes a picture*)   1. The user presses add picture of shave button 2. The user is prompted to take a photo or choose one from the phone’s default gallery 3. The user selects to take a photo 4. The phone’s default camera application opens. 5. The user takes a photo. 6. A confirmation or cancel choice is presented with the just taken phone, if the user selects cancel, revert to step 6. 7. The user confirms the photo 8. The photo is added to the shave record 9. The normal flow is resumed |
| Exceptions: | The user chooses not to select an item from the shave den |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As frequently as the user shaves |
| Business Rules: | Items shall not be shared to third party applications without permission from the user |
| Special Requirements: | None |
| Assumptions: | mySOTD has permission to use the phone’s camera  mySOTD has permission to access phone’s gallery pictures |
| Notes and Issues: | Selective permissions may prevent the photo functionality of creating an item. |

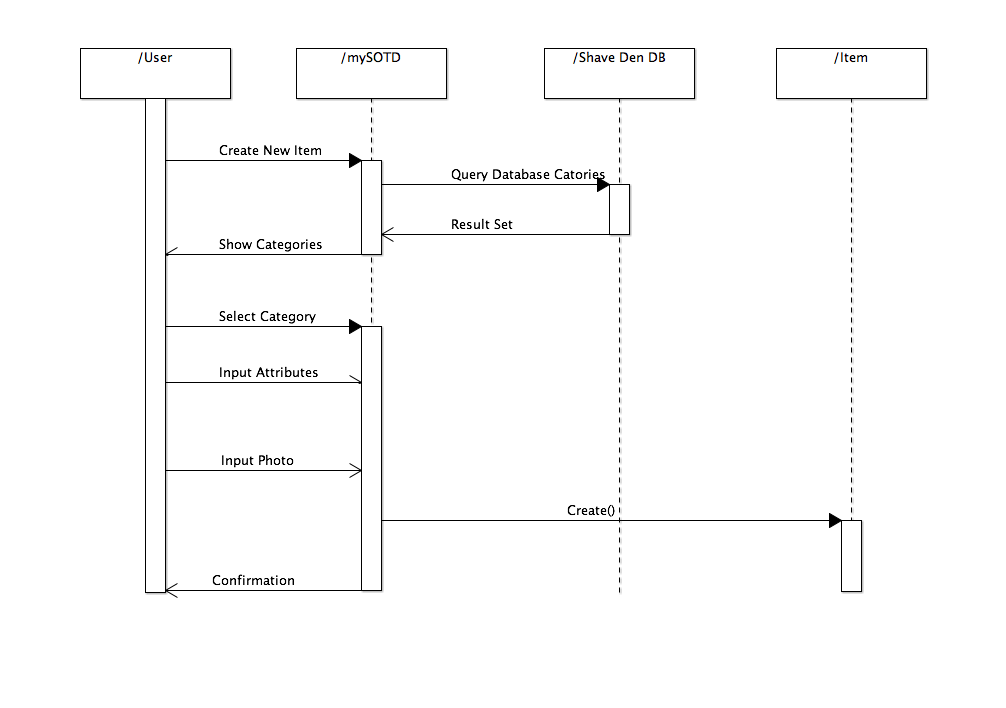


Figure Sequence Diagram, Create Item

### 7 - The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item.

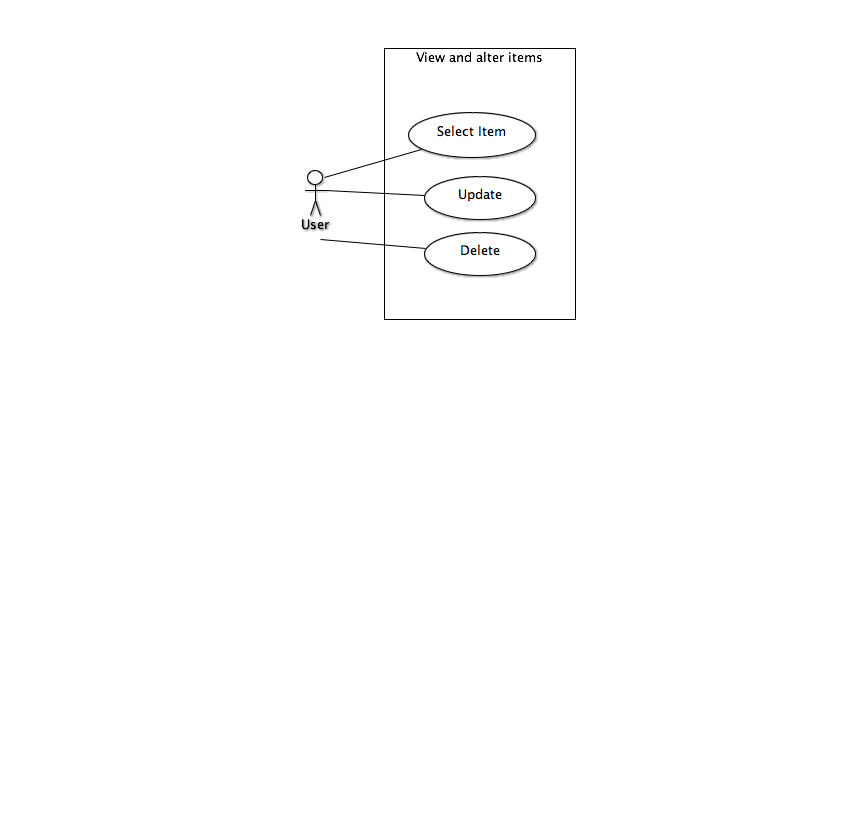
**

Figure Detailed Use Case, View and alter items

|  |  |
| --- | --- |
| Actors: | User |
| Description: | The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item. |
| Trigger: | The user clicks ‘My Items’ tab. |
| Preconditions: | 1. The user is logged into their mySOTD account. |
| Postconditions: | 1. The user knows their shaving inventory |
| Normal Flow: | 1. The user clicks ‘My Shaves’ tab.  2. The user can browse their item inventory starting from his last entry.  3. The user will see a list of item they created. Each entry consists of: name, purchase date, price, uses, if retired, category, and an image (if exist) of a product.  4. The user can select any item to see more detailed information such as description of a product.  5. The user can delete selected item by clicking ‘Delete’ button. If deletion confirmed, the entry will be removed from the database.  6. The user can edit a selected item by clicking ‘Edit’ button. The user saves changes by clicking ‘Update Item’ button.  7. All edit and delete actions will flash notice if successful or failed. |
| Alternative Flows: | N/A |
| Exceptions: | My Items will be blank if no shave records exists. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As requested by user. |
| Business Rules: | The shaving items of the user shall not be shared without proper authorization. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall be registered. * The user shall be logged in. * The user should have categories in the Shave den database. |
| Notes and Issues: | N/A |

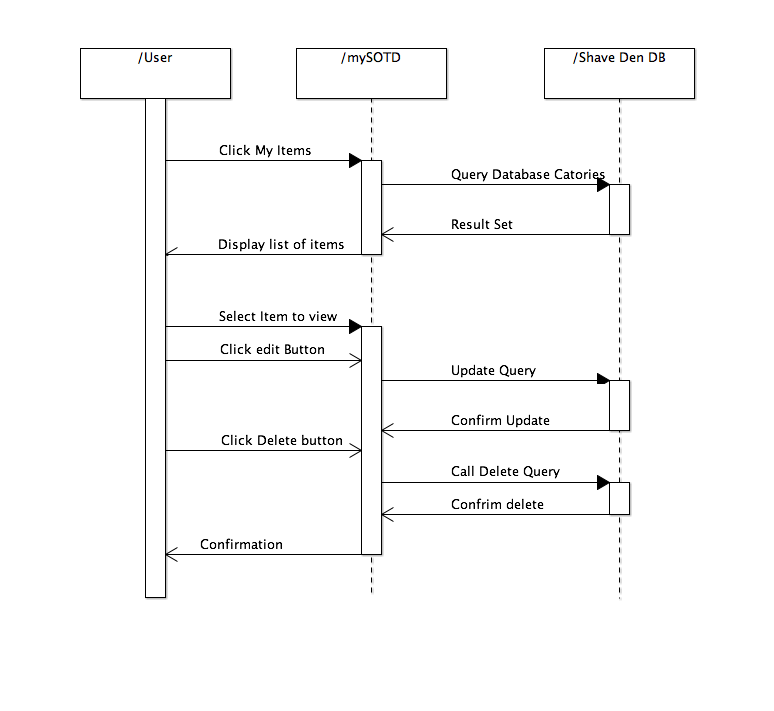
**

Figure Sequence Diagram, View and alter items

### 8 - The user shall be able to view and alter shaving categories in the database.

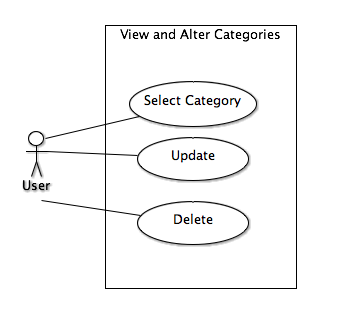
**

Figure Detailed Use Case, View and alter categories

|  |  |
| --- | --- |
| Actors: | User |
| Description: | The user shall be able to view categories from the ‘shave den’ database and can edit of delete the selected category. |
| Trigger: | The user clicks ‘Categories’ tab. |
| Preconditions: | 1. The user is logged into their mySOTD account. |
| Postconditions: | 1. The user knows their available categories. |
| Normal Flow: | 1. The user clicks ‘Categories’ tab.  2. The user can browse a list of their categories.  3. Each category consists of a name.  4. The user can select any category  5. The user can delete selected category by clicking ‘Delete’ button. If deletion confirmed, the category will be removed from the database.  6. The user can edit a selected category by clicking ‘Edit’ button. The user saves changes by clicking ‘Update Category’ button.  7. All edit and delete actions will flash notice if successful or failed. |
| Alternative Flows: | N/A |
| Exceptions: | My Items will be blank if no shave records exists. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As requested by user. |
| Business Rules: | The catgories of the user shall not be shared without proper authorization. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall be registered. * The user shall be logged in. |
| Notes and Issues: | N/A |

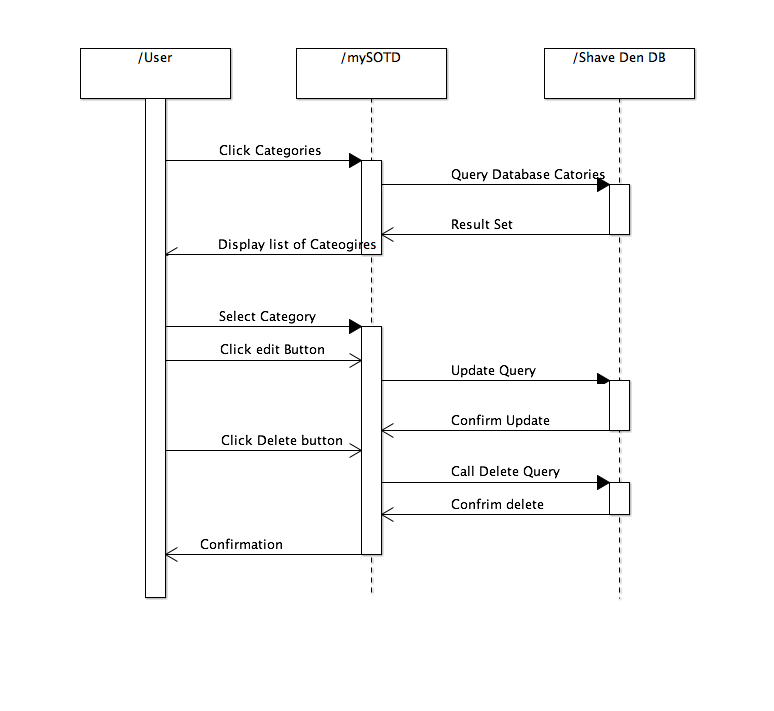
**

Figure Sequence Diagram, View and alter categories

### 9 - The user shall be able to track his frequency of usage of a selected item.

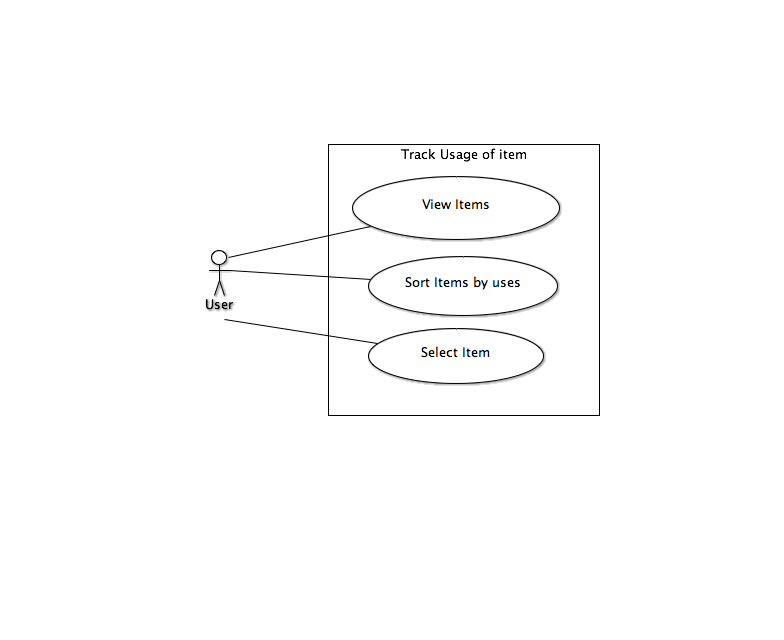
**

Figure Detailed Use Case, Track item usage

|  |  |
| --- | --- |
| Actors: | User |
| Description: | The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item. |
| Trigger: | The user clicks ‘My Items’ tab. |
| Preconditions: | 1. The user is logged into their mySOTD account. |
| Postconditions: | 1. The user knows their shaving inventory usage. |
| Normal Flow: | 1. The user clicks ‘My Shaves’ tab. 2. The user can browse their item inventory starting from his last entry. 3. The user will see a list of item they created. Each entry consists of: name, purchase date, price, uses, if retired, category, and an image (if exist) of a product. 4. The user can select any item to see more detailed information such as description of a product. 5. The user clicks the sort button ‘Uses’, the items are displayed in order of most uses to least uses. |
| Alternative Flows: | 1. The user clicks the sort button ‘Uses’ again, sorting each item by least to most uses. |
| Exceptions: | My Items will be blank if no shave records exists. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | As requested by user. |
| Business Rules: | The shaving items of the user shall not be shared without proper authorization. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall be registered. * The user shall be logged in. * The user should have categories in the Shave den database. |
| Notes and Issues: | N/A |

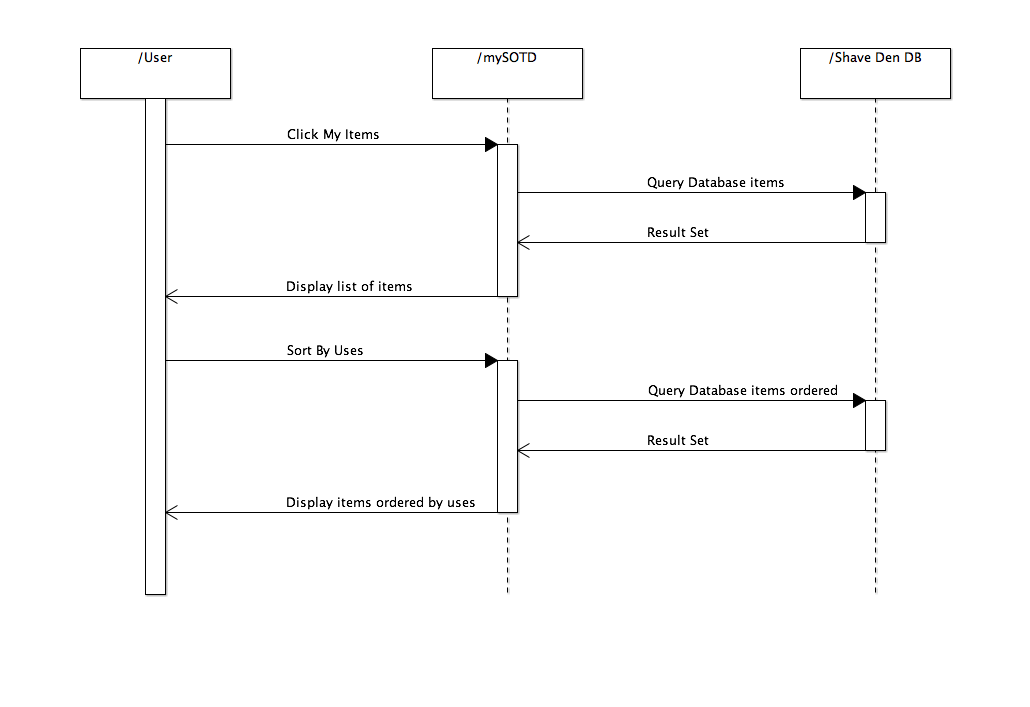
**

Figure Sequence Diagram, Track item usage

### 10 - The user shall be able to register an account.

**

Figure Detailed Use Case Diagram, Register Account

|  |  |
| --- | --- |
| Actors: | User |
| Description: | The user shall be able to register an account with my shave of the day. |
| Trigger: | The user clicks ‘Login’ button. |
| Preconditions: | 1. The user has no mySOTD account. 2. The user is not signed in. 3. User has twitter account for alternative flow. |
| Postconditions: | 1. The user has a mySOTD account. |
| Normal Flow: | 1. The user clicks ‘Login’ Button. 2. The user clicks the ‘Sign up’ link 3. The user enters in their email and chosen password. 4. User signed in and redirected to home page. Flash notice successful sign up. |
| Alternative Flows: | 1. The user clicks ‘Sign in with Twitter’ link. 2. User redirected to twitter sign in page. 3. User enters their twitter credentials. 4. User redirected to mySOTD, add email page. 5. User enters an email. 6. User signed in and redirected to home page. Flash notice successful sign up. |
| Exceptions: | * The user uses existing credentials stored in database. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | Once per user. |
| Business Rules: | The users password shall be encrypted. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall not be registered. * The user shall not be logged in. * Not use identical credentials to an existing account. |
| Notes and Issues: | N/A |

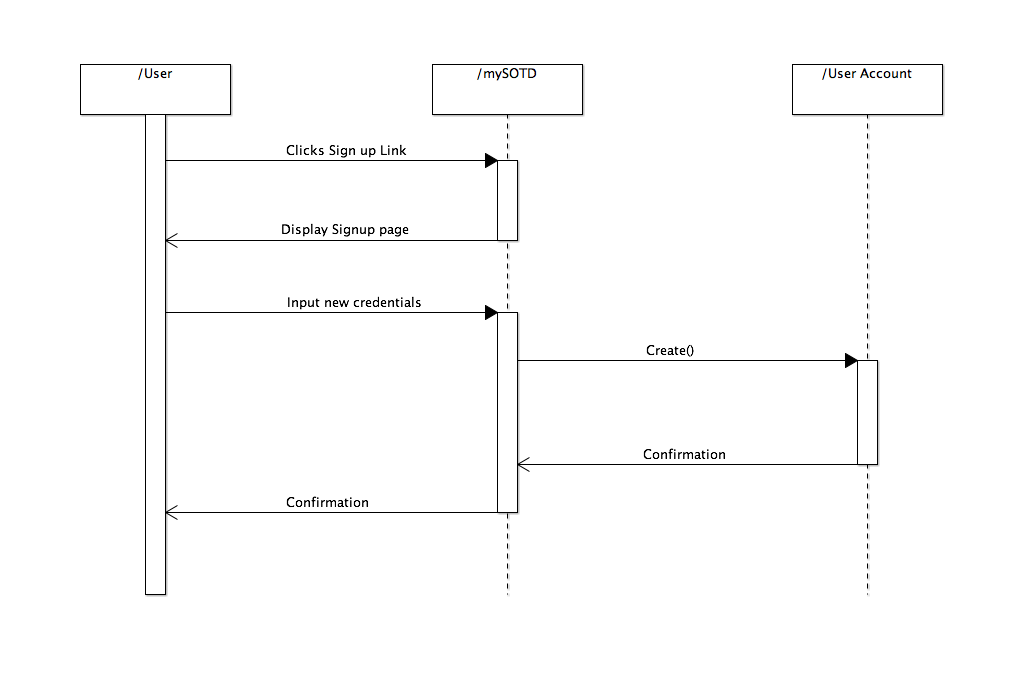
**

Figure Sequence Diagram, Register new user.

### 11 - The user shall be able to login with a registered account.

**

Figure Detailed Use Case Diagram, Login

|  |  |
| --- | --- |
| Actors: | User |
| Description: | The user shall be able to login with a registered account.t |
| Trigger: | The user clicks ‘Login’ button. |
| Preconditions: | 1. The user has a mySOTD account. 2. The user is not signed in. 3. User has twitter account for alternative flow. |
| Postconditions: | 1. The user has a mySOTD account. |
| Normal Flow: | 1. The user clicks ‘Login’ Button. 2. The user enters in their email and chosen password. 3. User signed in and redirected to home page. Flash notice successful sign up. |
| Alternative Flows: | 1. The user clicks ‘Sign in with Twitter’ link. 2. User redirected to twitter sign in page. 3. User enters their twitter credentials. 4. User signed in and redirected to home page. Flash notice successful sign up. |
| Exceptions: | My Items will be blank if no shave records exists. |
| Includes: | N/A |
| Priority: | High |
| Frequency of Use: | Only when session does not exist. |
| Business Rules: | The users password shall be encrypted. |
| Special Requirements: | N/A |
| Assumptions: | * The user shall be registered. * The user shall not be logged in. |
| Notes and Issues: | N/A |

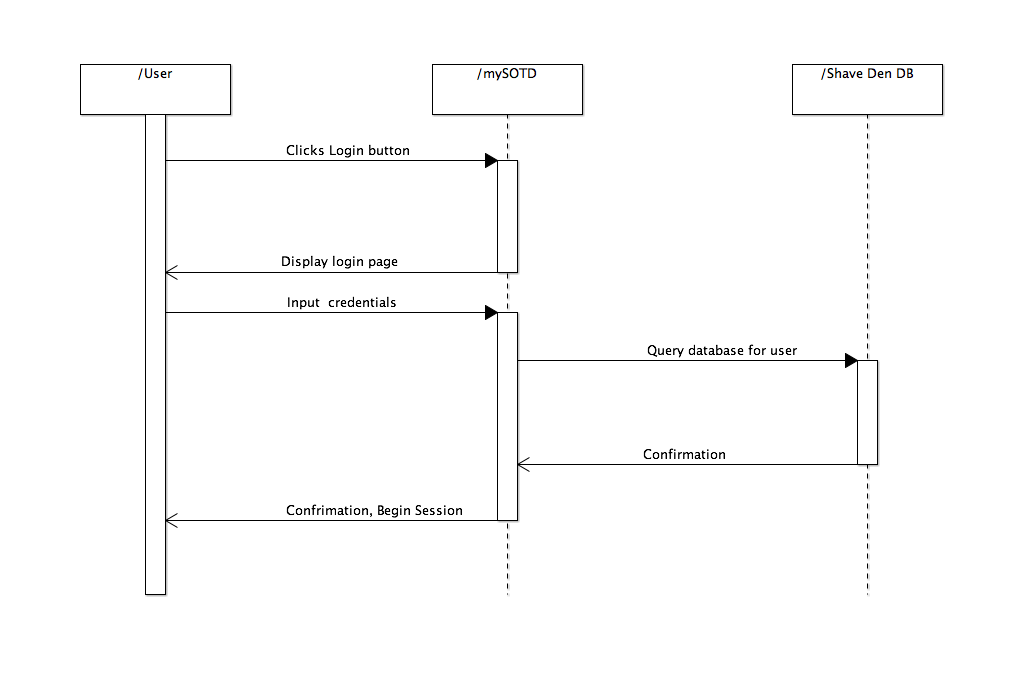
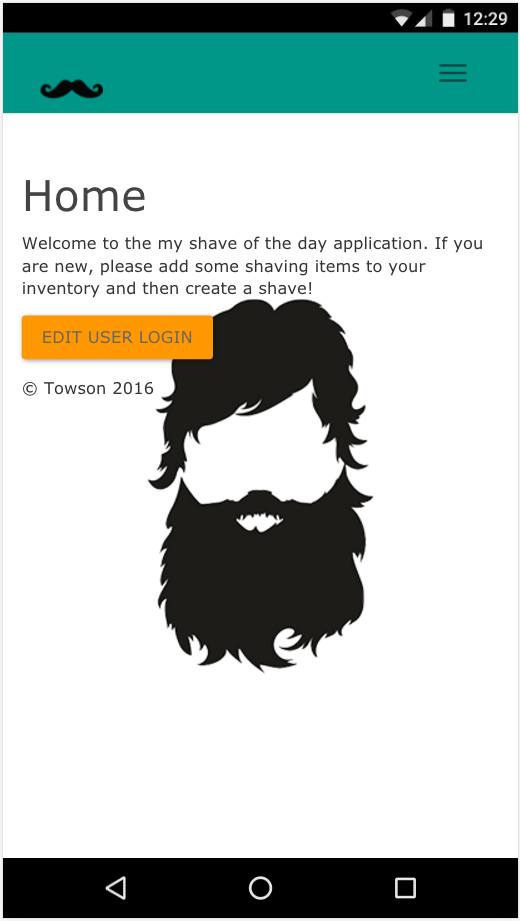
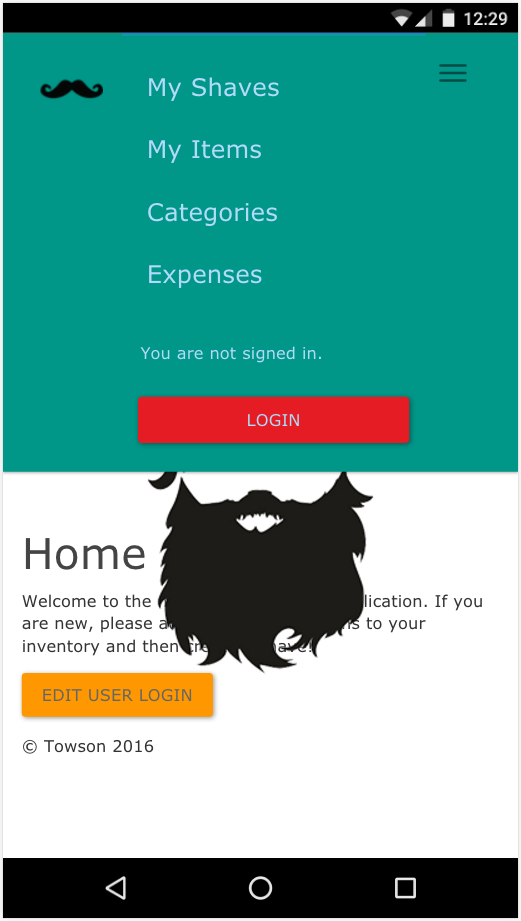
**

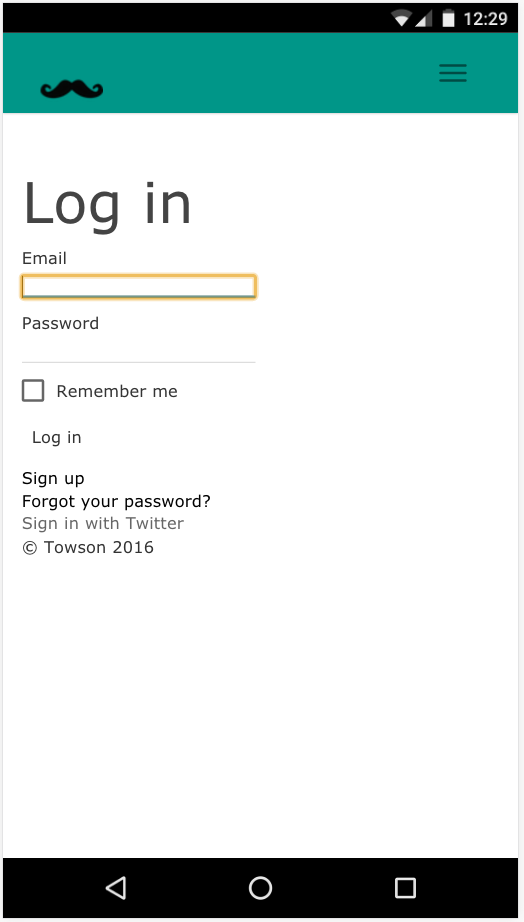
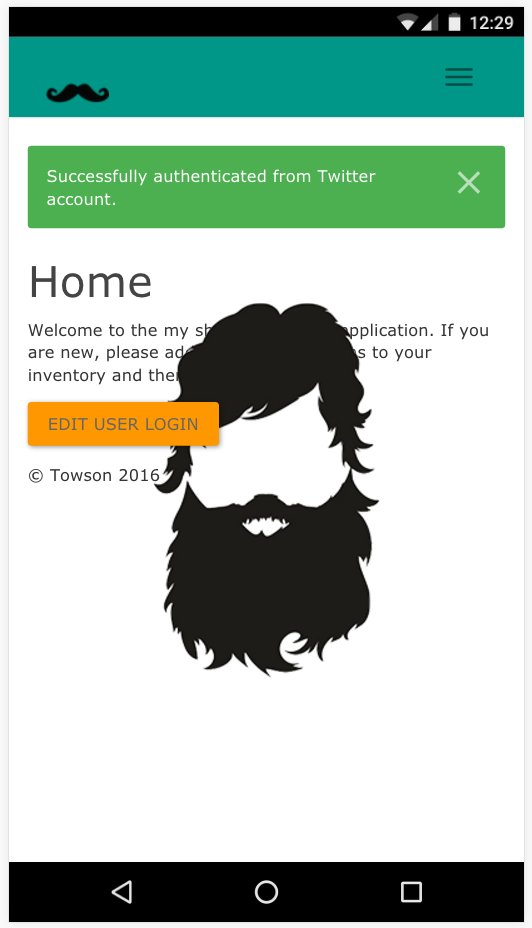
Figure Sequence Diagram, Login

## Screen Shots

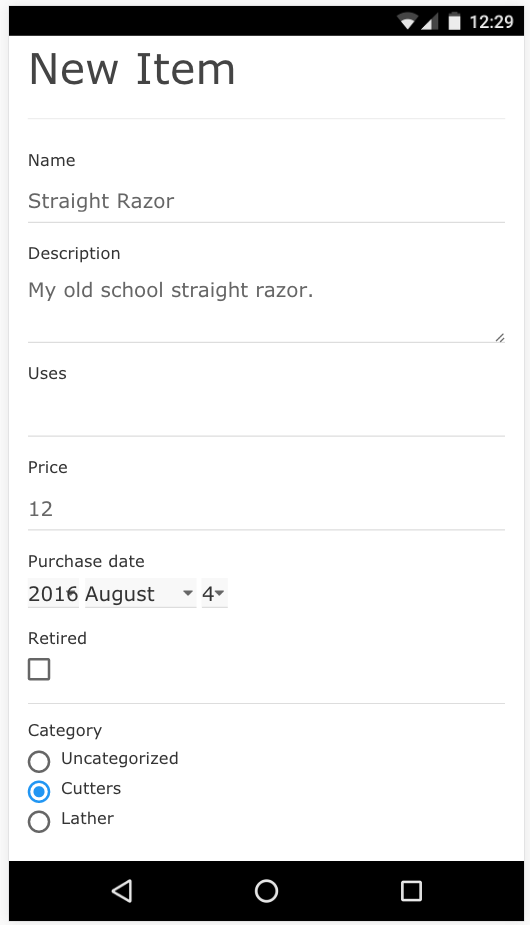
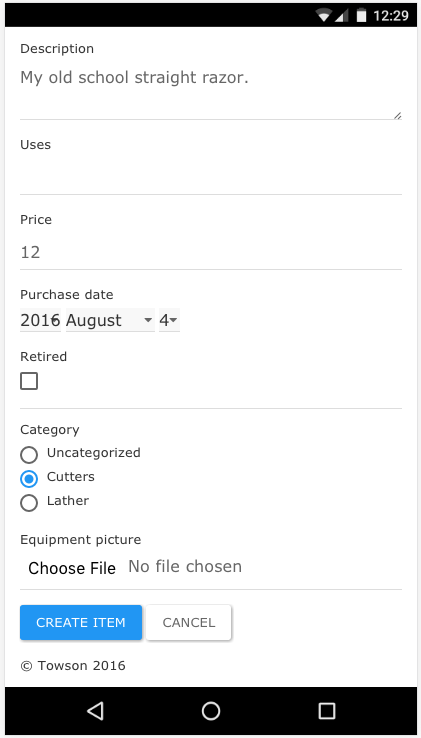
The following images are screen shots of the My Shave of the Day application in various states.

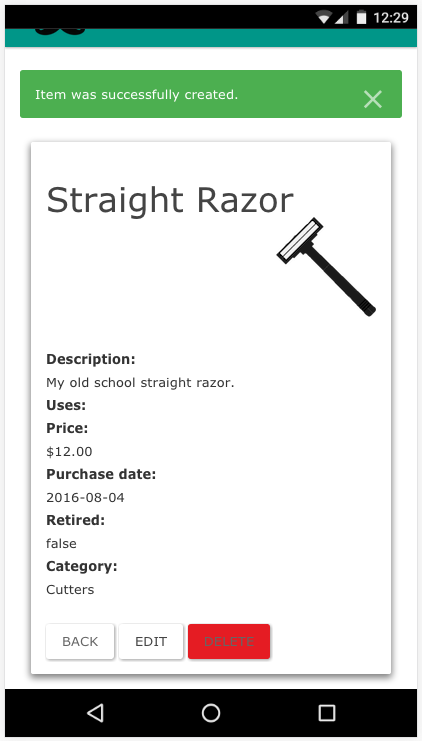
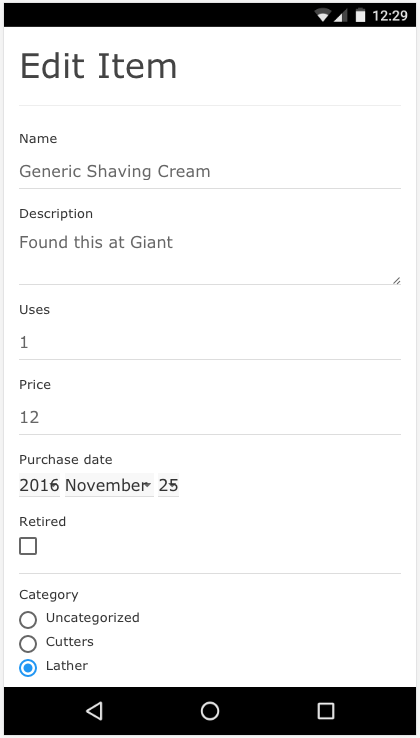
Screenshot Home Page Screenshot Drop Down Menu

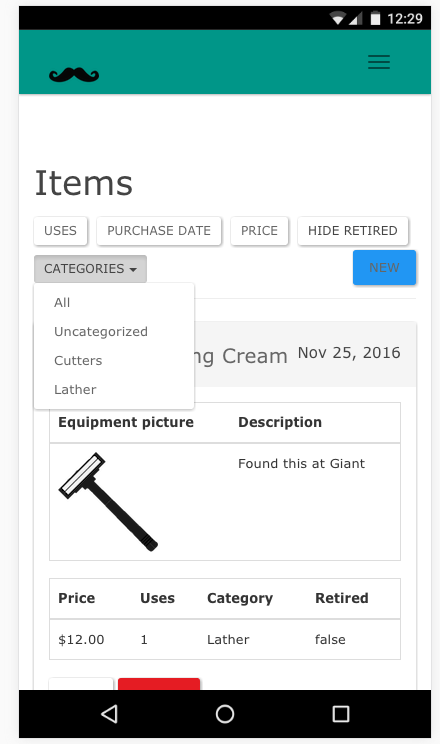
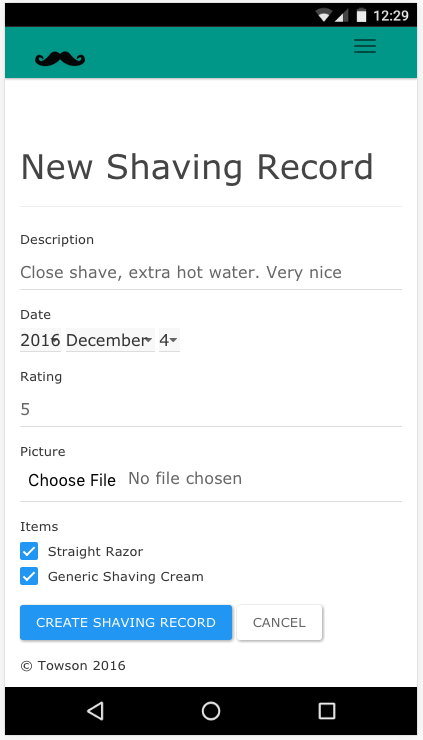
Screenshot Login Successful Screenshot Login Screen

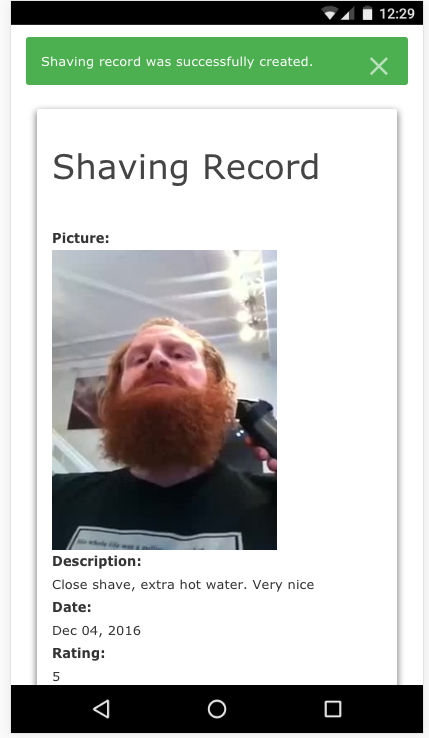
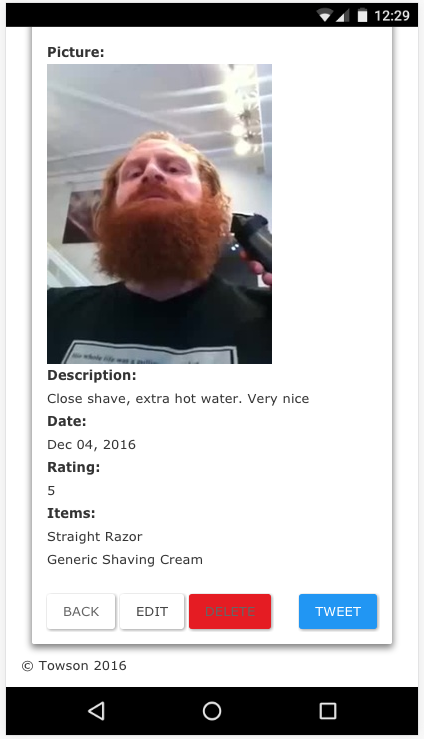
Screenshot New Item Screenshot New Item

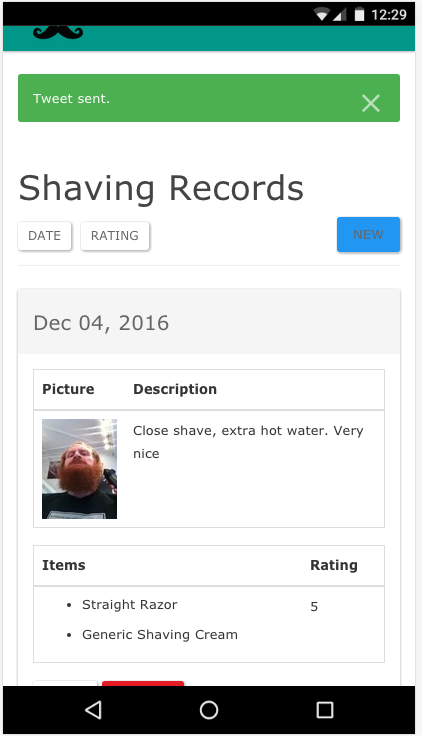
Screenshot New Item Created Screenshot Edit Item

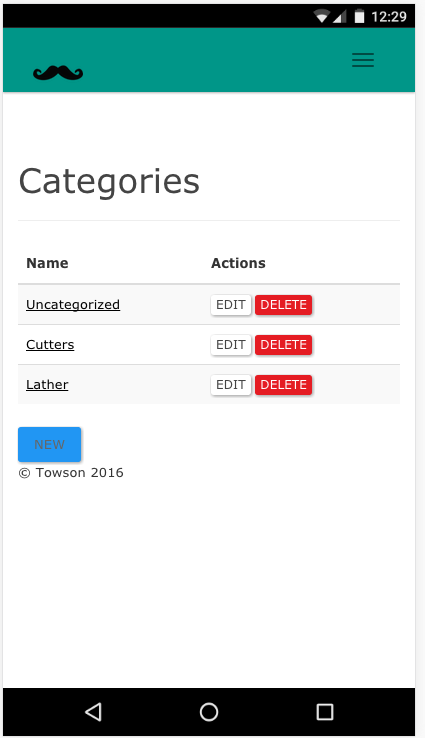
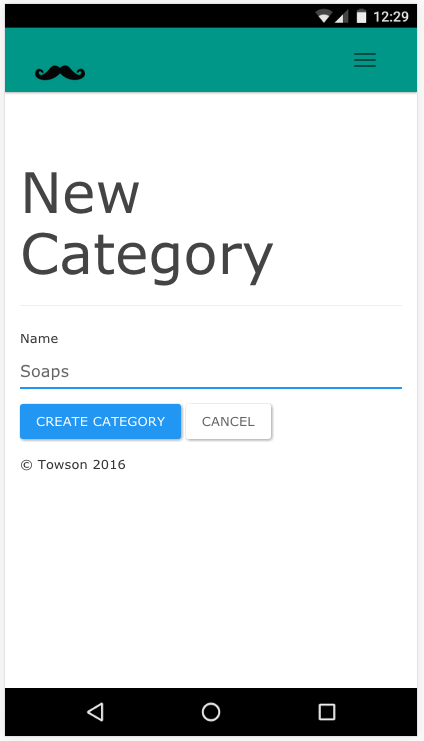
Screenshot Item Index View Screenshot New Shave Record

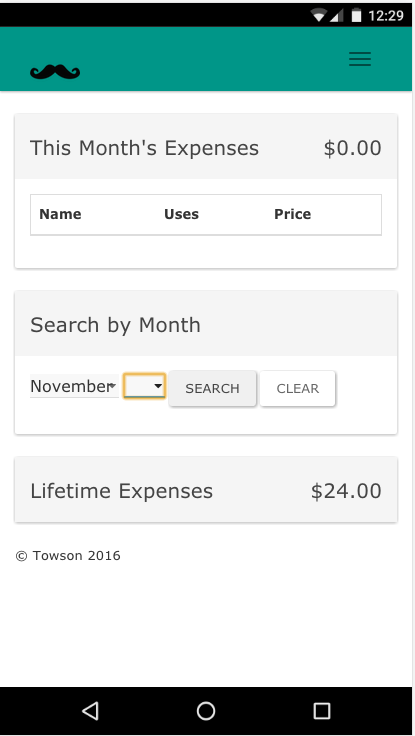
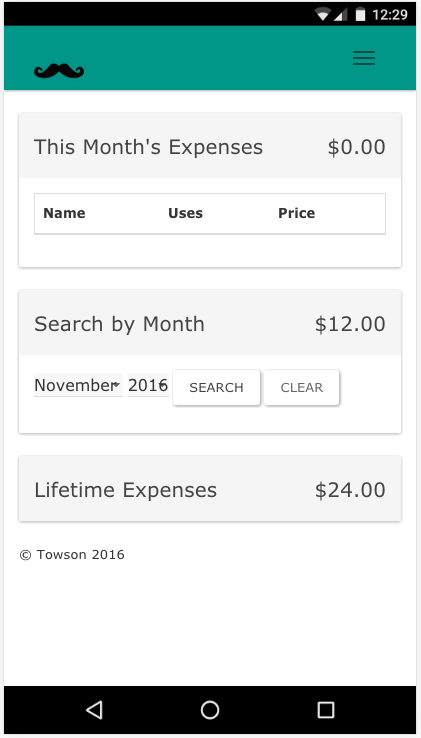
Screenshot Shave Record Created Screenshot View Shave Record

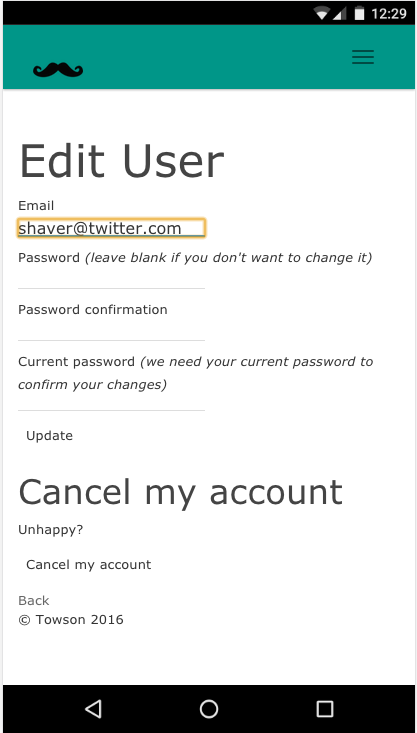
Screenshot Share record to twitter Screenshot Record on twitter

Screenshot Categories Index Screenshot New Category View

Screenshot Expenses View Screenshot Expenses Search by Month



Screenshot Edit User Account

# Test Cases

The following table contains test cases that are verification tests. Each test must be successful completed in order to conduct the next test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **Description** | **Expected Response** | **User Requirement** | **Pass / Fail** |
| Registration | Validation of creation of user account for mySOTD application. | Registration form is completed fully without error. Receive success notice and redirect to home page. | 10 - The user shall be able to register an account. | Pass |
| User Log-In | User validation of authorized access to mySOTD mobile application | Application accepts user’s input of log-in information, and grants user system access. | 11 - The user shall be able to login with a registered account. | Pass |
| Add Category | Validation of the creation of a new category record. | A new category record is created. Verify new category is created with correct user foreign key. | 5- The user shall be able to create an item category. | Pass |
| Edit Category | Validation that changes made to the selected category record are saved. | Category record reflects changes made. | 8 - The user shall be able to view and alter shaving categories in the database. | Pass |
| Delete Category | Validation that the deleted category record no longer exists. | Category record no longer exist in database. | 8 - The user shall be able to view and alter shaving categories in the database. | Pass |
| Create Item | Validation of the creation of a new item record. | A new item record is created in database and is correct. | 6 - The user shall be able to categorize and store shaving items in the ‘shave den’ database. | Pass |
| Edit Item | Validation that changes made to the selected item are saved. | Item reflects the changes made when queried from the database. | 7- The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item. | Pass |
| Delete Item | Validation that the item record no longer exists. Upon request of the user. | Shave den database no longer contains deleted item. | 7- The user shall be able to view items from the ‘shave den’ database and can edit of delete the selected item. | Pass |
| Create Shave Record | Validation of accurate display of shave record, upon user input of new shave record | Application processes the user’s entry input, and displays new shave entry | 1 - The user shall be able to create a shaving record for SOTD. | Pass |
| Edit Shave Record | Validation of accurate display of shave record, upon the shave data modification input, by the user | Application processes the shave data modification, and displays the updated shave data | 3 - The user shall be able to browse his shaving history and can edit or delete it. | Pass |
| Delete Shave Record | Validation of shave record purge, upon the request of the user | Application prompts the user of purge confirmation, and purges the entry, upon the user’s confirmation to delete record | 3 - The user shall be able to browse his shaving history and can edit or delete it. | Pass |
| Browse History | Validation of user being able to view shave record data | Application displays a list of all items already stored in database, allowing the user to browse through the items | 3 - The user shall be able to browse his shaving history and can edit or delete it. | Pass |
| Verify Expenses Correct. | Validation that the item prices the current month accurately reflect the sum of items purchased that month. | Month expense equal sum of items purchased that month. | 2 -The user shall be able to track their shaving expenses. | Pass |

# Conclusion

## Challenges Faced / Lessons Learned

I found getting the My Shave of the Day service quite a challenging endeavor. I started Towson University’s Computer Science Master program with little prior programming experience. My entire base of knowledge used on this project was based on what I have learned at Towson. During the initial design stage of the project, I spent much time drawing out all of the possible ways to establish relationships between the necessary entities. I knew by designing it correctly the first time, I could avoid any major rewrites during implementation.

In the initial coding phase, I ran into a very strange rails bug when creating the shave records scaffold. I initially wrote the model as shave, however rails set some of the controller variables to “shafe”. It was a very annoying bug, which forced me to use the model name shaving\_record. Another issue I had trouble with was with version control. I initially set up the repository to use GIT. When I implemented Heroku, my working files went out of sync with my remote repo at GitHub. After many failed attempts to reset GIT, I ended up copy the local project to a new directory and creating a new GitHub and Heroku remote repos. The original GitHub repo using GIT can be found at: <https://github.com/jrehakTow/MYSOTD.git>. I managed to avoid tracking conflicts by using SVN to communicate to GitHub and GIT for Heroku. Another challenging aspect was getting nested attributes to work for the many to many relationships. It took a lot of trouble shooting and browsing Stack Exchange to get mySOTD to show all the items a user had available to them when the created a shave, as well as getting those items to save to the shaving\_items database.

Android Studio was more challenging than I expected, I found myself going through lots of tutorials to get the web view application to work. One thing I found surprising was the additional methods required to get file upload to work with web view. When running mySOTD on a chrome browser, file upload works perfectly. It even prompts to use the camera. Web view in android will not upload files or access the camera without additional methods to handle the permissions. I initially planned to use the android client to store the views, but found no advantage to this because I had all logic operations handled on the server side. This ultimately led to the decision to treat the Android application as a very thin client.

One bug I could not get around was getting my unit tests to work for mySOTD. After some research, I found Devise, the Authentication gem, was causing foreign key violation errors with my shave den database. This prevent a test user account from being created, which in turn stopped all of the other unit tests from running because all entities belong to the user. I had to resort to manual testing, whenever changes were made.

## Experience Retrospective

As stated before, the majority of my programming experience has been established at Towson. I found many of the classes I completed before this project to help me greatly. Software Engineering helped me identify and refine the requirements for this project as well as choose the not deviate from the Architecture I choose. It is very easy to revert to Cowboy Coding. Software Testing and Maintenance encouraged me to think about the maintainability of this project and to refactor and cut waste where ever possible. Database Management Systems helped me envision the correct ways each entity should relate to each other as well as help me debug ORM, whenever a record was being retrieved or saved as expected. Advanced Web Development aided me greatly in my Ruby base knowledge and how to work with rails and certainly supplemented what I had learned about HTTP requests in Computer Networks. Human Computer Interaction always made me remember to consider the usability of mySOTD, when picking layouts and ways to navigate the UI. I would not have had the knowledge I needed to complete these projects without these classes.

## Future work

As with all software projects, there is always room for improvement. Introducing mySOTD on iOS would be a great way to expand mySOTD user base. As the application is currently configured, it would be possible to simply build it as a web application wrapper, much in the same way the Android Application is. This would allow for fast deployment. I did not try to develop for iOS because I currently do not have the iOS devices to conduct any testing. Currently I have only enabled one Social Media Platform, Twitter. Adding Facebook, Goolge+, and Instagram would greatly expand users sharing capabilities, as well as give more login options.

Additionally, it would be possible to introduce a whole contained social media platform to mySOTD. This social media platform would essentially operate as a twitter clone, allowing users to friend other users as well as post and share content within the app. To improve responsiveness, mySOTD could be converted to a single page application, only using JavaScript to update the interacted with portions of the application.

# Proposal

**Towson University**

**Department of Computer & Information Sciences**

**COSC 880**

**Student’s Name:** James Rehak **Student’s No:** 0229300

**Address:** 350 Bee Tree RD **Phone:** 410-960-0217

**City:** Parkton **State:** MD **Zip:** 21120 **E-mail:** jrehak1@students.towson.edu

**Tentative Project Title:** Face Groomer

**Graduate Credits Completed:** 42

**Semester to Register:** Fall X Spring\_\_\_\_ Summer\_\_\_\_ Year 2016

**Advisor’s Name:** Dr. Josh Dehlinger

**Project Description/Abstract:**

For the graduate project, I propose to build a web service with an Android front end that allows users to track their shaving habits and usage. This mobile application is targeted at shaving enthusiasts, who enjoying the ritual of a good shave. The application will be split into two core functionalities. One of the core functions is an inventory tracker for shaving equipment and consumables. Users will be able to keep track of items such as razors and shaving cream to see how much they are spending on shaving and to never be caught in a situation where they have to shave with a dull razor. The second core functionality is the shave record keeper. Users will be able to store their shaves as records with photos, text descriptions, and ratings. The users can indicate what equipment and consumables they used for the shave that will update their shaving inventory. Users will also have the option to show off their shaves to various social media platforms such as Facebook, Instagram, and/or Twitter. There is also the option to make the shaving application a self-contained social platform for shaving enthusiasts by implementing a forum and post / comment structure. There is currently not much market penetration for an application such as this in the Google Play store. So far I have only found other two applications that have similar functionality. The application with the most downloads appears to have been abandoned two years ago. None of the applications appear to have any social media elements in them.

This project was initially started as a group project for COSC 612 resulting in Software Requirements and Design Specification documents. I plan to use those documents as a starting point for the implementation of the project. Requirements and design elements may be modified from their original specifications. Additional use cases will be generated. The application will be developed using MVC architecture. I plan to build the application as a web service using Ruby on Rails and then connect to that service by creating an Android application to act as the front end. The service will use a PostgreSQL database to store persistent data. The android application will target 6.0.1 and follow material design guidelines for the GUI. I currently plan to deploy the web service using Heroku. The end goal is to have the application deployed on the Google Play store.

**Project Topics to be Studied**:

1. Java for Android
2. Ruby on Rails
3. PostgreSQL database
4. Bootstrap
5. Ajax
6. Version control

**Project Objectives:**

1. Develop a web application / service that is interface-able with an android front end.
2. Learn the process of developing a web service
3. Learn how to develop android UX.

**Deliverables:**

1. An online repository of the source code
2. Deployed on the world wide web
3. Deployed on the Play store

**Anticipated Project Timeline:**

|  |  |
| --- | --- |
| Gather Requirements | Done |
| Develop User Stories | June, 2016- August 2016 |
| Develop Web Service | August, 2016- December 2016 |
| Develop Android Front End | November 2016 – December 2016 |
| Advisor Meetings, Once a month or more | September 2016 – December 2016 |
| Present Final Project | December 2016 |

**Reading/References List:**

1. Ruby, Sam, Dave Thomas, and David Heinemeier Hansson. "Agile Web Development with Rails 4." (2013).
2. Sommerville, Ian. "Software Engineering. International computer science series." *ed: Addison Wesley* (2004).
3. Chacon, Scott, and Ben Straub. *Pro git*. Apress, 2014.
4. Kiessling, Manuel. "The node beginner book." *Available at [last accessed: 21 May 2015]: http://www. nodebeginner. org* (2011).
5. Phillips, Bill, and Brian Hardy. *Android Programming: The Big Nerd Ranch Guide*. Pearson Education, 2013.
6. Elmasri, Ramez. *Fundamentals of database systems*. Pearson Education India, 2008.
7. Valim, Jose. *Crafting Rails Applications: Expert Practices for Everyday Rails Development*. Pragmatic Bookshelf, 2011.
8. Hogan, Brian P. *HTML5 and CSS3 : level up with today's web technologies*. Dallas: The Pragmatic Bookshelf, 2013. Print.

**Completed Graduate Courses:**

*Provide a list of completed graduate coursework using the provided table.*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Number*** | ***Course Name*** | ***Semester*** | ***Grade*** |
| COSC 501 | Fundamentals of Data structures and Algorithms | Spring 2013 | A |
| COSC 502 | Computer Organization Assembly Language | Spring 2013 | S |
| COSC 519 | Operating System Principles | Fall 2014 | A- |
| COSC 600 | Advanced Data Structures and Algorithms | Fall 2014 | B+ |
| COSC 650 | Computer Networks | Fall 2014 | B+ |
| COSC 578 | Database Management Systems I | Spring 2015 | A- |
| COSC 605 | Human Factors/ Computer Interaction | Spring 2015 | A |
| COSC 611 | Computer Simulation | Spring 2015 | A |
| COSC 609 | Software Project Management | Fall 2015 | B+ |
| COSC 612 | Software Engineering I | Fall 2015 | A |
| COSC 645 | Applied Cryptology | Fall 2015 | B |
| COSC 603 | Software Testing and Maintenance | Spring 2016 | A |
| COSC 617 | Advanced Web Development | Spring 2016 | A |

**Projects Done in Other Courses:**

*Provide a list of relevant course projects and their descriptions completed in other courses.*

|  |  |
| --- | --- |
| **Course Number** | Project Description |
| COSC *519* | Modified Linux USB mouse driver and compiled custom Linux kernel. |
| COSC *600* | 1. Implemented a stack from scratch (no aid from Java stack library) 2. Created path finding program that listed different paths to visit the all States of the USA using DFT and BFT. |
| COSC 650 | Developed java application that uses TCP and UDP |
| COSC 578 | Developed an Apartment Management application using MySQL database and java swing front end. |
| COSC 605 | 1. Created a study to compare the usability of two different air travel websites. 2. Wrote a literature review examining the research done on Older Adults’ IT use. |
| COSC 611 | Learned to use Simscript to create various simulation scenarios. |
| COSC 609 | Created a mock RFP and proposal for an IT solution for an auto parts warehouse. |
| COSC 612 | Created SRS SDS documents from project proposal. |
| COSC 645 | Implemented the Digital Cash Protocol proposed by Bruce Scheiner. |
| COSC 603 | 1. Rewrote legacy program written in Fortran to Java. 2. Learned to use Eclipse’s refactoring tools to make a java program more maintainable. 3. Learned how to write JUnit test cases for OO application. 4. Learned how to use various eclipse plugins to aid in creation of unit tests and check code coverage. 5. Learned how to use mutation testing to improve test coverage. 6. Refactored a large open source game using the tools and knowledge learned from the previous projects. |
| COSC 617 | Developed an interactive education website using Ruby on Rails. |

**Project Requirements and Evaluation:**

1. Reading and presentation *0%*

2. Implementation *60%*

3. Examination *0%*

4. Written report to the instructor *30% (required regardless of % points)*

5. Public Presentation \* *10% (required regardless of % points)*

\* The instructor/student will announce the presentation time and place to the Department faculty and students. Those interested in the topic may attend the presentation.

I, \_\_\_\_\_James Rehak\_\_\_\_\_ propose to complete this project during the \_\_*fall* \_\_ semester of \_\_*2016*\_\_ and understand that this project and its derived materials (e.g., source code, written reports, presentation slides) are to reflect my own work, unless explicitly and appropriately referenced. Furthermore, I understand that plagiarism or other unattributed use of material not written by me is completely unacceptable, and **will be considered sufficient cause for a failing grade on the project.** For additional information on academic integrity policy at Towson University, I will visit [www.towson.edu/provost/resources/studentacademic.asp](http://www.towson.edu/provost/resources/studentacademic.asp) .

Student’s Signature: 

Instructor’s Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Graduate Program Director’s Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Works Cited

|  |  |
| --- | --- |
| [1] | S. Ruby, D. Thomas and D. Heinemeier Hansson, Agile Web Development with rails 4, Dallas, Texas: THe Pragmatic Bookshelf, 2013. |
| [2] | I. Sommerville, Software Engineering, 9th ed., Boston: Addison Wesley, 2004. |
| [3] | Heroku, "What is Heroku," SalesForce, 1 1 2016. [Online]. Available: https://www.heroku.com/what. [Accessed 3 12 2016]. |
| [4] | J. Drake, "Open Source Database Feature Comparison Matrix," QuinStreet Enterprise, 14 10 2005. [Online]. Available: http://www.devx.com/dbzone/Article/29480. [Accessed 3 12 2016]. |
| [5] | S. Chacon and B. Straub, Pro Git, 2nd ed., Apress, 2014. |