TEAM ROCKERS
PROJECT

Submitted to Dr. Peter Bodorik
Faculty of Computer Science
Dalhousie University

Submitted by Team Rockers
Date – 29 July, 2019

NAME	BANNER ID
AMAN ARYA	B00816348
ANIRUDDHA CHITLEY	B00808320
JESSICA CASTELINO	B00804805
NIKHIL TYAGI	B00809791
NITISH BHARDWAJ	B00811535
SAMARTH RAVAL	B00812673

CONTENTS

LIST OF FIGURES

PROJECT OVERVIEW:

The purpose of this project is to create four different portals for a customer looking to purchase a house and is applying for a mortgage. The whole process is supposed to be automated. The four main portals that need to be developed are MBR, EMP, INSinc, and RE.

MBR is a mortgage broker portal to apply for a mortgage. EMP is the employer portal where the employee who wants to apply for the mortgage needs to log in and provide the MortID and URL of the MBR. RE portal has two pages. One to raise a request for an appraisal by the customer and another for the Broker to enter details of the appraisal. Finally, the INSinc portal is used to verify the details of the insurance. In the end, the customer can download a copy of the mortgage document from the MBR portal.

The primary requirement is to make the whole process automated. It is also ensured that once the customer raises a request for a mortgage on the MBR portal then the order of the task execution on the portal does not matter.

Sails.js app is used to build the web service present on the server and the client-side application. Then, the web service is moved to Azure Cloud. Additionally, for the purpose of testing the web service parameters and output, Apache JMeter application has been used.

PROJECT PLAN:

For the efficient management of the project, the team brainstormed and finalized a set of tools to maintain the different deliverables. Figure 1 shows the list of tools used in the management of this assignment [1].

The project is planned to be divided into two major parts i.e. web-service and client-side code. **Group 9: Team Rockers**, having six members, were divided into two internal sub-teams consisting of three members each. The design of both parts was designed by individual teams and finalized as a group. In the next phase of development, the user portal, web services were developed and checked-in to the GitHub repository. The regular meetings and project management tool: Trello helped the group to flag the object based on their priority. It helped the team to identify and mitigate the risk associated with each object [1].

With the completion of each object, it was continuously tested by another group member who was not associated with the development of that object. This helped to identify more bugs in the objects and coherent development and testing. After the unit testing of each object, all the objects were tested i.e. the integration testing of the objects was performed. Eventually, after the reduction of bugs, the code was deployed on Azure cloud. Figure 2 depicts the overall project plan for Project [1].

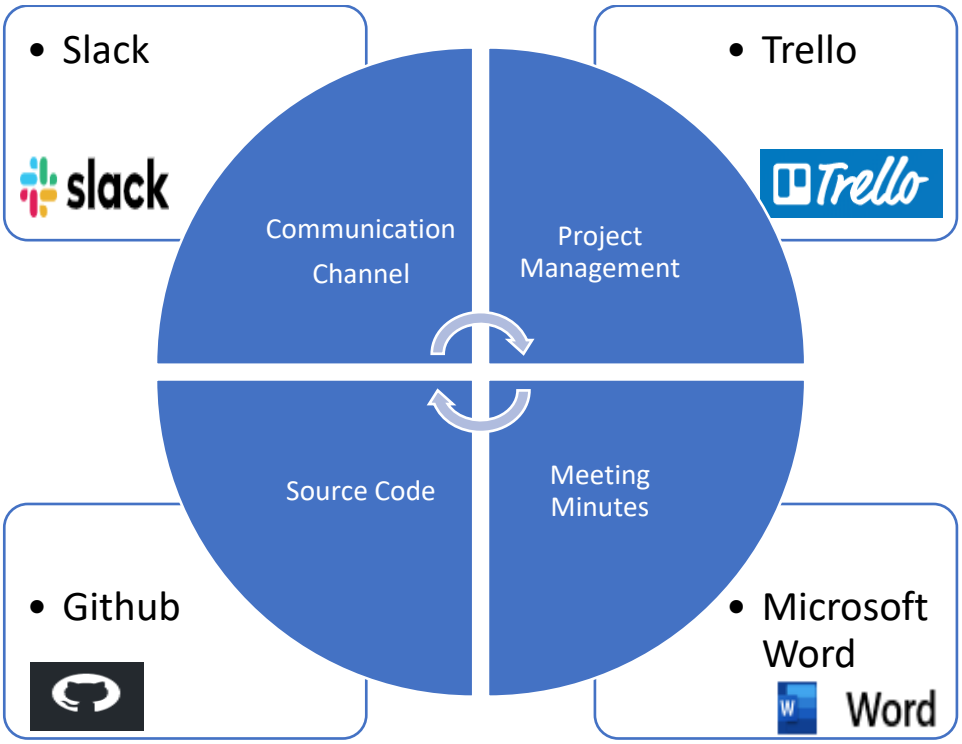


Figure 1: Tools used for the efficient management of the Project [1]

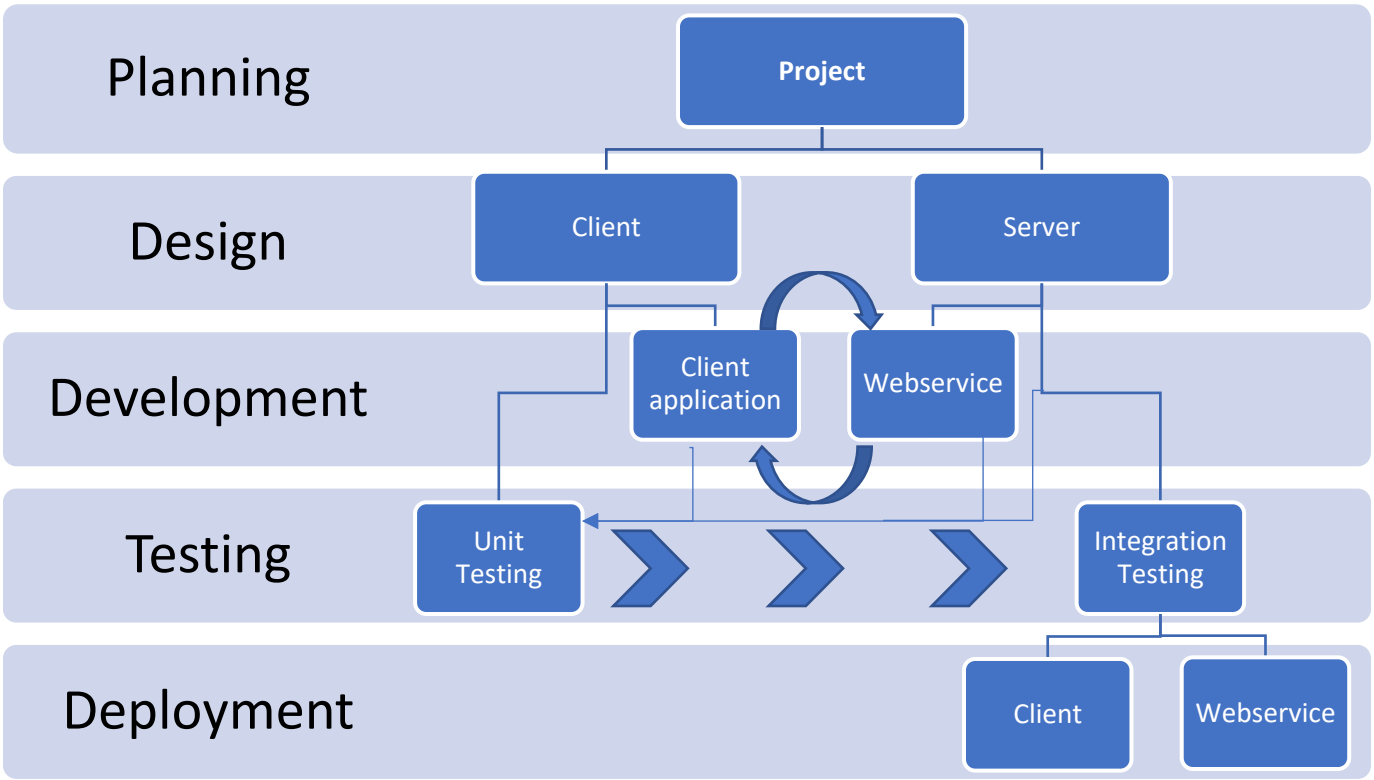


Figure 2: Overall Project plan [1]

RESOURCES USED:

Various libraries, software, tools, languages and frameworks are used in the execution of this assignment. This section contains the list of resources used in the completion of the assignment.

HARDWARE & SOFTWARE

All nodes must have[2] :

- Linux kernel version 3.10 or higher
- CS Docker Engine 1.13, or EE Daemon 17.03 and higher
- 8.00 GB of RAM for manager nodes or nodes running DTR
- 4.00 GB of RAM for worker nodes
- 3.00 GB of available disk space
- A static IP address

TOOLS

Load Testing tools: JMeter

API Testing tool: Postman

Load Balancer: NGINX

LIBRARIES

XMLHttpRequest – This library helps to retrieve the data from a URL without performing a full-page refresh. Thus, the web page can update a part of a page without disrupting what the user is doing [1].

Bootstrap: It is a free and responsive CSS framework used to create highly responsive applications. Bootstrap allows creating the application in a shorter duration of time. This ensures better usage of time and resources [1].

LANGUAGES AND FRAMEWORK:

JavaScript - JavaScript® is a lightweight, interpreted, object-oriented scripting language with first-class functions. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles [1].

HTML - Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. HTML elements are the building blocks of HTML pages. [2] With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. It provides a means to create structured documents [1].

Node.js - Node.js is an open-source, cross-platform JavaScript runtime engine to develop server-side applications. It helps the developers to create robust and distributed applications as a result of its event-driven architecture [1].

Express.js - Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications [1].

Sails.js - Sails.js is a Model-View-Controller (MVC) web application framework which is developed on top of the Node.js environment. As it is built on top of Node.js and Express.js, it enables applications to be built purely in JavaScript. This not only includes models, views, and controllers; but also configuration files and adapters (e.g., database) [1].

CSP SERVICES USED:

CSP (Cloud Service Provider): **Microsoft Azure** is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

In this project, the IaaS software service is used to create the Virtual machine and deploy the backend web services (Sails.js) and load balancer module (Nginx) to provide the required functionality of scaling and load-balancing the client requests.

FUNCTIONALITIES PROVIDED OVER AND ABOVE THE REQUIREMENTS:

Token-based authentication is implemented on Employer portal. Instead of having to authenticate with username and password for each protected resource (like consent form to allow the employer to share details with mortgage broker), the user is given a token which offers access to a specific resource.

The homepage that lists URLs to various portals. This is created to help the TAs to effectively navigate through the various portals.

Additionally, we have created a sign-up API to allow the employee to sign up on the employer portal.

IDENTIFICATION OF ANY CODE AND TOOLS USED:

The following code snippet is used to generate logs by accessing the request and response parameters. The reference for this code was taken from StackOverflow [3].

```
// https://stackoverflow.com/a/19215370
var oldWrite = res.write,
    oldEnd = res.end;

var chunks = [];

res.write = function (chunk) {
    chunks.push(chunk);

    oldWrite.apply(res, arguments);
};

res.end = function (chunk) {
    if (chunk)
        chunks.push(chunk);

    var body = Buffer.concat(chunks).toString('utf8');
    line = '\n######################## Response ####################\n';
    line += res.statusCode + '\n' + res.message;
    line += '\n' + res._header;
    line += '\n' + body;
    fs.appendFile(logfile, line, () => { console.log('logged') });

    oldEnd.apply(res, arguments);
};
```

The code snippet shown in figure 3 is taken from MDN Web Docs [1]. The purpose of using this code is to be able to send an asynchronous request to the server, fetch the response and display it on frontend without page reload. XMLHttpRequest object is used in this case to help with data exchange. The readyState property monitors the status of this object, and it calls an event handler present in the onreadystatechange property which gets called when the readyState attribute changes. This code is used in the JavaScript functions of the pages used in the Employee portal after suitable modifications are made to validate the inputs and pass them as parameters for the web service call.

Example

```
1  var xhr = new XMLHttpRequest(),
2      method = "GET",
3      url = "https://developer.mozilla.org/";
4
5  xhr.open(method, url, true);
6  xhr.onreadystatechange = function () {
7    if(xhr.readyState === 4 && xhr.status === 200) {
8      console.log(xhr.responseText);
9    }
10 };
11 xhr.send();
```

Figure 3: XMLHttpRequest object for sending data to web services and fetching the response without page reload

Cloud Computing – CSCI5409

TRELLO BOARD: https://trello.com/b/xYAXiGI3/project

GITHUB: Frontend → https://github.com/amanarya0/cloud-proj-portals
Backend
MBR → https://github.com/nikhiltyagi1991/mbr-portal
EMP → https://github.com/nikhiltyagi1991/employer-backend
RE → https://github.com/nikhiltyagi1991/real-estate-portal
INSinc → https://github.com/nikhiltyagi1991/insurance-portal

SLACK: macs-2018.slack.com

Azure: Backend
MBR → http://rockers-mbr.eastus2.cloudapp.azure.com/
EMP → http://rockers-employer.centralus.cloudapp.azure.com/
RE → http://rockers-realestate.southcentralus.cloudapp.azure.com/
INSinc → http://rockers-realestate.southcentralus.cloudapp.azure.com/

Heroku: Frontend → https://project-portals.herokuapp.com/

Meeting minute files:
1. https://macs-2018.slack.com/files/UD31WAJ76/FLULZGL2Y/15th-july-2019_mom.pdf
2. https://macs-2018.slack.com/files/UD31WAJ76/FLNQRHSGZ/22nd-july-2019_mom.pdf
3. https://macs-2018.slack.com/files/UD31WAJ76/FLX147W0P/27th-july-2019_mom.pdf
4. https://macs-2018.slack.com/files/UD31WAJ76/FLX9SU5TQ/28th-july-2019_mom.pdf

OVERVIEW OF FRONTEND:

There are four portals developed as part of this project. Each portal contains various pages. The description of each portal is given below:

MBR Mortgage Broker portal

MBR is a portal provided by the broker to let the customer enter the details of the mortgage. This helps the broker to understand that the customer is interested in which mortgage. Customer can enter details like name, contact number, address, employee id, employer name, Mortgage Value and M1sID. If the details are correct, the customer receives a MortID that can have to be used on other portals. Customer also receives information of the portal where the status of the application can be checked. Figure 4 shows the screenshot of the MBR portal. Figure 5 shows the screenshot to view the status of the MBR application.

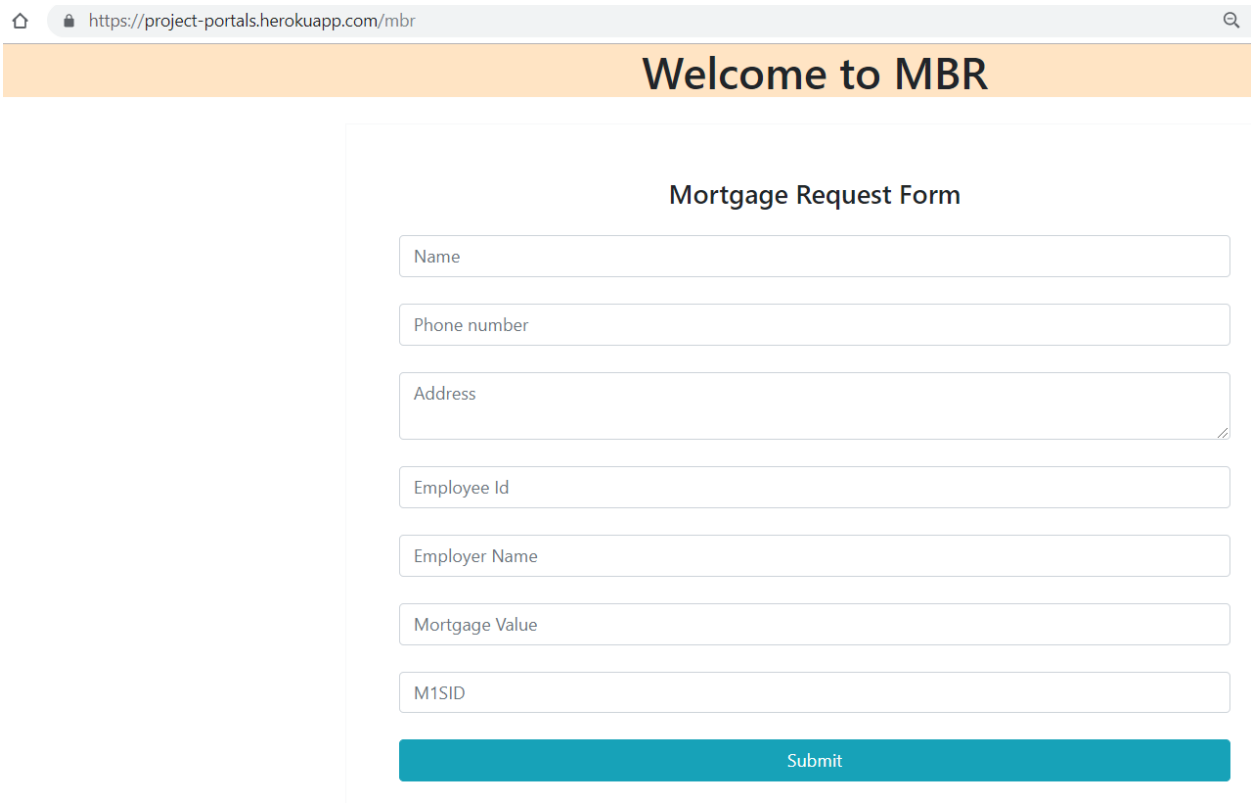


Figure 4: Screenshot of MBR Portal

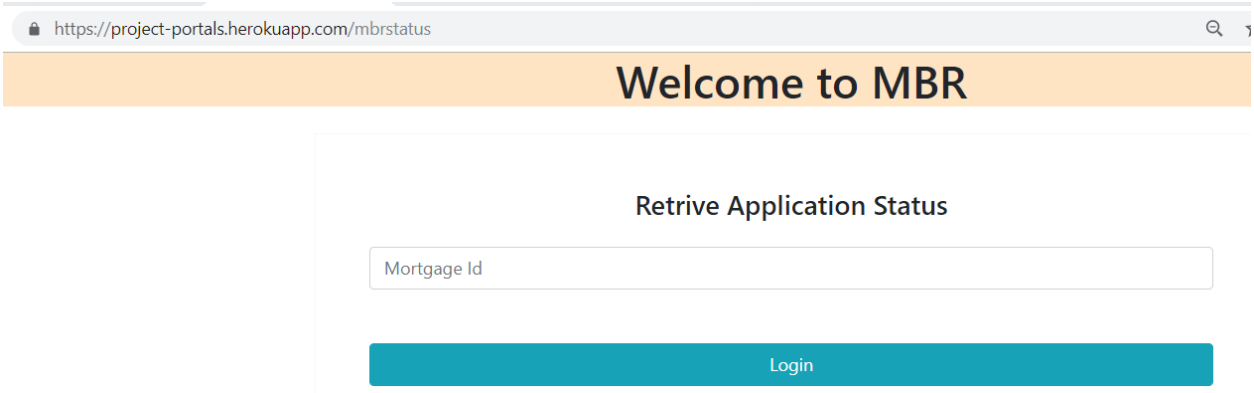


Figure 5: Screenshot to view status of MBR application

Real Estate Broker portal

Real estate portal has two pages named application and an update appraisal page. In the real estate application page, there are three fields for the input Mortgage unique ID, Mortgage value and Name. Validation has been done on the backend, the appropriate message has been displayed if fields are blank or invalid data has been given as input. If all the validations are satisfied, the application page will display a successful message.

The update appraisal page is shown in the figure. In this page, it has three different fields named appraisal value, mortgage value and comment. All this field must have value before requesting it otherwise it will through an exception. If all the values are valid it will submit the updated appraisal form. A successful message will also be displayed on the screen. Figure 6 shows the screenshot of the Real Estate Application form to request an appraisal. Figure 7 shows the screenshot of Real Estate appraisal form used by the broker.

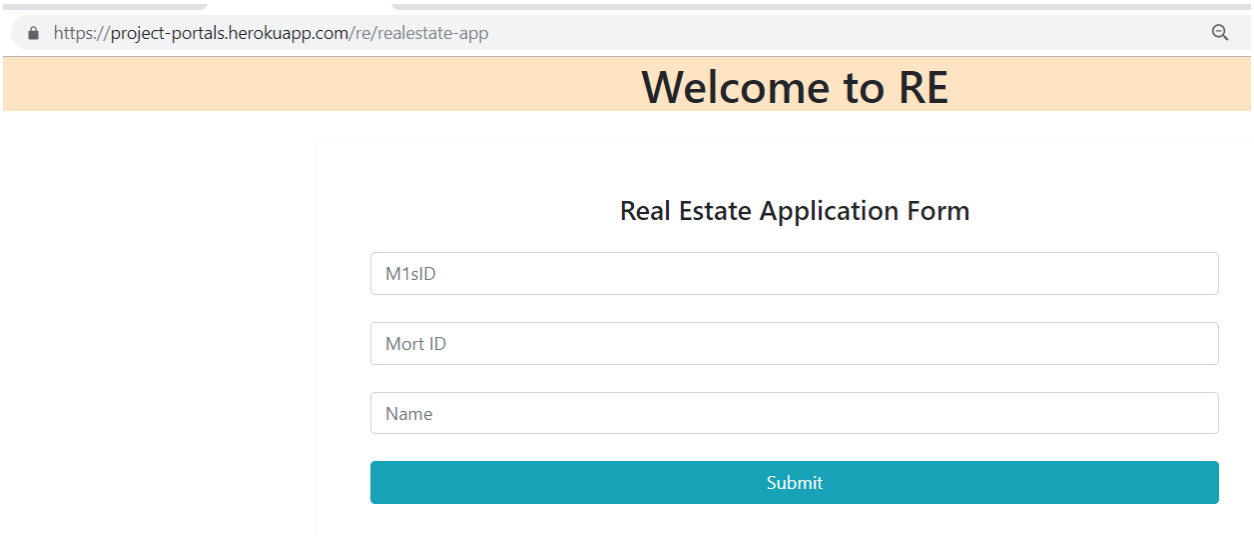


Figure 6: Screenshot of Real Estate Application Form to request an appraisal by the Employee

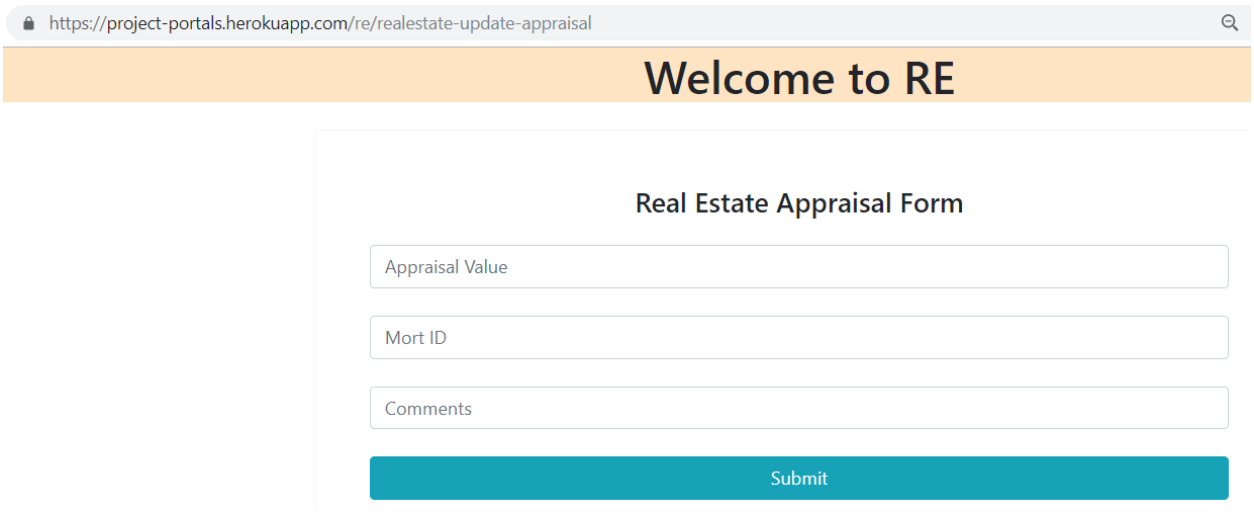


Figure 7: Screenshot of Real Estate appraisal form used by the broker

Employer portal

The Employer Portal has a login page as shown in the figure which takes as input two fields – the employee ID and the password. Appropriate JavaScript validations are present to ensure that the employee ID and password are not blank. In case validations are not satisfied, appropriate error messages are displayed on the screen while blocking form submission. If all validations are satisfied, the

employee credentials are passed to a web service that verifies the credentials and redirects the employee to a consent form on successful authentication. If authentication fails, an error message is displayed on the screen.

The consent form, as shown in the figure, takes as input two fields – the mortgage application ID (as provided by the broker) and the web service endpoint URL of the Broker, along with consent in the form of a checkbox. Appropriate JavaScript validations are present to ensure that the application number and web service endpoint are not blank. It is also mandatory for the customer to give consent by clicking on the checkbox. In case validations are not satisfied, appropriate error messages are displayed on the screen while blocking form submission. If all validations are satisfied, a web service call is made to the URL for the application number provided by the customer and the required employer details are sent. On successful submission, the customer is notified through a message on the screen. Figure 8 shows the screenshot of the Employer portal where the user can log in. Figure 9 is the screenshot of the page where the user can enter the MortID and URL of MBR to submit to the employer.

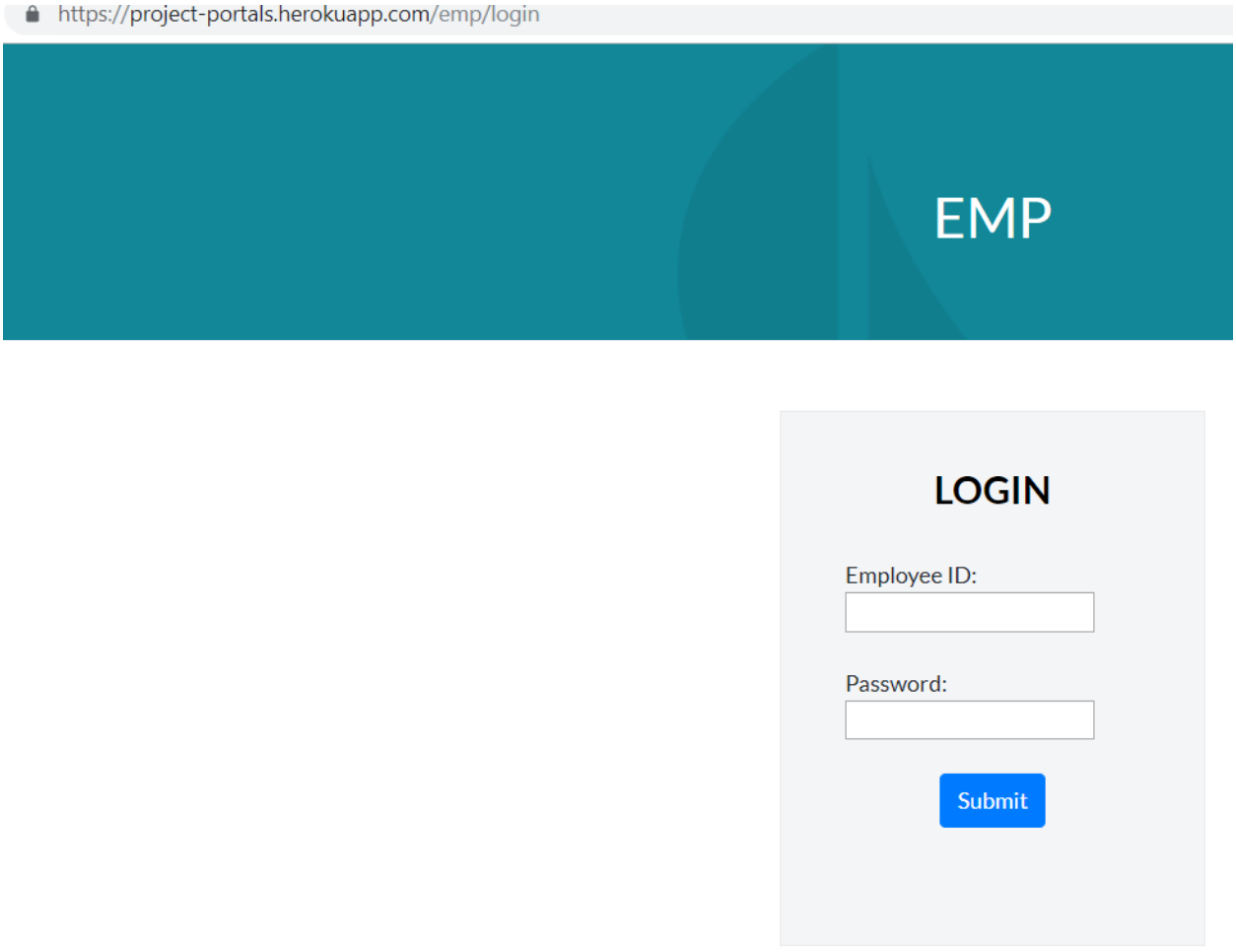


Figure 8: Screenshot of the Employer portal where an employee can log in

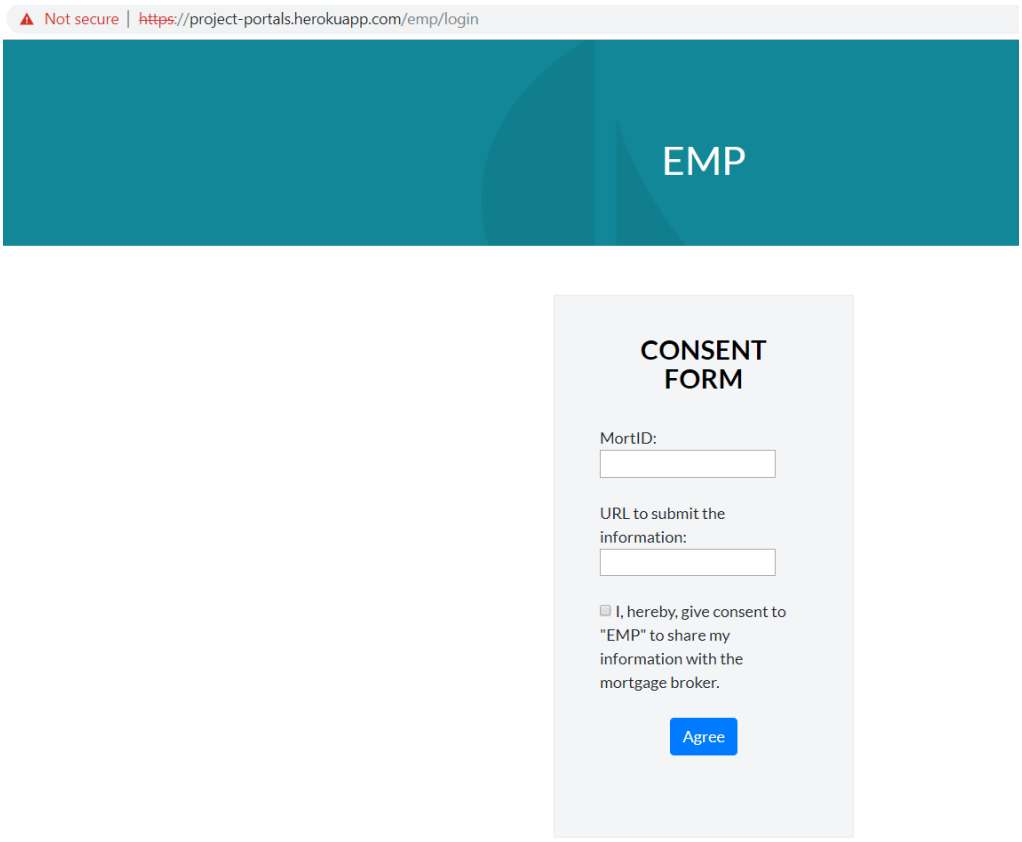


Figure 9: Screenshot of the page where the user can enter the MortID and URL of MBR to submit to the employer

Insurance portal

The insurance portal constructs an insurance quote once it receives a request from the real estate broker. It then sends the quote to the MBR broker portal. Additional functionality has been added to

this portal that allows the user to view the quote to get the state of their application as shown in figure 10.

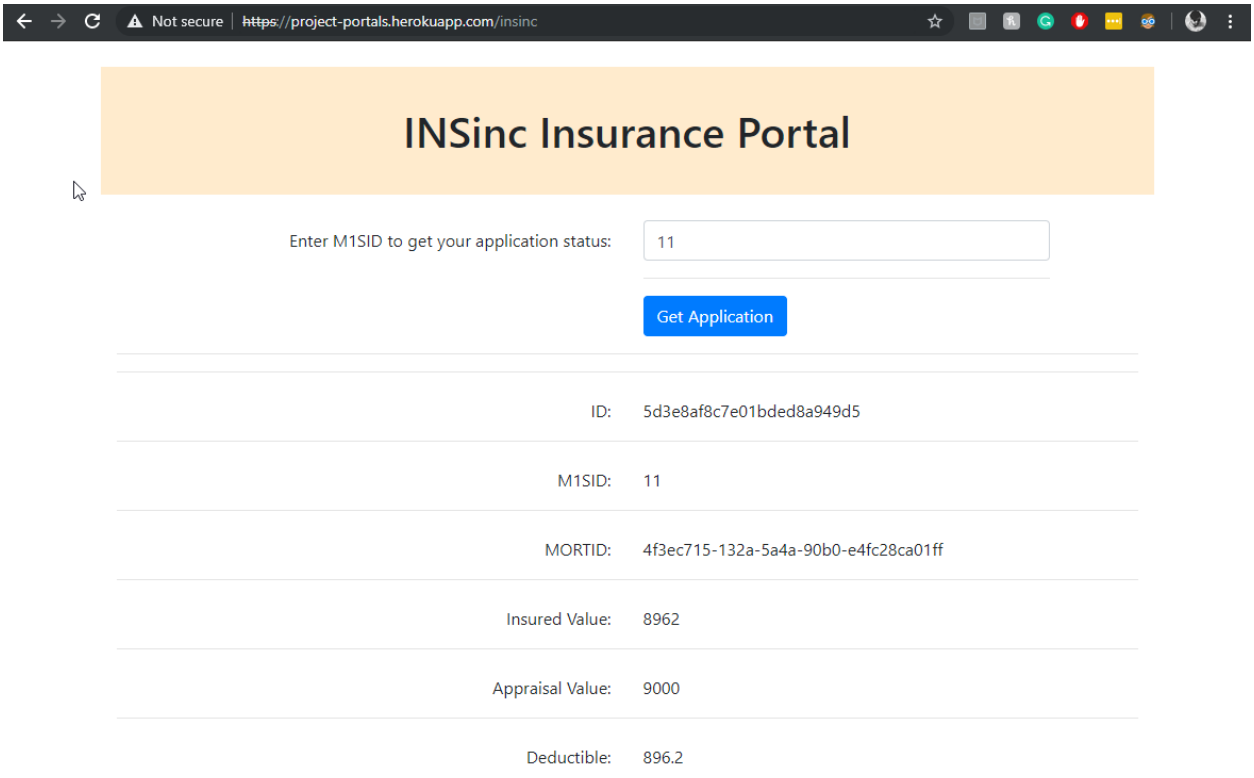


Figure 10: Status check at Insurance Portal

SAMPLE SCREENSHOTS OF SLACK AND TRELLO

Figure 11 to 16 shows the screenshot of the Slack channel. Please note that as Slack is used as the communication channel, so only the sample screenshots are provided in this report. However, complete communication can be viewed at Slack.

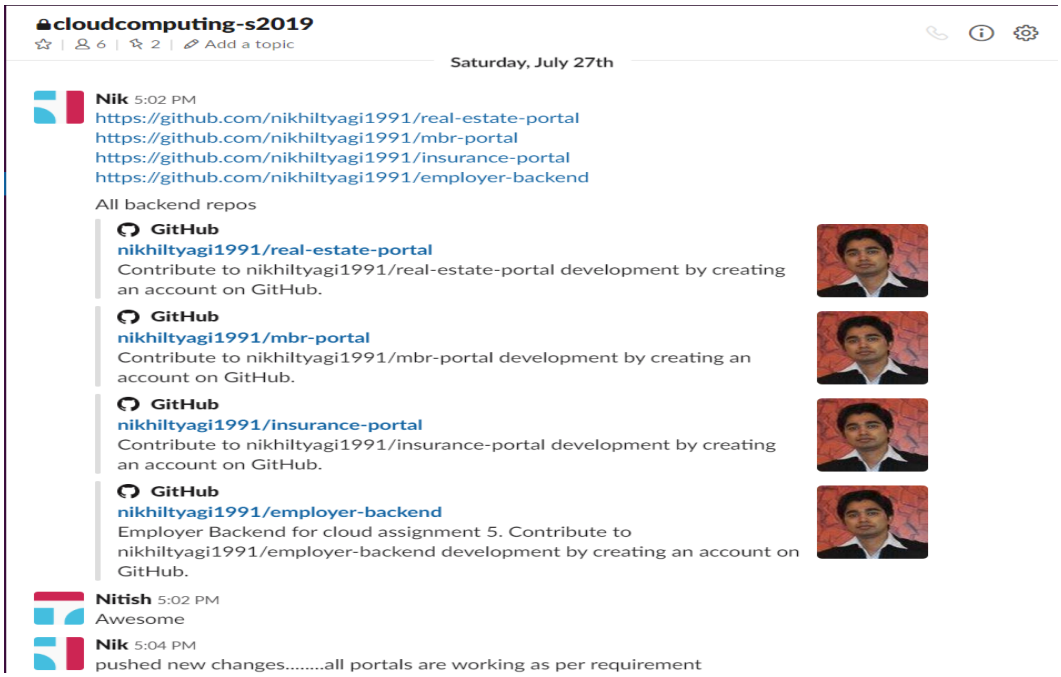


Figure 11: Screenshot 1 for Slack

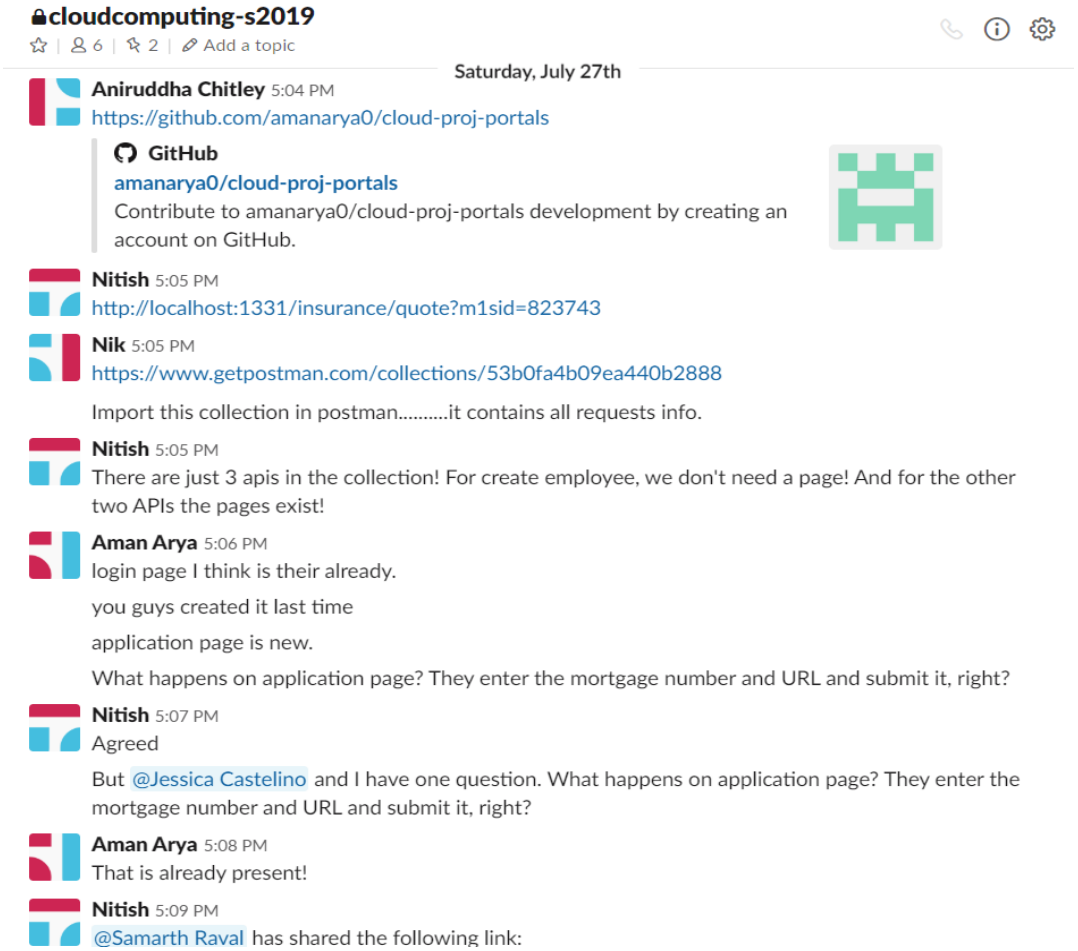


Figure 12: Screenshot 2 for Slack

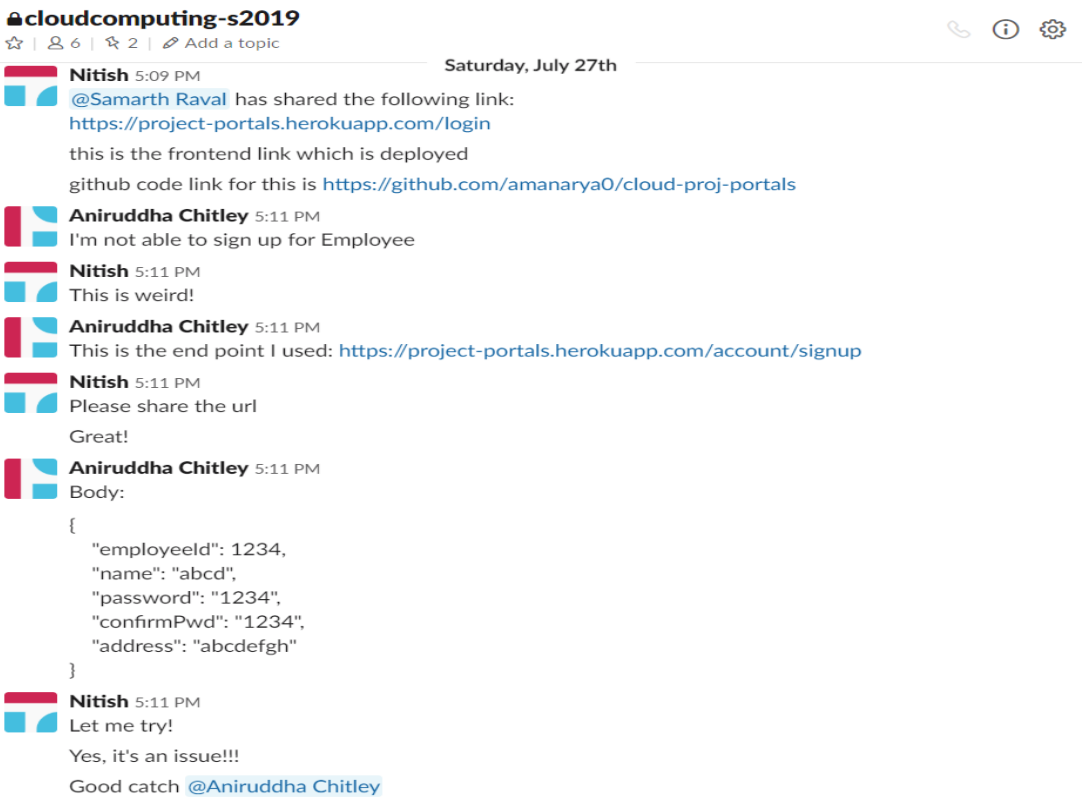


Figure 13: Screenshot 3 for Slack

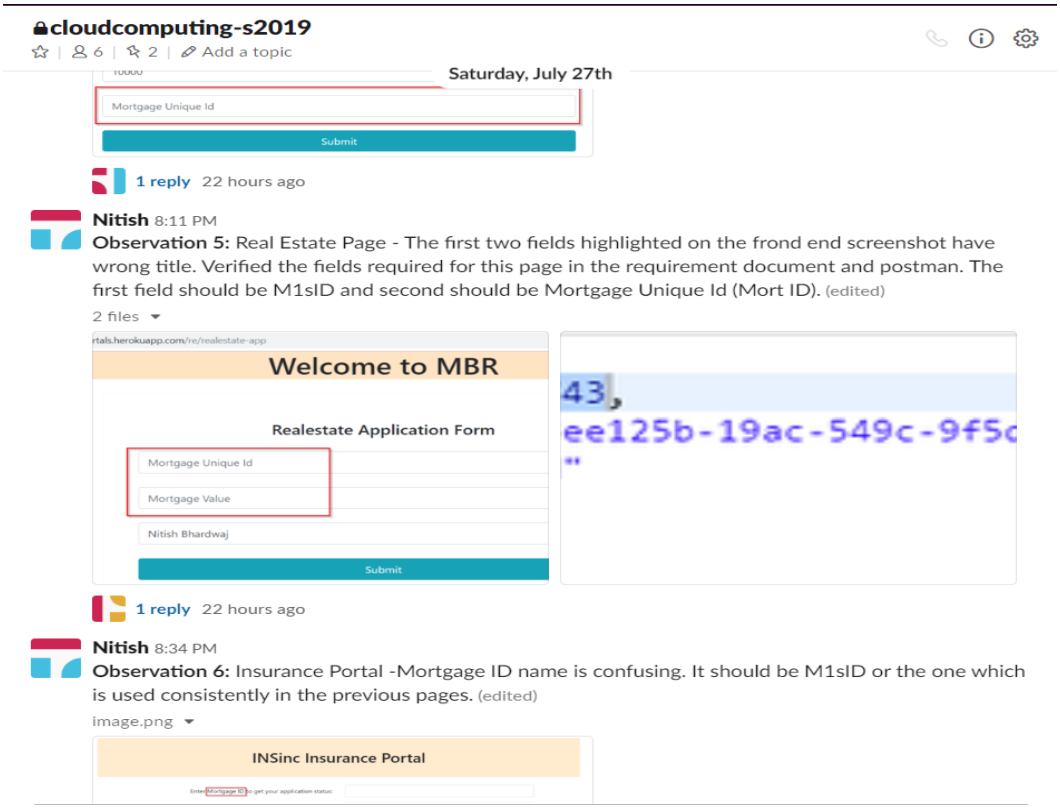


Figure 14: Screenshot 4 for Slack

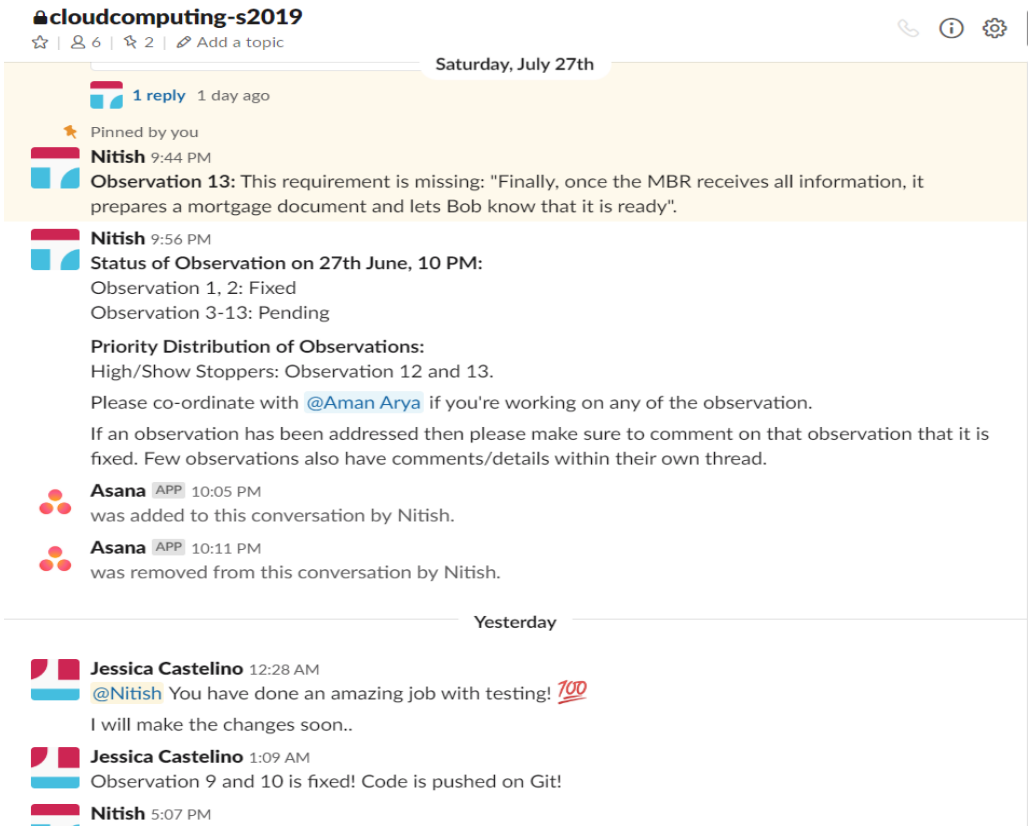


Figure 15: Screenshot 5 for Slack

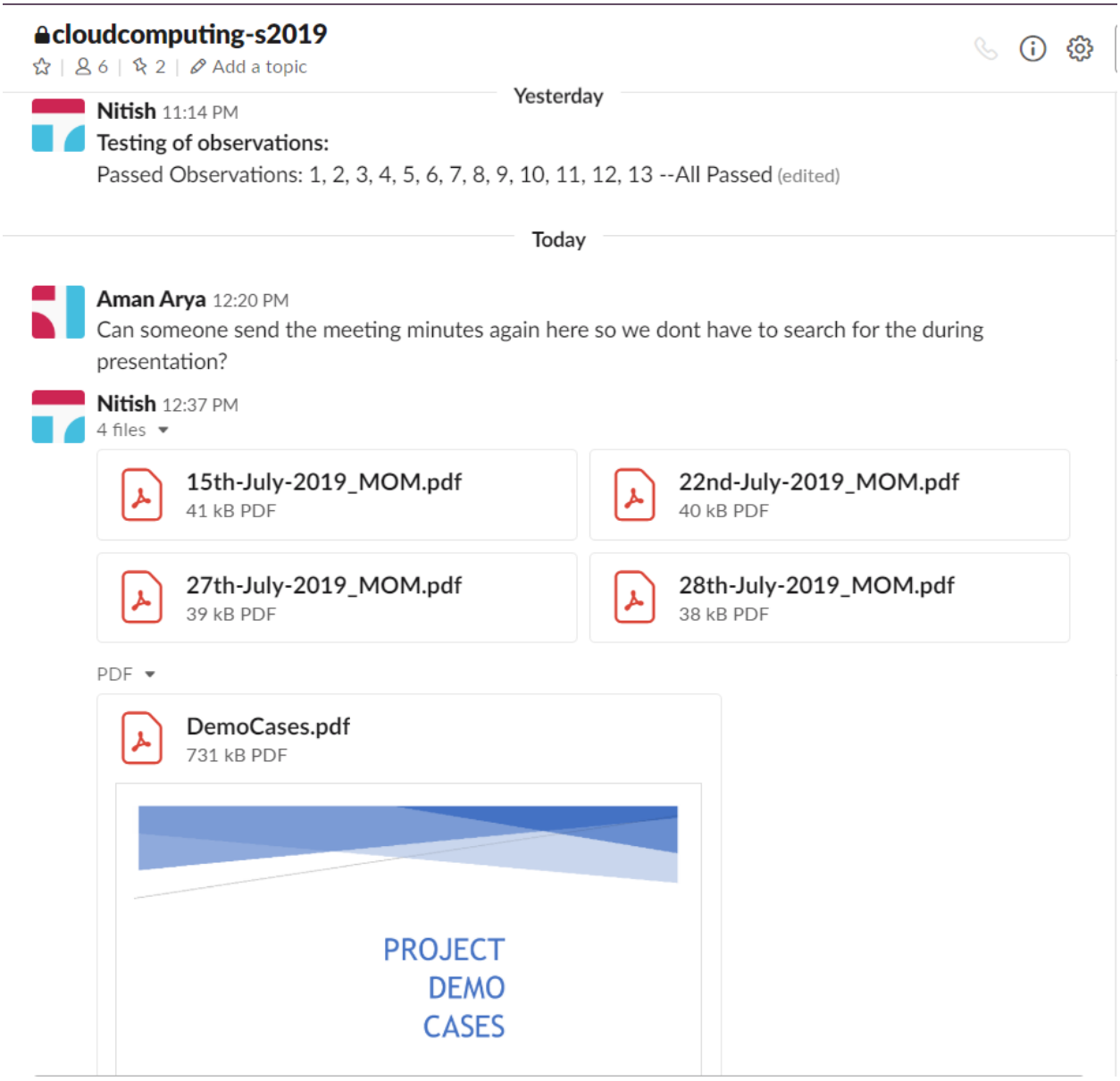


Figure 16: Screenshot 6 for Slack

Figure 17 to 19 shows the screenshot of the Trello board for project management.

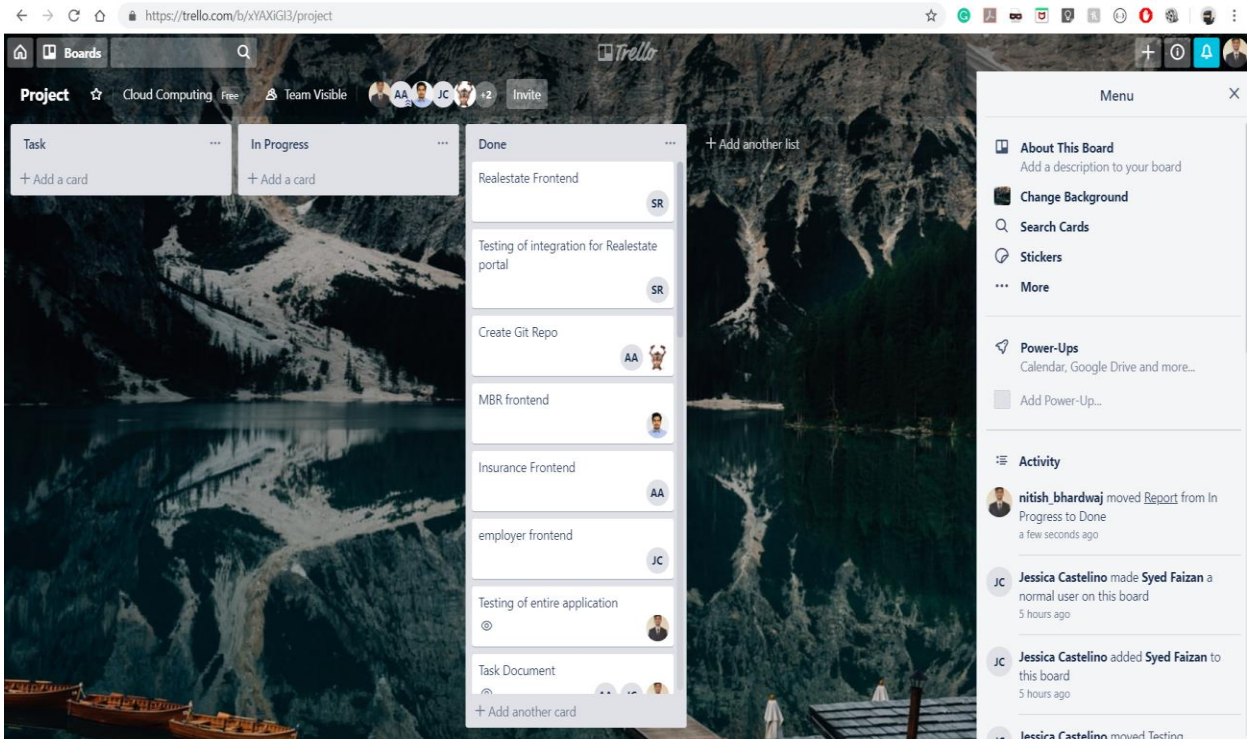


Figure 17: Screenshot 1 for Trello

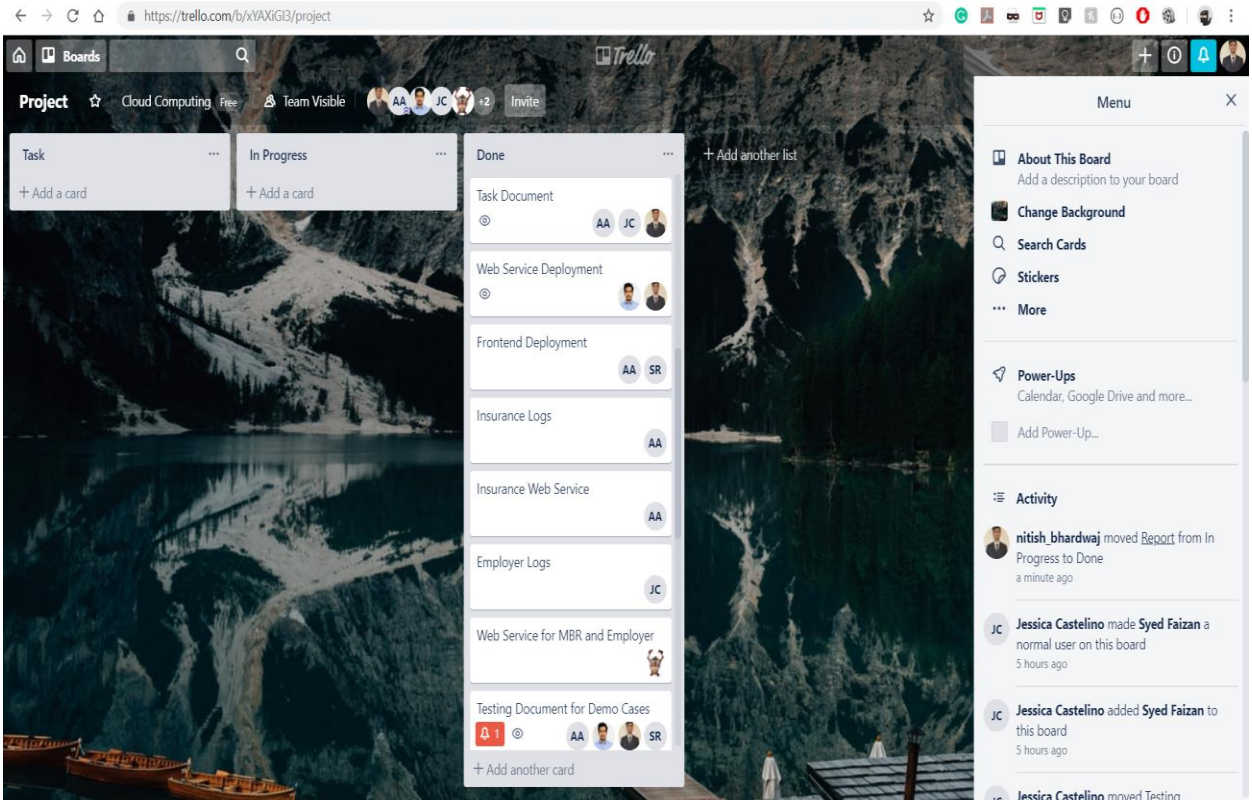


Figure 18: Screenshot 2 for Trello

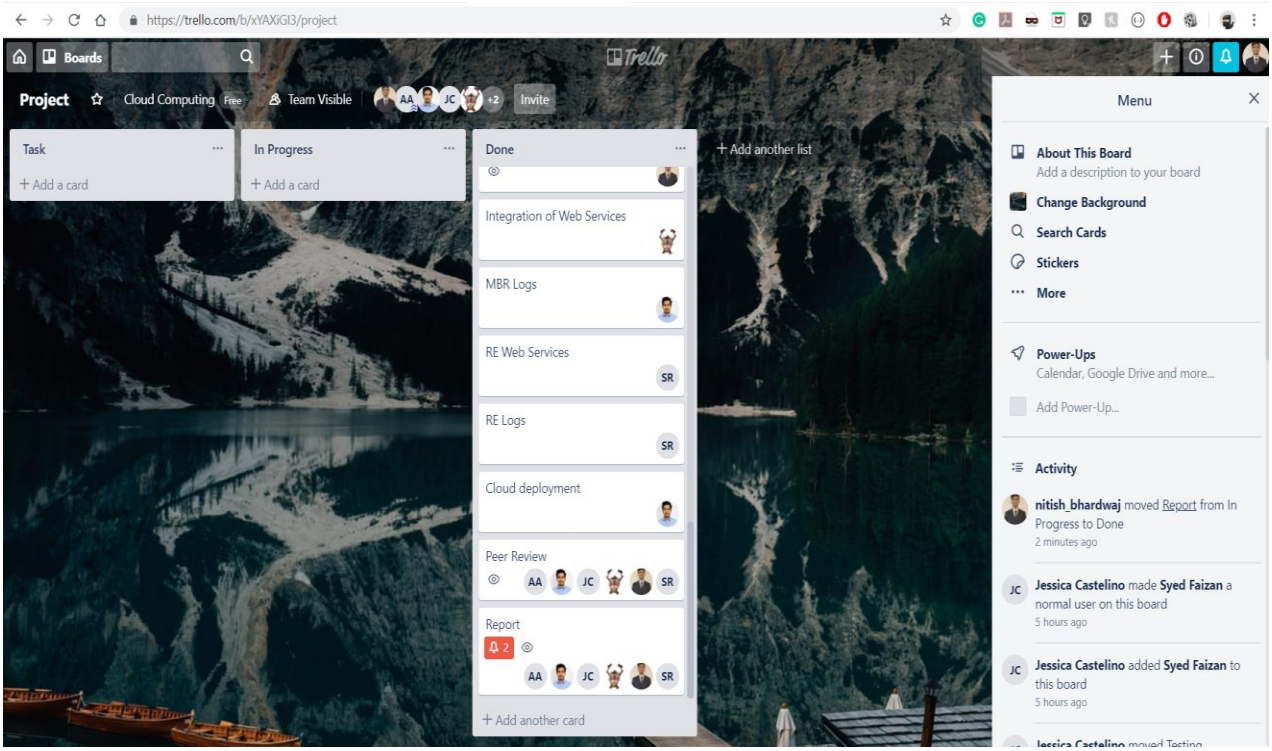


Figure 19: Screenshot 3 for Trello

DEFICIENCIES:

The application is developed as specified in the assignment. The classwork and lab material was sufficient to complete the assignment without any major deficiencies.

The project portal runs on HTTP port instead of HTTPS. Hence, even for the testing purpose, it is requested to run the system on HTTP. As shown in figure 20, please select the “Load unsafe scripts” to perform testing on all the portals.

Figure 20: Screenshot of HTTP portals

However, secure communication using the authentication token has been implemented in the Employer (EMP) portal.

IMPLEMENTATION:

To handle the bulk incoming requests from the cloud service consumer, a third-party reverse proxy server Nginx is used. It is deployed on Virtual machine and acts as the load balancer. It distributes the

incoming requests among the various web services instances running on Virtual Machine behind the Nginx module in round-robin manner so that no server is overloaded with requests.

In order to achieve the scalability, a new Virtual Machine is manually spun up and added to Nginx config, now the subsequent requests are distributed among all the initial web service instances and newly added Virtual Machine.

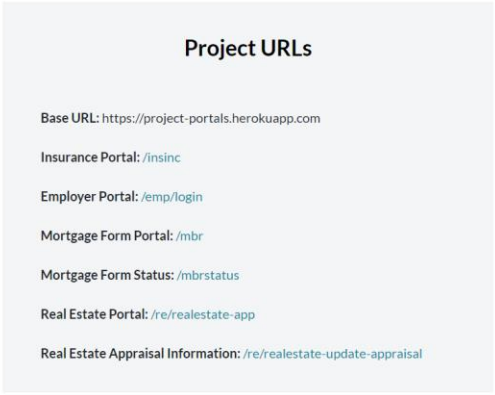
EXPERIMENTATION RESULTS:

This section contains the result of our experiments based on the specifications mentioned the assignment requirement document.

Description of the experiment

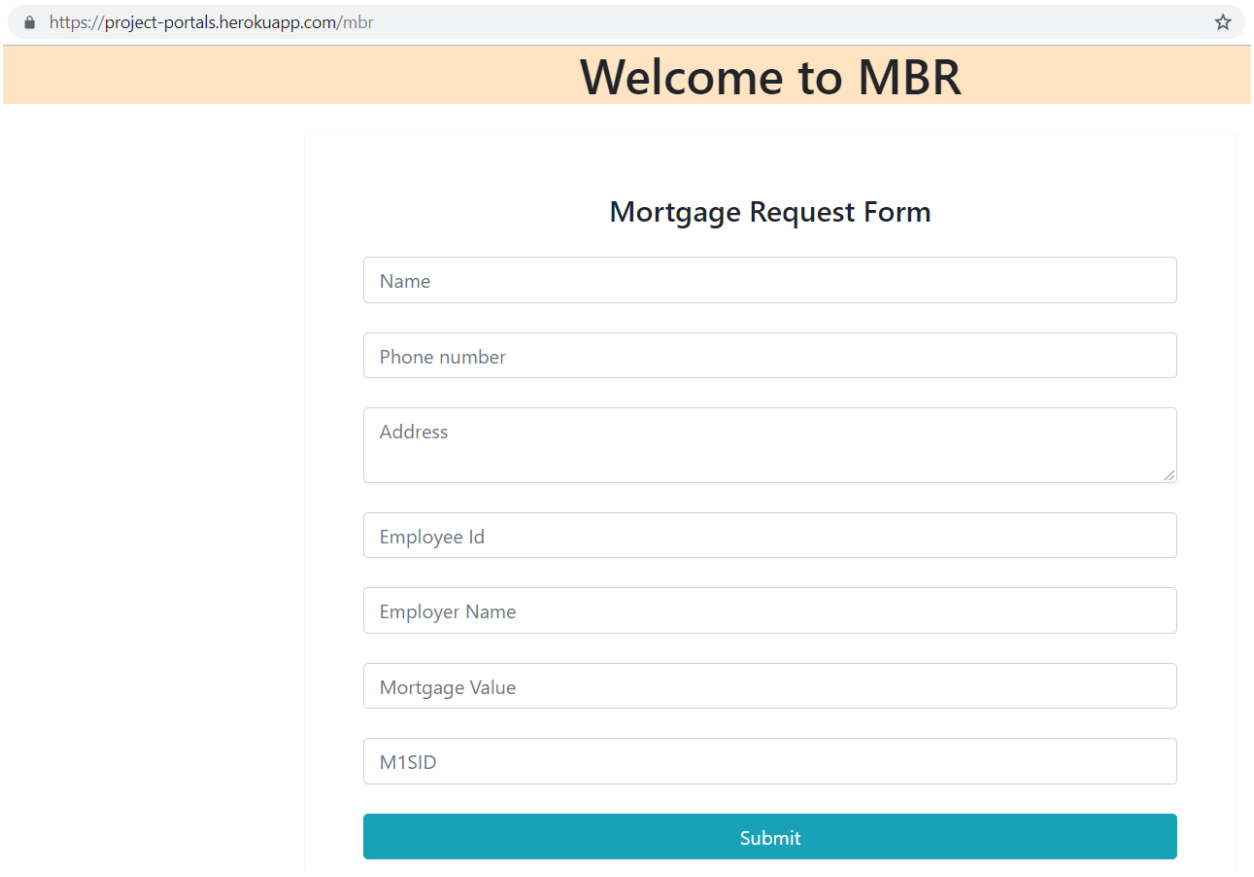
Test Case 1: Verify the main page for the project urls.

Tested Output: Success



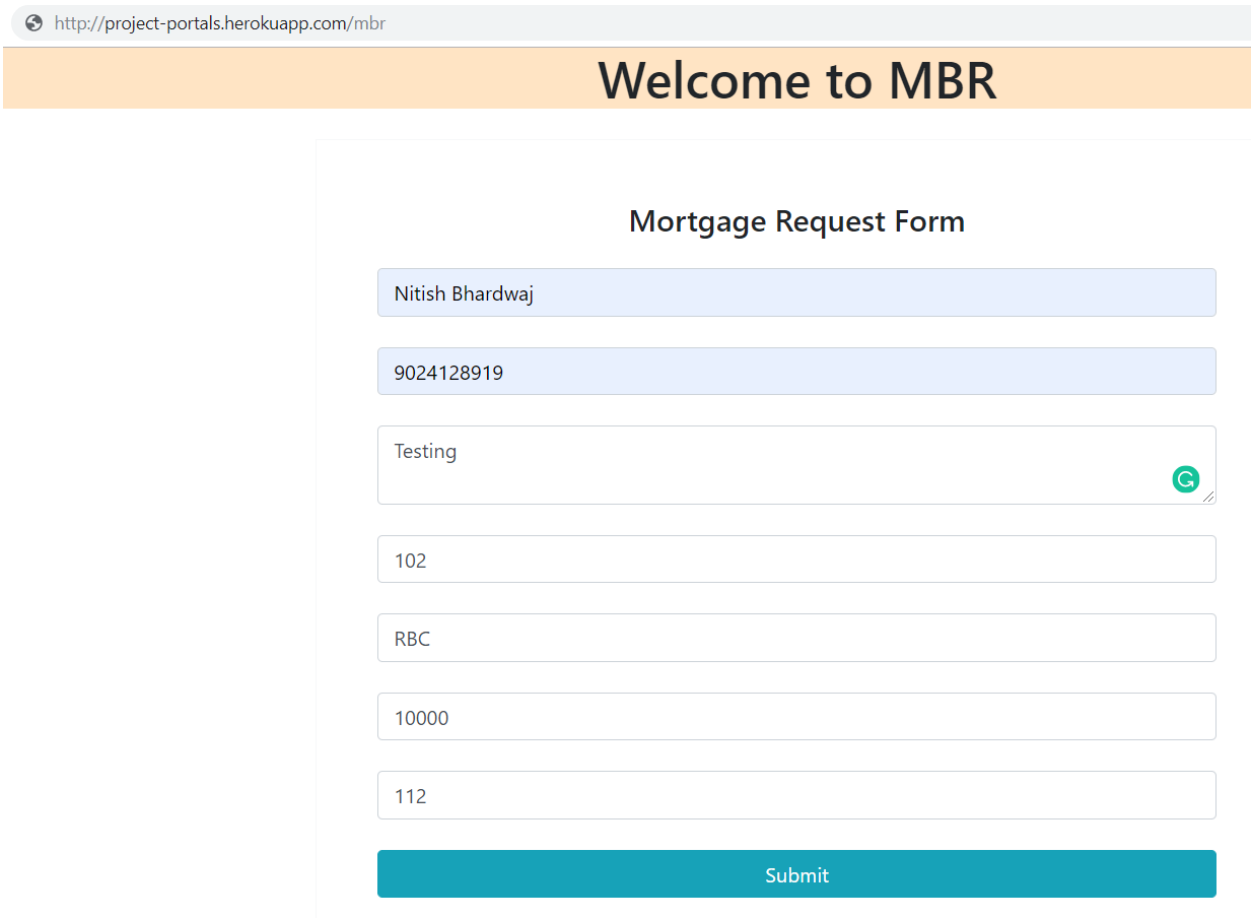
Test Case 2: Verify if the Mortgage Form is being redirected from the main page.

Tested Output: Success



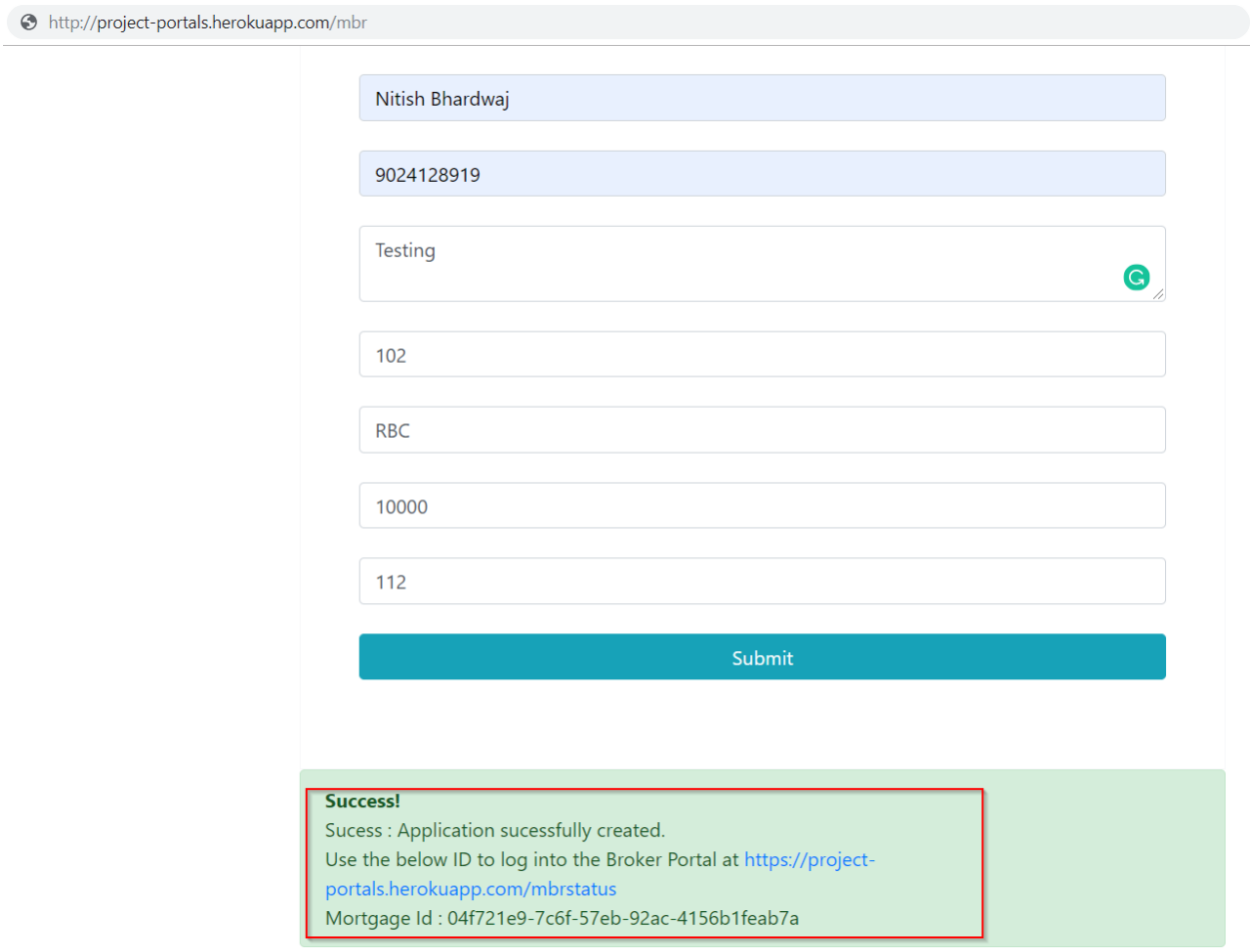
Test Case 3: Fill the details of the MBR and submit the form

Tested Output: Success



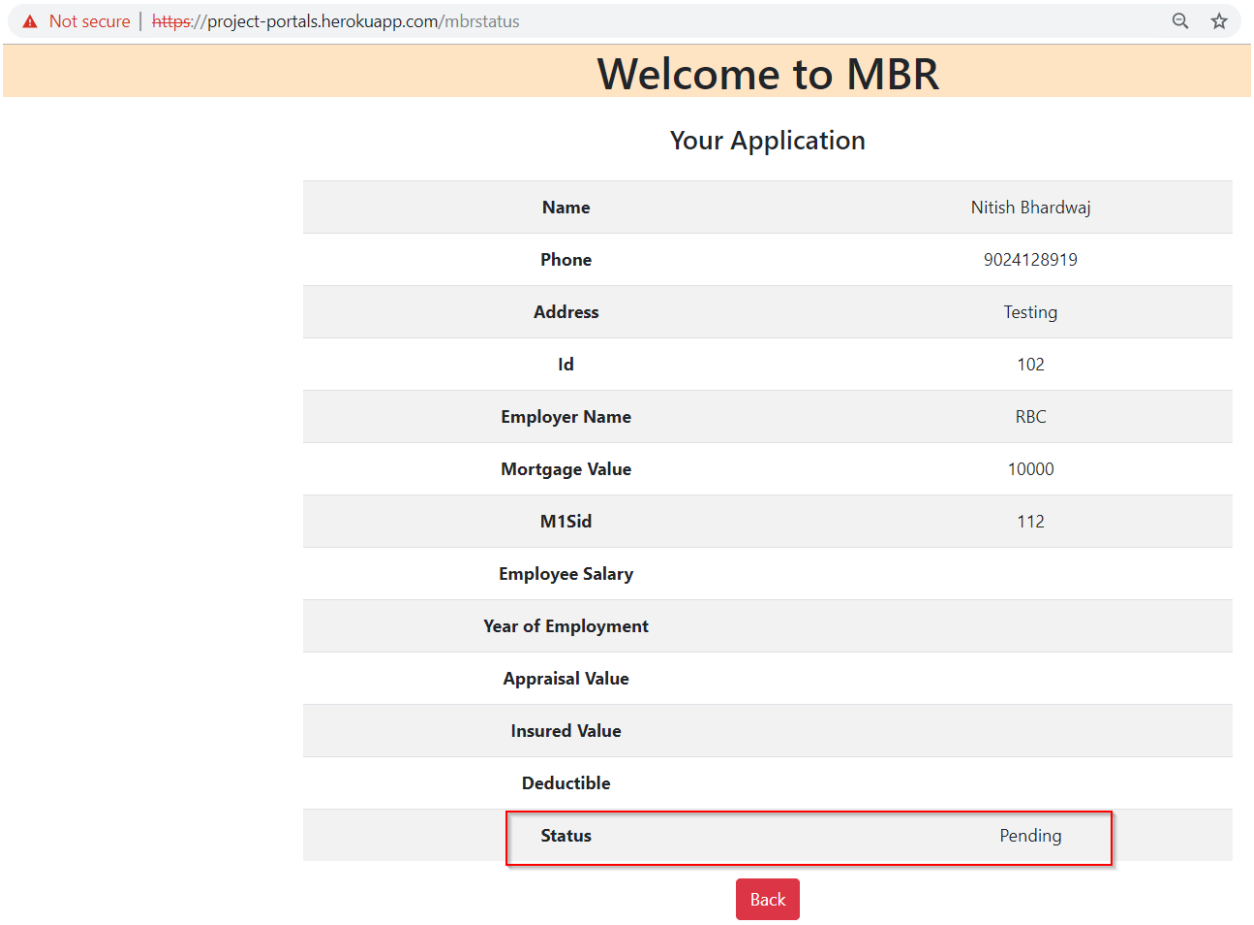
Test Case 4: Verify that the MBR page has shown a success message and the Mortgage id:

Tested Output: Success



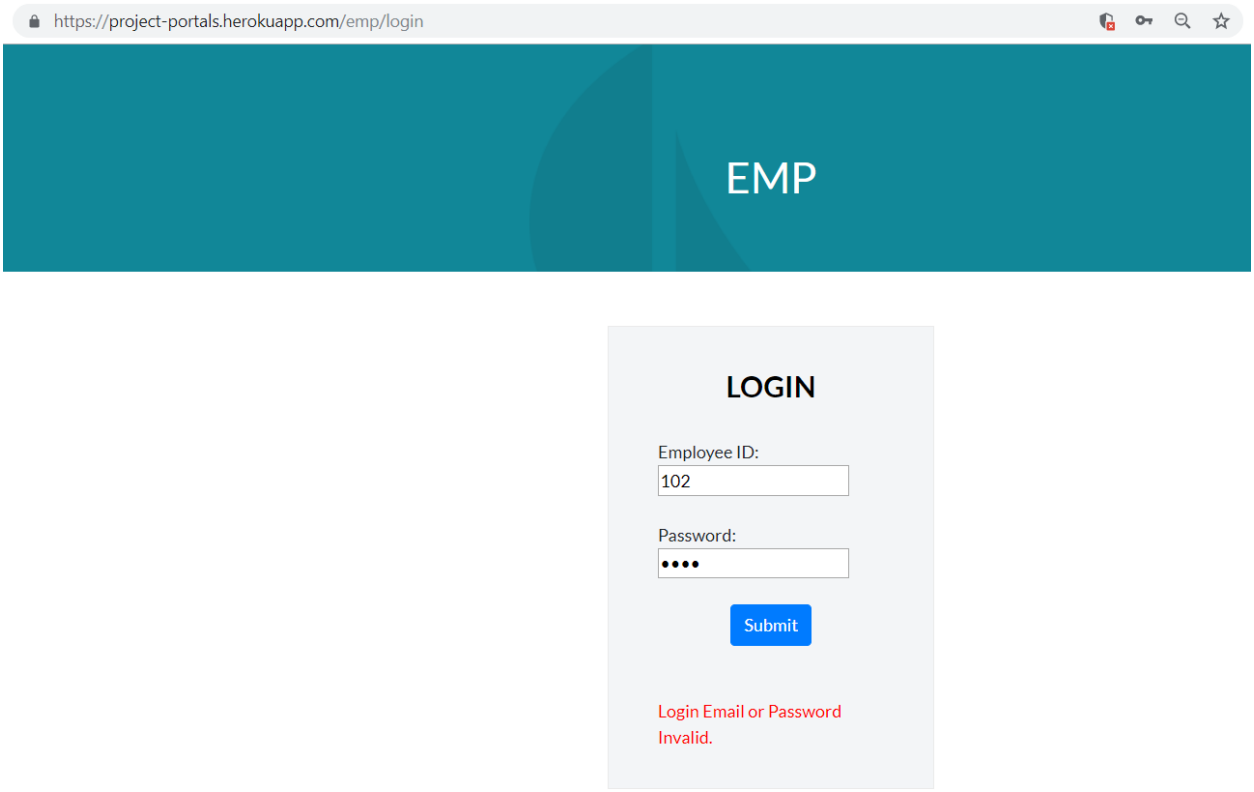
Test Case 5: Check the status of the application. It should be “Pending”.

Tested Output: Success



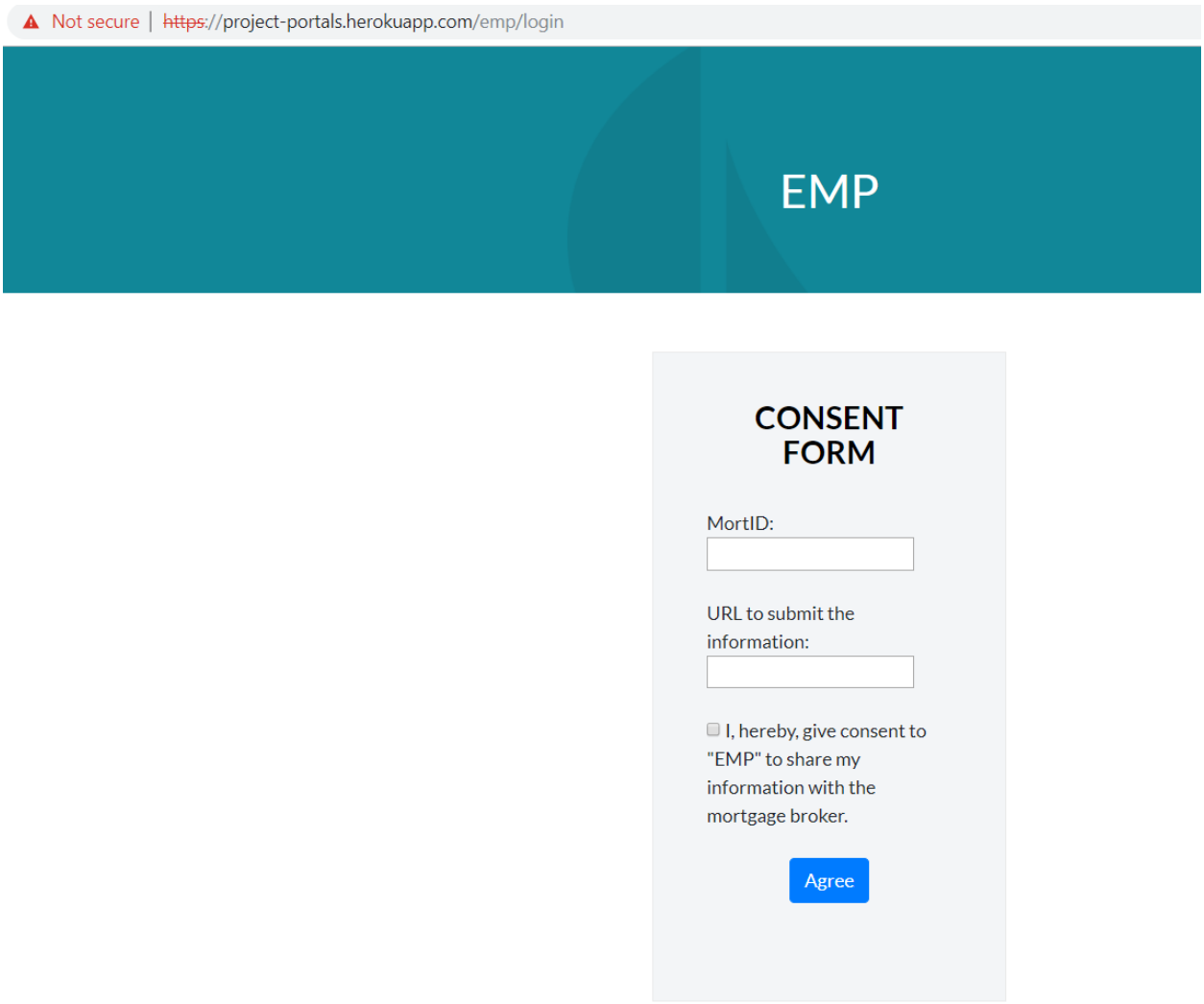
Test Case 6: Login to the Employer portal. Give wrong credentials and verify that a message is shown to the user for the wrong credentials entered.

Tested Output: Success



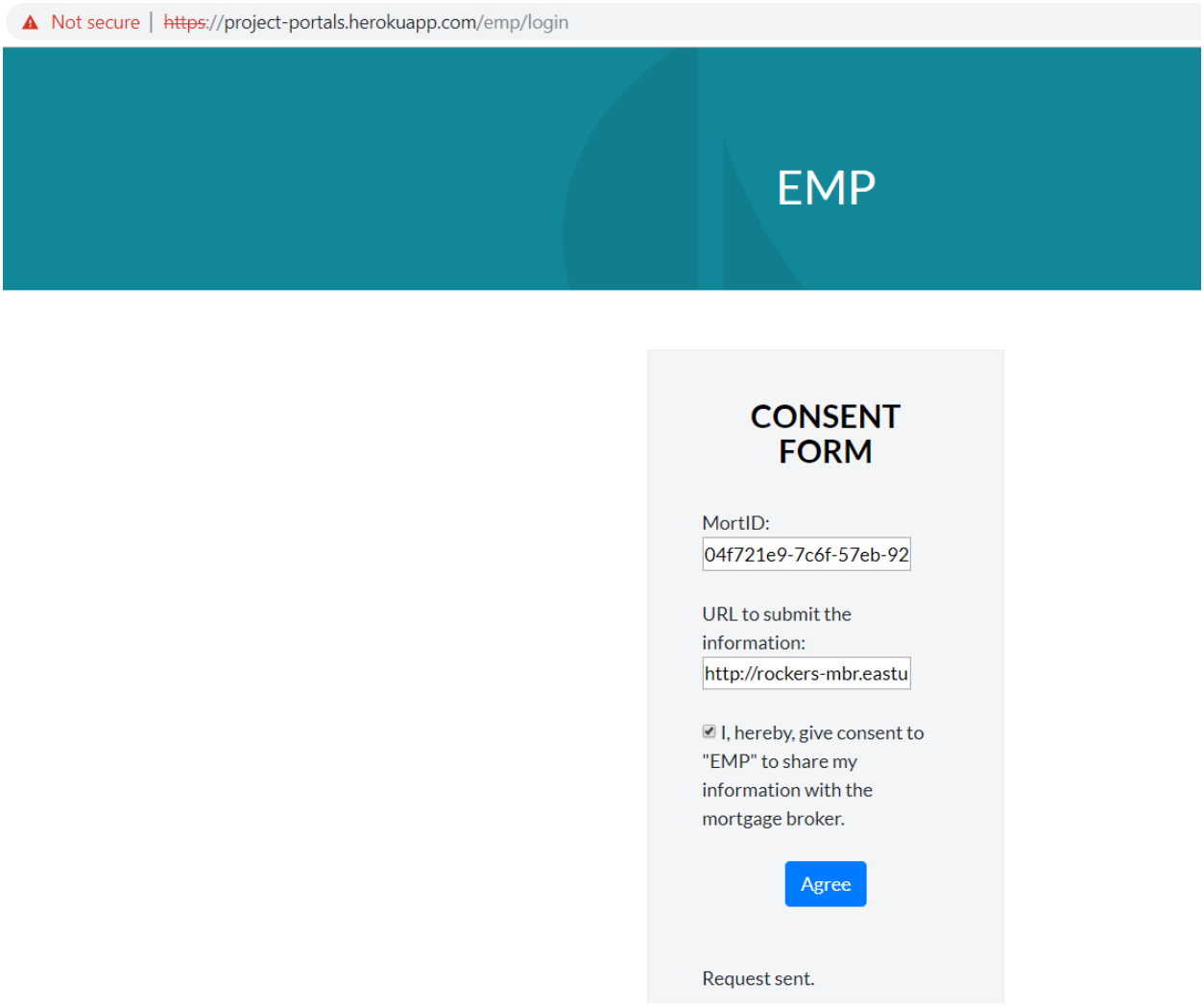
Test Case 7: Login to the Employer portal. Give correct credentials and verify a user is redirected to a page to enter MortID and URL.

Tested Output: Success



Test Case 8: Login to the Employer portal. Provide correct MortID and URL to submit the information. Verify that a success message is displayed to the user.

Tested Output: Success

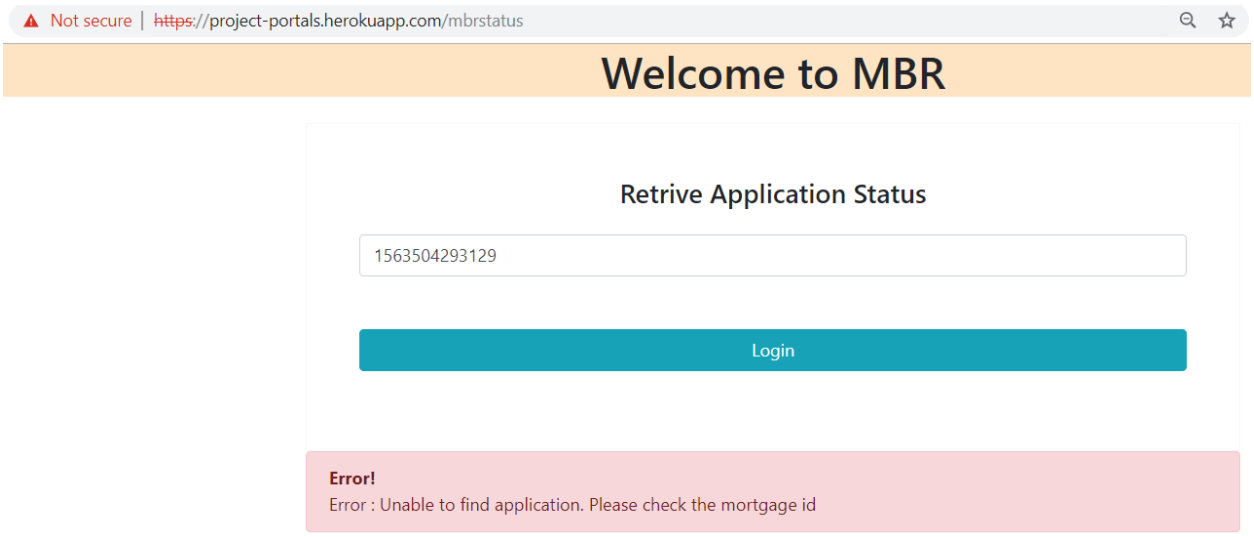


Test Case 9: Check the status of the application. It should be “Pending”. The employee salary and year of employment should be displayed to the user.

Tested Output: Success

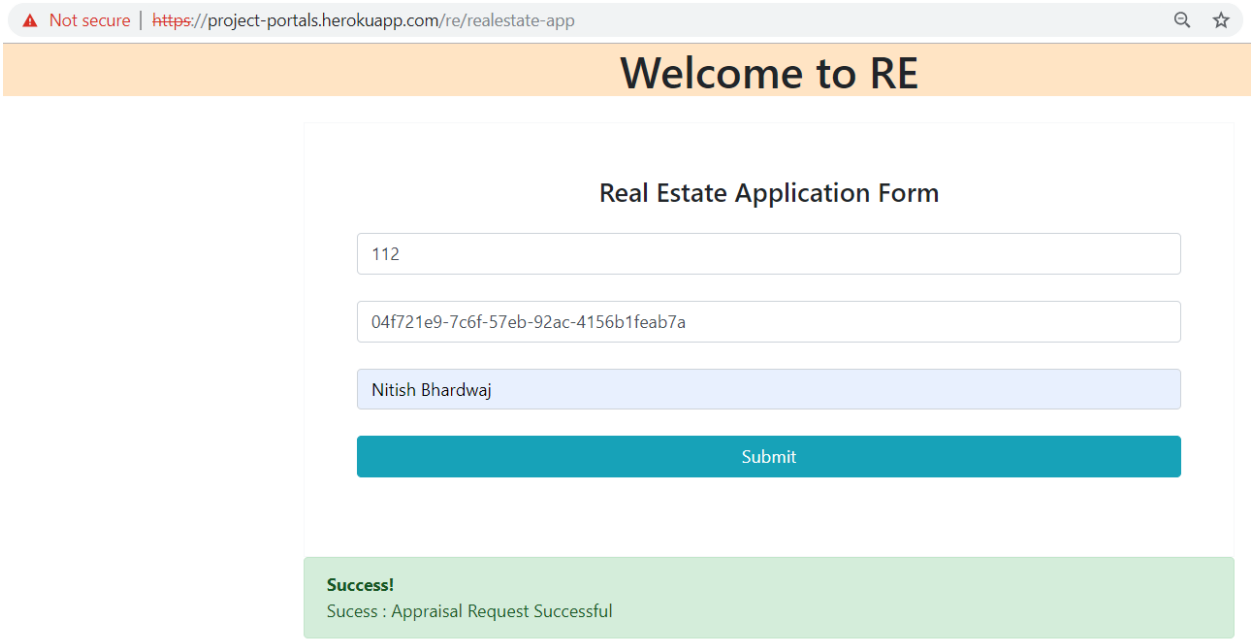
Test Case 10: Check the status of the application with a wrong MortID. User should receive an error that the application wasn’t found.

Tested Output: Success



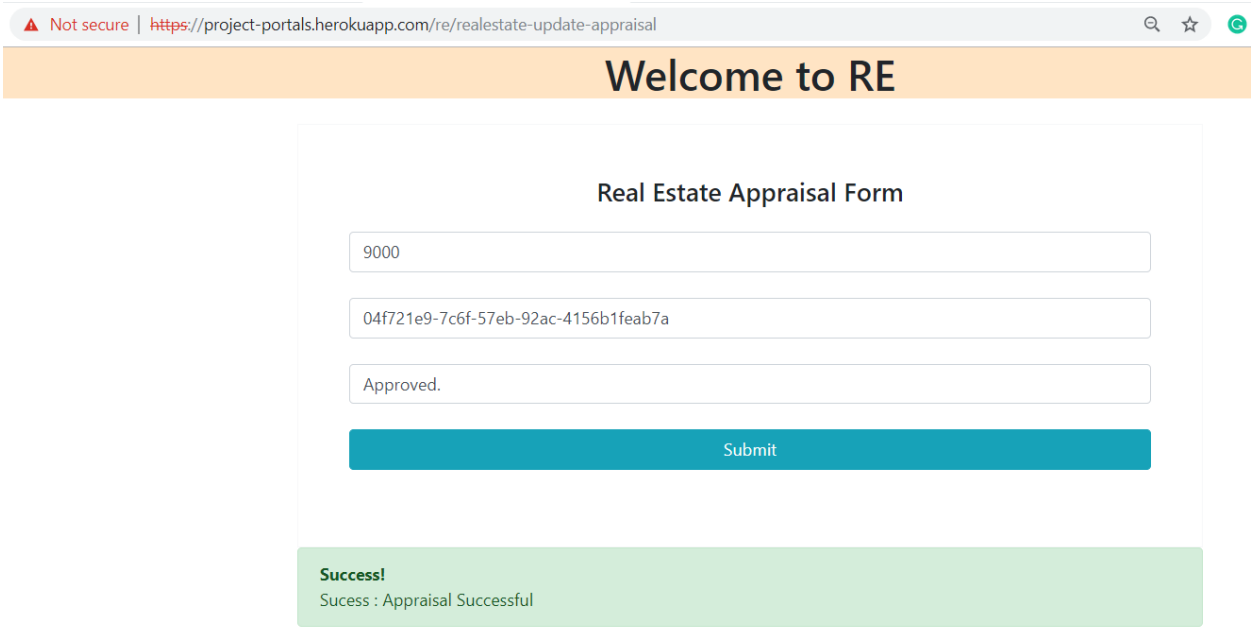
Test Case 11: Go to the Real Estate portal and request for the appraisal of the property.

Tested Output: Success



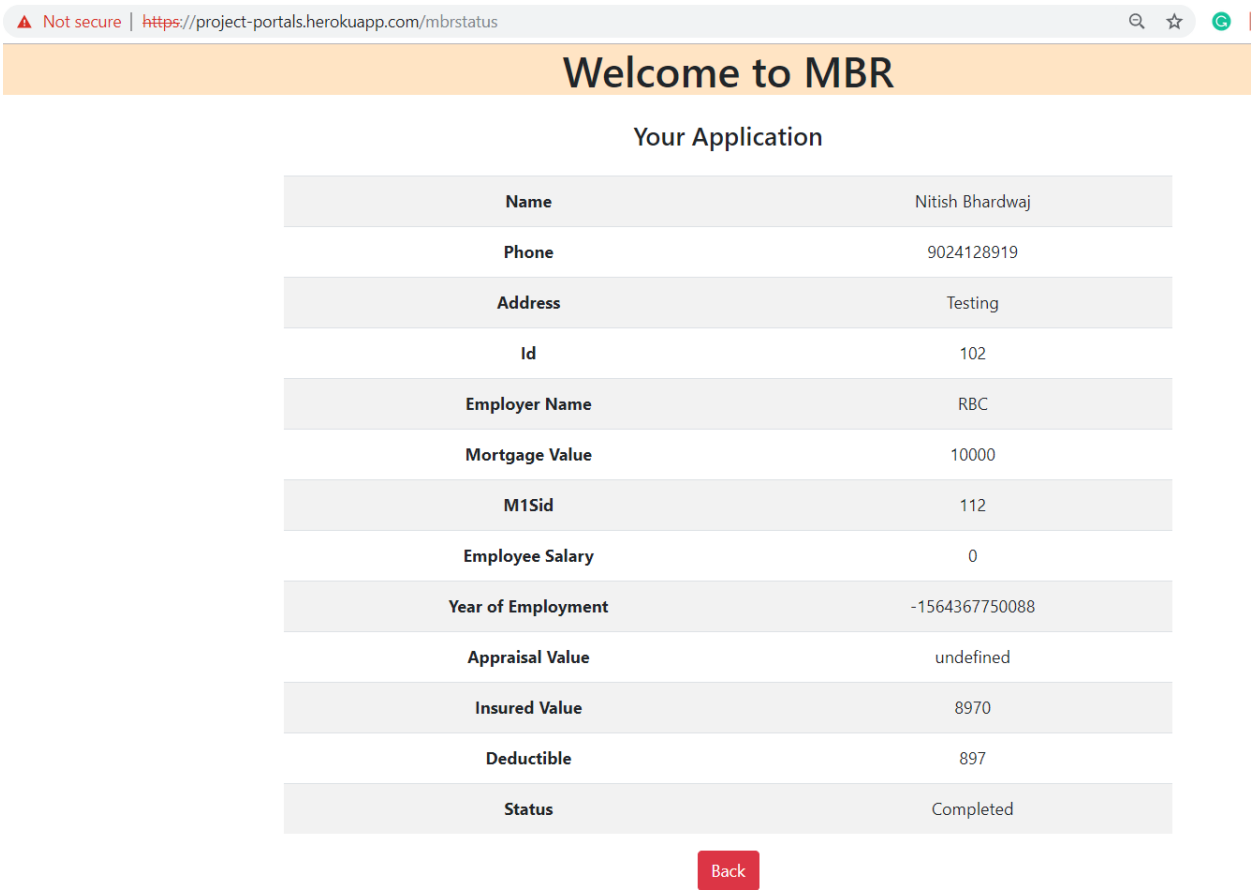
Test Case 12: Go to the Real Estate Appraisal Information page and submit the details including the M1sID and MortID.

Tested Output: Success



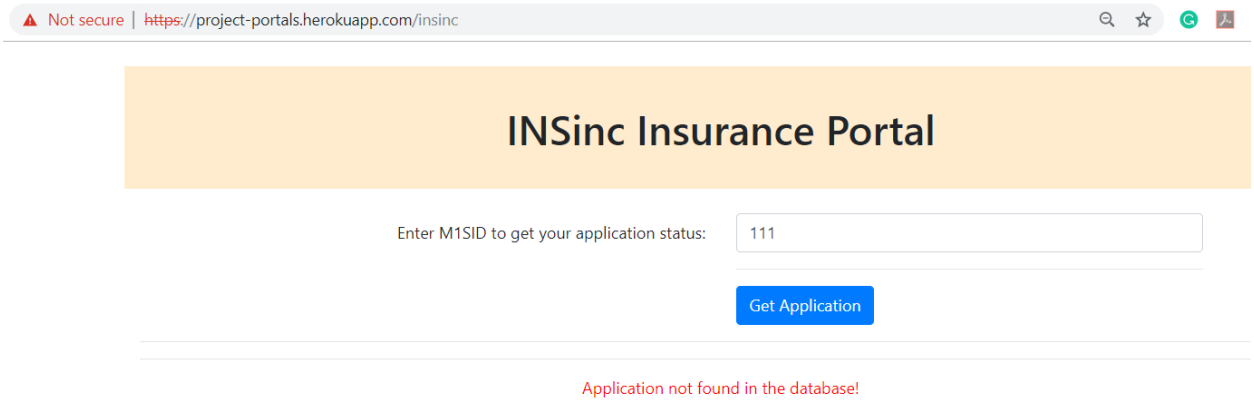
Test Case 13: Check the status of the application. It should be "Completed".

Tested Output: Success



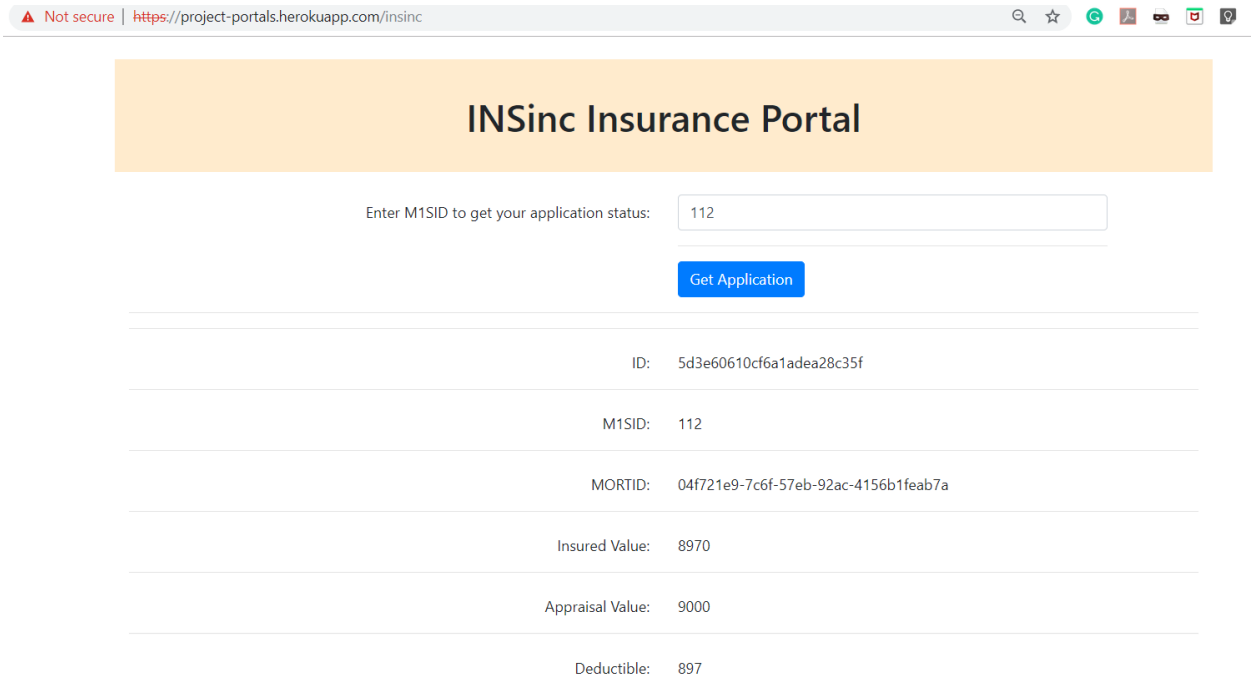
Test Case 14: Go to the Insurance Portal and give the wrong M1sID. Verify that the system shows an error that the application not found.

Tested Output: Success



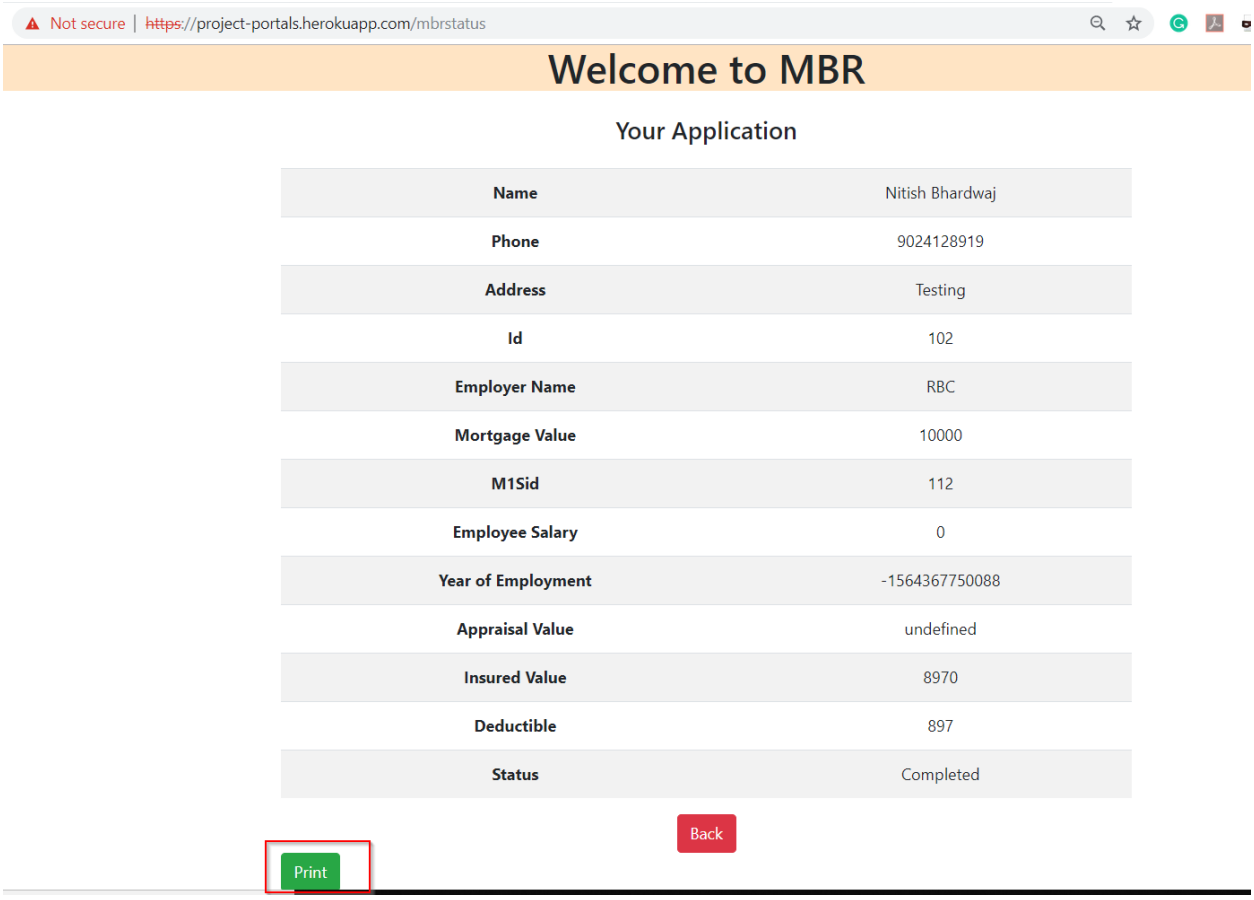
Test Case 15: Go to the Insurance Portal and give correct M1sID. Verify that the system shows an error that the application not found.

Tested Output: Success



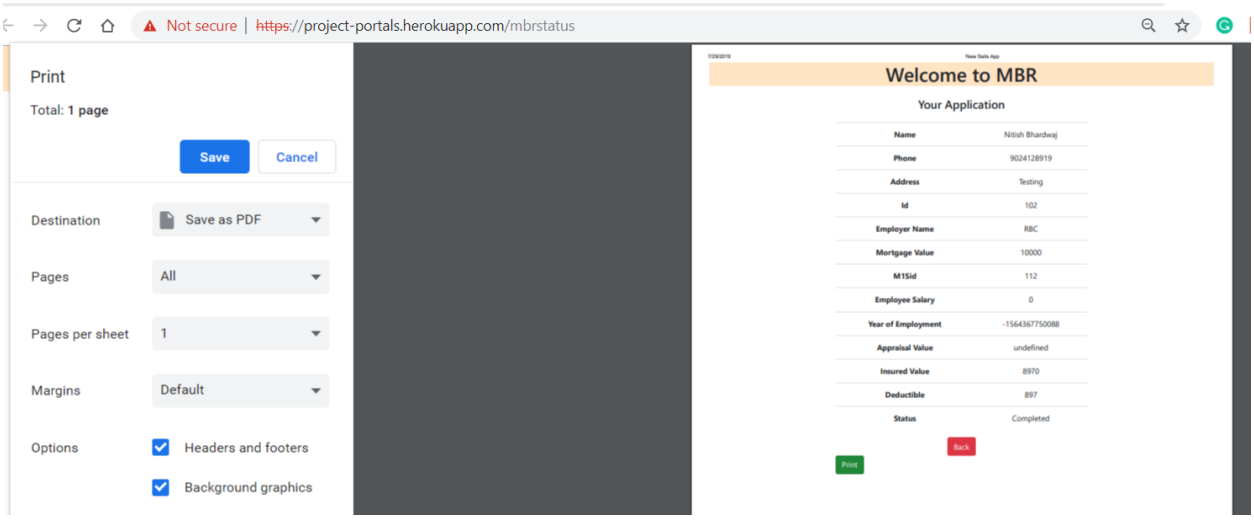
Test Case 16: Check the status of the application. It should be “Completed”. An option to download the mortgage document should be shown to the user.

Tested Output: Success



Test Case 18: Based on Test Case 17, click on Print button and verify that the print option of document is displayed.

Tested Output: Success



REFERENCES:

[1] N. Bhardwaj, "Assignment 6_Group-Rockers", Halifax, 2019.

[2] "UCP System requirements", Docker Documentation, 2019. [Online]. Available: https://docs.docker.com/v17.09/datacenter/ucp/2.1/guides/admin/install/system-requirements/#hardware-and-software-requirements. [Accessed: 27- Jul- 2019].

[3] L. Perrin, "express logging response body", Stack Overflow, 2013. [Online]. Available: https://stackoverflow.com/a/19215370. [Accessed: 29- Jul- 2019].

APPENDIX:

CLIENT CODE

HOMEPAGE

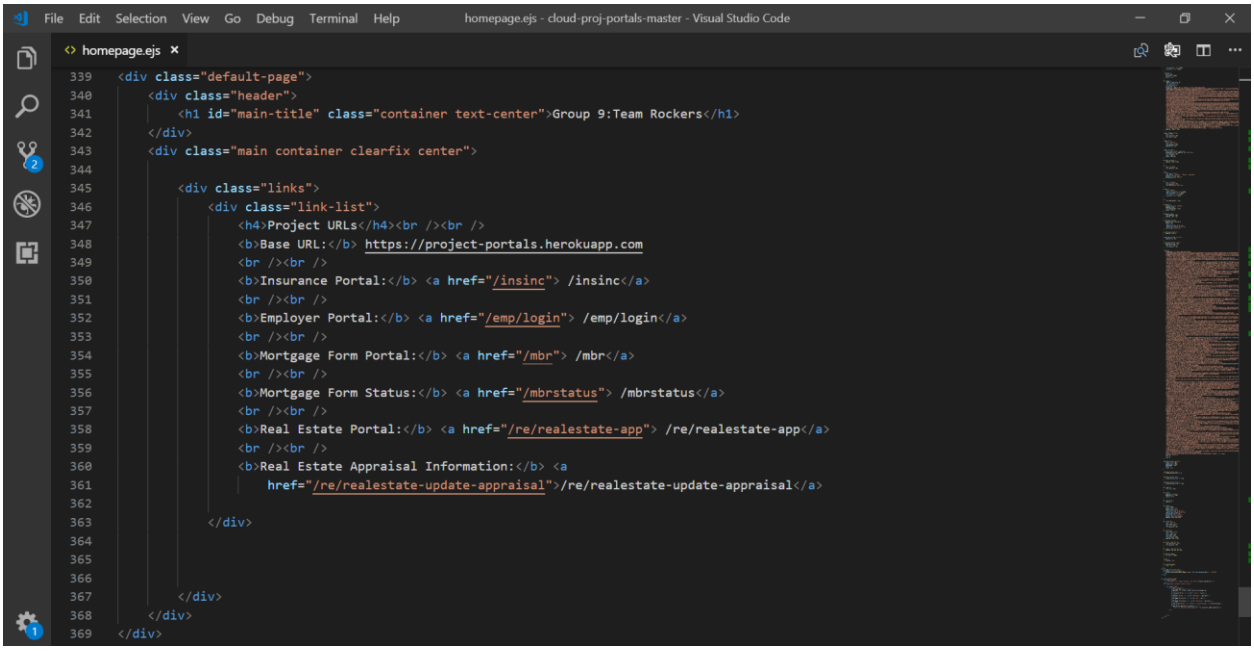


Figure 21: homepage.ejs (Provides links to various portals of the projects)

MBR PORTAL

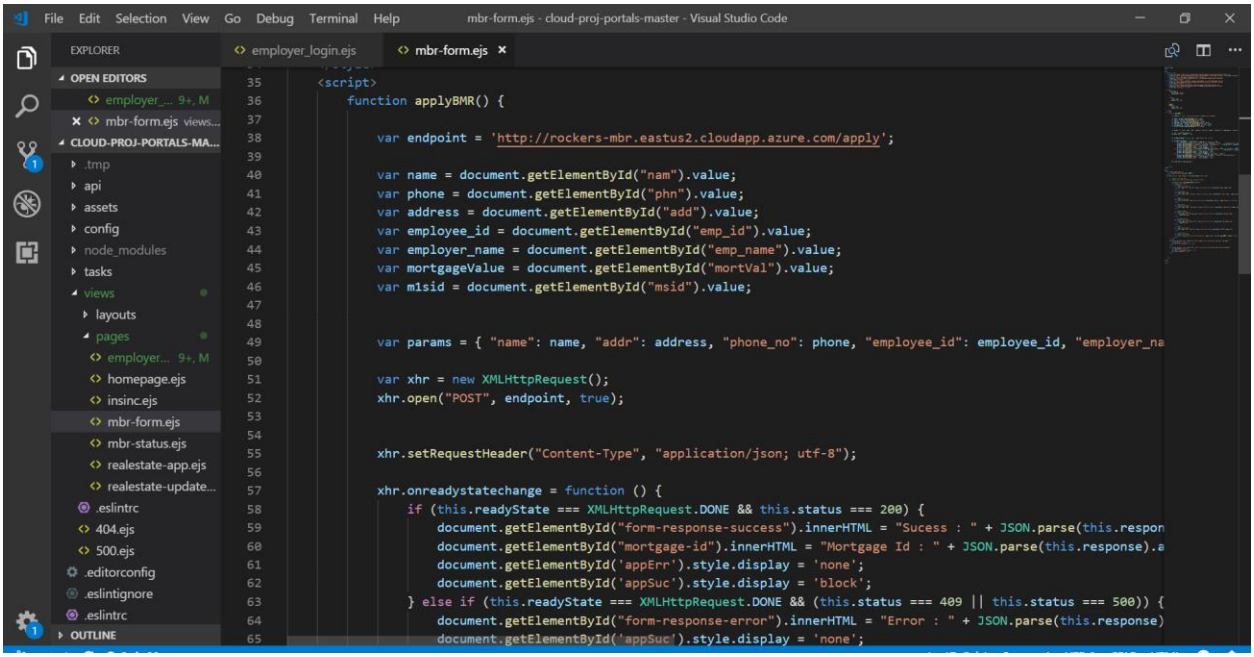


Figure 22: mbr-form.ejs (Form for mortgage application)

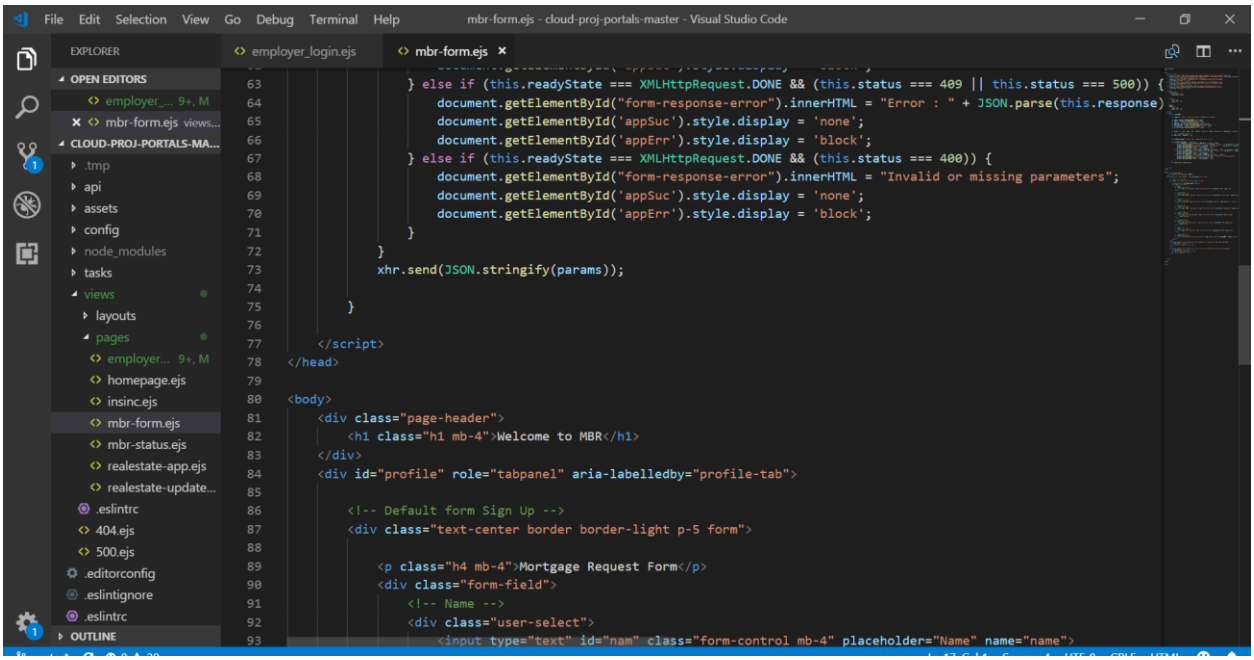


Figure 23: mbr-form.ejs (contd.)

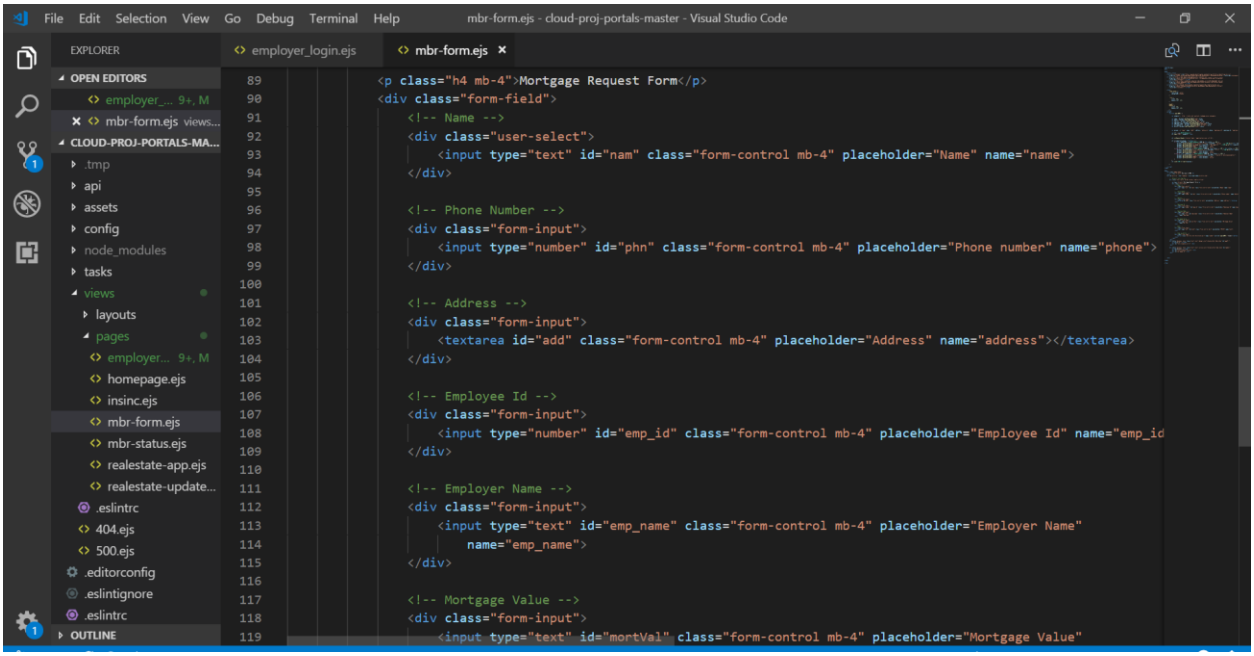


Figure 24: mbr-form.ejs (contd.)

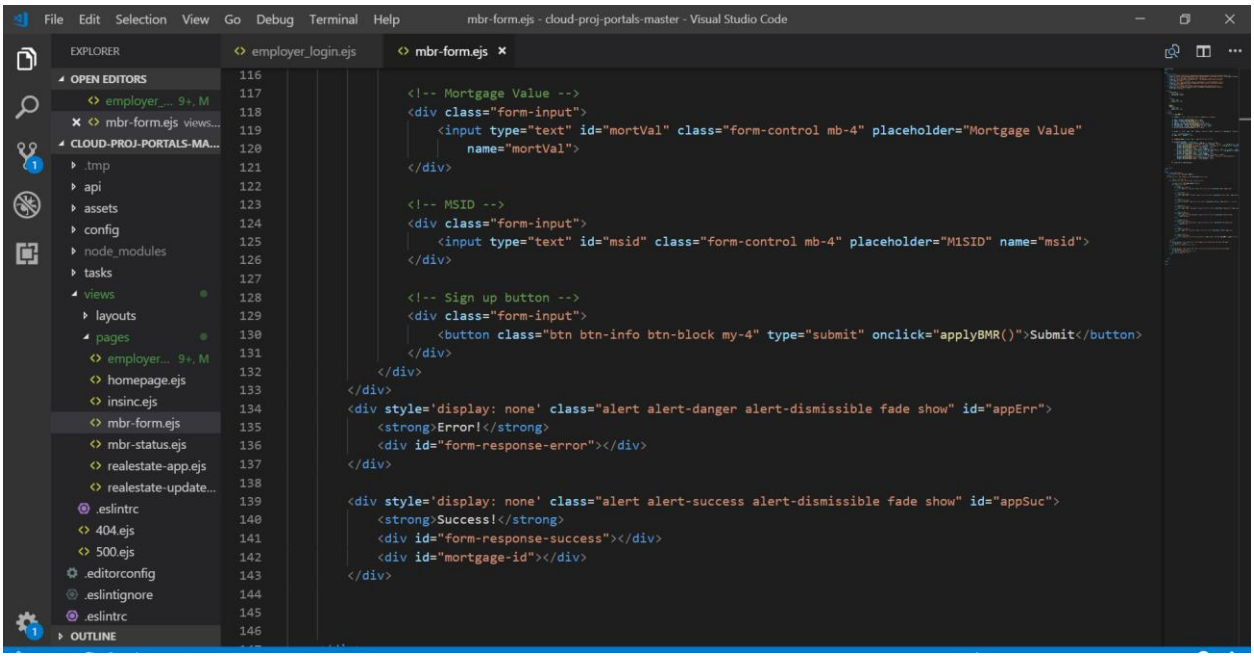


Figure 25: mbr-form.ejs (contd.)

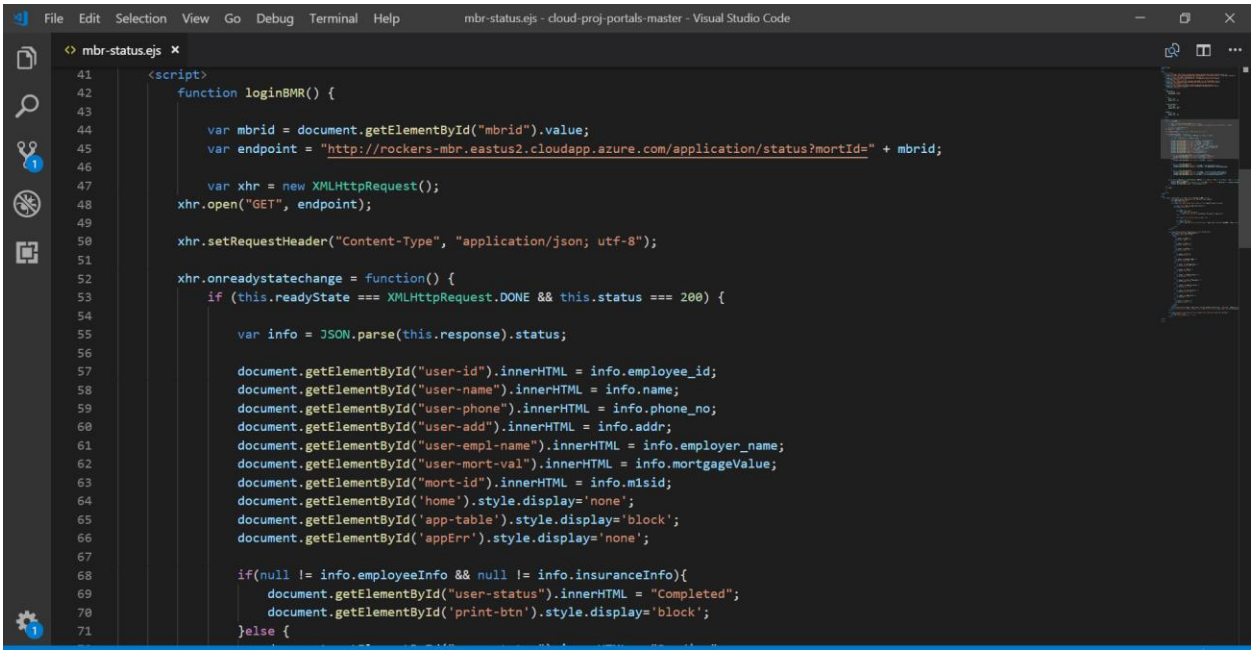


Figure 26: mbr-status.ejs (Portal to check the status of the mortgage application)

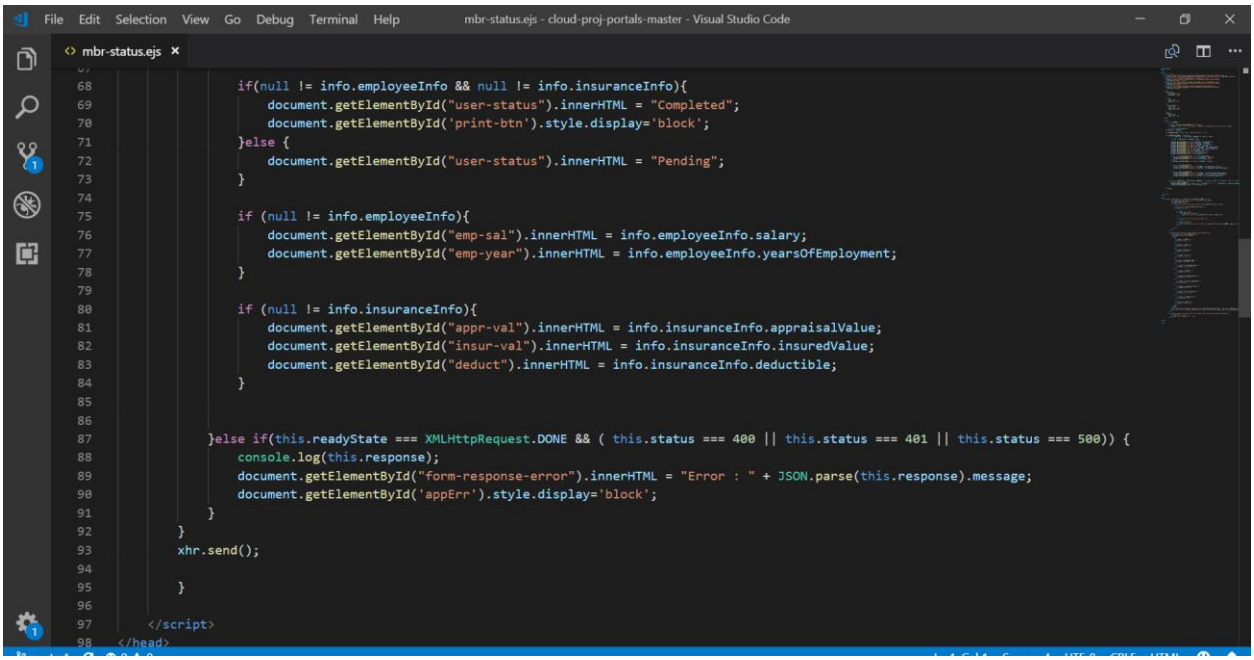


Figure 27: mbr-status.ejs (contd.)

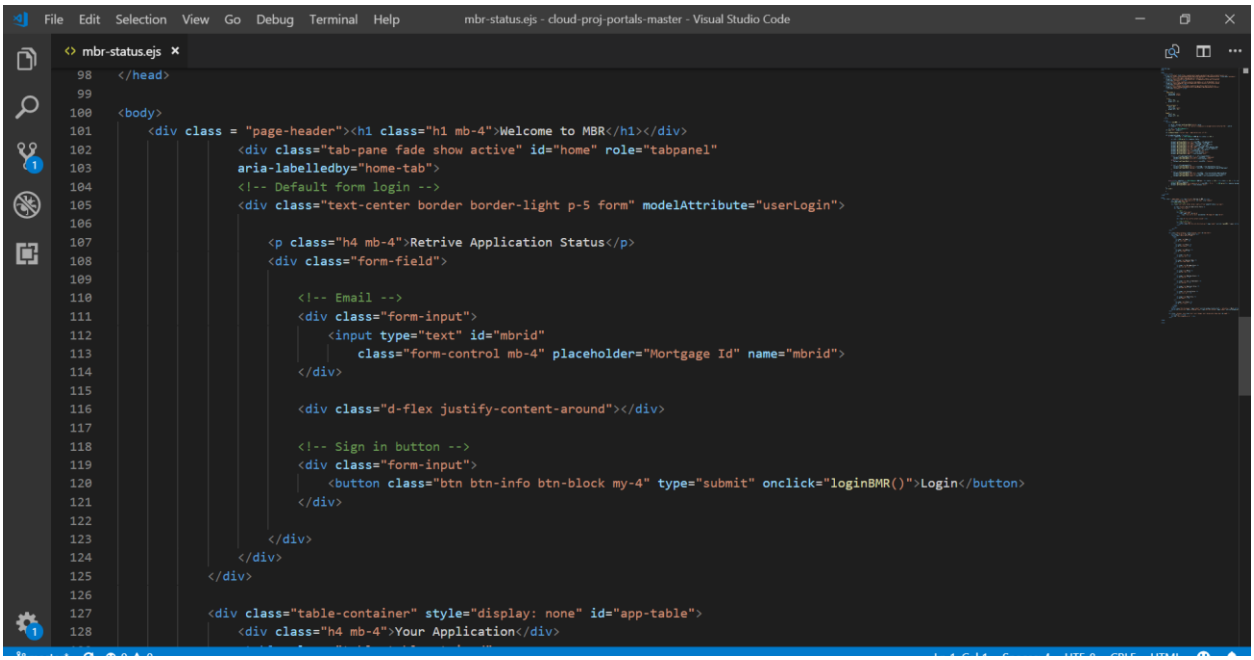


Figure 28: mbr-status.ejs (contd.)

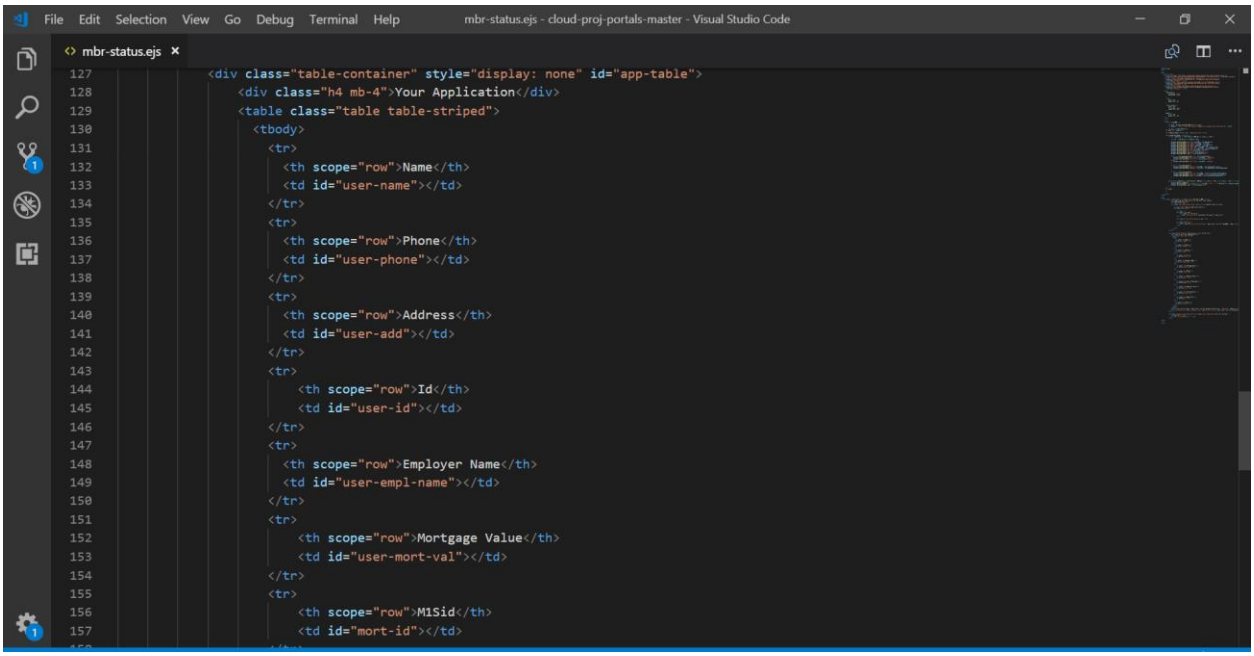


Figure 29: mbr-status.ejs (contd.)

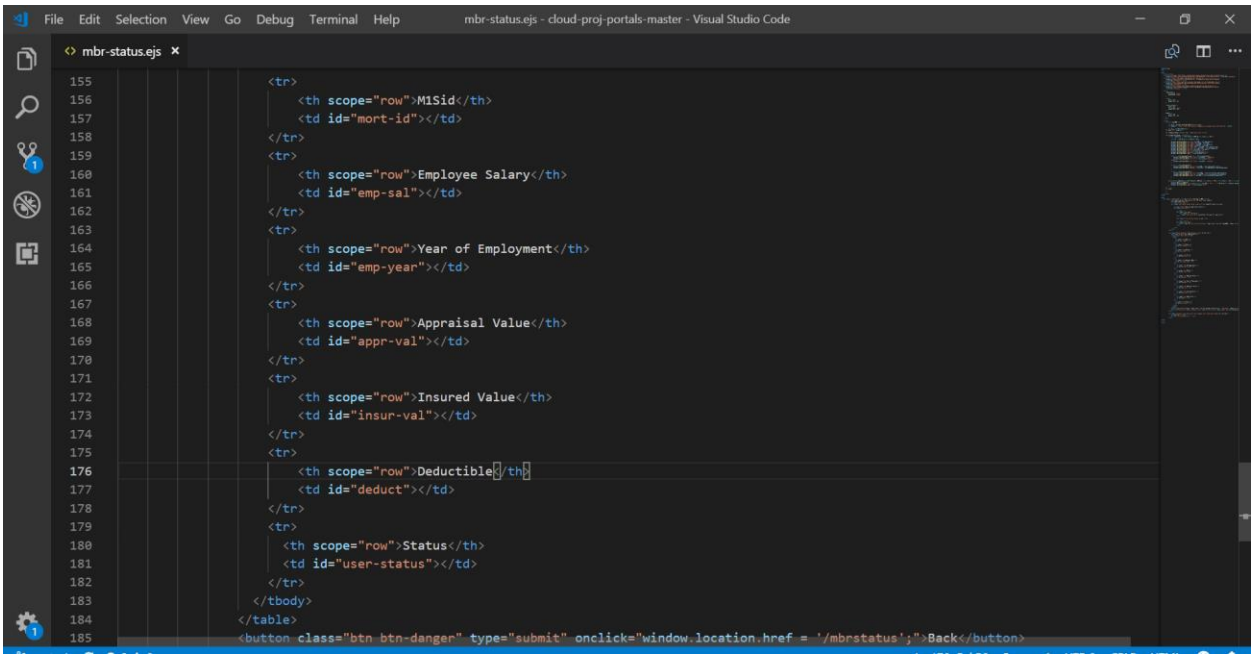


Figure 30: mbr-status.ejs (contd.)

EMPLOYER PORTAL

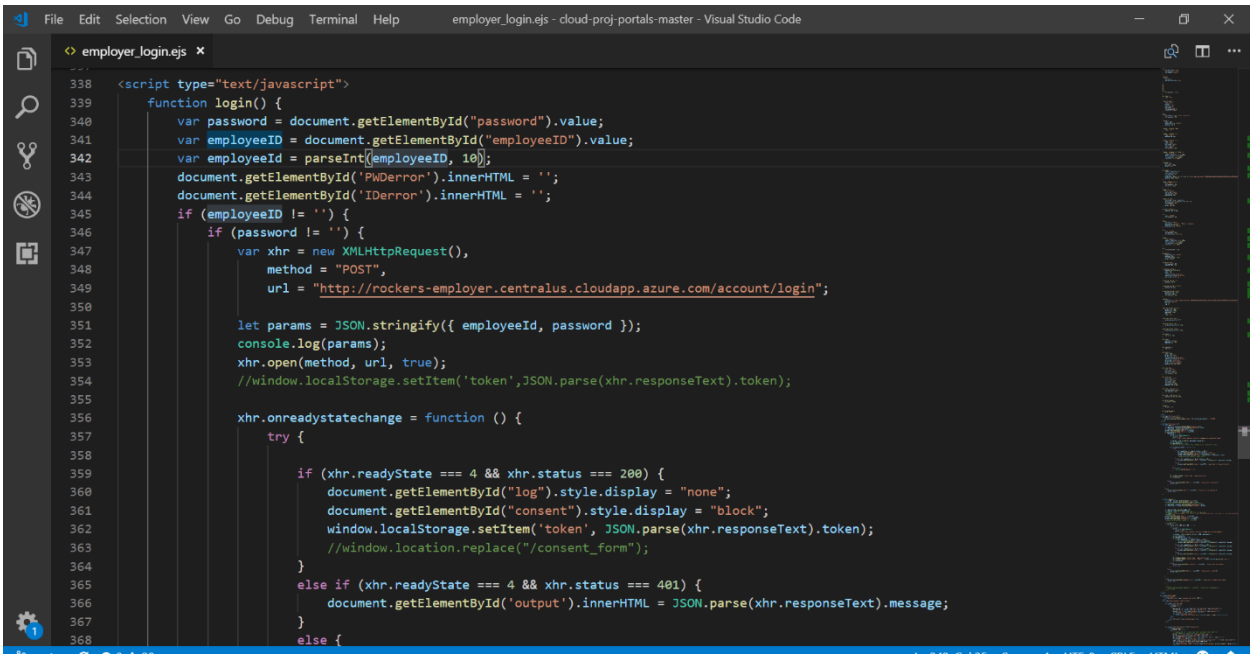


Figure 31: employer-login.ejs (Employer portal)

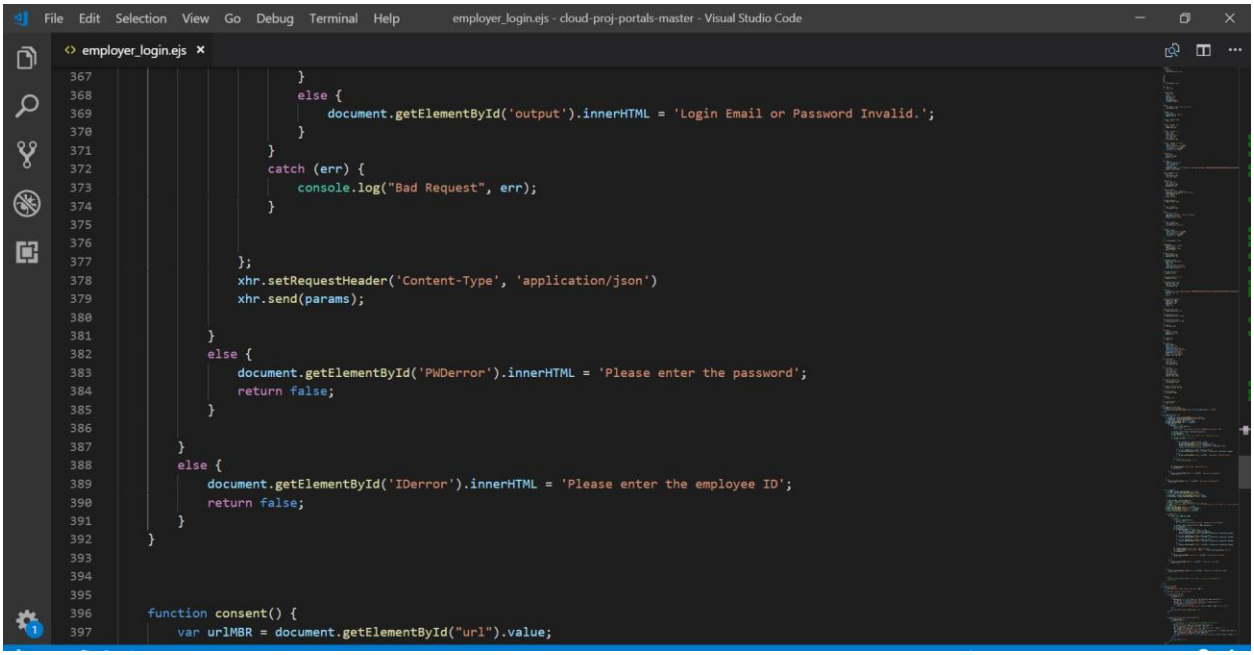


Figure 32: employer-login.ejs (contd.)

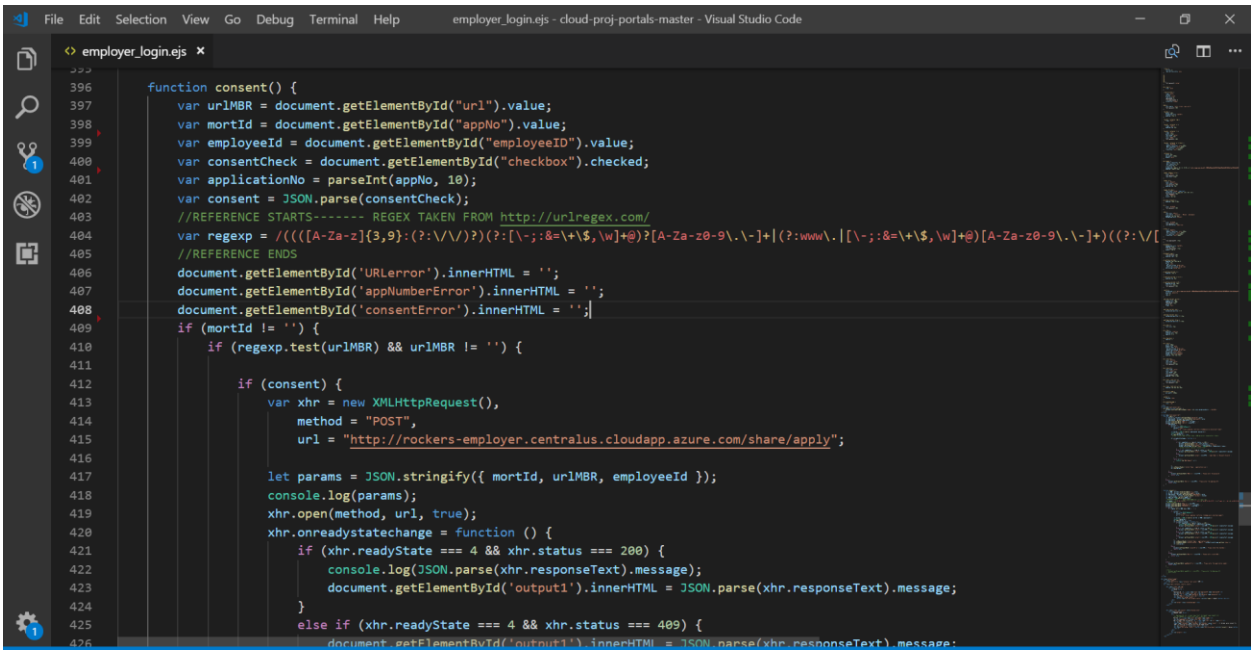


Figure 33: employer-login.ejs (contd.)

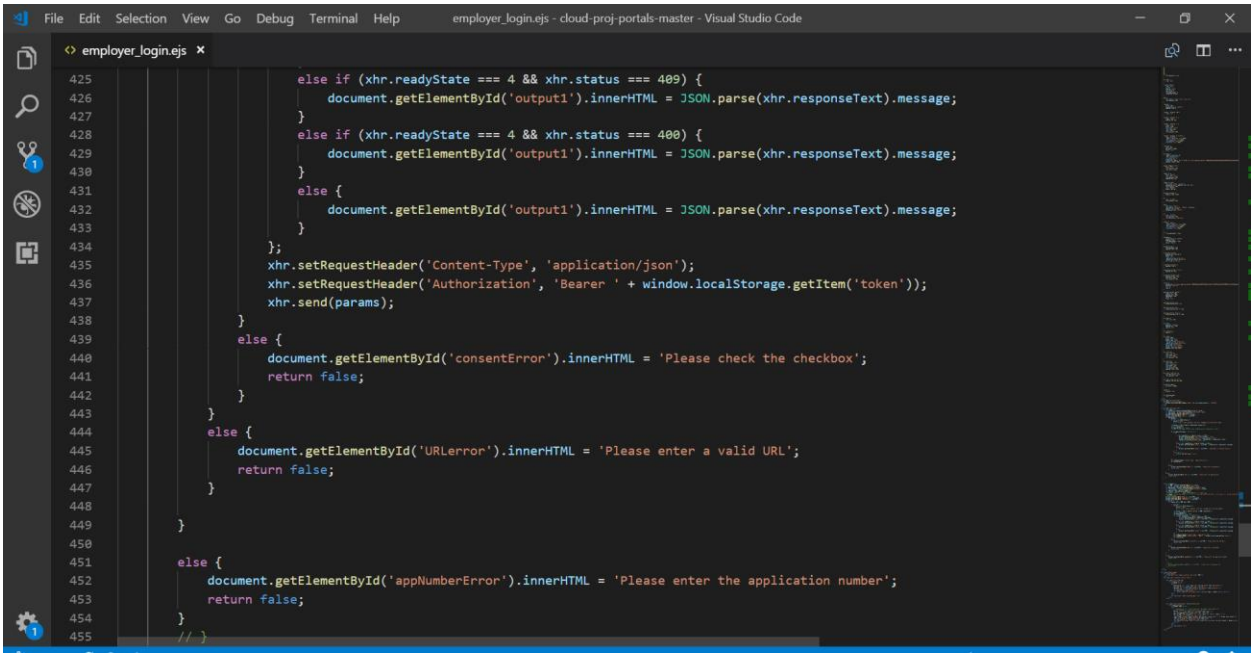


Figure 34: employer-login.ejs (contd.)

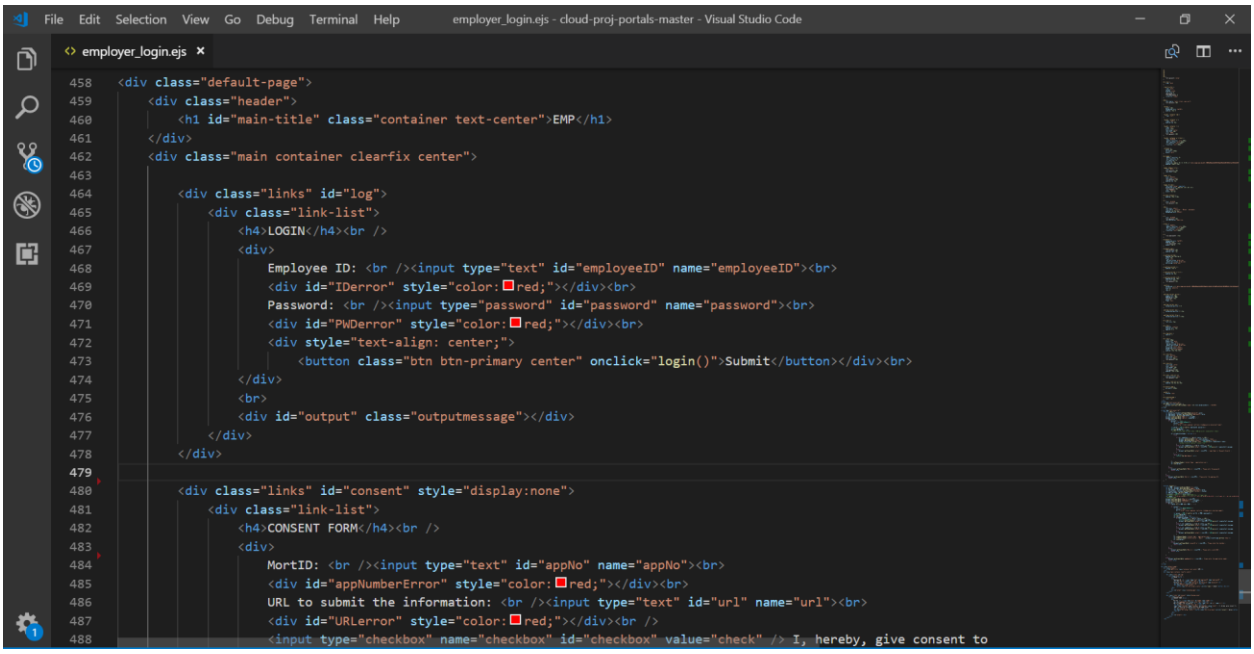


Figure 35: employer-login.ejs (contd.)

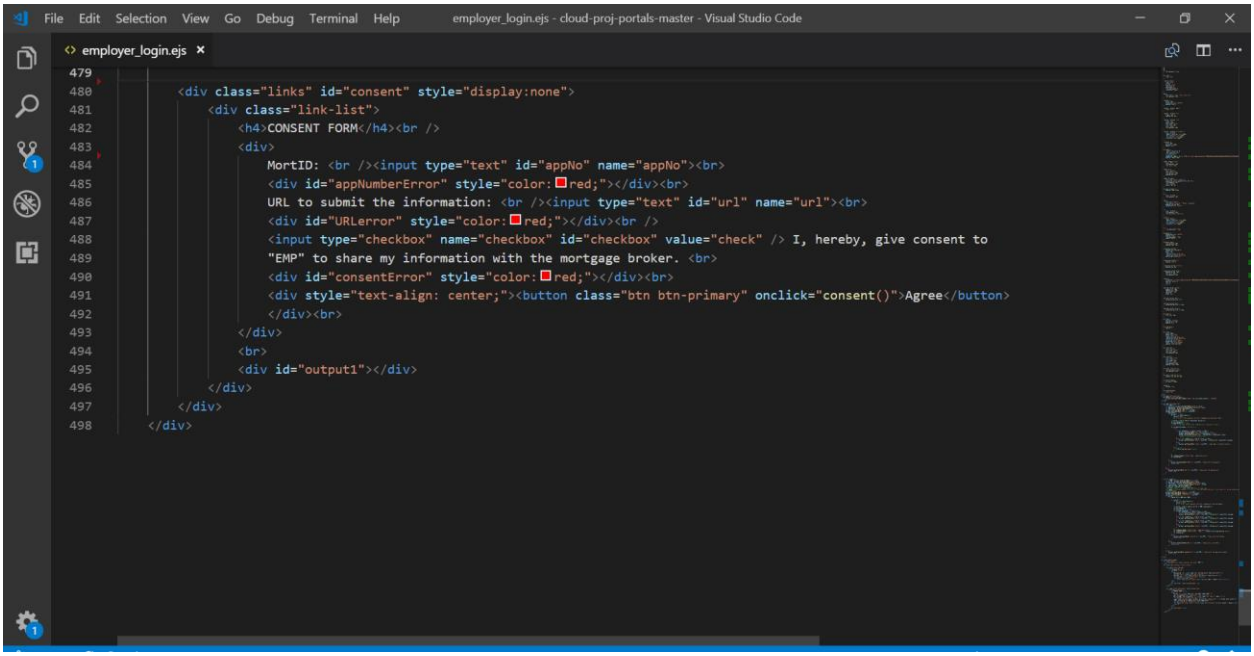


Figure 36: employer-login.ejs (contd.)

REAL ESTATE PORTAL

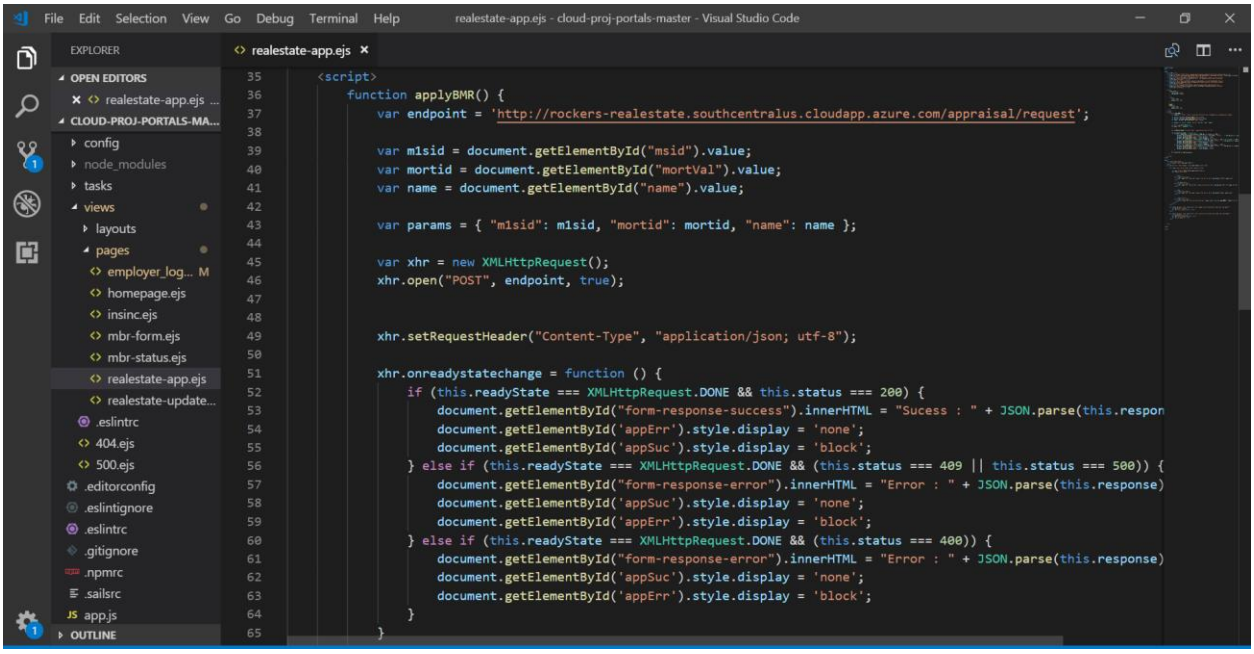


Figure 37: realestate-app.ejs (Real Estate appraisal application)

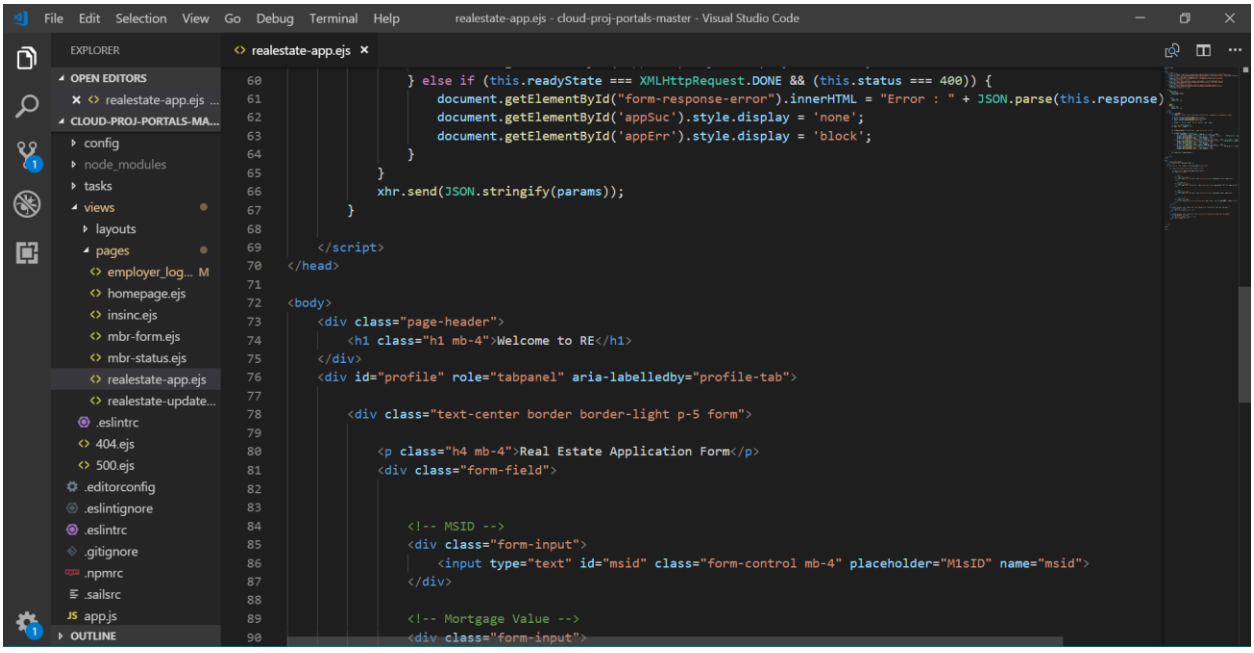


Figure 38: realestate-app.ejs (contd.)

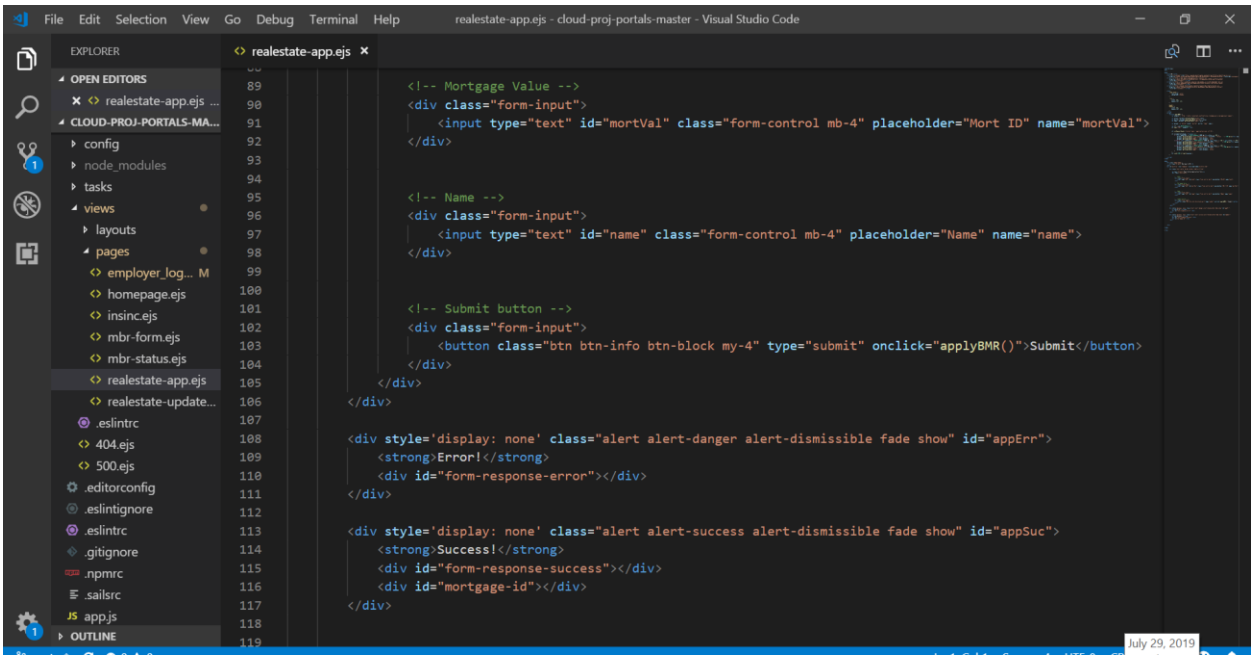


Figure 39: realestate-app.ejs (contd.)

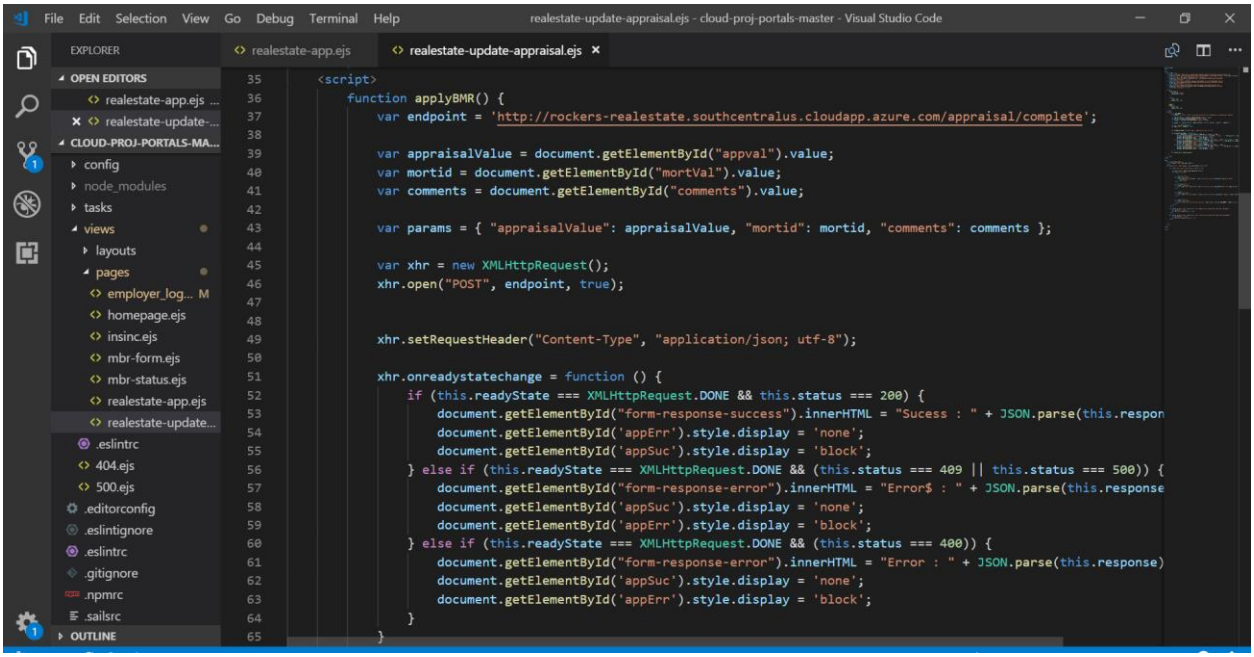


Figure 40: realestate-update.ejs (Appraisal Evaluation)

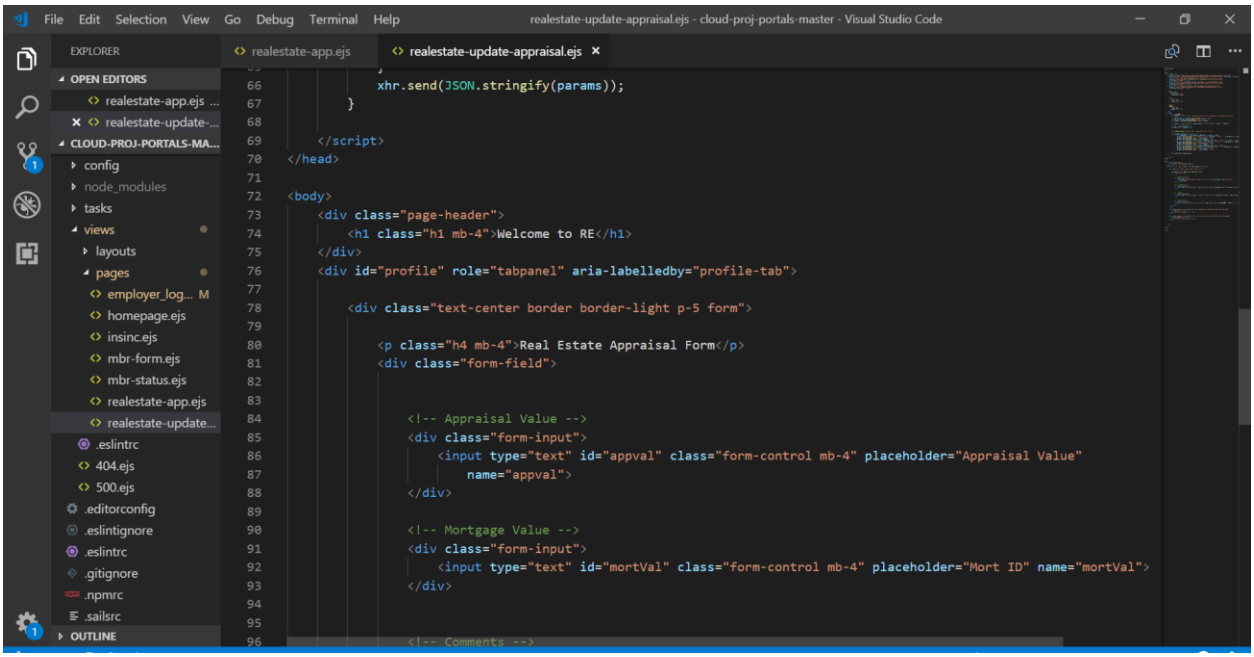


Figure 41: realestate-update.ejs (contd.)

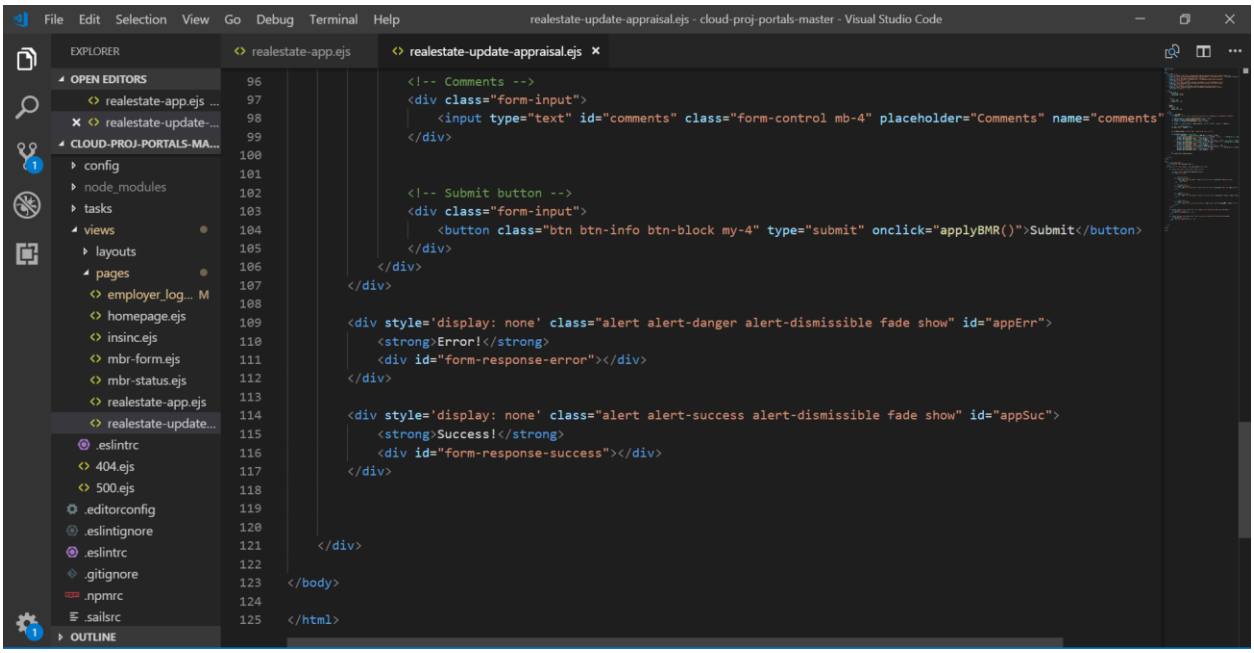


Figure 42: realestate-update.ejs (contd.)

INSURANCE PORTAL

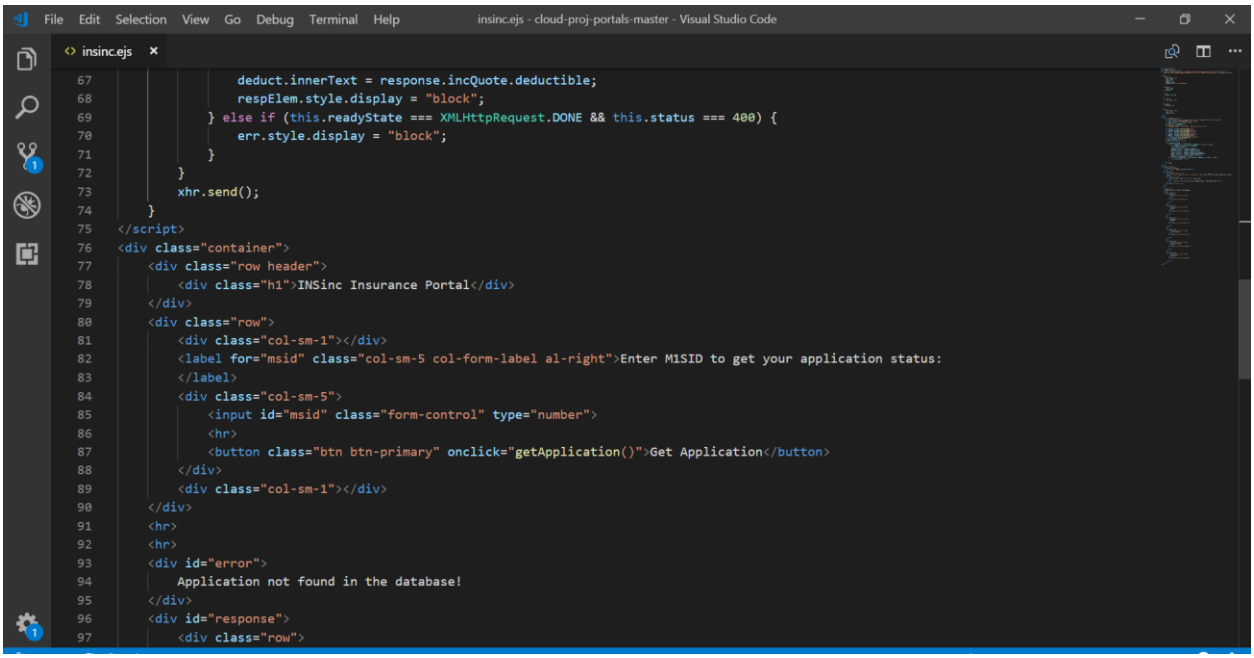


Figure 43: insinc.ejs (Insurance Portal)

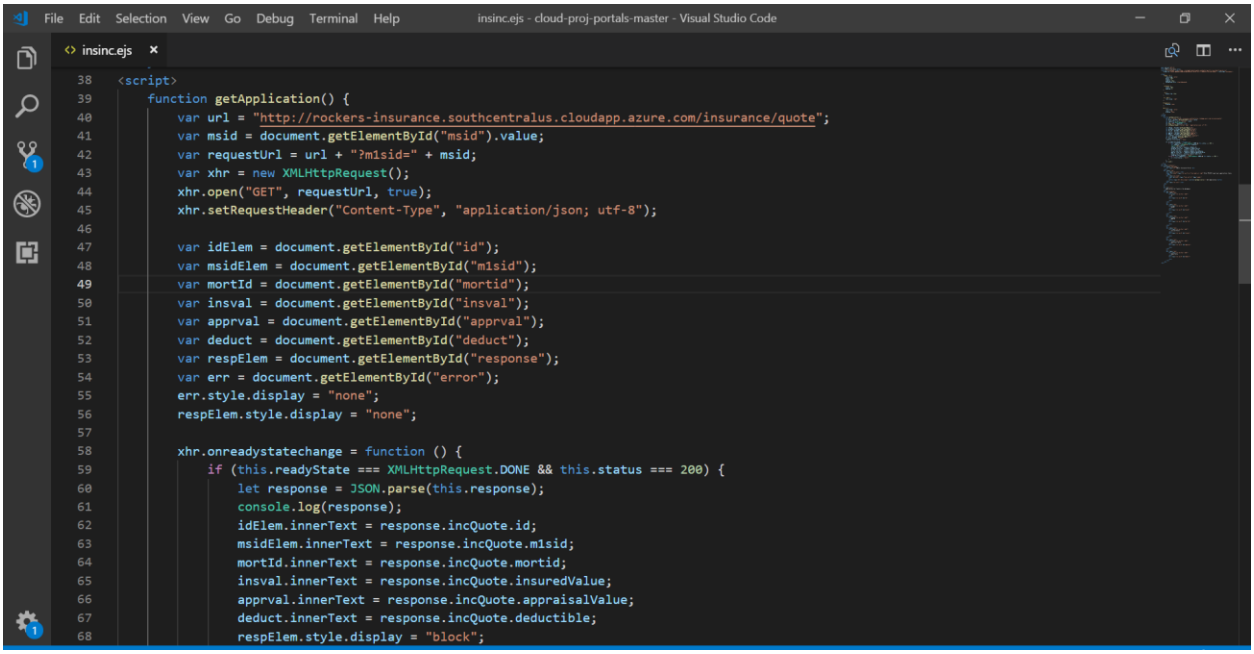


Figure 44: insinc.ejs (contd.)

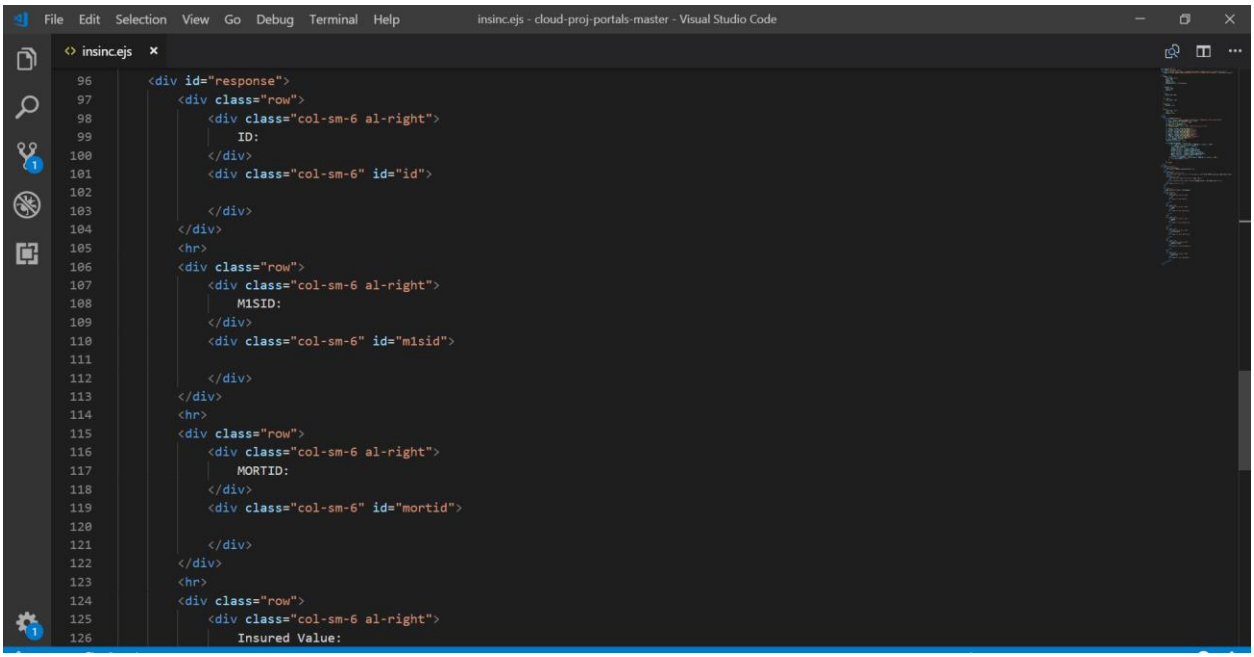


Figure 45: insinc.ejs (contd.)

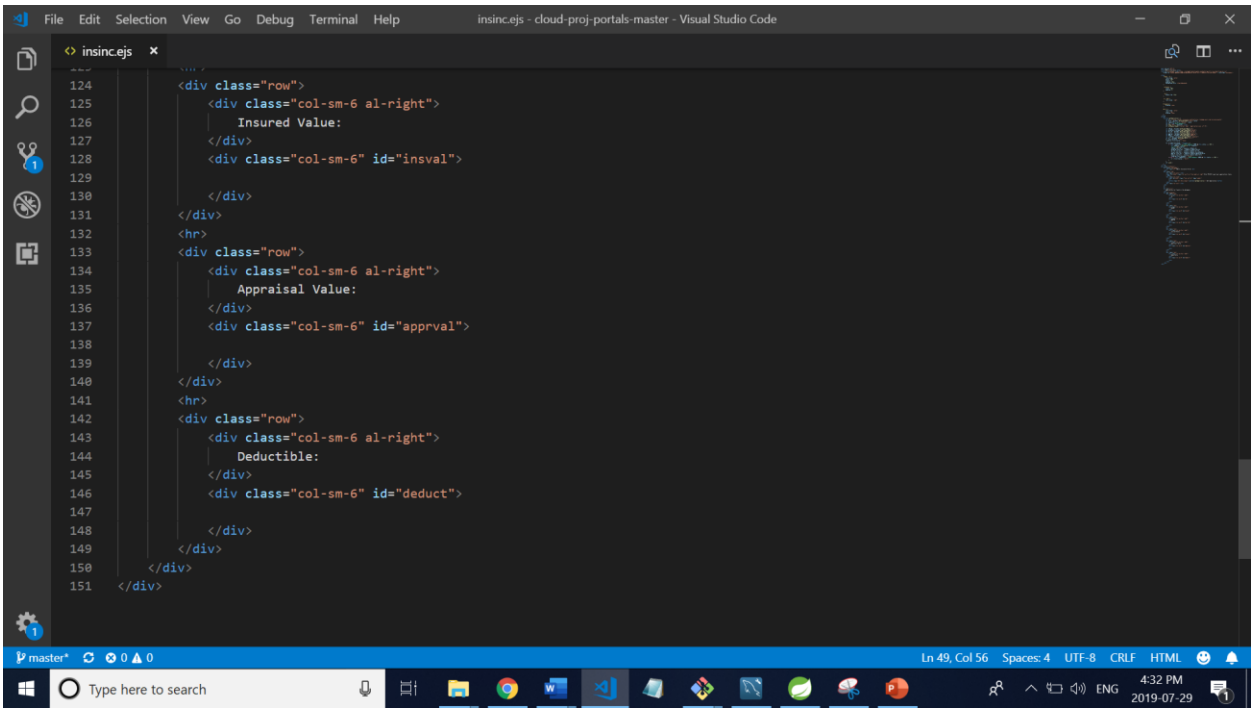


Figure 46: insinc.ejs (contd.)

WEB SERVICE CODE

EMPLOYER SERVICE

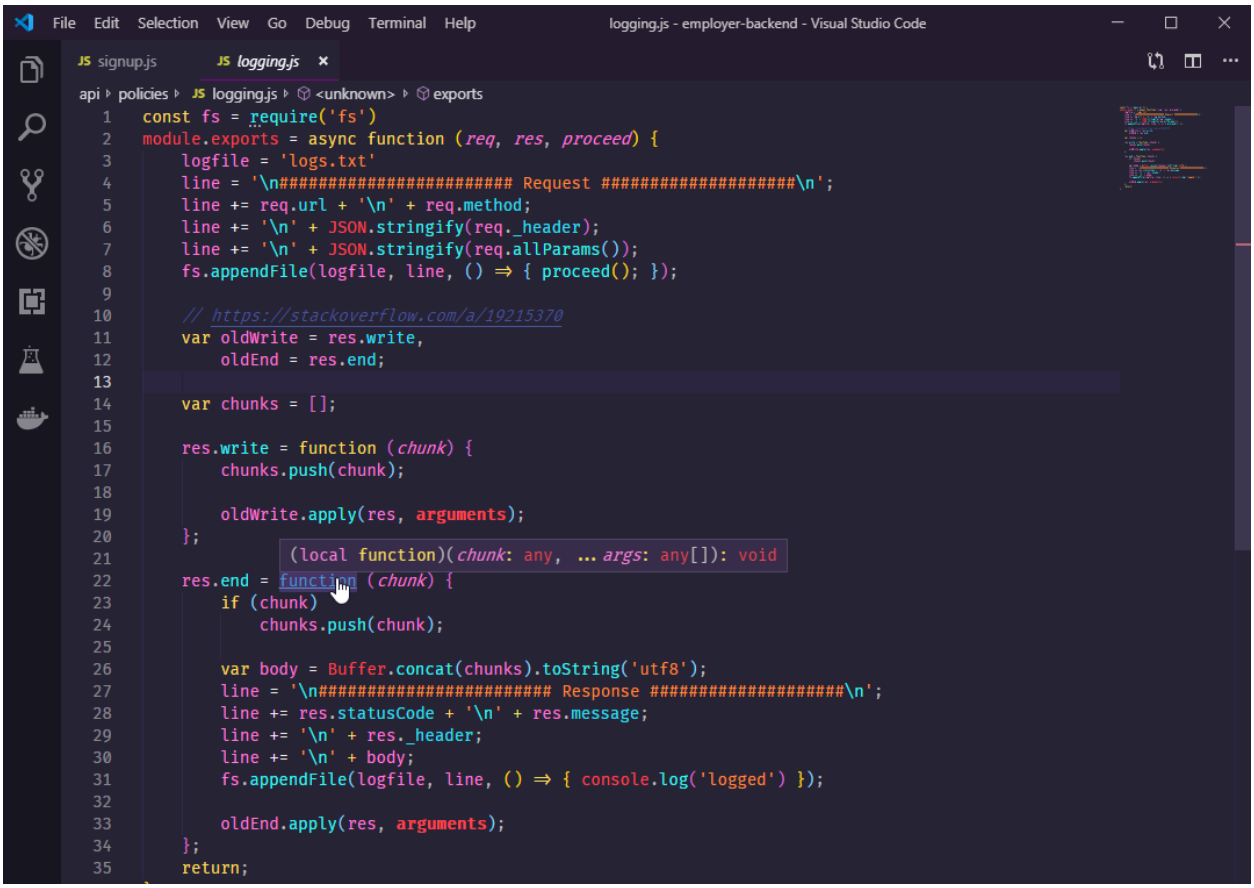


Figure 47: logging.js

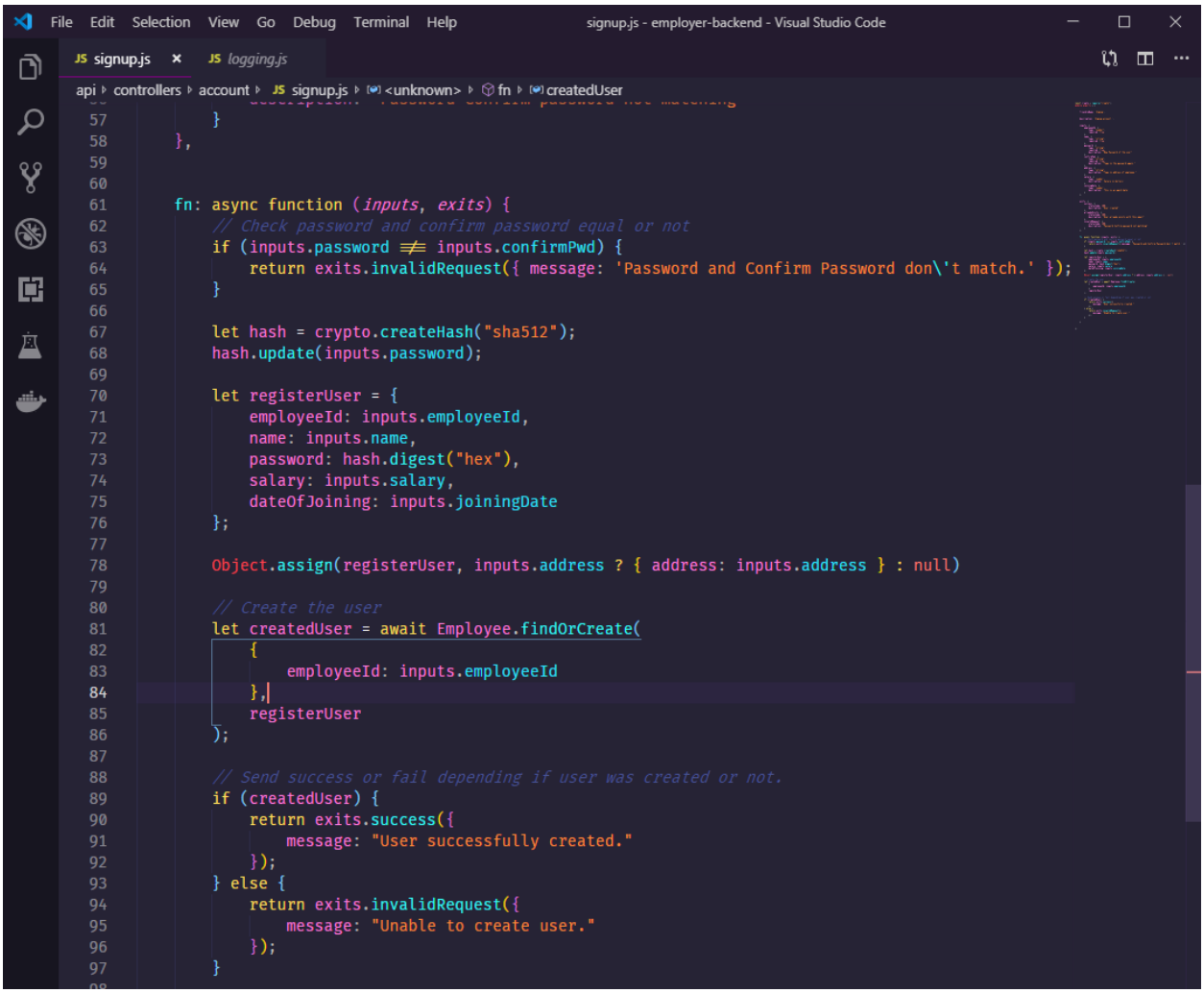


Figure 48: signup.js

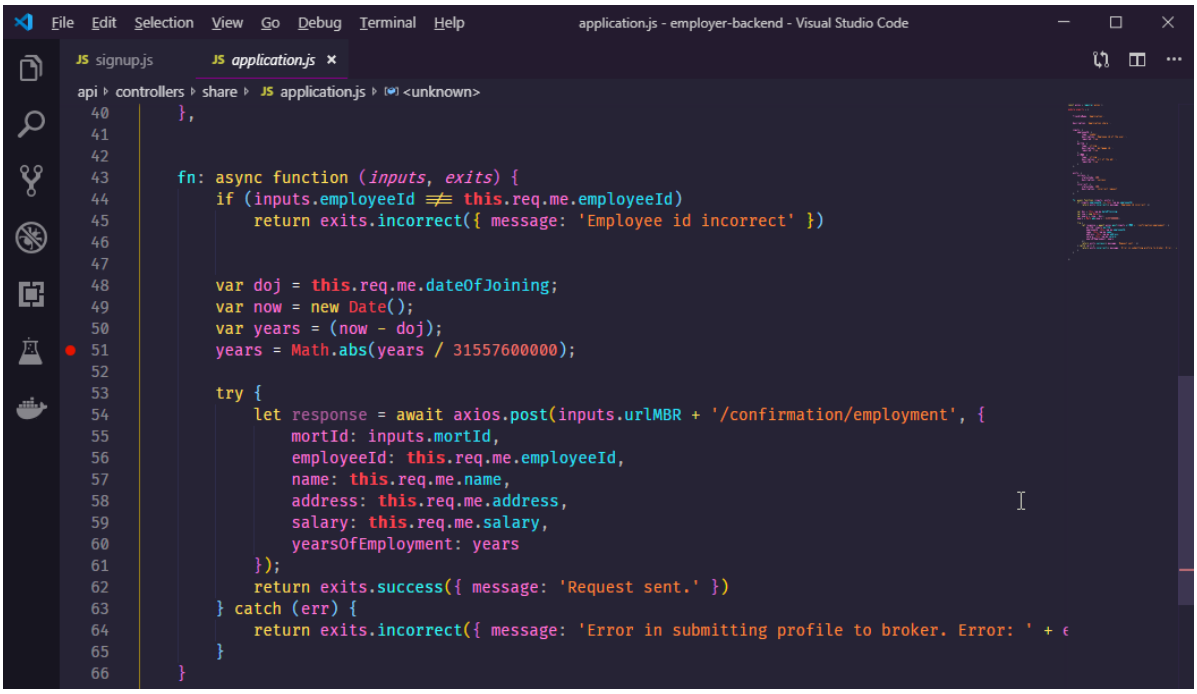


Figure 49: application.js

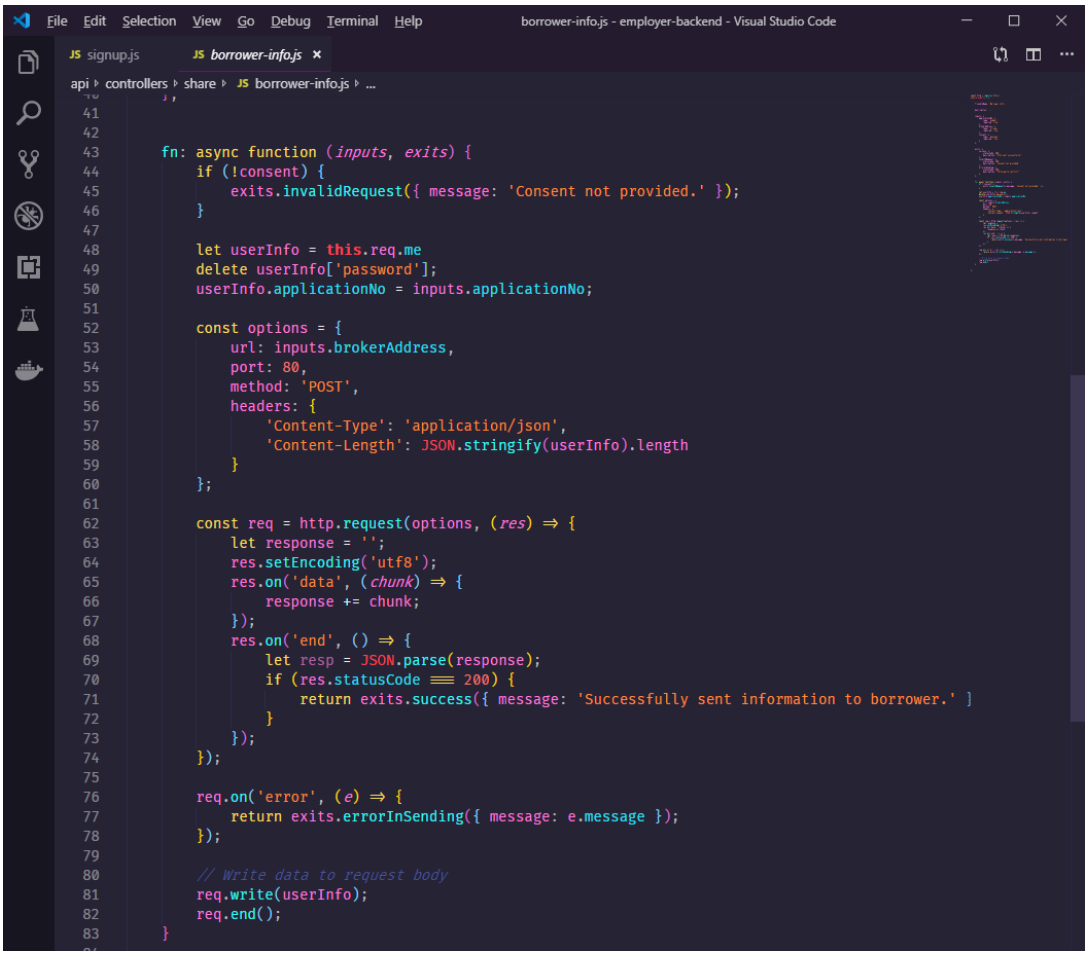


Figure 50: borrower-info.js

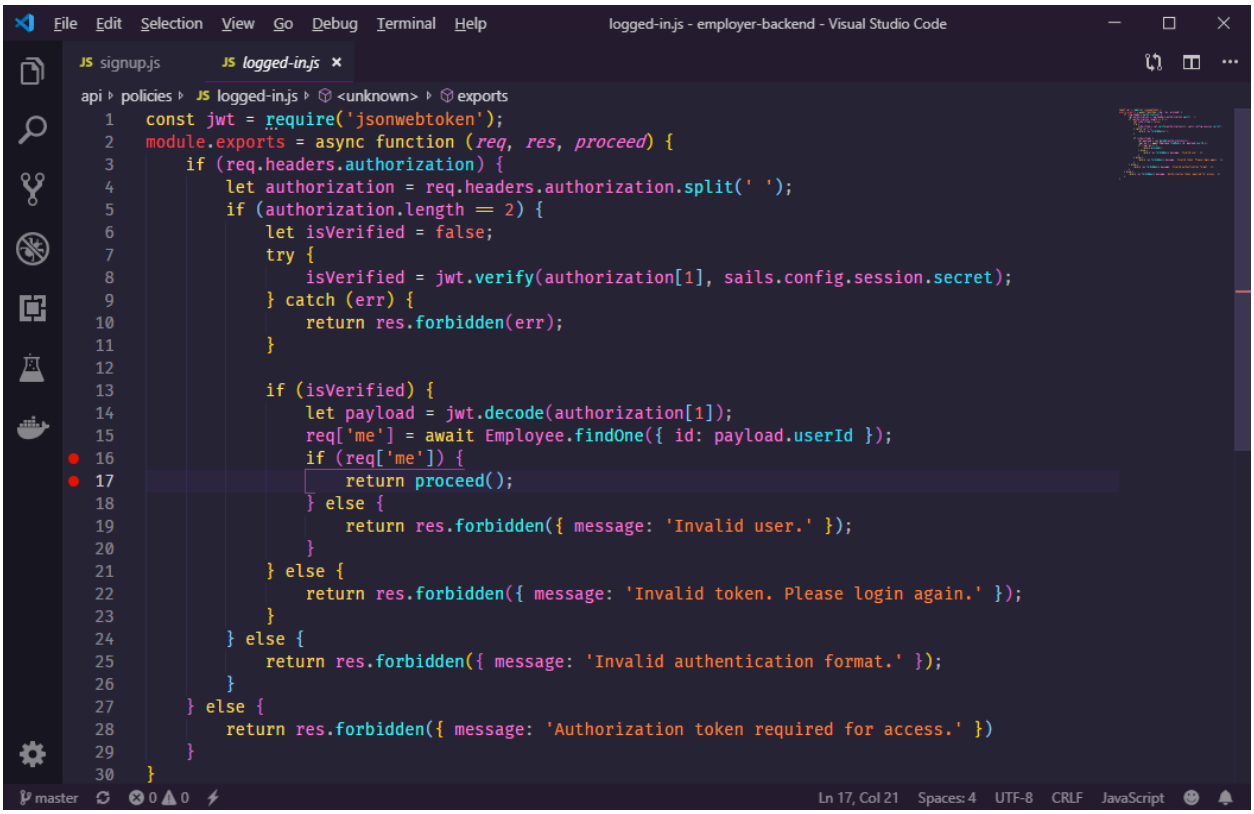


Figure 51: logged-in.js

INSURANCE SERVICE

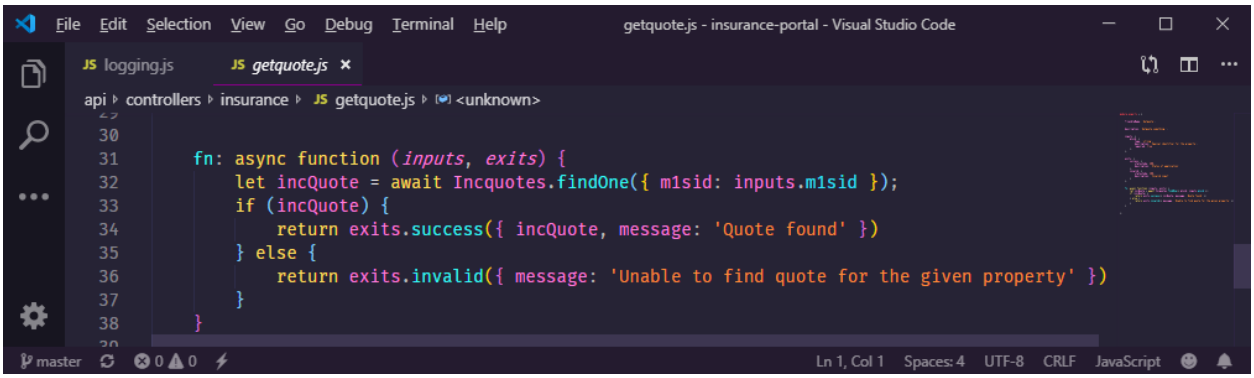


Figure 52: getquote.js

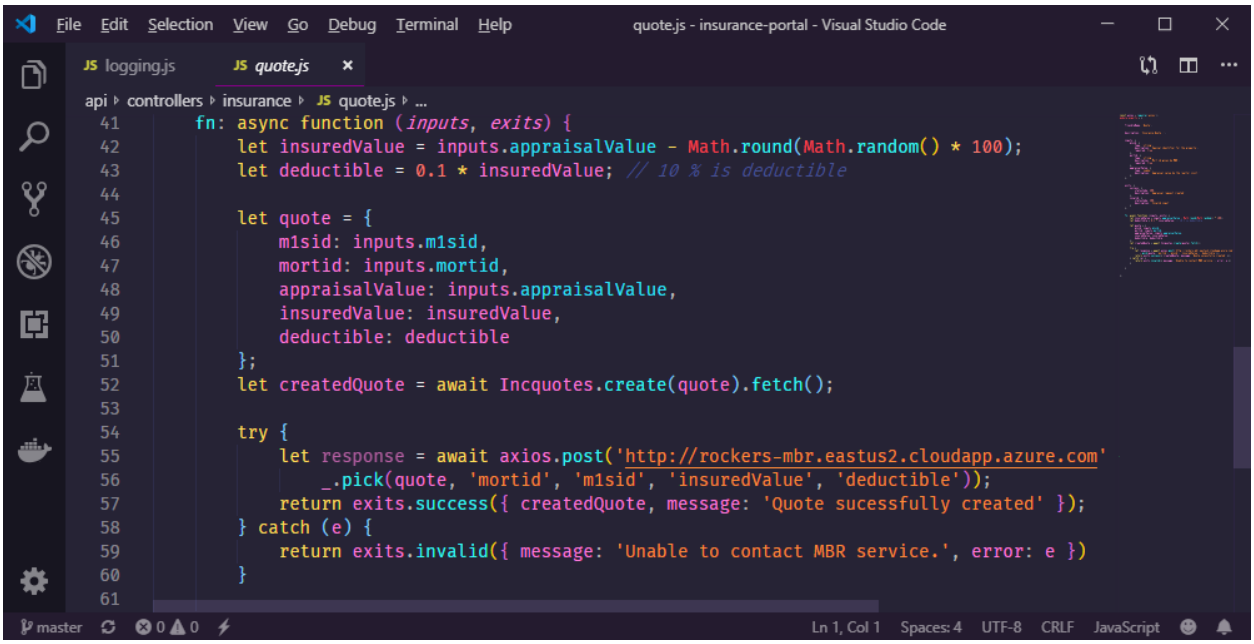


Figure 53: quote.js

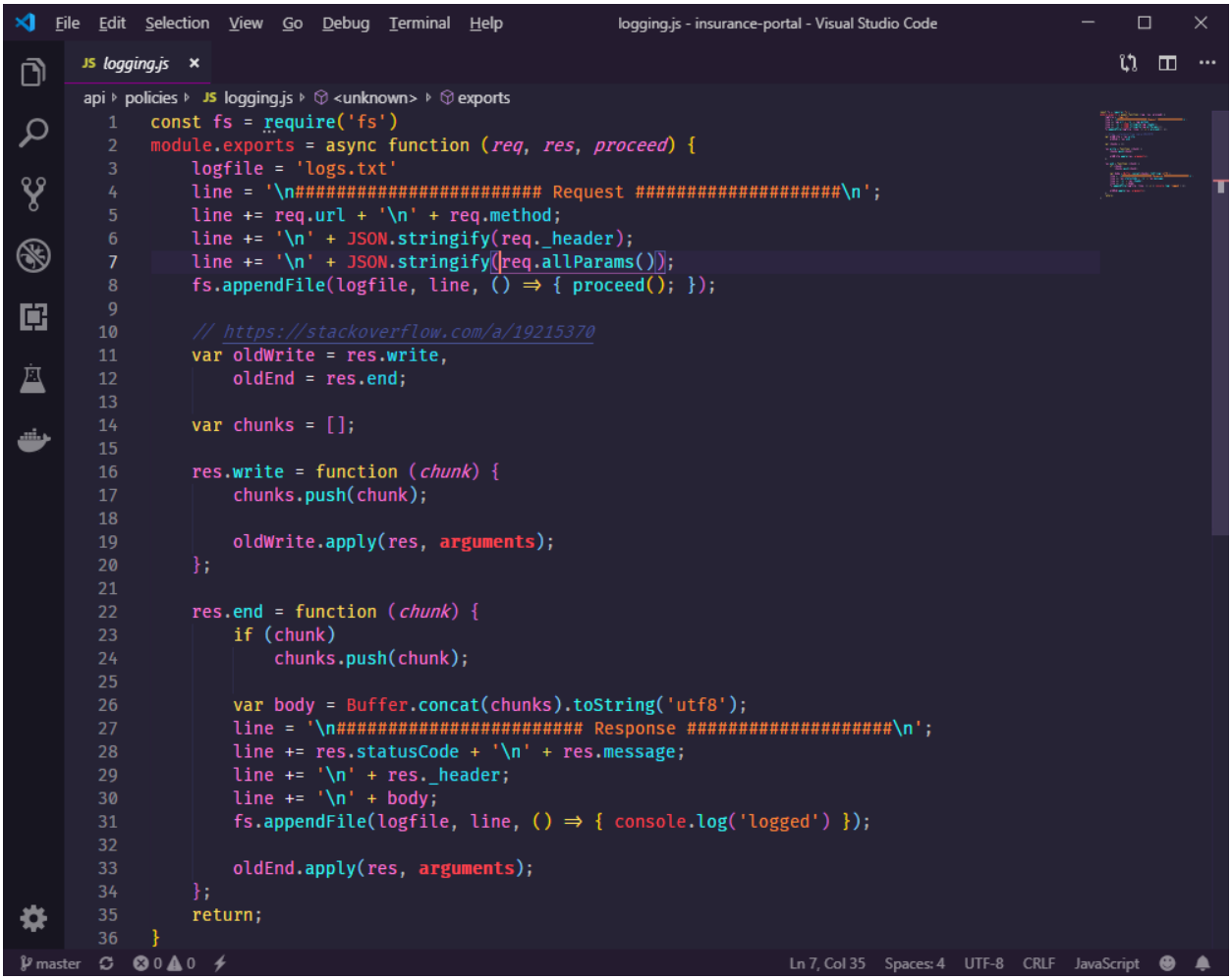


Figure 54: logging.js

MBR BROKER SERVICE

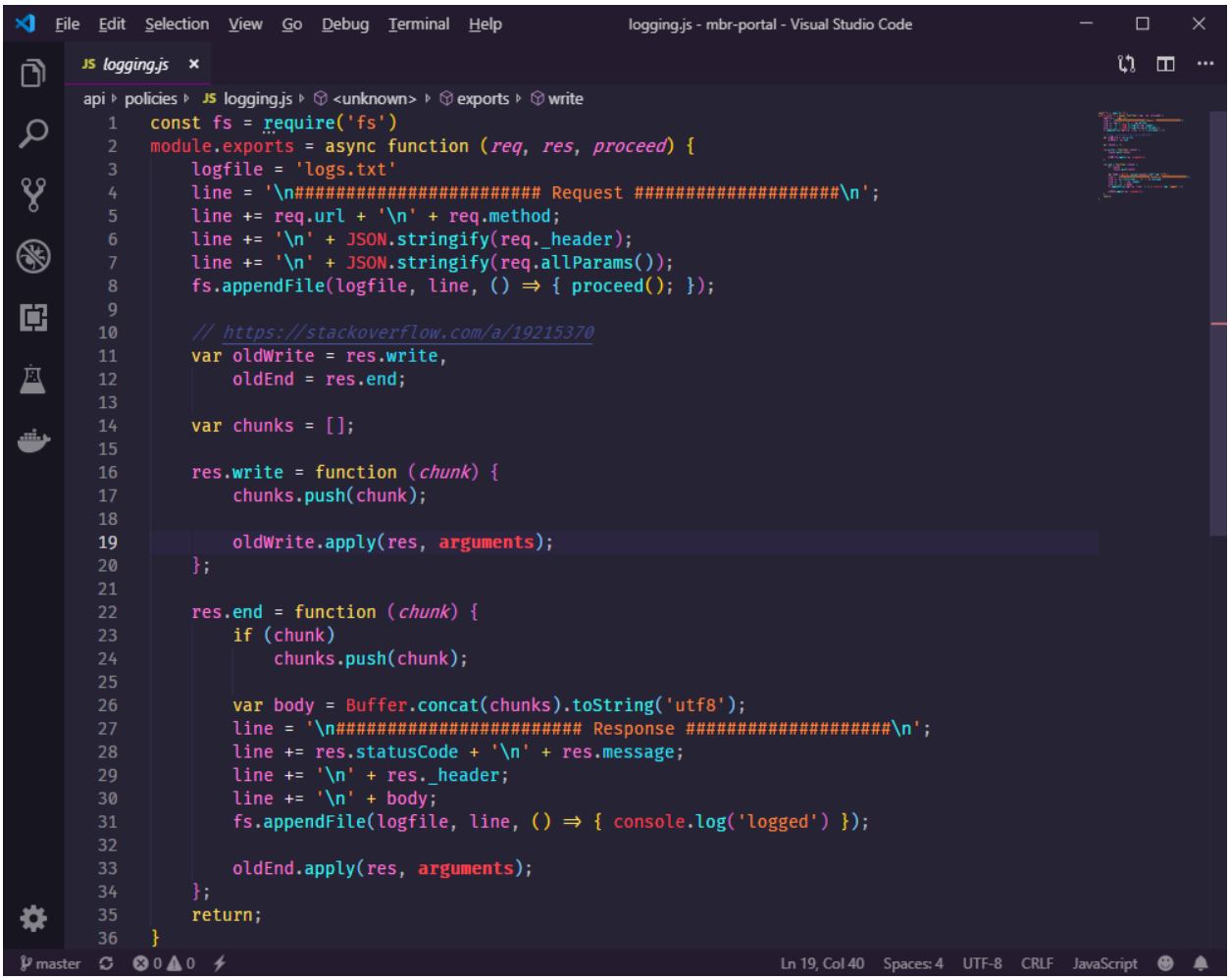


Figure 55: logging.js

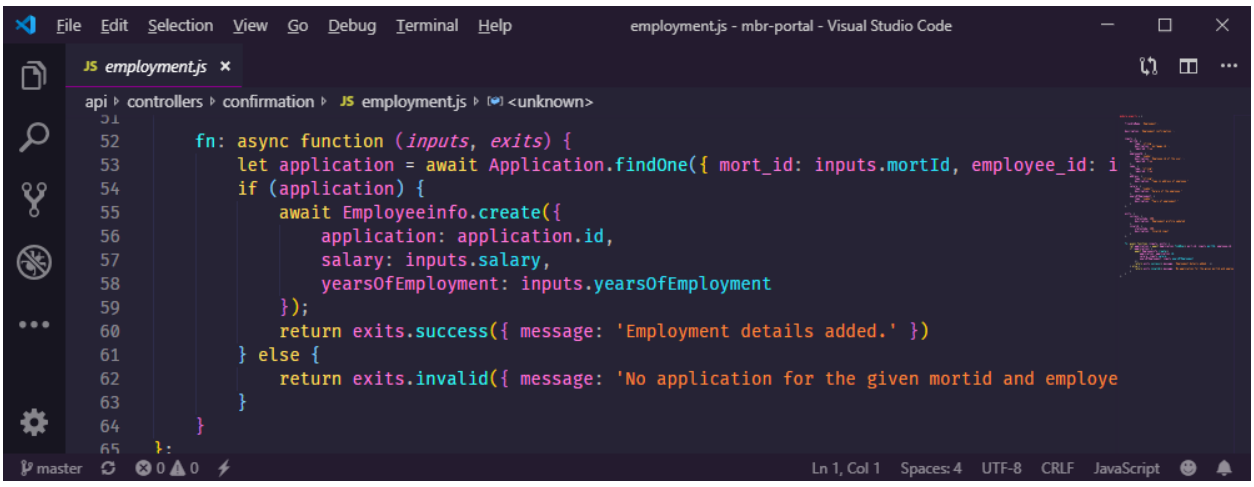


Figure 56: employment.js

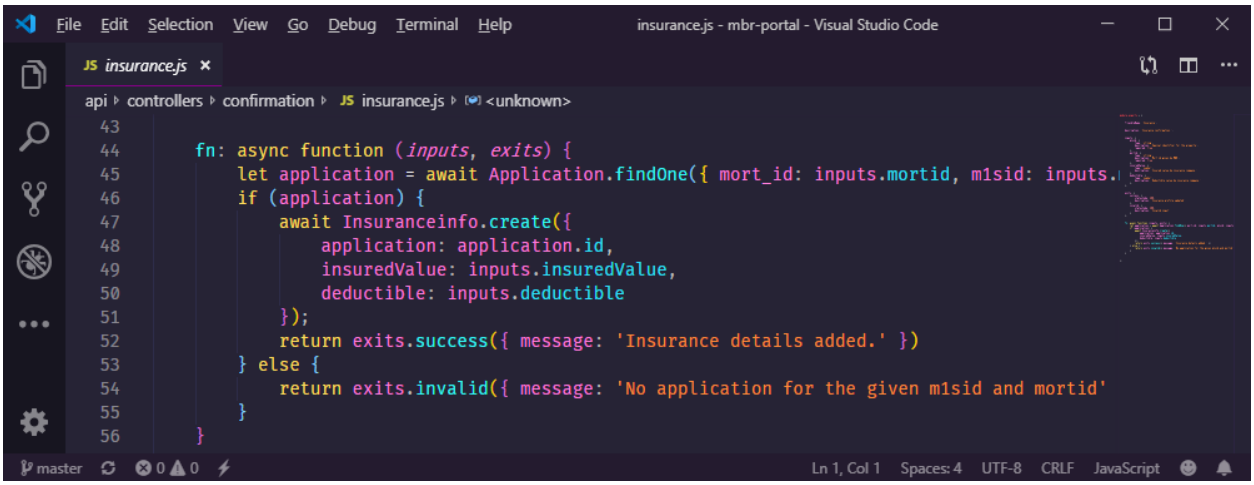


Figure 57: insurance.js

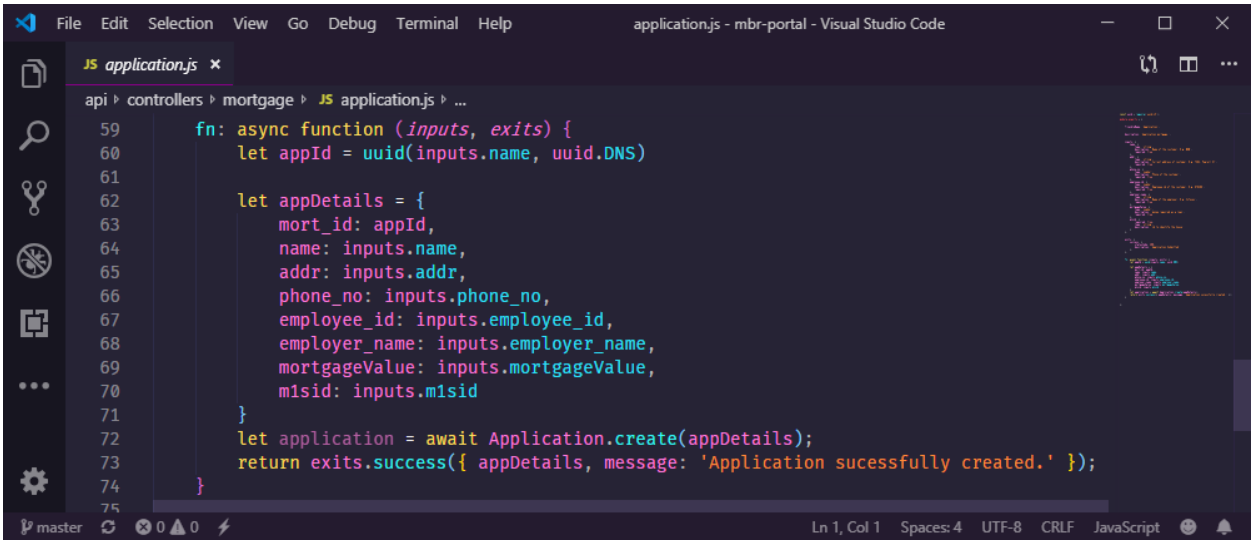


Figure 58: application.js

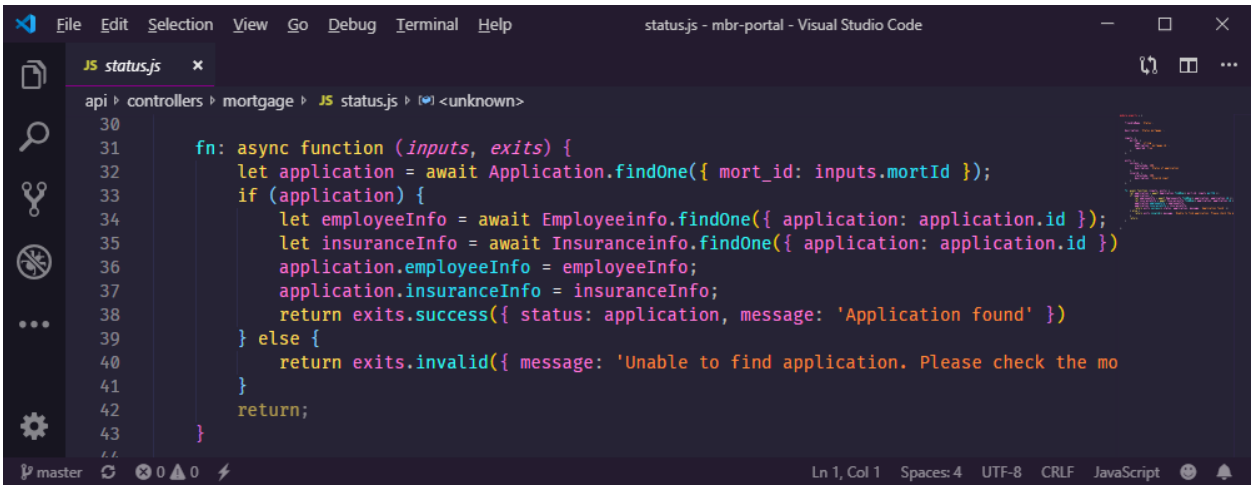


Figure 59: status.js

REAL ESTATE BROKER WEB SERVICE

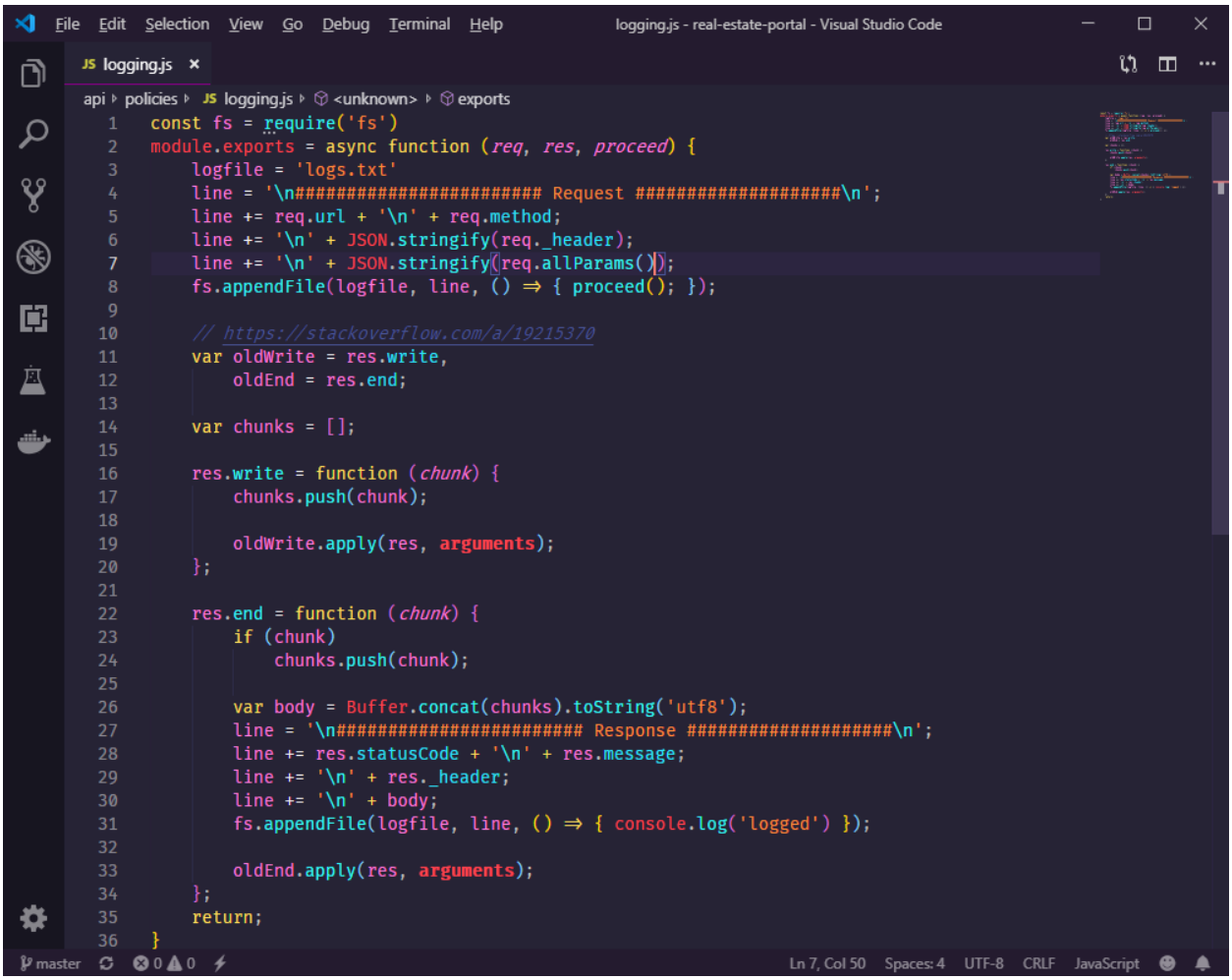


Figure 60: logging.js

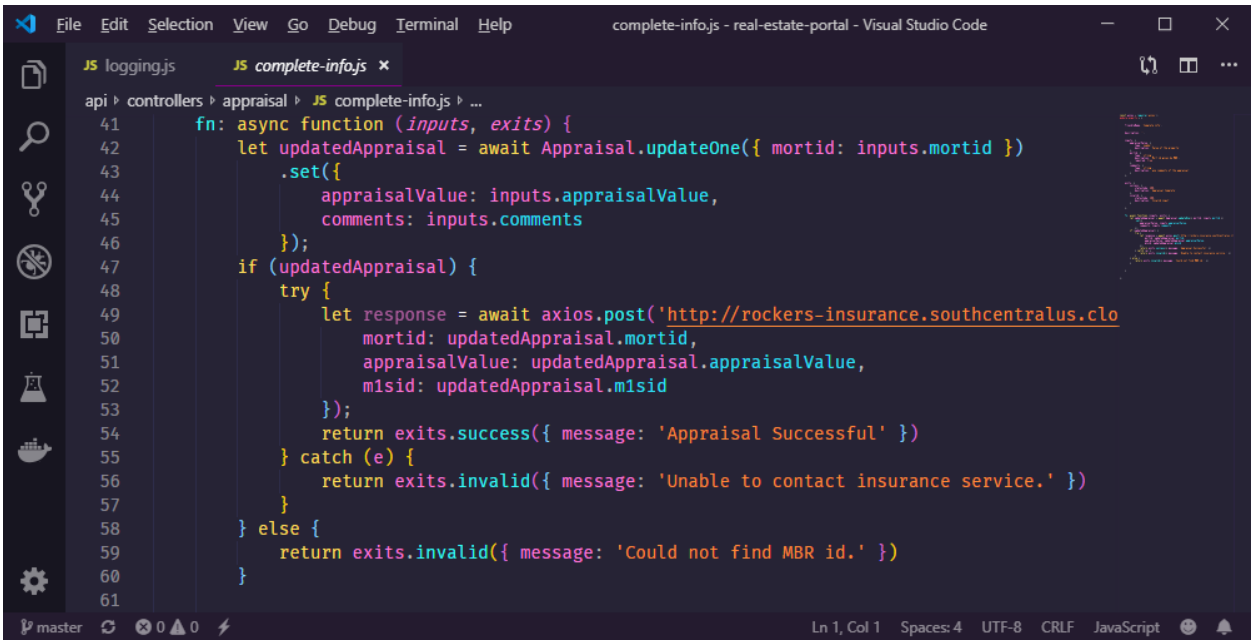


Figure 61: complete-info.js

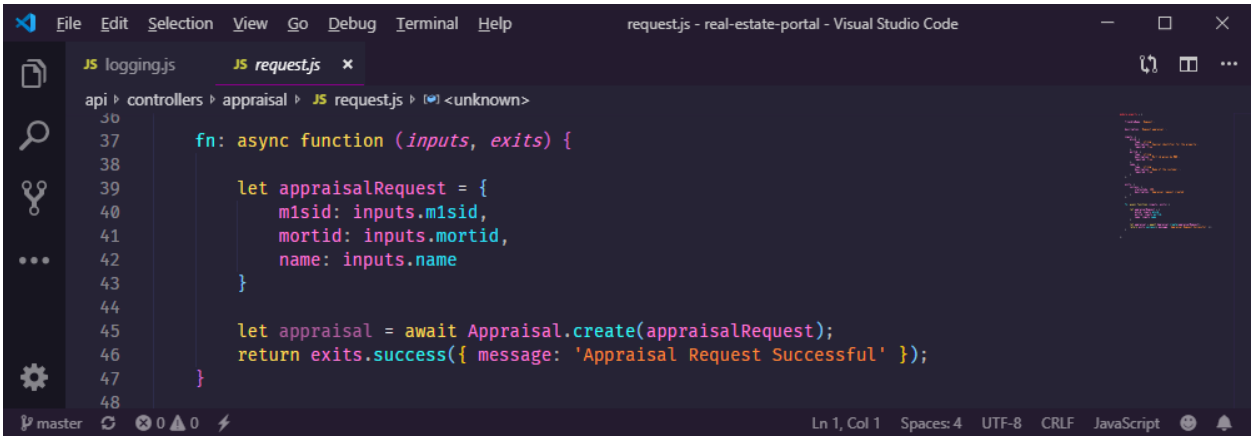


Figure 62: request.js