**COIT13230 Application Development Project**

Application Project Description

Table of Contents

[Project Summary 1](#_Toc147493903)

[Project Background 1](#_Toc147493904)

[Project Objectives 1](#_Toc147493905)

[User Stories 2](#_Toc147493906)

[Test Result 4](#_Toc147493907)

[User acceptance test - Guest persona 4](#_Toc147493908)

[User acceptance test - Volunteer persona 5](#_Toc147493909)

[User acceptance test - Staff persona 6](#_Toc147493910)

[User Manual 9](#_Toc147493911)

[Java Setup 9](#_Toc147493912)

[Database Setup 9](#_Toc147493913)

[GlassFish Setup 10](#_Toc147493914)

[Staff User Setup 15](#_Toc147493915)

[Volunteer Login 16](#_Toc147493916)

[Volunteer Search 17](#_Toc147493917)

[Staff Login 18](#_Toc147493918)

[Staff Adding Categories 19](#_Toc147493919)

[Staff Updating Categories 20](#_Toc147493920)

[Staff Removing Categories 21](#_Toc147493921)

[Staff Adding Services 22](#_Toc147493922)

[Staff Updating Services 23](#_Toc147493923)

[Staff Removing Services 24](#_Toc147493924)

[Project Reflection 25](#_Toc147493925)

[References 26](#_Toc147493926)

[Appendix A 27](#_Toc147493927)

[Implementation details 27](#_Toc147493928)

[Account Creation, Login and Authentication 27](#_Toc147493929)

[Image Uploading Methods 29](#_Toc147493930)

[Java Mail 31](#_Toc147493931)

[Page Authentication Filter 33](#_Toc147493932)

[Add and edit 35](#_Toc147493933)

# Project Summary

## Project Background

Central Queensland University (CQUniversity) has over 30,000 students studying online and on-campus throughout Australia (CQUniversity 2023). They offer more than 250 undergraduate, postgraduate, research and certificate courses in areas such as apprenticeships, psychology, information technology and more (CQUniversity 2023).

Many study areas require the participation of volunteers in order for students to practise learnt skills (Brown 2023). These volunteers are recruited from the public and are currently managed by each department individually on an “ad hoc” basis (Brown 2023), as timings are very specific. The CQUniversity Booking System was proposed to simplify this process by providing a centralised location for the storage of volunteer and service details. Users will be able to access the system via a website and select services to register their interest in based on campus location.

## Project Objectives

The main goal of our project was to increase the efficiency of CQUniversity’s existing volunteer management processes through the introduction of a system permitting each party involved in volunteering access to relevant services and information. As a result, more of the public will be able to take advantage of the inexpensive services on offer, leading to greater learning opportunities for students.

We will achieve this by:

* Providing a central location to store details of volunteers, services, categories, locations and staff members;
* Allowing public access to the database of offered services via a website;
* Allowing the search of services via a range of factors (ie. location, category) and select those they are interested in;
* Allowing administrative functionality to CQUniversity staff in order to modify available services as required; and,
* Allowing CQUniversity staff to send automated emails to registered volunteers with details about specific sessions and the booking confirmation process.

# User Stories

As a guest, I want to open the application, so that I can view a read-only version.

As a guest, I want to choose the campus location, so that I can view the services available on that campus.

As a guest, I want to view a list of categories, so that I can select a category.

As a guest, I want to select a category from the list, so that I can view the services available in that category.

As a guest, I want to view a list of services without searching for a location or category so that I can select a service.

As a guest, I want to select a service I am interested in so that I can view the details for that service.

As a guest, I want to create a login, so that I can access the full version of the application.

As a volunteer, I want to sign in to the website, so that I can store my contact information and preferences.

As a volunteer, I want to choose the campus location, so that I can view the services available on that campus.

As a volunteer, I want to view a list of categories, so that I can select a category.

As a volunteer, I want to select a category from the list, so that I can view the services available in that category.

As a volunteer, I want to view a list of services without searching for a location or category, so that I can select a service.

As a volunteer, I want to select a service I am interested in so that I can register for that service.

As a volunteer, I want to register for a particular service, so that I can be serviced.

As a volunteer, I want to deregister for a service I had previously registered for so that I will no longer be contacted about that service.

As a staff member, I want to sign in to the website, so that I can manage the services my department offers.

As a staff member, I want to access the admin side of the application, so that I can manage the services my department offers.

As a staff member, I want to add categories with services offered by my department, so that users can register for that service.

As a staff member, I want to remove categories of services that are no longer offered by my department, so that users are unable to register for that service.

As a staff member, I want to update categories with services offered by my department, so that users can register for that service.

As a staff member, I want to add services offered by my department, so that users can register for that service.

As a staff member, I want to update the services offered by my department, so that users can register for those services.

As a staff member, I want to remove services that are no longer offered by my department, so that users are unable to register for that service.

As a staff member, I want to save the changes I have made to services offered by my department so that users can register for that service.

As a staff member, I want to send an email to all volunteers who have selected to be informed about specific opportunities, so that users can make a booking for that service.

# 

# Test Result

## User acceptance test - Guest persona

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | Expected Outcome | Actual Outcome | Pass/Fail |
| As a guest, I want to open the application so that I can view a read-only version. | A guest user can open the application and view a read-only version. | *Figure 1:Success opening the web application* | Pass |
| As a guest, I want to choose the campus location, so that I can view the services available on that campus. | A guest user can select a campus location to view the services available on that campus. | *Figure 2: Success selecting a location* | Pass |
| As a guest, I want to view a list of categories, so that I can select a category. | A guest user can view a list of categories. | *Figure 3: Success viewing a category list* | Pass |
| As a guest, I want to select a category from the list so that I can view the services available in that category. | A guest user can select a category to view the services available in that category. | *Figure 4:Success selecting a category* | Pass |
| As a guest, I want to view a list of services without searching for a location or category so that I can select a service. | A guest user can view a list of services. | *Figure 5: Success viewing a list of services* | Pass |
| As a guest, I want to select a service I am interested in so that I can view the details for that service | A guest user can select a service and view the details for that service. | *Figure 6: Success viewing service details* | Pass |
| As a guest, I want to create a login, so that I can access the full version of the application. | A guest user can create a login account. | *Figure 7: Volunteer sign up page* | Pass |

## 

## 

## User acceptance test - Volunteer persona

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | Expected outcome | Actual outcome | Pass/Fail |
| As a volunteer, I want to sign in to the website, so that I can store my contact information and preferences. | A volunteer user can sign in to the website with the correct credentials. | *Figure 8: Volunteer login page* | Pass |
| As a volunteer, I want to choose the campus location, so that I can view the services available on that campus. | A volunteer user can choose the campus location and view the services available on that campus. | *Figure 9: Success selecting a service location* | Pass |
| As a volunteer, I want to view a list of categories, so that I can select a category. | A volunteer user can view a list of categories. | *Figure 10: Success viewing a category list* | Pass |
| As a volunteer, I want to select a category from the list, so that I can view the services available in that category. | A volunteer user can select a category and view the services in that category. | *Figure 11: Success selecting a category* | Pass |
| As a volunteer, I want to view a list of services without searching for a location or category, so that I can select a service. | A volunteer user can view a list of services. | *Figure 12: Success viewing a list of services* | Pass |
| As a volunteer, I want to select a service I am interested in so that I can register for that service. | A volunteer user can select a service and see details relevant to that service. | *Figure 13: Success viewing service details* | Pass |
| As a volunteer, I want to register for a particular service, so that I can be serviced. | A volunteer user can register for a service. | *Figure 14: Service registration* | Pass |
| As a volunteer, I want to deregister for a service I had previously registered for so that I will no longer be contacted about that service. | A volunteer user can deregister or unsubscribe from a service. | *Figure 15: Unsubscribing from a service* | Pass |

## 

## 

## User acceptance test - Staff persona

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | Expected outcome | Actual outcome | Pass/Fail |
| As a staff member, I want to sign in to the website, so that I can manage the services my department offers. | A staff user can sign in to the website with the correct credentials. | *Figure 16: Staff login page* | Pass |
| As a staff member, I want to access the admin side of the application, so that I can manage the services my department offers. | A staff user can access the admin side of the application and view and manage the categories and services offered by their department. | *Figure 17: Staff user viewing admin side of the application* | Pass |
| As a staff member, I want to add categories with services offered by my department, so that users can register for that service. | A staff user can add categories for services offered by their department. | *Figure 18: Add category page* | Pass |
| As a staff member, I want to remove categories of services that are no longer offered by my department, so that users are unable to register for those services. | A staff user can remove categories for services offered by their department. | *Figure 19: Remove category page* | Pass |
| As a staff member, I want to update categories with services offered by my department, so that users can register for that service. | A staff user can update categories for services offered by their department. | *Figure 20: Update category page* | Pass |
| As a staff member, I want to add services offered by my department, so that users can register for that service. | A staff user can add services offered by their department. | *Figure 21: Add services page* | Pass |
| As a staff member, I want to update services offered by my department, so that users can register for that service. | A staff user can update services offered by their department. | *Figure 22: Update services page* | Pass |
| As a staff member, I want to remove services no longer offered by my department, so that users are unable to register for that service. | A staff user can remove services offered by their department. | *Figure 23: Remove services page* | Pass |
| As a staff member, I want to save the changes I have made to services and categories offered by my department so that users can register for those service. | A staff user can save changes and they are persisted in the database. | *Figure 24: Saving a category to the database* | Pass |
| As a staff member, I want to send an email to all volunteers who have been selected to be informed about specific opportunities, so that users can make a booking for that service. | A staff user can send an email to registered volunteers about a specific service. | *Figure 25: Send email page* | Pass |

# 

# User Manual

The following instructions pertain to the setup of CQUBS and its necessary components on a Windows machine. The application and other components mentioned in this manual can be found on [CQUBS’s Github repository](https://github.com/amygreenwood03/CQUniversity-Booking-System).

## Java Setup

CQUBS is built on Oracle’s Java 17 and hence requires the presence of a Java 17 installation in order to run on a machine. The GUI installer for Java 17 can be downloaded from [Oracle’s website](https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html). Please download the Windows 64-bit installer before proceeding with this section.

Once the executable has been downloaded, run it and allow administrative permissions if prompted. The installer will start up and guide you through the installation of Java 17. You will receive a notification once the installation is complete, after which you can continue working through this manual.

## 

## Database Setup

1. MySQL Server Installation

A MySQL database is required for this application. For guaranteed compatibility, it is recommended that you install a version under the MySQL 8.x banner.

The installer for all MySQL 8.x applications, **MySQL Installer - Community**, can be downloaded [here](https://dev.mysql.com/downloads/installer/). Please download and run this executable to install the MySQL 8.x Installer. Once this process is completed, run MySQL Installer - Community to begin the installation of MySQL components.

CQUBS requires MySQL Server and the Java connector, so ensure that these components are selected on the configuration screen at a minimum. Once you have selected the necessary applications, continue to follow the installer’s prompts. If the installer identifies any actions that need to be completed before the installation of the chosen components, follow the instructions given to resolve these.

The installer will ask to configure MySQL Server during installation. This is where you will set up credentials for the root MySQL user, as well as the port number MySQL listens to and whether to automatically start the MySQL service on machine startup. Make the changes you wish, but be sure to note the port number and account credentials of the root user, as these will be used at a later stage in this manual. The default port number (3306) is also acceptable.

You will be notified by the installer once all components have been installed and configured successfully.

1. Test Data Implementation

An SQL script has been provided on CQUBS’s Github repository to automatically create the necessary database structure and populate it with test data. Please download and run this script using the MySQL Command Line Client.

## 

## GlassFish Setup

1. Download and Installation

This application requires a server compatible with Jakarta EE 10 to be installed and operated. In this case, Eclipse GlassFish is used as the Java server solution. GlassFish can be downloaded from the official website at [glassfish.org](https://glassfish.org/) as ZIP files. GlassFish version 7.0.7 or higher is recommended as the earlier versions of GlassFish 7 are known to make errors with the administration page, not allowing systems to progress further setups.

Once the ZIP file is downloaded, extract the files in a designated directory. It is recommended that you use a directory such as your user account’s folder to avoid permission issues if you are not the administrator of your machine.

Once all files are extracted, a domain needs to be created. Please start the command prompt or terminal as an administrator and change the directory to **“(GlassFish directory)\bin”**. You can create a new domain or start an existing domain with asadmin command. By default, if GlassFish gets an order to run a domain with the “asadmin start-domain” command, GlassFish starts a domain named “domain1” and opens a UDP port 4848 for administration. You may specify other domain names and ports for administration by using different commands if needed. Please check GlassFish administration guidelines chapter 3 at   
<https://glassfish.org/docs/latest/administration-guide.html> for details.

Once a domain is created and started, you can access (server address):(admin port) using a web browser to progress further. In this case, the administrator page can be accessed at localhost:4848.

1. JDBC Connection Pool

Once GlassFish is installed, a JDBC connection pool needs to be created to allow the server to communicate with the MySQL database. However, the MySQL Java connector that was installed in the previous section needs to be added first to Glassfish.

Copy the connector file from your MySQL installation. This is typically located in the **“\MySQL\Connector J [version]”** directory in **“Program Files (x86)”**. Navigate to your Glassfish installation’s **“\glassfish\lib”** directory and paste the connector inside. Restart Glassfish if it is currently running, before accessing the admin console.

Please go to Resources/JDBC/JDBC Connection Pool and click the “New” button. Set the pool name to “CQUBSPU”, the resource type to “javax.sql.DataSource”, and the database driver vendor to “MySQL”. Once these details have been filled, select “Next”.

On the second screen, change the data source class name to   
**“com.mysql.cj.jdbc.MysqlConnectionPoolDataSource”** using the text box. At the bottom of the page, a table for additional properties is displayed. Add the following entries to this table, then select “Finish”.

* Name: DatabaseName, Value: app\_dev\_database
* Name: User, Value: root (or the name of the MySQL user you would like to connect with)
* Name: Password, Value: [password of MySQL root account or chosen user account]
* Name: PortNumber, Value: [port MySQL listens on, typically 3306]
* Name: useSSL, Value: false

The new connection pool will appear in the connection pools table if successful.

A close-up of a screen

Description automatically generated

*Figure 26: Connection Pool additional properties (example)*

1. JDBC Resources

Once the JDBC connection pool is created, a JDBC resource also needs to be created. Please go to Resources/JDBC/JDBC Resources and click the “New” button. Set the JNDI name “jdbc/AppDevProject”, select CQUBSPU as a pool name, and then click the OK button.

1. Create an Image Folder

A folder that saves the uploaded custom images needs to be created on the domain. Please create a folder named “images” in your domain docroot directory at the **“(GlassFish directory)/domains/(domain name)/docroot”** folder. Once the folder is created, change the String values on the source code for the directory to the images folder to the directory of the folder that you created as well. The classes that include the String value to be changed are as follows:

* ServiceAddController.java
* DetailsStaffController.java
* CategoryAddController.java
* CDetailsStaffController.java.

The directory value is stored in the string imagesPath at the uploadImg() function.

1. Importing Gmail’s SMTP Certificate

Glassfish refers to its own private keystore, cacerts.jks, to determine whether to refute or accept a connection based on the digital certificate it presents. In order to make use of CQUBS’s email notification functionality, Gmail’s SMTP certificate must be imported into this keystore.

Please download the file **“gmail.cert”** from CQUBS’s Github repository. Please note that at the time of writing, the certificate provided is valid; however, this will not always be the case once its expiry date is reached. A new certificate will need to be sought in this scenario.

Once completed, open a command prompt window in order to run the required import commands. Java’s keytool application will be used to import the digital certificate, which is located in the “\bin” directory of your JDK installation. If this directory has been added as a system environment variable, the command prompt can access the keytool application from any directory. If this is not the case, or you are unsure about your system’s environment variables, manually navigate to the “\bin” folder in the console via the cd command.

Type the following command to import the certificate, replacing any directories with the corresponding directories on your machine:

**keytool -import -trustcacerts -alias smtp.gmail.com -file C:\path\to\gmail.cert -keystore C:\path\to\glassfish7\glassfish\domains\[domain name]\config\cacerts.jks**

Glassfish’s keystore is located in the “\config” directory of the domain that you created during installation. If prompted to provide a password for the import action, enter “changeit”, which is the default password for “cacerts.jks”. It is recommended that this is changed at a later date.

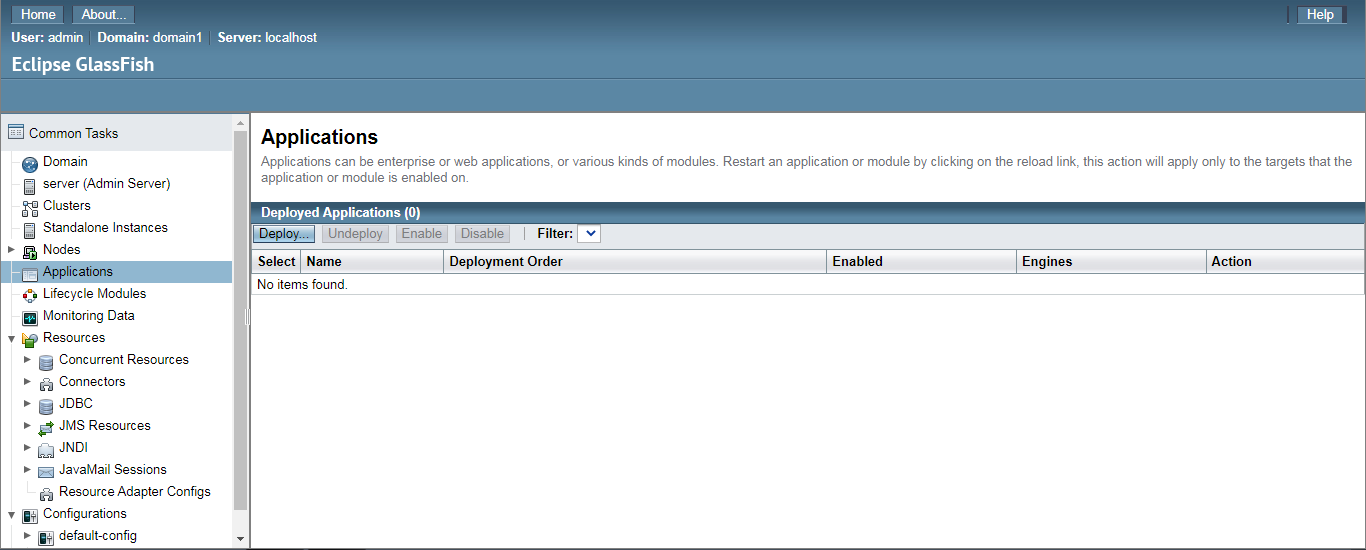
If prompted to trust Gmail’s SMTP certificate, type “yes” into the console. A message should then appear confirming that the import has been completed successfully. If Glassfish is currently running, restart to begin using the updated keystore.

When the SMTP certificate is successfully installed, you can type in your email login details, email address and app password, to allow the system to log in and send emails with successful creation of SMTP sessions. The String values that need to be changed are senderAddress and senderPassword on the NotifController.java class.

1. Deploy Application

The compiled web application is provided on the GitHub repository as a WAR file. You will need to download this file before proceeding.

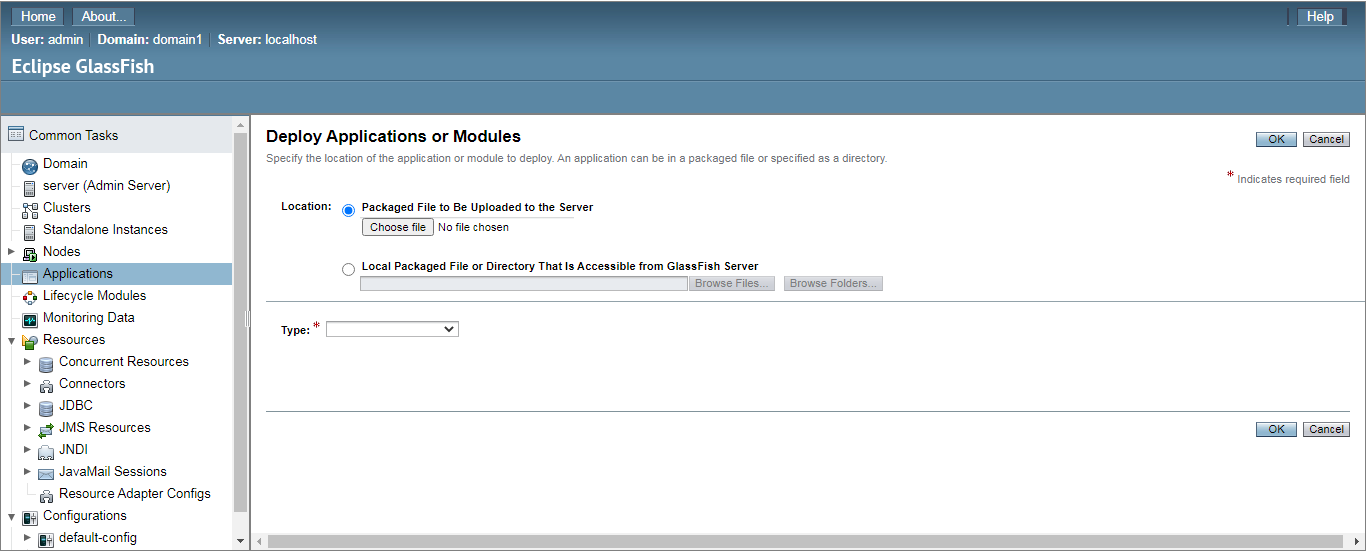
Start Glassfish if it is not already running, and navigate to the admin console via your web browser. On the side panel, click on the “Applications” option to bring up the list of currently deployed applications on this domain (Figure 27).



*Figure 27: Glassfish Applications menu (empty)*

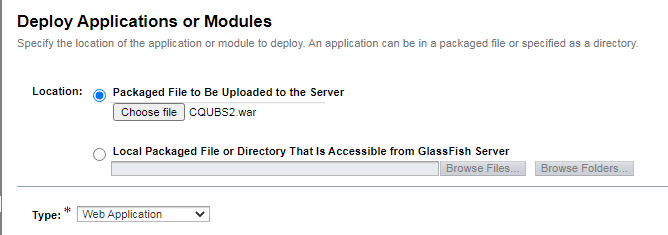
Select the option at the top of the table, “Deploy” to begin the deployment process.

Glassfish provides multiple ways to deploy an application. Since the WAR file has been downloaded locally on your machine, we will upload it to Glassfish using the provided file uploader (Figure 28).



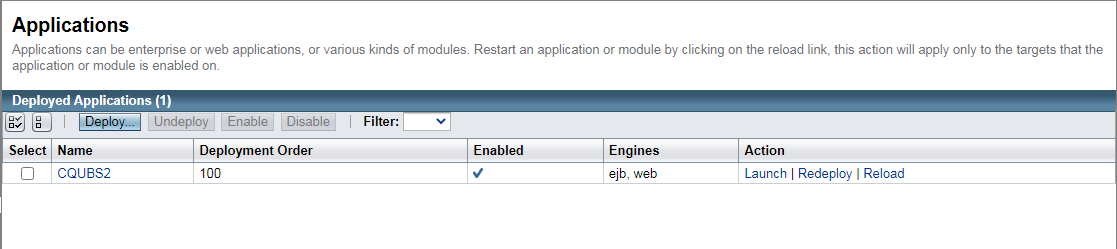
*Figure 28: Initial deploy applications screen*

The file is successfully uploaded when the name appears in the file uploader box (Figure 29), after which you can select “OK” at the top right of the screen to start deployment.



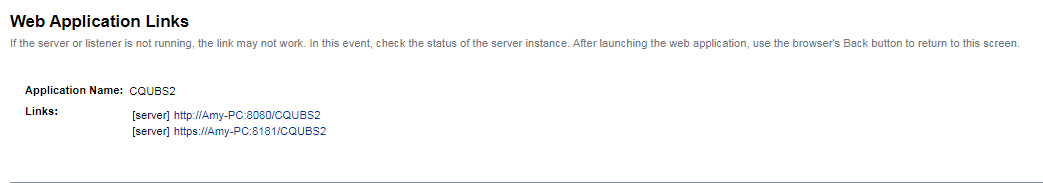
*Figure 29: Successful WAR file upload*

A successful deployment will redirect you back to the Applications page. You should now be able to see the CQUBS application displayed in the deployed application table (Figure 30).



*Figure 30: Applications menu after successful deployment of CQUBS*

From here, select the “Launch” option to launch CQUBS. When prompted to select a link from which to access the application, select the first (using the HTTP protocol), as shown in Figure 31.



*Figure 31: Links to launch CQUBS*

The home page of CQUBS should then load, indicating a successful launch.

## 

## Staff User Setup

This prototype does not have the facility to add new Staff users while the application is running. It is possible to add new Staff users or change existing data via a SQL script however, this would require a salt and password hash to be completed for the new user prior to entering it into the database. Four Staff users were created in the test data and the relevant data is in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Email | Password | Department |
| Amy Greenwood | amy.greenwood@cqumail.com | PhahYBHT179W | Allied Health |
| Aidan Petre | aidan.petre@cqumail.com | 4Tski4itt0uX | Nursing |
| Kahlia Heimann | kahlia.heimann@cqumail.com | T8XNT09KJcik | Trades School |
| Sangin Kim | sangin.kim@cqumail.com | cRU3XTLYaHom | Trades School |

## 

## Volunteer Login

|  |  |
| --- | --- |
| *Figure 32: Home page of the web application* | 1. Open the web application and select the User Login button |
| *Figure 33: Volunteer login page* | 1. Enter your registered email address. 2. Enter your password 3. Select the Login button |

## Volunteer Search

|  |  |
| --- | --- |
| *Figure 34: Volunteer home page* | 1. After logging in select the Search Services button to open the Find a Service page |
| *Figure 35: Find a service page (* | 1. Select the Location button to view the drop-down list of locations 2. Select one or more of the locations in the list 3. Select the Search icon |
| *Figure 36: Find a service page (categories)* | 1. To further filter the available services select the Category button to view the drop-down list of categories 2. Select one or more of the categories in the list 3. Select the Search icon |
| *Figure 37: Result of search* | 1. Select a Service to view the details of that Service |

## Staff Login

|  |  |
| --- | --- |
| *Figure 38: Web application home page* | 1. Open the web application and select the User Login button |
| *Figure 39: Staff login page* | 1. Enter your registered email address. 2. Enter your password 3. Select the Login button |

## Staff Adding Categories

## 

## 

|  |  |
| --- | --- |
| *Figure 40: Staff view of web application* | 1. After logging in select Categories from the navigation bar |
| *Figure 41: Staff category view* | 1. To add a category, select the Add Category button |
| *Figure 42: Add category page* | 1. Enter a Category Name 2. To add an image select the Browse button and select the image from your device 3. Select the Save button to add the new Category to the database |

## 

## Staff Updating Categories

## 

|  |  |
| --- | --- |
| *Figure 43: Staff view of web application* | 1. After logging in select Categories from the navigation bar |
| *Figure 44: Staff category view* | 1. To update a Category, select one from the list |
| *Figure 45: Category update page* | 1. To update a Category select the Edit Category button |
| *Figure 46: Edit category page* | 1. Modify the Category name if required 2. To change or replace the associated image select the Browse button and select the image from your device 3. Select the Save button to update the Category in the database |

## 

## 

## Staff Removing Categories

## 

|  |  |
| --- | --- |
| *Figure 47: Staff view of web application* | 1. After logging in select Categories from the navigation bar |
| *Figure 48: Staff category view* | 1. To remove a Category, select one from the list |
| *Figure 49: Edit category page* | 1. To remove a Category select the Remove Category button |
| *Figure 50: Prompt to remove category* | 1. At the prompt select the OK button to remove the category from the database |

## 

## Staff Adding Services

## 

## 

|  |  |
| --- | --- |
| *Figure 51: Staff view of web application* | 1. After logging in select Services from the navigation bar |
| *Figure 52: Staff services page* | 1. To add a Service select the Add Service button 2. Enter a Service name for the new Service 3. Select the Location button to select a Location from the drop-down list 4. Select the Category button to select a Category from the drop-down list 5. Select a price for the Service 6. Enter a description for the Service 7. Select the Browse button to upload an image from your device 8. To save the new Service select the Save button |
| *Figure 53: Add service page* |

## 

## Staff Updating Services

## 

## 

|  |  |
| --- | --- |
| *Figure 54: Staff view of web application* | 1. After logging in select Services from the navigation bar |
| *Figure 55: Staff services page* | 1. To update a Service select one from the list |
| *Figure 56:Edit services page* | 1. To update a Service select the Edit Service button 2. If required update the Service name 3. Select the Location button to change the Location via the drop-down list 4. Select the Category button to change the Category from the drop-down list 5. Select or alter a price for the Service 6. Enter or update a description for Service 7. Select the Browse button to upload an image from your device 8. To save the updated Service to the database select the Save button |
| *Figure 57: Edit services page* |

## 

## Staff Removing Services

## 

## 

|  |  |
| --- | --- |
| *Figure 58: Staff view of web application* | 1. After logging in select Services from the navigation bar |
| *Figure 59: Staff services page* | 1. To remove a Service select one from the list |
| *Figure 60: Edit services page* | 1. To remove a Service select the Remove Service button |
| *Figure 61: Prompt to remove a service* | 1. At the prompt select the OK button to remove the service from the database |

## 

# 

# Project Reflection

The main goal of our project was to increase the efficiency of CQUniversity’s existing “ad-hoc” management style for the process of locating and sourcing the required volunteers to assist students with learning practical skills. We aimed to do this by introducing a system that permits all parties involved in the volunteering process to access relevant services and information.

To achieve this goal, we developed our prototype to provide a central location to store the details of members of the public interested in receiving free or cheap services. Additionally, the web application we developed allows CQUniversity staff to add to a database of services by adding, updating, and removing both categories and services at various CQUniversity locations. These services are then available to all members of the public who can then search the available options by location, category or service type. Finally, we provided the ability for CQUniversity staff to send an email to all potential volunteers who have registered for a particular service, outlining service availability dates, booking procedures and other relevant information.

During the development of our prototype, we encountered several obstacles that required research to overcome. One example is the implementation of login and authentication for the web application which included using the SHA-512 hash algorithm to create a password hash. Another example involves the use of certificates and authentication regarding the JavaMail implementation to send the required emails for the application. Additionally, we encountered multiple issues surrounding the selected application server, GlassFish, that ranged from the inability to open the administration console to difficulties deploying the application.

Overcoming these challenges has formed an integral part of the prototype development and enabled us to achieve our project objectives. We were also given the opportunity to interact with the project stakeholders throughout the project, from requirements analysis to the completion of the finished prototype. This experience will assist us in our future professional lives when we need to communicate requirements and project updates with relevant clients. In addition, it has given us the opportunity to perform user acceptance testing with actual users of the web application.

The prototype we created meets the requirements we identified, stayed within the scope, and performed well on the user acceptance test. It also met the project objectives we previously outlined and achieved our goal of creating a prototype to improve the efficiency of volunteer management for CQUniversity courses.

# 

# References

Brown, R. (2023). *Possible student Capstone Project: Client Bookings Database* [Project Idea Notes]. CQUniversity.

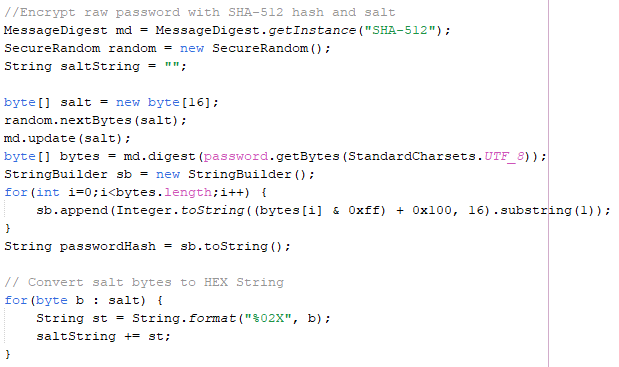
CQUniversity 2023, *CQUniversity*, viewed 26 July 2023, <https://www.cqu.edu.au/>

# Appendix A

## Implementation details

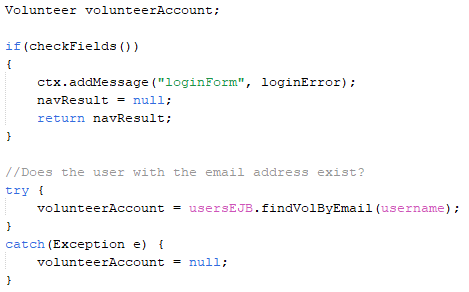
### Account Creation, Login and Authentication

When a guest user creates a volunteer account, the String value of the user password gets converted from UTF-8 standard character sets to a byte array. A random value of 16 bytes of byte array gets generated simultaneously and mixed with the password value converted into byte array. The mixture of password value and salt value gets encrypted with the SHA-512 hash algorithm. Mixing the raw password value with the salt value and the encryption process is handled by a MessageDigest object. The use of salt was decided to prevent possible brute-force attacks. After encryption, the hash value and the salt value will be saved on the user table of the database as an HEX string when the user account gets stored on the database.



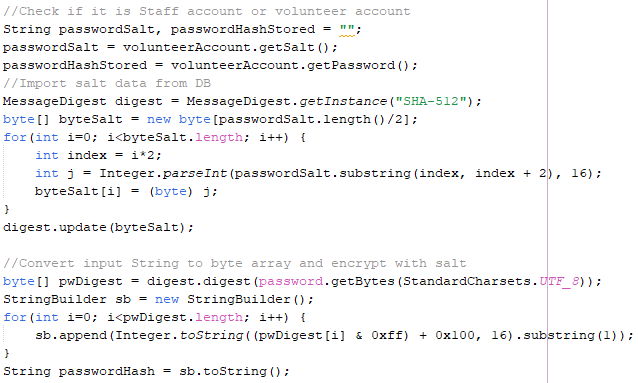
*Figure 62: Password encryption code from SignUpController.Java class for the creation of a volunteer account*

When the user attempts to log in using an account, the system searches for user accounts associated with the email address provided on the login page by using the search function of user EJB. The staff login page searches for staff accounts only, and the volunteer login page searches for volunteer accounts only.



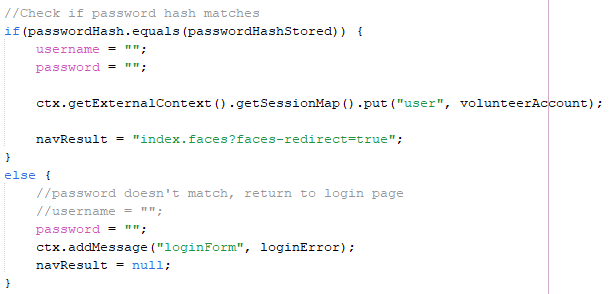
*Figure 63: Volunteer account search code from LoginController.java class using EJB search function*

Once the user details are found in the database, the system brings the salt and hash data dedicated to the user account. The system converts the converted salt HEX string back to byte arrays, combines the salt with the password input value, and then compares the new encrypted value to the previously stored hash value.



*Figure 64: The code on the LoginController.java class for encrypting the password input value with previously saved salt data from the database*

If the value of the new hash matches the hash value previously stored on the database, it means that the input password value is the same as before and thus the authentication for the user account is complete. The users who complete the authentication process can proceed to use the system with their accounts. However, if the new hash value does not match the previously stored hash value, then it means that the user typed in the wrong password for the account and thus the authentication has failed. The user who fails the authentication process stays on the login page.

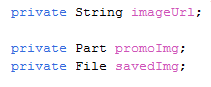


*Figure 65: The code on the LoginController.java class that compares two hashes and redirects the user to the volunteer home page if the hash values match*

When users log in to the system successfully, their account details get stored on the user session of the system. The system will remember the devices owned by the users that successfully logged in using their accounts with the FacesContext user sessions until the users log out.

### Image Uploading Methods

Image uploading functionality is included on the add and edit pages for both categories and services. The logic illustrated here has been taken from the CategoryAddController but is identical to the logic used in the other controller classes. Once a user uploads an image file through the UI, the file is stored in a variable of type Part, shown below alongside relevant controller attributes.



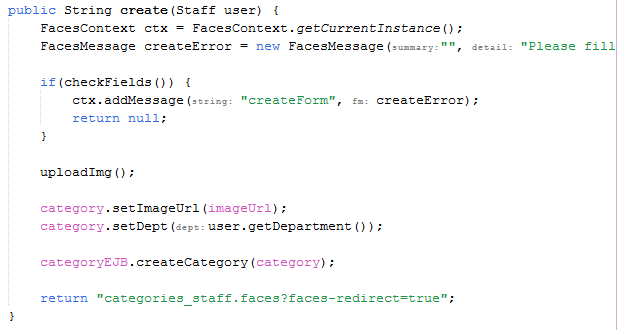
*Figure 66: Image-related attributes in controller*

The uploadImg() method in the controller is responsible for saving the image to the server and generating its URL string, which is stored as a part of its corresponding service/category in the database. This process is displayed in the following figure.



*Figure 67: uploadImg() method in controller*

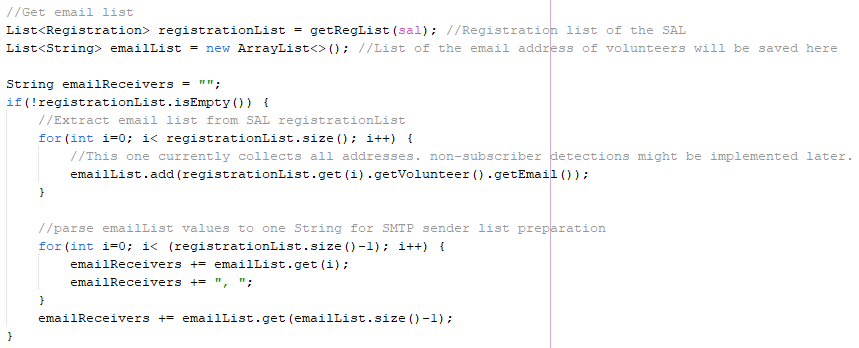
When the user submits the create/edit form, the appropriate method is executed within the controller. After checks have been made to ensure data has been entered correctly, uploadImg() is called. The generated image URL is then set inside the category/service object in preparation to be saved to the database. The excerpt below depicts where uploadImg() is called inside the create() method of the CategoryAddController.



*Figure 68: CategoryAddController’s create() method*

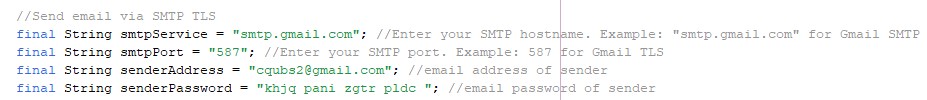
### Java Mail

Every ServiceAtLocation object, the object that has the details of an event at a specific location, has the list of volunteers registered for the specified events. When a staff asks the system to send emails to volunteers about an event, the system collects the email addresses of volunteers from the list of volunteers registered for the event.



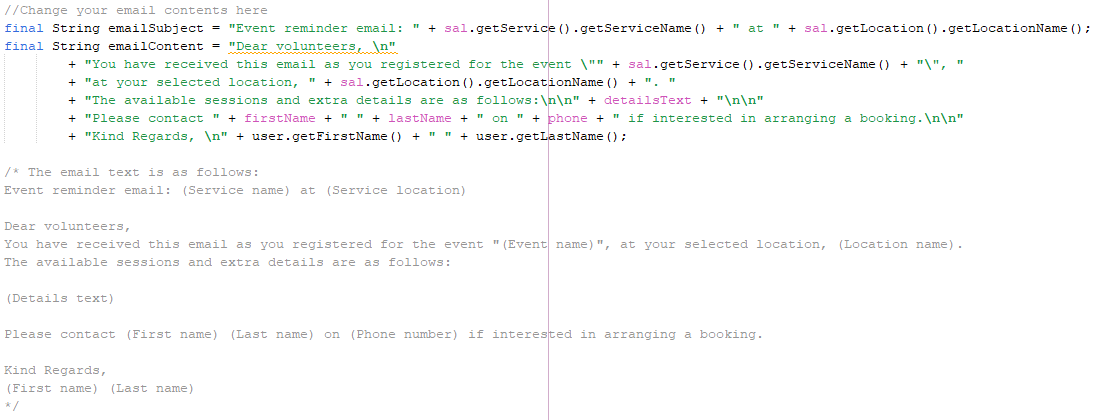
*Figure 69: The code on the NotifController.java class that collects the email addresses from the registered volunteers list*

As an attempt to send emails using the Java application, Jakarta Mail API was used for the implementation of the Simple Mail Transfer Protocol (SMTP) protocol. For testing, Google Gmail SMTP TLS service was used. The Gmail SMTP server has the hostname “smtp.gmail.com” and has port 587 open for TLS communication. The system administrator needs to include the sender address and app password for the system before compiling and deployment of the application. Please make sure that the password that needs to be typed into the code is an app password specially created for this application.



*Figure 70: The code on the NotifController.java class that initialises the email sender details*

The code below defines the text message of the email that is expected to be if the emails are correctly sent. The notation underneath is the preview of the expected message.



*Figure 71: The code on the NotifController.java class that initialises the email sender details*

With these processes, the information needed for sending email is prepared. The next step for the system is to create the properties for the SMTP session for sending emails with given properties. SMTP server login attempt with the use of the given email address and password is made when the session for SMTP is created.



*Figure 72: SMTP session creation code on the NotifController.java class*

The successful creation of an SMTP session means that the system has no issues with logging into the Gmail SMTP server and is thus ready to send email requests. A MimeMessage object will be created and handle the email send request.



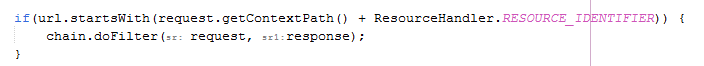
*Figure 73: SMTP email send request code on the NotifController.java class*

### Page Authentication Filter

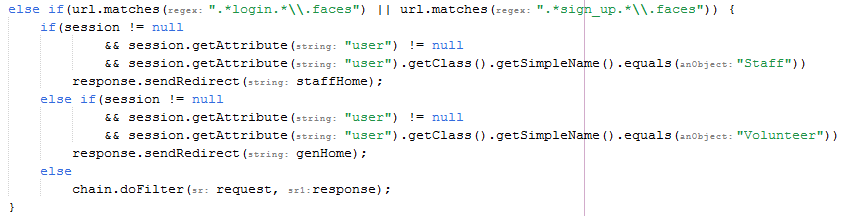
The AuthFilter class inherits from Filter, allowing for HTTP requests to be intercepted and served with the appropriate response depending upon the existing user session. This forms an important part of the application’s security, as it prevents guests from accessing admin pages and ensures the continued function of the session system.

The doFilter() method is responsible for responding to HTTP requests made by clients. It checks the URL of the requested page and compares it against multiple regular expressions to determine the page being visited. The existing session is then checked and if in violation of a rule (eg. a guest attempts to access the home page of a staff member) will send a redirect to the client.

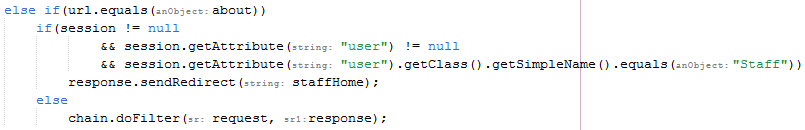
The following figures display the if-else statements used to filter requests inside doFilter().



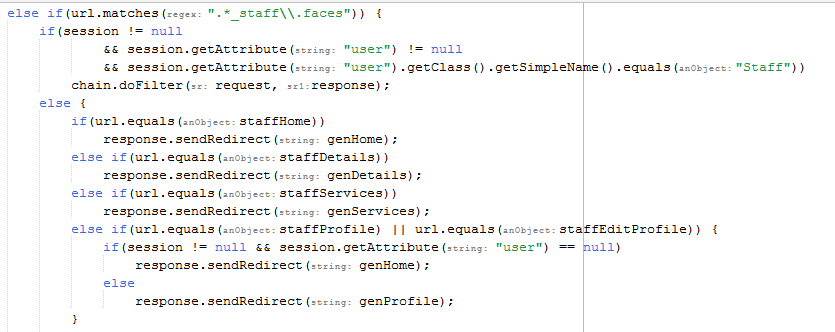
*Figure 74: Rule allowing for requests to any resource to be served*

**

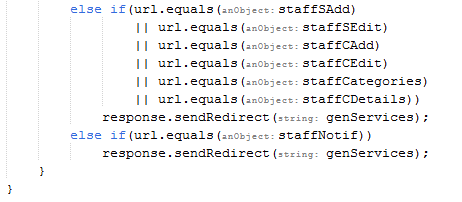
*Figure 75: Rule preventing a signed-in user from accessing login and sign-up pages*

**

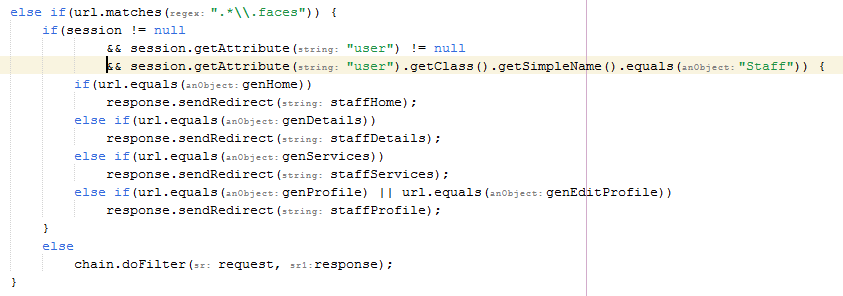
*Figure 76: Rule preventing staff accounts from accessing the About page*

**

*Figure 77: Rule preventing non-staff accounts from accessing staff pages (Part 1)*

**

*Figure 78: Rule preventing non staff accounts from accessing staff pages (Part 2)*

**

*Figure 79: Rule preventing staff accounts from accessing non-staff pages whilst still signed in*

**

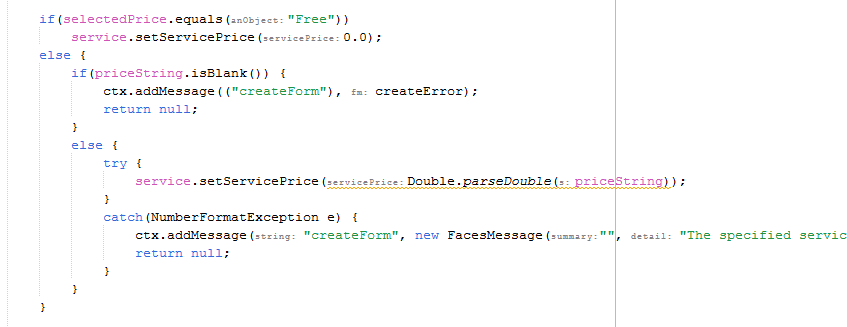
*Figure 80: Rule serving all other requests*

### Add and edit

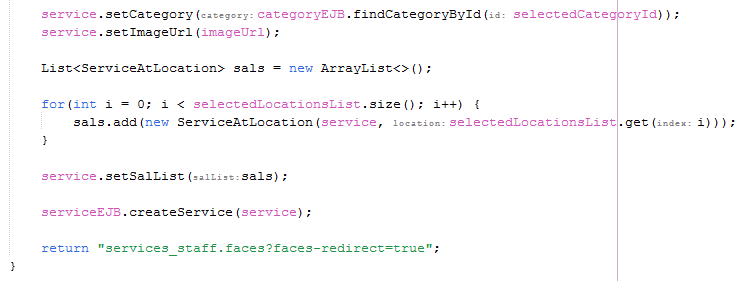
A key aspect of the admin functionality is the ability to add and edit both services and categories. These processes occur within a method in the controller that is executed once the user submits the corresponding form by clicking the save button. The create() method, used to add new services to the database, is shown below.



*Figure 81: create() method in ServiceAddController (Part 1)*

**

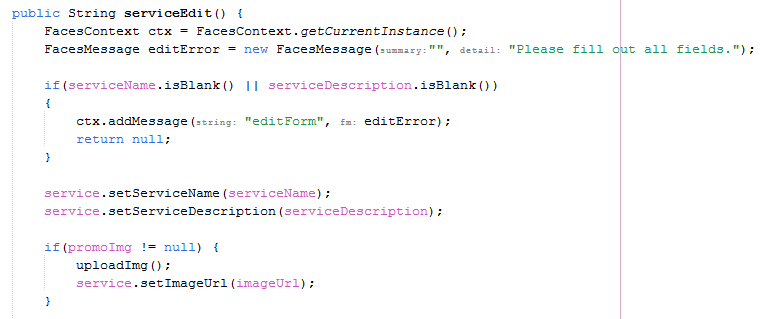
*Figure 82: create() method in ServiceAddController (Part 2)*

**

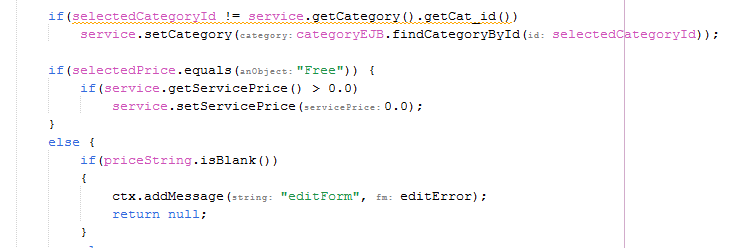
*Figure 83: create() method in ServiceAddController (Part 3)*

As illustrated, the create method first checks for empty fields and/or invalid data entered by the user. Attributes of the new service (ie. imageUrl, servicePrice, serviceDescription, category, ServiceAtLocations and serviceName) are acquired from the entered data and set in the Service object, which is then passed as a parameter to the createService method of ServiceEJB and persisted into the database. A successful creation redirects the user back to the staff services page, where the new service will be displayed.

Editing a service applies a similar logic, as displayed in the following figures.



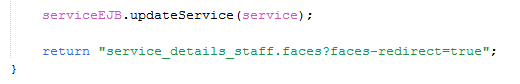
*Figure 84: serviceEdit() method in DetailsStaffController (Part 1)*

**

*Figure 85: serviceEdit() method in DetailsStaffController (Part 2)*

**

*Figure 86: serviceEdit() method in DetailsStaffController (Part 3)*

**

*Figure 87: serviceEdit() method in DetailsStaffController (Part 4)*