CS 40800: Incremental and Regression Testing

**Splytr**

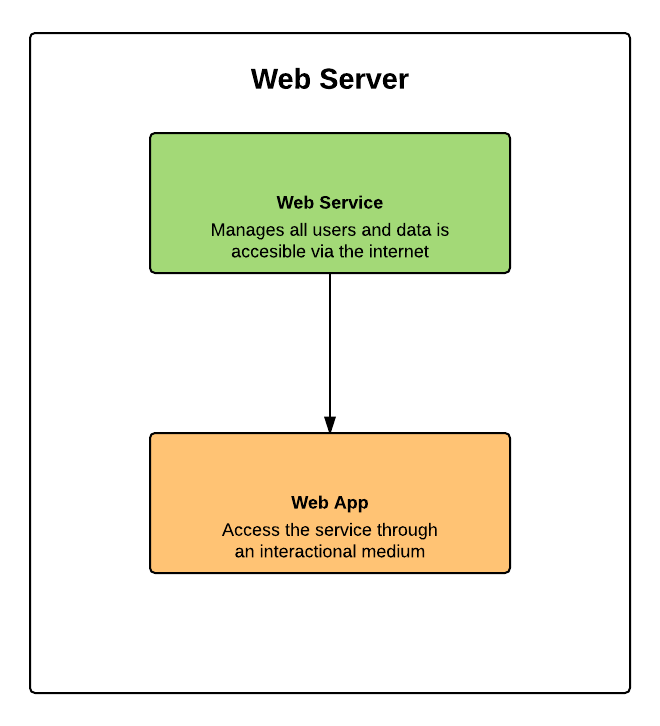
Sahil Pujari, Keenan Wresch, Akshit Gudoor, Hari Krishnam Raju, Aritra Samanta

horizontal line

**Incremental and Regression Testing**

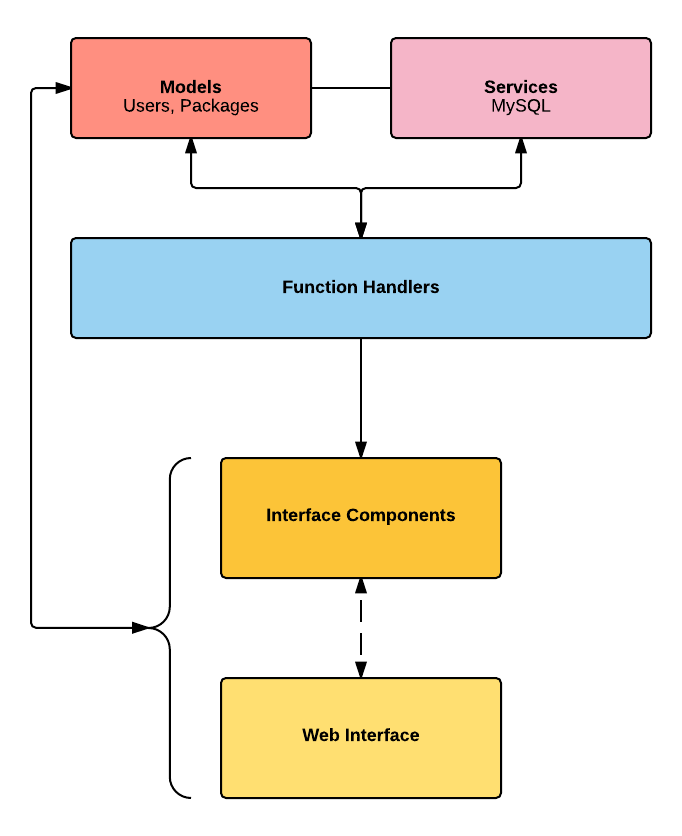
Classification of Components

**Overall Architecture and Major System Components**



The overall system consists of two major components. A web service in the center helps manage the user system and the package data and as well as help with the logical processing required for the system. As an addition to this, the web app allows users to access the service through an interactional medium based on the web which would require visiting the service using a web browser.

* Web Service
  + Input: Actions from the web app.
  + Output: Data to display on the web app.
  + Dependencies: See sections below
  + Dependents: Web app
* Web app
  + Input: User actions on the GUI
  + Output: Displaying data and taking actions based on the user commands.
  + Dependencies: ExpressJS (for web interface and routing)
  + Dependents: Users

**Web Service and Web App (Web Server)** 

The web server provides the core services of the system as well as interfaces to access these services. From a user­-facing perspective, there are two interfaces: a web­-based interface for use in a web browser and a service at the backend which handles all actions done on the web-based interface.

The web service consists of two components. A routing service which reads requests from clients and performs necessary actions while routing the user to a web page containing result of the action. Further, a database that stores all required data needed for performing actions.

The web interface is created in Javascript using the NodeJS framework. ExpressJS is used for routing, while an EJS-Engine is used to generate the UI for web-pages. These UIs may consist of custom components, such as a login form.

Function Handlers provide easy access to the core functionality of the system. Methods provide functionality to add users, check user credentials, add expenses, change expense, and more. These methods perform appropriate database transactions as well as other functions such as sending emails. They help to ensure that consistent functionality is provided between the web service and web app.

Finally, several services, such as password reset email service and MySQL connectors are used by the function handlers in order to perform their tasks. Models for users and packages are used by the function handlers, services, and other portions of this component of the system.

* Function Handlers
  + Input: Method calls by actions done on the web app
  + Output: Perform necessary actions with given data and return the result
  + Dependencies: Services, Models
  + Dependents: Web App
* Services
  + Input: Method calls by function handlers
  + Output: Perform necessary actions and return result
  + Dependencies: Service Providers (SendGrid, MySQL Connectors)
  + Dependents: Function Handlers
* Models
  + Input: Construction of objects
  + Output: Accessing attributes
  + Dependencies: None
  + Dependents: Function Handlers, Web Service

**Form of Incremental Testing Followed**

We have used only top-down testing methodologies, and were solely focused on the top-down testing. This helped us to ensure that the major components of our system were working properly and were deployed as soon as possible, while allowing us to focus on smaller details at a later time when they could be more easily tested as part of the system.

**Incremental and Regression Testing**

|  |
| --- |
| Incremental Testing |

**Web APP**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | Users can reach an invalid location. | 3 | Set up proper bounds and set auth tokens |
| 2 | Users can reach a location without proper resources required by the route. | 3 | Set up auth tokens and prop checks to limit this from happening. |
| 3 | Facebook registration does not always work. | 2 | Adapted to the changes introduced by FB in the SDK during the course of this sprint. |
| 4 | Adding friends to a splitting group crashes. | 2 | Fixed relations in the database and created a Group object that stores all splitters and acts as a container. |
| 5 | Users are allowed to enter a username that does not exist in the system. | 1 | Added method if the username provided is valid. |

**Web Service**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | Web app crashes for pulling user info sometimes. | 2 | Serialize database entries to follow proper syntax needed as by the server. |
| 2 | User credentials hashing sometimes throws an exception and app crashes. | 3 | Use a different hashing library and move to SHA512 which prevented this. |
| 3 | Server doesn’t pull the correct information for a group of splitters. | 1 | Fixed relations in the database. |
| 4 | String could not be null | 3 | Check for not null before accepting. |

|  |
| --- |
| Regression Testing |

**Web Services**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | User credentials entered may not match with the formatting guidelines. | 3 | Used validity checking for credentials before they go through further. |
| 2 | While using a performance testing script, routing sometimes times out. | 3 | Optimized the code to remove redundancies in calls to Database. |
| 3 | Web app would crash if a particular attribute returned by the database is null. | 2 | Use default values for nulls to prevent a null exception. |
| 4 | While returning the error we could not differentiate which error was returned where | 3 | Added the print statements to differentiate which error is which. |
| 5 | When adding a new package data, if the user information is incorrect, the server won’t go further to create a new package data in the database. | 2 | Error checking to make sure the user is always valid before modifying the package information. |

**Web App**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Users were redirected to login although their auth token was valid. | 1 | Modify login system design to account for active cookies and auth tokens. |
| 2 | Group deletion does not clear the relations in the database. | 3 | Implement an additional instruction set for deletion of groups adding an instruction to clear relations in DB. |
| 3 | Payment Splitter app didn’t display the divided amount among the members | 3 | Added a function to divide the amount among the desired people and then retrieve the name and the amount it was split between. |
| 4 | Changes made in the User Information were not saved in the database. | 3 | Ensure the current user’s information is changed and data is recorded properly. |