Project Report

CIS 4930 – Summer 2020

Umar Baloch

Table of Contents

Introduction:

This project is a console social media application called ‘Flixer’. Initially, the application was supposed to be a full-stack web application built using ReactJS and Firebase. The plan was to learn Firebase, JavaScript, HTML, ReactJS, and then build the web app from scratch. Due to time constraints, I could not learn all the components in time and decided to make this project using c++ as I have worked with c++ before.

Scope:

Many avid film fans in the world feel that they cannot find quality film reviews to guide them in their film-watching experience. This application solves that problem as it allows users to not only write their reviews but browse through reviews of all users. If a user likes the reviews posted by another user, they can easily follow them and have their reviews displayed in the 'Review Feed' menu. Our app also has a database management system that is updated in real-time to ensure that users see all the new reviews posted.

Specifications:

Files:

1. User-Data.txt
2. Reviews-Data.txt
3. Follower-Data.txt
4. Following-Data.txt
5. Data.h
6. Data.cpp
7. Reviews.h
8. Reviews.cpp
9. User.h
10. User.cpp
11. main.cpp

Compile and Run Commands:

```
g++ Data.h Data.cpp Reviews.h Reviews.cpp User.h User.cpp main.cpp
```

```
./a.out
```

Software Requirements:

This project can be compiled and run on a Linux shell. I used bash but visual studio may also be able to compile and run the code.

Constraints:

- Since this is a console application that runs locally, all users in the database must share the same PC and can only access the application one at a time.
- Due to time constraints, in certain menus and situation the app may not validate input which can lead to crashes.
- Due to time constraints, the user cannot edit or delete his/her posted reviews.
- Due to time constraints, the user can only unfollow a user in the current session, since the unfollow is not registered in the database. In the next session the user will still be following someone they unfollowed in previous session.
- The database is not very secure, and if directly altered, it can cause data corruption.

Assumptions and Dependencies:

- The app assumes that user will enter the correct input.
- The app assumes the data files will be stored in the same directory as the code files.
- The app cannot confirm if user’s email is real and will take any string as an ID.
- The app assumes that empty inputs are real inputs.
- The app assumes that code and data files will not be tampered with in the future.

Functional Requirements:

- Allow user to create an account using and ID string and password.
  - Prompt user for ID and password
  - Store ID and password in database
- Allow user to login using the one of the ID/password combinations in database.
  - Prompt for ID, password and compare to database
  - Notify user if ID or password are incorrect
- Allow user to change their password
  - Prompt for new password twice and check if the two strings match

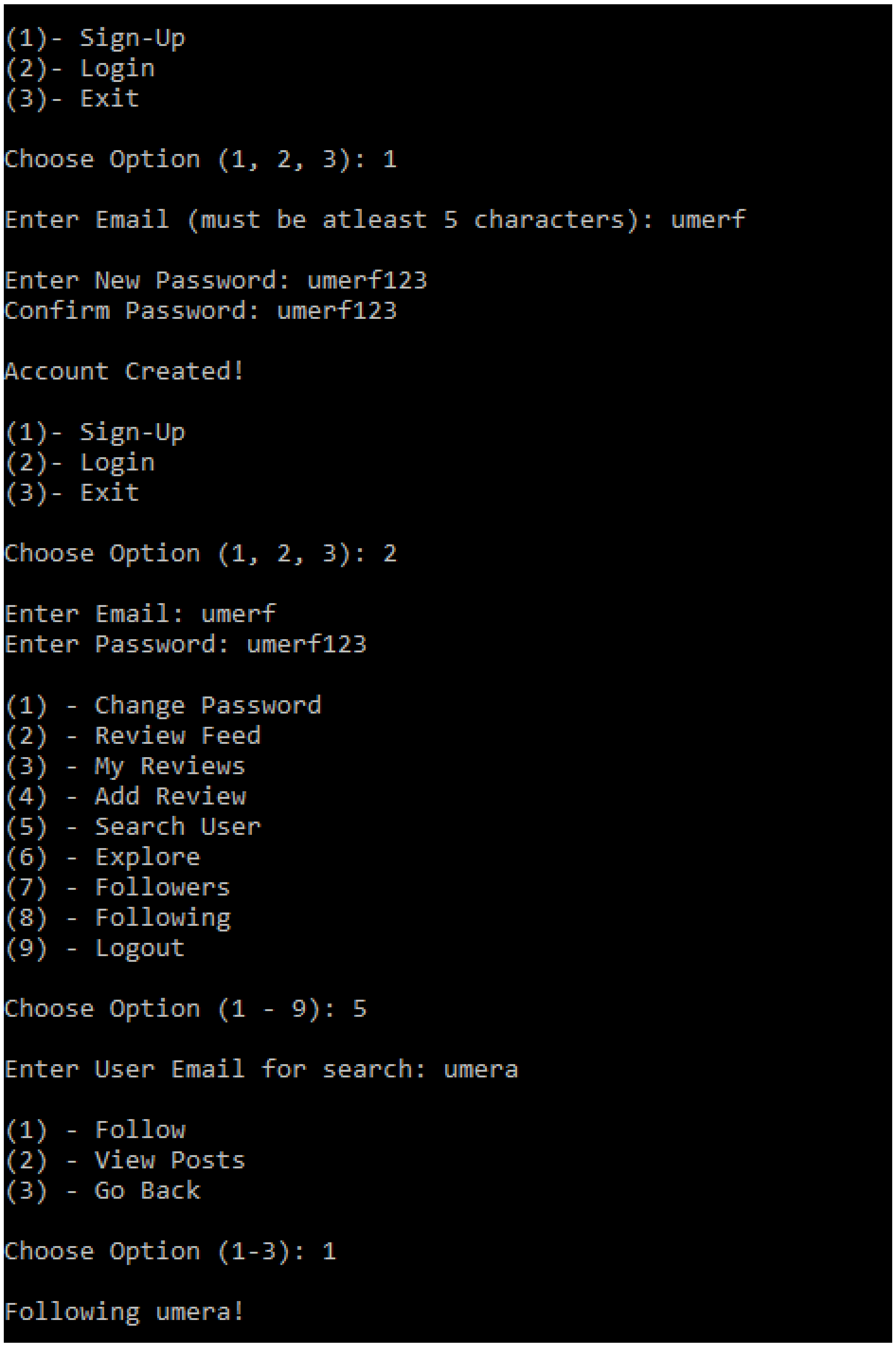
    - Update the password in database
- Allow user to see their ‘Review Feed’
    - Display all reviews by current user and users they follow
- Allow user to see their reviews.
- Allow user to add a new review.
    - Store the review along with current time to the database
- Allow user to search other users using their ID
    - The user can see reviews by the searched user
    - The user can follow or unfollow the searched user
- Allow user to see reviews by all users in the Explore option.
- Allow user to see their followers.
    - User can follow back a follower
    - User can see the reviews by a follower
- Allow user to see users they follow.
    - User can choose to unfollow a user
    - User can see the reviews by a user they follow
- Allow user to logout
    - Go back to the sign-up/login screen

Coding Style:

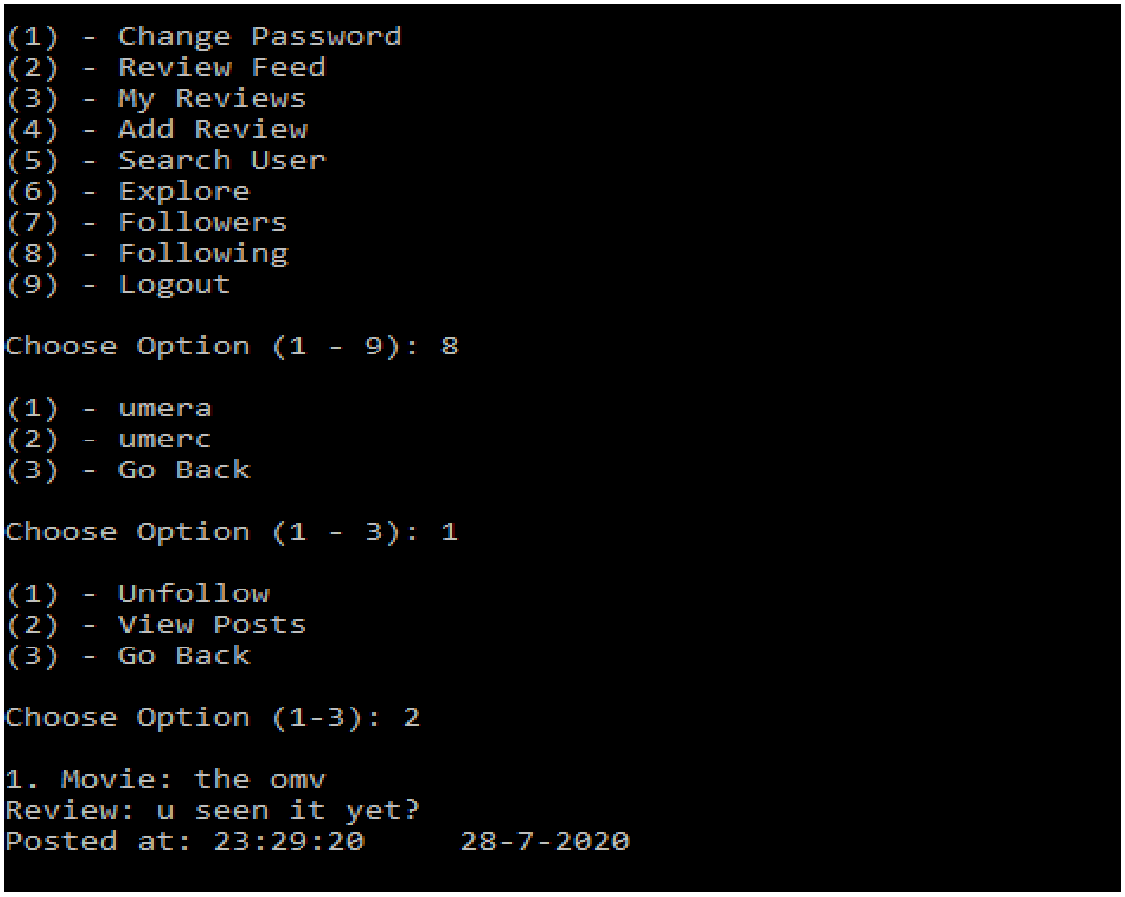
The coding style for this project is mostly object-oriented. The program treats users, reviews, and data files as objects. The main function implements menu levels through various loops and if statements. Then depending on the user input, it calls functions from the user, reviews, and data classes. User and review objects are stored in a universal vector accessible by the main function or Data class. Then those vectors are traversed through to get to the desired objects.

Data files are updated by either appending to the current files or making a copy with the new data added and then deleting the original file.

User Interface:



```
(1)- Sign-Up
(2)- Login
(3)- Exit

Choose Option (1, 2, 3): 1

Enter Email (must be atleast 5 characters): umerf

Enter New Password: umerf123
Confirm Password: umerf123

Account Created!

(1)- Sign-Up
(2)- Login
(3)- Exit

Choose Option (1, 2, 3): 2

Enter Email: umerf
Enter Password: umerf123

(1) - Change Password
(2) - Review Feed
(3) - My Reviews
(4) - Add Review
(5) - Search User
(6) - Explore
(7) - Followers
(8) - Following
(9) - Logout

Choose Option (1 - 9): 5

Enter User Email for search: umera

(1) - Follow
(2) - View Posts
(3) - Go Back

Choose Option (1-3): 1

Following umera!
```



```
(1) - Change Password
(2) - Review Feed
(3) - My Reviews
(4) - Add Review
(5) - Search User
(6) - Explore
(7) - Followers
(8) - Following
(9) - Logout

Choose Option (1 - 9): 8

(1) - umera
(2) - umerc
(3) - Go Back

Choose Option (1 - 3): 1

(1) - Unfollow
(2) - View Posts
(3) - Go Back

Choose Option (1-3): 2

1. Movie: the omv
Review: u seen it yet?
Posted at: 23:29:20     28-7-2020
```

Test Plan:

This app was tested in the following ways:

- Provide incorrect input to see the response of the app
- Create several users and follow several users to check if data files get corrupted at any time
- Traverse through reviews and users to check that data structures are not corrupted or accessed out of bounds
- Check to see if data files are correctly updated after altering data, because if they are not updated in the right format, the next time you run the app, it gives multiple errors.
- Run all functions and sub menus to see if program gets corrupted at any point.

Maintenance and Enhancements:

The only time this project would need maintenance is if one of the data files gets corrupted. In that case, you need backups for your data files so you can restore your data. On the other hand, a lot of enhancements can be made to this project to get a more optimal social media app.

The first enhancement would be to add edit/delete functions for reviews and a function to update data files when a user unfollows another user. These functions were part of the original plan, and I tried to implement many different algorithms for them. Unfortunately, I was getting small errors in some of the runs, and those easily corrupted the data files over several sessions of the app. Ultimately, I decided to omit these functions due to time constraints.

Other enhancements could include adding notifications to update users and likes and comments on the reviews. Given enough time these could also be implemented to our app. But at the end of the day, a real social media app needs internet to connect users. This is a big limitation for our app since it cannot share data across multiple devices and must store data locally.