CS 40800: Incremental and Regression Testing

**Splytr**

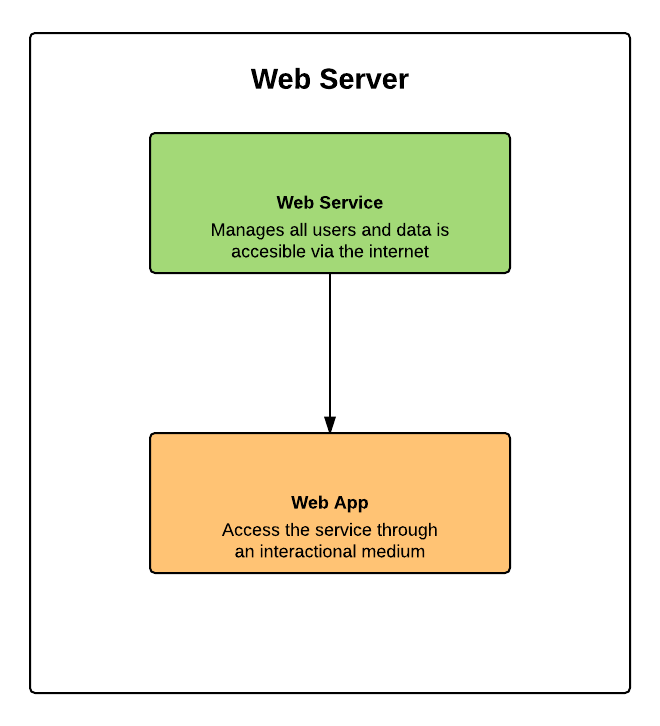
Sahil Pujari, Keenan Wresch, Akshit Gudoor, Hari Krishnam Raju, Aritra Samanta

horizontal line

**Incremental and Regression Testing**

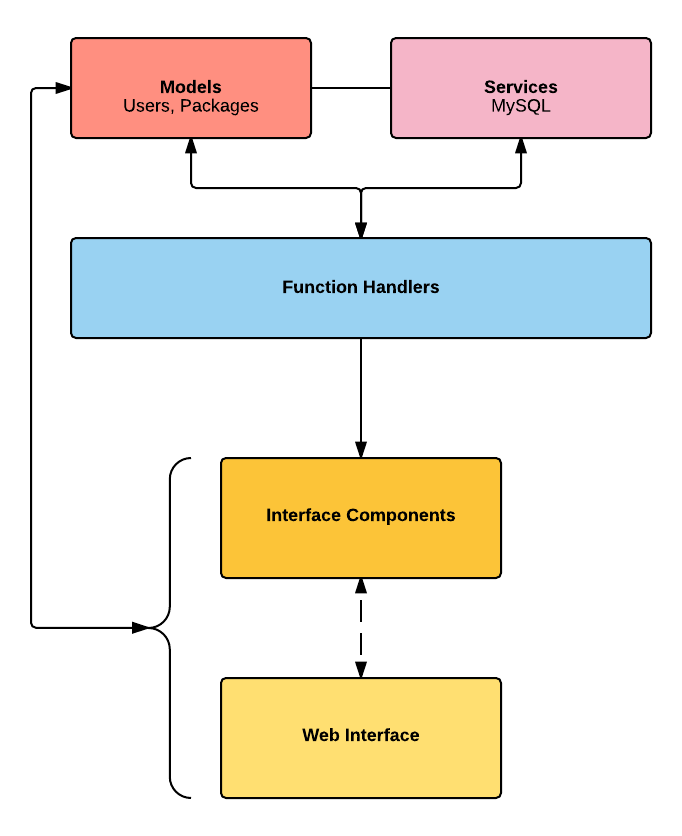
Classification of Components

**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

**Incremental and Regression Testing**

|  |
| --- |
| Incremental Testing |

**Web APP**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | Web app won’t always deploy | 1 | Added implementation in the app.JS file to pick a port that is available. |
| 2 | Successful login redirects to the login page again | 2 | Set separate routes and configure passport. |
| 3 | Facebook registration won’t work. | 3 | Configure passport-facebook and fix the facebook strategy app ID |
| 4 | Custom registration would not sync with the database | 3 | Configure the database properties to point it towards the correct connection ID |

**Web Service**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | Clients do not get any messages or responses from the server | 2 | Ensured that the client always gets a message back no matter what happens in the server. |
| 2 | If the client send a command with garbage values to the server it could crash | 1 | Checking the validity of the command before sending and executing it. |
| 3 | Client doesn't get error notification from the server that the login information is incorrect. | 2 | Checking the sent username and password in the database. If not matched sent an error message to the client. |

|  |
| --- |
| Regression Testing |

**Web Services**

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was corrected |
| 1 | Users can do a code reuse attack | 2 | Add access rights to internal methods to prevent this from happening |
| 2 | Adding the client timeout didn’t allow the server to perform longer operations | 3 | Increased the timeout amount to give the server more time to perform the action |
| 3 | Handling requests on separate threads created a race condition. | 1 | Use calls to database functions which has an effect of making any code that changes the database atomic. |

**Web App**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Registering new users with an existing username | 2 | Before adding the entry to the database, check if exists in the database. |
| 2 | Successful login redirects to the login page again | 2 | Set separate routes and configure passport. |
| 3 | Custom registration would not sync with the database | 3 | Configure the database properties to point it towards the correct connection ID |