



# National College of Ireland

BSHCE

Software Development

2020/2021

Aaron Reilly

x17124719

[x17124719@student.ncirl.ie](mailto:x17124719@student.ncirl.ie)

CompaniesUnite

Technical Report

## Contents

Executive Summary.....	4
1.0 Introduction .....	5
1.1. Background .....	5
1.2. Aims.....	6
1.3. Technology.....	6
1.3.1 Java.....	6
1.3.2 JSON .....	7
1.3.3 ClearDB MySQL .....	8
1.3.4 JSP .....	8
1.3.7 Heroku.....	8
1.3.8 Eclipse .....	9
1.3.9 GitHub .....	9
1.4. Structure .....	10
1.4.1 System.....	10
1.4.2 Conclusion.....	10
1.4.3 Further Development or Research .....	11
1.4.4 References .....	11
2.0 System.....	11
2.1. Requirements.....	11
2.1.1. Functional Requirements.....	11
2.1.1.1. Use Case Diagram .....	11
2.1.1.2. Requirement 1 <Add Company Details>.....	12
2.1.1.3. Description & Priority.....	12
2.1.1.4. Use Case .....	12
2.1.1.5. Requirement 2 <Edit Company Details> .....	14
2.1.1.6. Description & Priority.....	14
2.1.1.7. Use Case .....	14
2.1.1.8. Requirement 3 <Delete Company Details>.....	16
2.1.1.9. Description & Priority.....	16
2.1.1.10. Use Case .....	16
2.1.1.11. Requirement 4 <Search Companies>.....	18
2.1.1.12. Description & Priority.....	18
2.1.1.13. Use Case .....	18

2.1.1.14.	Requirement 5 <User Registration>.....	20
2.1.1.15.	Description & Priority.....	20
2.1.1.16.	Use Case .....	20
2.1.2.	Data Requirements .....	22
2.1.2.1	ClearDB MySQL .....	22
2.1.2.2	JSON .....	24
2.1.3.	User Requirements .....	24
2.1.4.	Environmental Requirements .....	25
2.1.4.1	Heroku.....	25
2.1.4.2	ClearDB MySQL .....	26
2.1.4.3	Eclipse .....	26
2.1.4.4	Postman .....	26
2.1.5.	Usability Requirements.....	26
2.1.5.1	Performance .....	26
2.1.5.2	Accessibility.....	27
2.2.	Design & Architecture .....	27
2.3.	Implementation .....	28
2.3.1	Login Controller.....	28
2.3.2	Modify Company List Controller .....	29
2.3.3	Register Account Controller .....	31
2.3.4	Login JSP .....	32
2.3.5	Modify Company List JSP .....	33
2.3.6	Company Search JSP .....	34
2.3.7	Company Form JSP .....	35
2.3.8	Accounts Register.....	37
2.3.9	Web XML.....	38
2.3.10	Company Model Class.....	39
2.3.11	Companay Service .....	41
2.3.12	Company Resource .....	42
2.3.13	Account Model.....	43
2.3.15	Database Class .....	46
2.3.16	Update Company DAO .....	47
2.3.17	Account Details DAO.....	49
2.3.18	Hibernate.cfg.xml.....	49
2.4.	Graphical User Interface (GUI).....	50

2.4.1 Login Page .....	50
2.4.2 Register Company .....	50
2.4.3 Add New Company .....	51
2.4.4 Edit New Company.....	51
2.4.4 Register New Administrator Account .....	52
2.4.5 Search for Businesses.....	53
2.4.6 About Page .....	53
2.5. Testing.....	54
2.5.1 Unit Tests .....	54
2.5.2 Integration .....	58
2.5.3 End User Testing .....	62
2.6. Evaluation .....	68
3.0 Conclusions .....	69
4.0 Further Development or Research .....	70
5.0 References .....	70
6.0 Appendices.....	71
6.1. Project Plan .....	71
7.0 Objectives.....	73
8.0 Background .....	73
9.0 Technical Approach.....	73
10.0 Project Plan .....	74
11.0 Technical Details .....	76
12.0 Evaluation .....	76
12.1. Ethics Approval Application (only if required) .....	77
12.2. Reflective Journals .....	77
Reflective Journal (October) .....	77
Reflective Journal (November) .....	77
Reflective Journal (December).....	78
Reflective Journal (January) .....	79
Reflective Journal (February) .....	79
Reflective Journal (March) .....	80
Reflective Journal (April) .....	81
12.3. Other materials used .....	81
12.3.1 Think Aloud .....	81

## Executive Summary

Throughout the course of the pandemic, thousands of companies have been forced to shut down as they don't provide a particular service. Hospitality sector for example have been impact the most financially.

There are many reasons why businesses are forced to shut down throughout the pandemic. Some examples are:

- They're sector of business is not considered essential.
- They're unable to provide social distancing.
- Depending on the service they provide they could be considered not to open for a long period of time.

Given the restrictions, we have seen some businesses adapt under these circumstances.

Businesses delivering they're services to customer homes, whereas previously that wasn't the case. We've also seen a huge increase in food trucks and café trucks, in some cases these maybe some new ventures for businesses but also brands are following in this direction for their business to survive.

This project is a platform to help those looking to adapt under these new circumstances. Providing customers with different methods of retrieving data. For example, customers can use a web application or retrieve data through json format if they prefer.

Customers of this project are businesses who don't provide the all the necessary services required to stay open. Allowing them to find businesses/people in their area willing to work together.

Examples would be.

- Food trucks or Café trucks looking for open area where they can park and provide a service.
- Pubs looking for catering service who can set up shop quickly on their premises.
- Gastropubs and restaurants looking for delivery service to deliver their products to customer homes.

Given the reasons mentioned above, I believe this project is necessary, as it gives businesses a platform for surviving lockdowns.

One of the key benefits of this project is to bring businesses closer together. When the pandemic is over, I think many businesses will change their approach and services.

## 1.0 Introduction

### 1.1. Background

Reason behind CompaniesUnite is the fact businesses have been forced to shut down throughout the pandemic.

Businesses have been impacted in different ways some not as lucky as others. Just about all businesses in the hospitality sector were forced to close their doors for a considerable amount of time. Many of those businesses will never open their doors again, due to not being able to withstand the money constraint over the past year.

Other businesses had the ability to serve outside and keep social distancing. This allowed some businesses to open their doors at some stage in the last year. We have seen cases where if a business did not serve food they had to completely shut down.

Businesses found it difficult to respond and adapt in very little time. Leaving them with no option but to close. Over time we seen some business adapt to keep their business afloat.

CompaniesUnite is a platform which allows end users to see what services other businesses are looking to provide in order to survive the pandemic.

Services are provided in two ways.

- Web application
- API Service

As mentioned, end users will have the ability to use certain functionality. There are two types of users, Administrator and Normal User.

#### **Administration Functionality**

- Login to the platform to gain additional privileges.
- Ability to interact with the system through API
- Register new administrator accounts.
- Add new businesses to the database which will display on the platform.
- Remove businesses who wish not to be on the platform.
- Edit existing business, a business may change location or would like to update their details to make their business more intriguing to others.
- Logout

## Normal User Functionality

- Search for business by typing into a search box, search is conducted on two fields Postcode and Sector. Postcode allows end users to see businesses in areas they're interested in, and sector if they know what services they need.
- About page provides a description of what we do also providing the end user with the option to send information about the business they would like to be added to the platform.
- Ability to view the full list of businesses.

### 1.2. Aims

Purpose of this project is to build a platform that provides businesses with the opportunity to get their information added to our system. Allowing end users to view their interests along with the skills or products they can provide to others. Helping businesses to adapt and survive throughout the pandemic. Creating new relationships outside of their organisation.

Platform provides administrators with an API system, allowing them to interact with the application or by API requests.

Customer data will be stored on the cloud platform Heroku using ClearDB MySQL. Ensuring data is secure, along with availability provided by Heroku.

End users without the administrator privileges can only search and view businesses interested working with others.

It's not best practice to allow any user to interact with customer data. Ensuring security on all business data as only admin users have the privilege to add/edit/remove data.

The platform provides customers with data on businesses with the same interest, allowing them to communicate and work together. Given a business feels that their interests align contact details is available on the platform.

### 1.3. Technology

Description on the technologies used for this project, listing some of the implementations made with them.

#### 1.3.1 Java

Java is the core programming language used for this project. Java classes developed in this language include,

- Controller classes also known as Servlets.

- DAO Classes – Data Access Object used to connect to the database and create the SQL statements to interact with the database.
- Model Classes – Classes are used to create object for when we look to interact with the database. Annotations used here were @Entity, @Table, @Id and @Column, allowing to interact with the correct schema, table, and fields on the database.
- Resources – Developed to create the path when using API option. Setting the media type to JSON. Here we set the HTTP methods type for each request GET, POST, PUT and DELETE. Importing service class to use business logic available there.
- Services – Provides functionality to the resources class. It's the bridge to connect to the DAO class allowing us to interact with the database. Providing us with the ability to add, update or delete.
- Database connection class which is imported and used by multiple DAO classes. It's the class that connects to the hosted database on Heroku. The methods here are used to establish the connection.
- Unit Tests – Junit tests were developed in Java and executed on Eclipse IDE. Using assertEquals to ensure we are returning the values expected.

As the platform provides an API system, allowing request to manipulate the data, it was decided to go with JAX-RS as its reliable.

JAX-RS gives support to web-services which is used for this project. It allows us to use annotations and interfaces to make it easier and more robust in building the REST APIs. Cosgrave, N., 2020. RESTful Web Services – Part III (JAX-RS in a nutshell). *Topic 7*, pp.21,22,23,24,25,26.

### 1.3.2 JSON

The platform allows administrators to transfer and manipulate the customer data using JSON (JavaScript Object Notation) format.

JSON seems to be the go-to for most IT companies these days as it's easy to understand and read.

JSON is considered to be a lightweight format, which is why I decided to use it for the transferring of data for this project.



### 1.3.3 ClearDB MySQL

ClearDB MySQL through the Heroku platform provides the services of MySQL and more. Our application can allow administrators to read, write and update and delete from the MySQL database.

Providing the platform as a database as a service, ensuring the data is secure. Also providing us with the comfortability of knowing we don't have to deal with such things as database servers, database failures and advance storage.

Additional benefits of using ClearDB MySQL are the free bandwidth along with the 24/7 support they provide. The cost of storing my data for this project is also very cheap which is a nice benefit. (ClearDB Developer Center - Welcome, 2021)

### 1.3.4 JSP

JSP (Java Server Pages) are used within this project as it provides us with the ability for server-side programming. As JSP can use of Java API's which we take advantage of to connect to the database and provide search capabilities. Allowing end users to search for companies through on the UI.

Our JSP files are built on top of our Servlets, as the two are in communication. Most of the business logic is implemented in our Servlet classes, following best practices when it comes to coding. (Tyson, 2019)

JSTL (Java Server Pages Standard Tag Library) is the JSP tag which allows the platform to gain the functionality it provides. We used Core Tags and SQL Tags which allowed us to connect to the hosted database. Along with other functionalities such as displaying and quiring the data from the database. (JSTL Tutorial, JSTL Tags Example - JournalDev, 2013)

HTML and Bootstrap are all used throughout the JSP files.

### 1.3.7 Heroku

Heroku are the cloud providers chosen to deploy the platform on. They provide a quick and efficient service when looking to deploy a web application and a MySQL database.

Using ClearDB MySQL for the database, proved to be the right decision for me to make.

Heroku is extremely easy to use providing clear navigation to information on the instances such as Resources, Deploy, Metrics, Activity, Access and Settings.

#### **Platform details**

- App Name: testprojectunite
- Region: Europe
- Stack: Heroku-20

- Framework: Heroku-maven-plugin
- Slug size: 147.2 MiB of 500 MiB

The screenshot shows the Heroku dashboard for the application 'testprojectunite'. At the top, there are navigation tabs: Overview (selected), Resources, Deploy, Metrics, Activity, Access, and Settings. Below the navigation, there are three main sections:

- Installed add-ons:** Shows one add-on, 'ClearDB MySQL Ignite' with ID 'cleardb-rectangular-78676'. The cost is \$0.00/month. A 'Configure Add-ons' link is present.
- Dyno formation:** Shows 'This app is using free dynos'. Below this, a table lists the formation:
 

web	java \$JAVA_OPTS -jar webapp-runner.jar \$WEBAPP_RUNNER_OPTS...	ON
-----	---	----

 A 'Configure Dynos' link is present.
- Collaborator activity:** Shows one collaborator, 'x17124719@student.ncirl.ie', with 16 deploys. A 'Manage Access' link is present.

### 1.3.8 Eclipse

Eclipse IDE used to develop the maven web application, where we would import all the necessary dependencies through our .pom file. They provide useful integration with GitHub and Apache Tomcat. Tomcat was very useful throughout development phase, providing us with a HTTP web server allowing us to review our latest changes immediately on the browser.

### 1.3.9 GitHub

GitHub became very important throughout this project, a couple of times I'd have to pull the previous project again. It is a version control platform which allows us to store our

project code. Luckily after pushing regular commits to the remote project repository, there was always a backup version available.

GitHub is great for keeping track of the updates made for each commit, which makes it easy to find the changes made. If I got some major issues after a couple of updates on my local project. I found comparing my local copy against the working remote branch very useful. Saved me a lot of time if I wanted to get back to when the program was working.

## 1.4. Structure

### 1.4.1 System

In the System section of this report, we cover the project requirements for both functional and non-functional. In detail we provide each of the functional requirements by using use case diagrams, scope, and description.

Additional areas covered in this section are data requirements where we discuss how we integrate ClearDB MySQL into our system.

Other topics discussed are user requirement, environmental requirements, and usability requirements.

Design and Architecture discusses the overall structure of the system and how everything is connected and importance of each instance.

Implementation we list important java classes and JSP files, we provide a clear description followed by a screenshot under the description.

Graphical User Interface similar layout to the implementation section, we provide a description of each web page and explain why it's there and how its needed to provide the full application.

Testing we cover different types of testing, each of which are very important. Three types of testing covered are Unit, Integration and End User Tests. We must prove what we're delivering is valid and tested for end users.

We discuss our evaluation of the project, covering different topics such as tests ran, Think Aloud techniques taken by end users.

### 1.4.2 Conclusion

We discuss in detail the advantages and disadvantages of the project. Issues faced with JSP not importing CSS or JavaScript as expected. Also covering some of the strengths and limitations of the project itself.

### 1.4.3 Further Development or Research

This section we discuss possible further developments that can be implemented in the future. Discussing the problems faced during the integration of JavaScript and possible newer frameworks such as Angular and REACT.

### 1.4.4 References

We list all necessary references used for this document, along with research used for the implementation of the application itself.

## 2.0 System

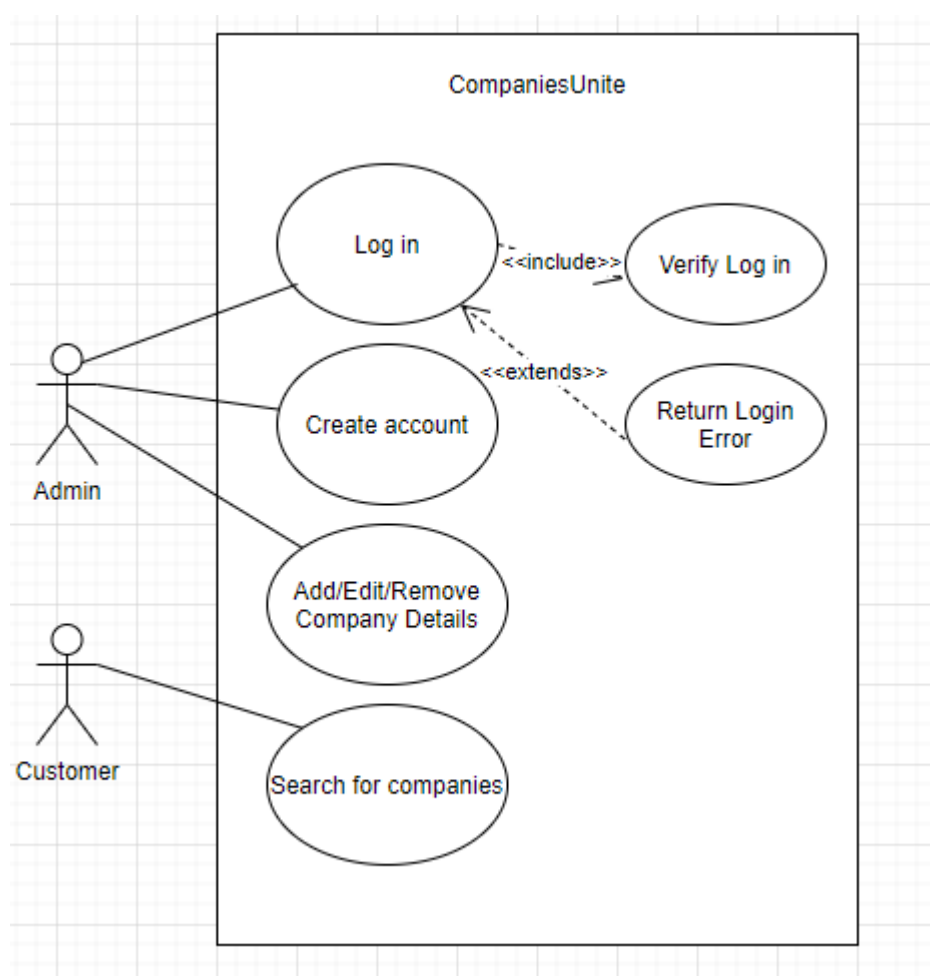
### 2.1. Requirements

#### 2.1.1. Functional Requirements

This section lists the functional requirements in **ranked order**.

##### 2.1.1.1. Use Case Diagram

The use case diagram below details an overview of the functional requirements.



## Figure 1: CompaniesUnite Use Case Diagram

### 2.1.1.2. Requirement 1 <Add Company Details>

### 2.1.1.3. Description & Priority

This use case describes how administrators can access the system and add important company data. Which in turn will be displayed to the public. This use case priority was set to critical as it covers the core functionality of the application.

### 2.1.1.4. Use Case

#### **ID**

UC02

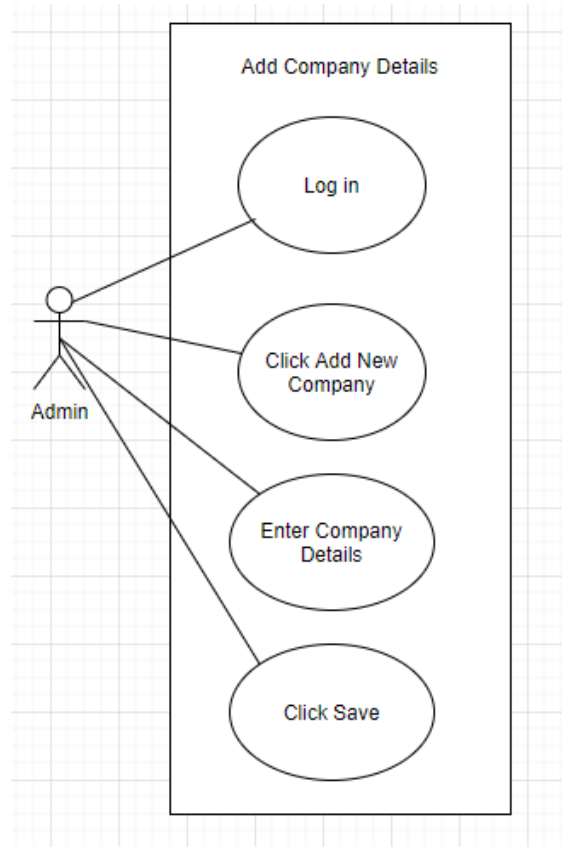
#### **Scope**

The scope of this use case is to add company information which later can be retrieved by end users.

#### **Description**

This use case allows the administrator to add a company's details to the database.

#### **Use Case Diagram**



**Figure 2: Add Company Use Case**

### **Flow Description**

#### **Precondition**

The administrator must be logged into the application using their registered account.

#### **Activation**

This use case starts when a user begins to write information in the form based on their company to the database.

#### **Main flow**

1. The user begins to fill out the form with company information (See E1)
2. The user presses the save button (See A1)
3. The system writes the information for that company to the database.
4. The user is brought to list of companies including the newly added company.
5. The system responds with company information.

#### **Alternate flow**

A1: <User fails to fill in mandatory fields>

1. The user unsuccessfully entered data to all mandatory fields.
2. The system will prompt a message telling the user which fields need to be filled in.
3. The user will then add to the mandatory fields.
4. The use case continues at position 2 of the main flow.

#### **Exceptional flow**

E1: <Server responds with failure>

5. The system will display a message if the system fails to write to the database.
6. The user will refresh the system.
7. The use case continues at position 1 of the main flow.

#### **Termination**

The system returns the user to the login page.

#### **Post condition**

The system stores the data sent to the server on the database.

#### [2.1.1.5. Requirement 2 <Edit Company Details>](#)

#### [2.1.1.6. Description & Priority](#)

This use case describes how administrators can edit company details. This can only happen by the request of that company. Companies might decide they haven't added enough details to be considered and try make themselves look more valuable.

This use case covers important functionality by providing the customer with the ability to change their detail at their wish. Priority of this use case is high.

#### [2.1.1.7. Use Case](#)

##### **ID**

UC03

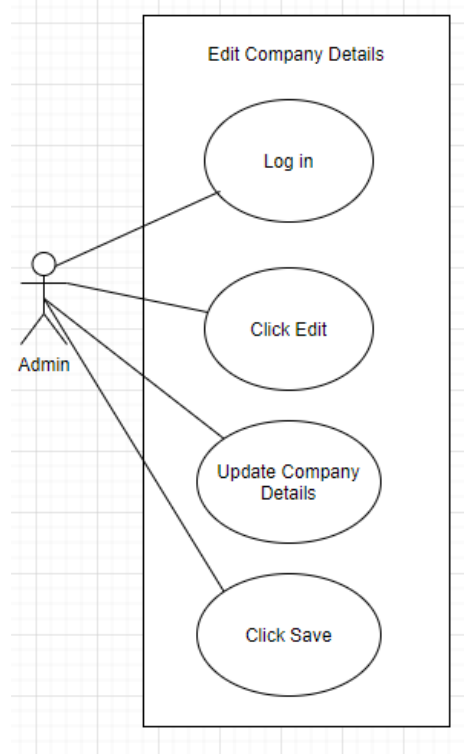
##### **Scope**

The scope of this use case is to edit the customer data which later can be retrieved by other users.

##### **Description**

This use case allows the administrators to edit their details to the database.

##### **Use Case Diagram**



**Figure 3: Edit Company Use Case**

### **Flow Description**

#### **Precondition**

The user must be logged into the application using their registered account.

#### **Activation**

This use case starts when a user clicks the edit option on the main administrator page.

#### **Main flow**

6. The user presses the edit button (See E1)
7. The user begins to fill out the form with company information (See A1)
8. The system writes the information for that company to the database.
9. The administrator is brought to list of companies including the newly updated company.
10. The system responds with company information.

#### **Alternate flow**

A1: <User fails to fill in mandatory fields>

8. The user unsuccessfully entered data to all mandatory fields.
9. The system will prompt a message telling the user which fields need to be filled in.
10. The user will then update the mandatory fields.



11. The use case continues at position 7 of the main flow.

#### **Exceptional flow**

E1: <Server responds with failure>

12. The system will display a message if the system fails to display the form.

13. The user will refresh the system.

14. The use case continues at position 6 of the main flow.

#### **Termination**

The system returns the user to the login page.

#### **Post condition**

The system stores the updated data sent to the server on the database

#### 2.1.1.8. [Requirement 3 <Delete Company Details>](#)

#### 2.1.1.9. [Description & Priority](#)

This use case describes how administrators can delete company details. If in the case a company chooses to no longer stay on the platform. A request will be sent to the administrator to remove them from the platform. Several reasons why a company may wish to be removed.

For example

- Found a match and successfully got what they needed from the platform.
- No longer have use for the platform, they may have to shut their doors for good.
- Decide they don't want their information out in the public.

This use case is extremely important as it covers concerns of the customer and core functionality making this highly important. Priority of this use case is high.

#### 2.1.1.10. [Use Case](#)

##### **ID**

UC04

##### **Scope**

The scope of this use case is to delete customer information which is no longer needed.

##### **Description**

This use case allows the administrator to delete a company's details when requested by that company from the database.

### Use Case Diagram

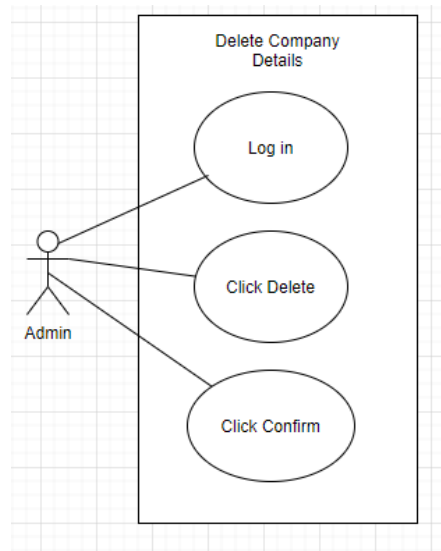


Figure 4: Remove Company details Use Case

### Flow Description

#### Precondition

The user must be logged into the application using their registered account.

#### Activation

This use case starts when a company wishes to have their details remove from the platform.

#### Main flow

11. The administrator Delete when successfully logged in.
12. The system removes the information for that company from the database (See E1)
13. The administrator is brought to list of companies (See A1)
14. The system responds with that company no longer available on the list of companies.

#### Alternate flow

A1 : <User fails to delete details>

1. The user unsuccessfully deletes the company data.
2. The user will retry the delete option.
3. The use case continues at position 12 of the main flow.

#### Exceptional flow

E1 : <System Fails to Remove Details>

4. The system will display a message telling the user there has been an error.
5. The user will check to see if their data is no longer available by querying.
6. The use case continues at position 11 of the main flow.

### **Termination**

The system presents the administrator with a warning if they wish to remove the data.

### **Post condition**

The system removes data from the database.

[2.1.1.11. Requirement 4 <Search Companies>](#)

[2.1.1.12. Description & Priority](#)

This use case explains how the end user can query the database and search for companies. There are two fields on which the end user can search from. Option to search by postcode if distance causes an issue for them. Also, they can search by sector, allowing them to see all businesses within the sector they're interested in. The priority of this ticket is high.

[2.1.1.13. Use Case](#)

### **ID**

UC05

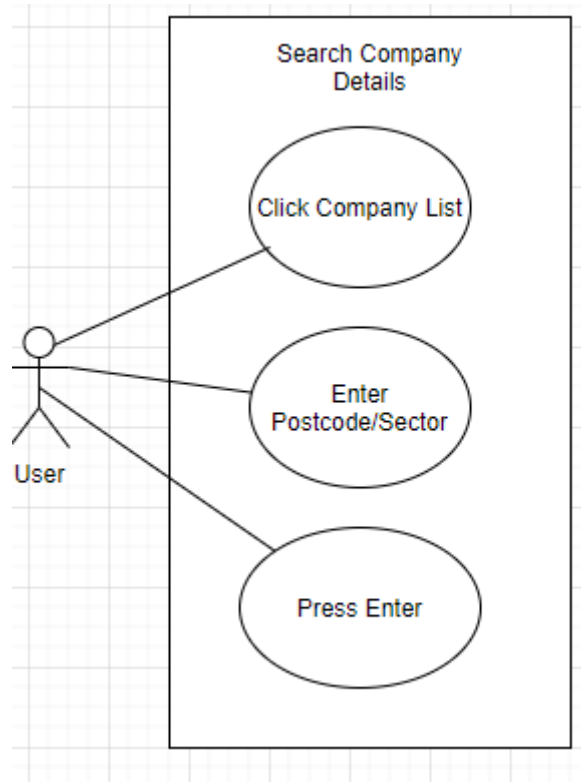
### **Scope**

The scope of this use case is to provide the ability to search for businesses within a postcode they enter or a sector they wish to enter.

### **Description**

This use case describes how users can search for businesses by postcode or sector.

### **Use Case Diagram**



**Figure 5: Search for Companies through postcode or sector Use Case**

**Flow Description**

**Precondition**

User is not required to be logged into the application.

**Activation**

This use case starts when a user types a postcode or sector into the search field.

**Main flow**

- 15. The user enters the postcode or sector they're interested in (See A1)
- 16. The user presses Enter to search (See E1)
- 17. The system identifies the postcode or sector the user entered.
- 18. The system returns the list businesses based on the postcode or sector entered.
- 19. The platform returns all businesses within that postcode or sector to the user.

**Alternate flow**

A1 : <No companies in that postcode>

- 7. The system responds with no businesses in the list (blank page)
- 8. The use case continues at position 15 of the main flow.

**Exceptional flow**

E1 : <Server Error>

9. The system responds with a message.
10. The user will be required to enter a postcode or sector.
11. The use case continues at position 17 of the main flow.

### **Termination**

The system presents with full list of businesses.

### **Post condition**

The system provides the details of the business within the postcode or sector.

[2.1.1.14. Requirement 5 <User Registration>](#)

[2.1.1.15. Description & Priority](#)

This use case explains how administrators can create an account through registration.

On login the username and password credentials will be compared to the ones we have stored on the database. Given a successful match the administrator will be granted access to the application. This use case is high priority as the administrator needs an account to login to use the application.

[2.1.1.16. Use Case](#)

### **ID**

UC01

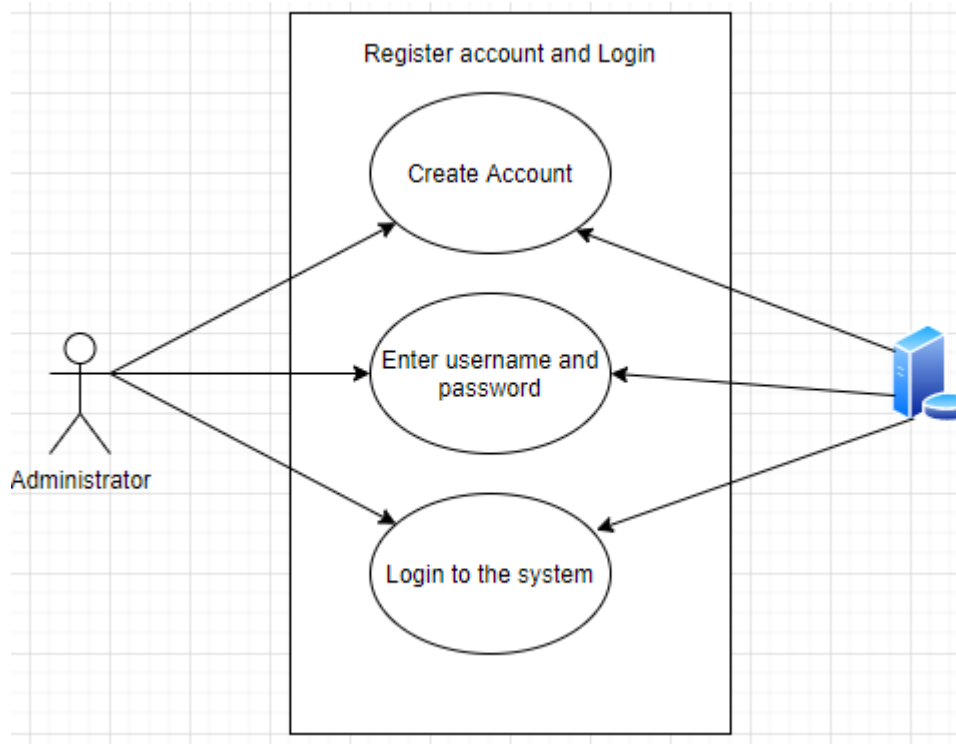
### **Scope**

The scope of this use case is to allow administrators to create an account and gain access to the system.

### **Description**

This use case describes how the administrators can register and sign into the system.

### **Use Case Diagram**



**Figure 6: Register Account Use Case**

### **Flow Description**

#### **Precondition**

The administrator must have access to the internet to create an account.

#### **Activation**

This use case starts when a user clicks registration option.

#### **Main flow**

20. The system will provide the administrator with the option to create an account (See E1)
21. The administrator will be required to enter in the user credentials for their account.
22. The system will add and store new user credentials to the database.
23. The system will prompt the user to login (See A1)
24. The user will enter their new credentials.
25. The system will confirm if they are valid credentials or not by comparing what is stored in the database.
26. The system will grant the user access and redirect them to a form they need to fill in.

#### **Alternate flow**

A1: <Wrong user credentials>

12. The user has entered the incorrect credentials.
13. The system will continue to show to login page.
14. The use case continues at position 24 of the main flow.

### Exceptional flow

E1: <User already stored in the database>

15. The user tries to register with credentials already stored in the database.
16. The system returns a message stating the user already exists.
17. The use case continues at position 21 of the main flow.

### Termination

The system presents the administrator with the login page.

### Post condition

The system stores new credentials in the database.

## 2.1.2. Data Requirements

### 2.1.2.1 ClearDB MySQL

All business and account details are stored remotely on ClearDB MySQL on Heroku.

Administrators will have the ability to Add, Update and Delete business information along with Account Details. All business data will be visible on the platform UI, where end users can search for the business location or their sector depending on what they choose.

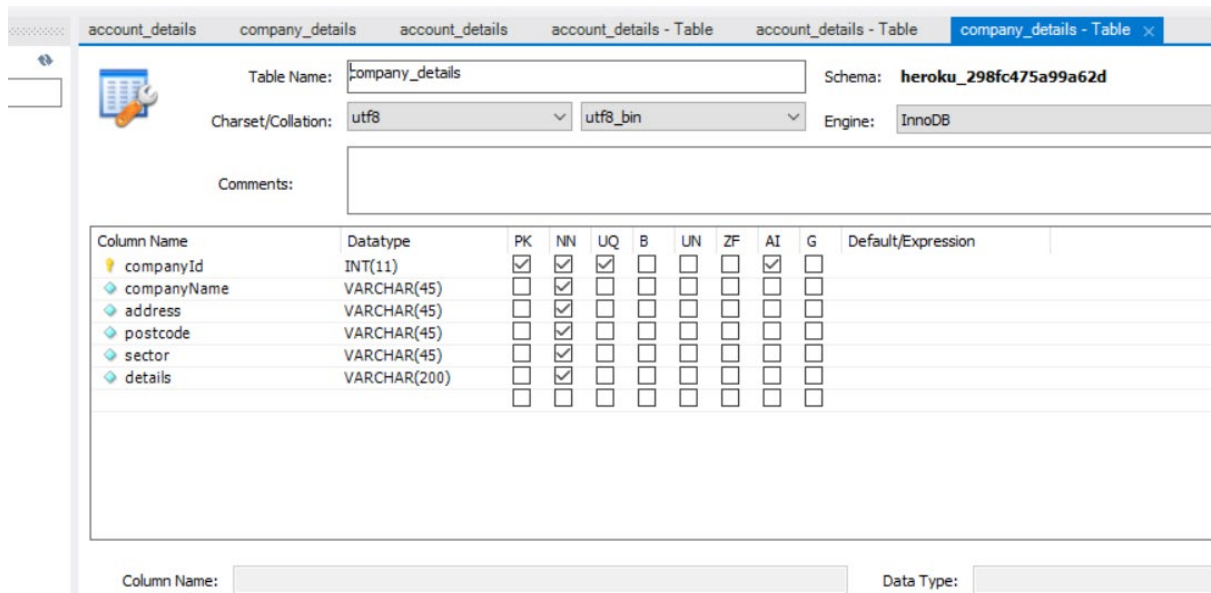
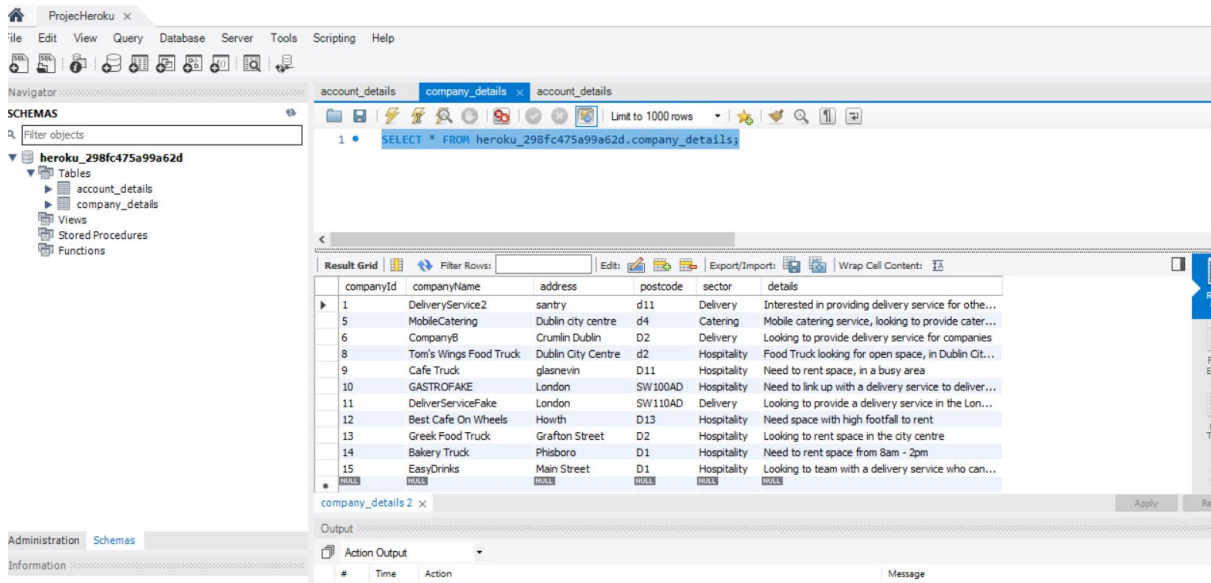
Screenshot of the hosted database on Heroku below.

A screenshot of the Heroku MySQL database management interface. At the top, there are tabs for 'Community Edition', 'Dev & Production Edition', 'Security', and 'How to Connect'. Below the tabs is a table with columns: Name, Cloud, Region, Status, Type, Last Updated, and Actions. The table contains one row with a green checkmark icon in the Name column, the name 'heroku\_298tc475a99a62d', 'Other' in the Cloud column, 'EU-West' in the Region column, 'Online' in the Status column, 'ignite' in the Type column, 'Sat May 08 2021 12:35:56 UTC' in the Last Updated column, and 'N/A' in the Actions column.

Name	Cloud	Region	Status	Type	Last Updated	Actions
✓ heroku_298tc475a99a62d	Other	EU-West	Online	ignite	Sat May 08 2021 12:35:56 UTC	N/A

Available below is a screenshot of the MySQL Heroku database. As you can see the schema name matches that of the hosted database shown in the previous screenshot.

Live company data is displayed when running the highlighted SQL query.



Screenshot below shows the admin table in our hosted database you can see the relationship between the two tables with the companyId.



The screenshot shows a database management interface with a query window containing the following SQL code:

```

1 • SELECT * FROM heroku_298fc475a99a62d.account_details;
2 • INSERT INTO heroku_298fc475a99a62d.account_details (companyId, first_name, last_name, username, password, phone, email) VALUES (
3 •

```

The result grid displays the following data:

accountId	companyId	first_name	last_name	username	password	phone	email
1	1	John	Smith	johnsmith	johnsmith123	0975644	johnsmith@fakeemail.com
2	1	Sarah	Foley	sfoley	sfoley123	5976755	sfoley@fakeemail.com
3	5	Bob	Sindair	bobsin	bobsin123	0686875	bobsin@fakeemail.com
4	5	Stephen	Johnson	stepjohn	ksjKfae	082905802	stepjohn@fakeemail.com
5	6	Rachel	Smith	raefakeaccount	sdgfsf	4323432	raefakeaccount@fakeemail.com
6	6	Carl	Paul	carPaul	carPaul	32466243	carPaul@fakeemail.com
7	6	Thomas	Corcoran	thomasCor	thomasCor	4678474	carPaul@fakeemail.com

Below the screenshot, the table structure for 'account\_details' is shown:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
accountId	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
companyId	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
first_name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
last_name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
phone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

### 2.1.2.2 JSON

JSON is the format we will use for transferring our data. Administrators can also use Postman which will return business data in JSON format if they're just interested in the data and not the UI.

### 2.1.3. User Requirements

The purpose of CompaniesUnite application is to provide businesses and others looking to venture into new professional areas with options to adapt to the current problems they're facing due to the pandemic.

Providing end users with a list of companies who share the same interests and are willing to work with others.

In this case the only major requirement is for the end user to have access to the internet, along with some interest in growing relationships outside of their organisation.

The end user can search the list by entering a postcode or by sector.

If sending their details to the administrators, they are required to be patient in the case they wish to have their details removed.

Users will need to be ok with having their data stored in ClearDB. Updates to the database should be visible on the application immediately.

The application will allow administrators to Add, Update and Delete businesses information on the hosted database at their request.



















Each user will have the ability to query for companies in a particular area or their sector.

## 2.1.4. Environmental Requirements

### 2.1.4.1 Heroku

This project will be hosted in the cloud using Heroku. By hosting this project in the cloud, we can ensure availability, portability, security and more. Below are the deployments made for this project to Heroku.

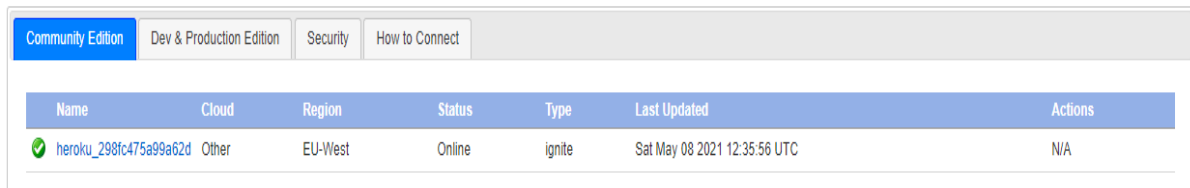
Latest activity [All Activity](#) ↕

		x17124719@student.ncirl.ie: Deployed <code>0.0.1-SNAPSHOT</code> Today at 10:38 AM · v21
		x17124719@student.ncirl.ie: <b>Build succeeded</b> Today at 10:37 AM · <a href="#">View build log</a>
		x17124719@student.ncirl.ie: Deployed <code>0.0.1-SNAPSHOT</code> May 10 at 7:58 AM · v20
		x17124719@student.ncirl.ie: <b>Build succeeded</b> May 10 at 7:57 AM · <a href="#">View build log</a>
		x17124719@student.ncirl.ie: Deployed <code>0.0.1-SNAPSHOT</code> May 10 at 7:50 AM · v19
		x17124719@student.ncirl.ie: <b>Build succeeded</b> May 10 at 7:50 AM · <a href="#">View build log</a>
		x17124719@student.ncirl.ie: Deployed <code>0.0.1-SNAPSHOT</code> May 8 at 9:01 PM · v18
		x17124719@student.ncirl.ie: <b>Build succeeded</b> May 8 at 9:01 PM · <a href="#">View build log</a>
		x17124719@student.ncirl.ie: Deployed <code>0.0.1-SNAPSHOT</code> May 8 at 8:55 PM · v17

#### 2.1.4.2 ClearDB MySQL

ClearDB should be always running online if we want to display the data to the end user. If we have a case where Heroku fails to provide data online, it's a Critical issue which should be addressed immediately.

Screenshot below show some details of the ClearDB instance used for this project.



The screenshot shows the ClearDB console interface. At the top, there are tabs for 'Community Edition', 'Dev & Production Edition', 'Security', and 'How to Connect'. Below the tabs is a table with the following columns: Name, Cloud, Region, Status, Type, Last Updated, and Actions. The table contains one row with the following data:

Name	Cloud	Region	Status	Type	Last Updated	Actions
heroku_298fc475a99a62d	Other	EU-West	Online	ignite	Sat May 08 2021 12:35:56 UTC	N/A

#### 2.1.4.3 Eclipse

Eclipse IDE proved to very useful IDE the integration with Apache and Github saved some time as I have experience with this IDE in the past. Java POJO Classes, Database Connection Class, Servlets, and JSP files were all implemented using this IDE.

#### 2.1.4.4 Postman

Postman used for our integration testing, ensuring the integration over multiple java classes while the data is being used as agreed. Confirming the major functionalities work by sending HTTP methods such as GET, POST, PUT and DELETE.

Administrators can also use this to add, update and delete company data quicker.

### 2.1.5. Usability Requirements

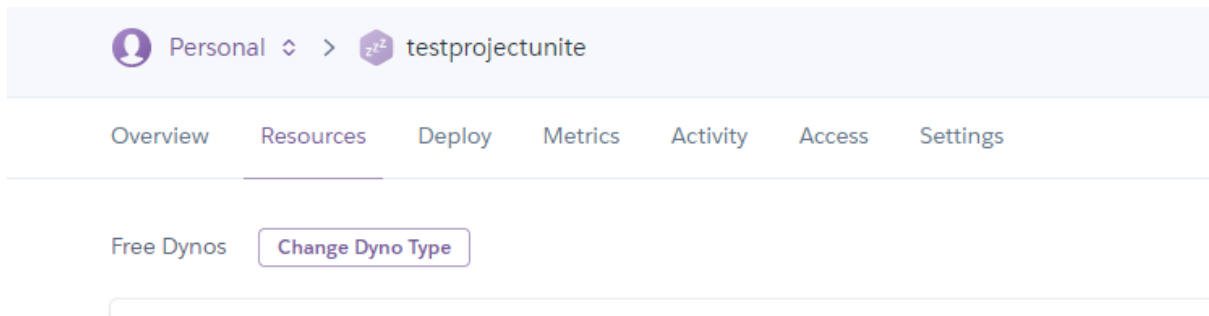
#### 2.1.5.1 Performance

For this project performance is guaranteed, using the Dyno Type seen in the screenshot below. We are given 512 MB of Memory (RAM), CPU Share 1x, Compute of 1x-4x.

It's enough at this moment for the purpose of this project.

If the decision is made to continue this project and add new features, Dyno type can be easily upgraded. For now, the Dyno type in use meets our usability requirements.

We have set our instance up in the EU, which provides less latency given our location.



### 2.1.5.2 Accessibility

Having hosted the application in Heroku cloud, we can ensure the hardware required to run the software is available when needed. Given that this is a project, we're using the free dynos option which can sleep. In the case of where there is no web traffic for 30 minutes the sleep will be set to yes. It provides the accessibility required at this time.

## 2.2. Design & Architecture

Separation of Concerns (SoC) was decided as a design pattern for Java project. Learning this design pattern in the Web Services and API Development module I was keen to use the material I was learning in the college.

Resource packages containing the classes used for building the paths using the API option, also importing the service class implementation. Allowing for the interaction with the database using different HTTP methods and JSON requests.

Our JSP files were used for server-side programming, getting the full use of Java API's. JSP files are built on top of our servlets, where a lot of the business logic is located.

Throughout the development of this project our data was stored on MySQL. While looking to deploy the project on the cloud, realising all the data will need to be migrated.

Heroku platform provides ClearDB MySQL, while also using Heroku in our Cloud Computing module it was decided to go with Heroku over AWS. All functionality is deployed successfully which is brilliant.

## Overview of the hardware architecture

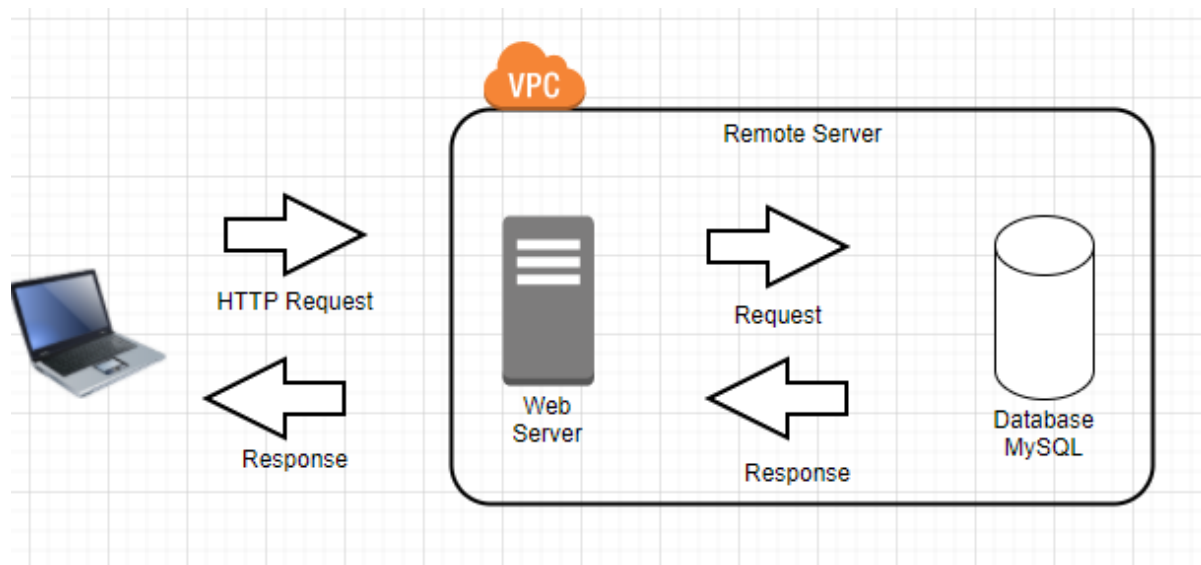


Figure 6: CompaniesUnite hardware architecture diagram

## 2.3. Implementation

### 2.3.1 Login Controller

Login Servlet class takes in the username and password entered through the Login.jsp. Checking if the username and password are valid by connecting to the Account Details DAO class.

We validate by creating an object of account details DAO using the validate method available in this class. We take in the username and password and pass the account object into the validate method. If that username and password are stored in the database as seen in the if statement.

If the username and password are valid the Administrator will be brought to the Modify Company List Controller. Else if the username and password doesn't exist in the database and are not valid, they will remain at the login page.

```

14 @WebServlet("/Login")
15 public class LoginController extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17     private AccountDetailsDAO accountDetailsDAO = new AccountDetailsDAO();
18
19
20     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
21         // TODO Auto-generated method stub
22         response.getWriter().append("Served at: ").append(request.getContextPath());
23     }
24
25
26     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27         String username = request.getParameter("username");
28         String password = request.getParameter("password");
29
30         Account account = new Account();
31         account.setUsername(username);
32         account.setPassword(password);
33
34         try {
35             if (accountDetailsDAO.validate(account)) {
36                 response.sendRedirect("ModifyCompanyList");
37             } else {
38                 response.sendRedirect("Login.jsp");
39             }
40         } catch (ClassNotFoundException e) {
41             e.printStackTrace();
42         }
43     }
44 }
45
46 }

```

### 2.3.2 Modify Company List Controller

Modify Company List servlet is developed to handle a couple of different interactions with the application. By using a switch statement, depending on the action taken through our JSP file we'll run the method assigned to that case in the servlet. We can call different actions in our JSP file.

On line 22 we create an object of type UpdatedCompanyDAO, giving us the functionality to interact with the database.

List of interactions with the database we can do from this servlet are listed below.

- Insert Company
- Update Company
- Delete Company

Within the switch statement are methods that also have no interaction with the database such as "/new" and "/edit". These methods display the forms necessary for us to add or update to the database.

```

18 @WebServlet("/")
19 public class ModifyCompanyList extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21
22     private UpdateCompanyDAO updateCompanyDAO;
23
24     public void init() {
25         updateCompanyDAO = new UpdateCompanyDAO();
26     }
27
28     protected void doPost(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException {
30         doGet(request, response);
31     }
32
33
34     protected void doGet(HttpServletRequest request, HttpServletResponse response)
35         throws ServletException, IOException {
36
37         String action = request.getServletPath();
38
39         try {
40             switch (action) {
41                 case "/new":
42                     showNewForm(request, response);
43                     break;
44                 case "/insert":
45                     insertCompany(request, response);
46                     break;
47                 case "/delete":
48                     deleteCompany(request, response);
49                     break;
50                 case "/edit":
51                     showEditForm(request, response);
52                     break;
53                 case "/update":
54                     updateCompany(request, response);
55                     break;
56                 default:
57                     listCompanies(request, response);
58                     break;
59             }
60         } catch (SQLException ex) {
61             throw new ServletException(ex);
62         }
63
64     }

```

```

66# private void showNewForm(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
67     RequestDispatcher dispatcher = request.getRequestDispatcher("companyForm.jsp");
68     dispatcher.forward(request, response);
69 }
70
71# private void insertCompany(HttpServletRequest request, HttpServletResponse response) throws IOException, SQLException {
72     String companyName = request.getParameter("companyName");
73     String address = request.getParameter("address");
74     String postcode = request.getParameter("postcode");
75     String sector = request.getParameter("sector");
76     String details = request.getParameter("details");
77     Company newCompany = new Company(companyName, address, postcode, sector, details);
78     updateCompanyDAO.insertCompany(newCompany);
79     response.sendRedirect("list");
80 }
81
82# private void deleteCompany(HttpServletRequest request, HttpServletResponse response) throws SQLException, IOException {
83     int companyId = Integer.parseInt(request.getParameter("companyId"));
84     updateCompanyDAO.deleteCompany(companyId);
85     response.sendRedirect("list");
86 }
87
88# private void showEditForm(HttpServletRequest request, HttpServletResponse response) throws SQLException, ServletException, IOException {
89     int companyId = Integer.parseInt(request.getParameter("companyId"));
90     Company existingCompany = updateCompanyDAO.selectCompany(companyId);
91     RequestDispatcher dispatcher = request.getRequestDispatcher("companyForm.jsp");
92     request.setAttribute("company", existingCompany);
93     dispatcher.forward(request, response);
94 }
95
96# private void updateCompany(HttpServletRequest request, HttpServletResponse response) throws SQLException, IOException{
97     int companyId = Integer.parseInt(request.getParameter("companyId"));
98     String companyName = request.getParameter("companyName");
99     String address = request.getParameter("address");
100    String postcode = request.getParameter("postcode");
101    String sector = request.getParameter("sector");
102    String details = request.getParameter("details");
103
104    Company updateCompany = new Company(companyId, companyName, address, postcode, sector, details);
105    updateCompanyDAO.updateCompany(updateCompany);
106    response.sendRedirect("list");
107 }
108
109# private void listCompanies(HttpServletRequest request, HttpServletResponse response) throws SQLException, IOException, ServletException {
110    List<Company> listCompanies = updateCompanyDAO.selectAllCompanies();
111    request.setAttribute("listCompanies", listCompanies);
112    RequestDispatcher dispatcher = request.getRequestDispatcher("modifyCompanyList.jsp");
113    dispatcher.forward(request, response);
114 }
115 }

```

### 2.3.3 Register Account Controller

Account Servlet class creates an object of type AccountDetailsDAO, which provides us with the ability to use their methods such as registerAccountDetails(). Line 57 we use this method passing the account object as a parameter. Which contains the new administrator account details, allowing for the registration of that new account.

It's required the administrator enters data to all necessary fields. Providing a successful insertion of a new administrator account the end user will be redirected to the list of companies' page. Line of code for this is outside the try catch on line 62.

WebServlet annotation is set to ("/register"), if you go to accountregister.jsp you can see we use the same path.



```

15 @WebServlet("/register")
16 public class AccountController extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     private AccountDetailsDAO accountDetailsDAO = new AccountDetailsDAO();
20
21     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
22
23         //Add the accounts to request object
24         response.getWriter().append("Served at: ").append(request.getContextPath());
25
26         //Get dispatcher
27         RequestDispatcher dispatcher = request.getRequestDispatcher("/accountregister.jsp");
28         //Forward the req and res objects
29         dispatcher.forward(request, response);
30     }
31
32     /**
33      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
34      */
35     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
36
37         String firstName = request.getParameter("firstName");
38         String lastName = request.getParameter("lastName");
39         String username = request.getParameter("username");
40         String password = request.getParameter("password");
41         String phone = request.getParameter("phone");
42         String email = request.getParameter("email");
43
44         System.out.println("firstName"+firstName);
45         System.out.println("lastName"+lastName);
46         System.out.println("email"+email);
47
48         Account account = new Account();
49         account.setFirstName(firstName);
50         account.setLastName(lastName);
51         account.setUsername(username);
52         account.setPassword(password);
53         account.setPhone(phone);
54         account.setEmail(email);
55
56         try {
57             accountDetailsDAO.registerAccountDetails(account);
58             //accountDetailsDAO.registerAccountDetails(accountDetails);
59         } catch (SQLException e) {
60             // TODO Auto-generated catch block
61             e.printStackTrace();
62         }
63         response.sendRedirect("Login.jsp");
64
65     }
66 }

```

### 2.3.4 Login JSP

Within this JSP we have our Login form, also we import a header and footer.

As you can see on line 13 we are using the action “request.getContextPath()” with /Login path appended at the end. This allows us to get the requested servlet in this case it’s the Login servlet. Providing a connection between the servlet and JSP gaining the functionality the Login Servlet provides.

```

1 |<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2 |   pageEncoding="ISO-8859-1"%>
3 |<!DOCTYPE html>
4 |<html>
5 |<head>
6 |  <meta charset="ISO-8859-1">
7 |  <title>Login</title>
8 |  <link href="css/bootstrap.css" rel="stylesheet" type="text/css">
9 |  <script defer src="password.js"></script>
10 |</head>
11 |<%@ include file="leastPrivilegeHeader.jsp"%>
12 |<body>
13 |<form id="form" action="<%=request.getContextPath()%>/Login" method="post">
14 |  <div class="container">
15 |    <div id="error"></div>
16 |    <div class="row">
17 |      <div class="col-lg-3"></div>
18 |      <div class="col-sm-6 col-xs-12">
19 |        <div class="jumbotron">
20 |          <h1 class="text-center">Login Page</h1>
21 |          <br>
22 |          <div class="form-group">
23 |            <label class="control-label" for="username">Username</label>
24 |            <input id="username" type="text" name="username" placeholder="Username" required>
25 |          </div>
26 |          <div class="form-group">
27 |            <label class="control-label" for="password">Password</label>
28 |            <input id="password" type="text" name="password" placeholder="Password" required>
29 |          </div>
30 |          <input type="checkbox" name="remember"> Remember Me?
31 |          <br><br>
32 |          <div class="pull-right">
33 |            <button type="submit" class="btn btn-outline-warning">Login</button>
34 |            <button type="reset" class="btn btn-outline-dark">Close</button>
35 |          </div>
36 |        </div>
37 |      </div>
38 |    <div class="col-md-3"></div>
39 |  </div>
40 |</div>
41 |
42 |</form>
43 |
44 |<script type="text/javascript" src="js/bootstrap.js"></script>
45 |<script type="text/javascript" src="js/jquery.js"></script>
46 |</body>
47 |<%@ include file="footer.jsp"%>
48 |</html>

```

### 2.3.5 Modify Company List JSP

Provides the UI with the ability to see full list of businesses and their information. Which is all retrieved from the hosted database.

Other options provided by this JSP are:

- Add New Company
- Edit Company Details
- Delete Company Details

On line 19 you can see the `getContextPath() /new` is matching that of the switch statement in the `ModifyCompanyList Servlet`. Which means if this button is clicked, the method `showNewForm()` under `"/new"` in the switch statement is executed, which provides us with the company form JSP.

```

1 |<% page language="java" contentType="text/html; charset=ISO-8859-1"
2 |   pageEncoding="ISO-8859-1"%>
3 |   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 | <!DOCTYPE html>
5 | <html>
6 | <head>
7 |   <meta charset="ISO-8859-1">
8 |   <title>Company List</title>
9 |   <link href="https://unpkg.com/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet" type="text/css">
10 | </head>
11 | <%@ include file="adminHeader.jsp"%>
12 | <body>
13 |
14 |   <div class="container">
15 |     <h3 class="text-center">Register Company</h3>
16 |     <hr>
17 |     <div class="container text-left">
18 |
19 |       <a href="<%=request.getContextPath()%>/new" class="btn btn-success">Add
20 |         New Company</a>
21 |     </div>
22 |     <br>
23 |     <table border = "1" class="table table-striped table-bordered">
24 |       <tr class="thead-dark">
25 |         <th>ID</th>
26 |         <th>Name</th>
27 |         <th>Address</th>
28 |         <th>Postcode</th>
29 |         <th>Sector</th>
30 |         <th>Additional Information</th>
31 |         <th>Modify/Delete</th>
32 |       </tr>
33 |       <c:forEach items = "${listCompanies}" var = "companyDetails">
34 |         <tr>
35 |           <td><c:out value="${companyDetails.companyId}" /></td>
36 |           <td><c:out value="${companyDetails.companyName}" /></td>
37 |           <td><c:out value="${companyDetails.address}" /></td>
38 |           <td><c:out value="${companyDetails.postcode}" /></td>
39 |           <td><c:out value="${companyDetails.sector}" /></td>
40 |           <td><c:out value="${companyDetails.details}" /></td>
41 |
42 |           <td><a href="edit?companyId=<c:out value='${companyDetails.companyId}' />">Edit</a>
43 |             / <a href="delete?companyId=<c:out value='${companyDetails.companyId}' />">Delete</a></td>
44 |         </tr>
45 |       </c:forEach>
46 |     </table>
47 |   </div>
48 | </body>
49 |
50 | <%@ include file="footer.jsp"%>
51 | </html>

```

(JSP - Standard Tag Library (JSTL) Tutorial - Tutorialspoint, 2021)

### 2.3.6 Company Search JSP

As you can see, we're importing the java libraries into this JSP file. These imports on line 1 and 2 allow us to write the code in the jsp file for connecting and filtering the data in the hosted database.

Line 40 we pass in what the end user searched as a parameter assigning to the variable result. Result is then used in the SQL query to filter the data in the database, whether a postcode or sector has been entered.

```

companySearch.jsp
1 <%@page import="java.sql.*" %>
2 <%@page import="java.sql.Connection" %>
3 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
4   pageEncoding="ISO-8859-1"%>
5 <!DOCTYPE html>
6 <html>
7 <head>
8 <meta charset="ISO-8859-1">
9 <title>Company List</title>
10 <link href="https://unpkg.com/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet" type="text/css">
11 </head>
12 <%@ include file="LeastPrivilegeHeader.jsp"%>
13 <body>
14
15 <div class="col-md-4 mb-2">
16 <form action="" method="get">
17 <input type="text" class="form-control" name="search" placeholder="Search postcode or sector"/>
18 </form>
19 </div>
20 <div class="container mb-5">
21 <table border="1" class="table table-striped table-bordered">
22 <thead>
23 <tr class="bg-info">
24 <th>Name</th>
25 <th>Address</th>
26 <th>Postcode</th>
27 <th>Sector</th>
28 <th>Additional Information</th>
29 </tr>
30 </thead>
31 <!--
32 Below code is connecting to the hosted database.
33 Its taking in the postcode or sector entered by the end user
34 then assigning that to variable result which is then used in the SQL query to query the database
35 assigning all matches to data, which returns all matching companies in a ResultSet
36 -->
37 <tbody>
38 <%
39 String host = "jdbc:mysql://eu-cdbr-west-01.cleardb.com/heroku_298fc475a99a62d";
40 Connection connection = null;
41 Statement statement = null;
42 ResultSet resultSet = null;
43 Class.forName("com.mysql.cj.jdbc.Driver");
44 connection = DriverManager.getConnection(host,"b0a60f8774be0e","13f98876");
45 statement = connection.createStatement();
46 String result = request.getParameter("search");
47 String query;
48 if(result!=null){
49   query = "select * FROM company_details where postcode like '%" + result + "%' or sector like '%" + result + "%'";
50 }else{
51   query = "select * FROM company_details order by companyId desc";
52 }
53 resultSet = statement.executeQuery(query);
54 while(resultSet.next()){
55   %>
56 <tr>
57 <td><%=resultSet.getString("companyName") %></td>
58 <td><%=resultSet.getString("address") %></td>
59 <td><%=resultSet.getString("postcode") %></td>
60 <td><%=resultSet.getString("sector") %></td>
61 <td><%=resultSet.getString("details") %></td>
62 </tr>
63 <%
64 }
65 %>
66 </tbody>
67 </table>
68 </div>
69 </body>
70 </html>

```

(JSP 6 CRUD - Search Data, 2017)

### 2.3.7 Company Form JSP

This file interacts with the ModifyCompanyList Servlet, line 18 and 21 allows us to display which form the end user wants. We know if the company form is not null, the user is looking to update company details.

Using the actions on line 18 and 21, the switch statement in the servlet mentioned will execute the corresponding method.

We are also calling the live data from the hosted database. As when an edit happens we display the data already stored on the database.

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <meta charset="ISO-8859-1">
8 <title>Register Company</title>
9 <link href="https://unpkg.com/bootstrap@4.5.0/dist/css/bootstrap.min.css" rel="stylesheet" type="text/css">
10 </head>
11 <%@ include file="adminHeader.jsp"%>
12 <body>
13   <div class="container col-md-5">
14     <div class="card">
15       <div class="card-body">
16
17         <c:if test="${company != null}">
18           <form action="update" method="post">
19             </c:if>
20         <c:if test="${company == null}">
21           <form action="insert" method="post">
22             </c:if>
23
24         <caption>
25           <h2>
26             <c:if test="${company != null}">
27               Edit New Company
28             </c:if>
29             <c:if test="${company == null}">
30               Add New Company
31             </c:if>
32           </h2>
33         </caption>
34
35         <c:if test="${company != null}">
36           <input type="hidden" name="companyId" value="<c:out value='${company.companyId}' />" /> />
37         </c:if>
38
39         <fieldset class="form-group">
40           <label>Company Name</label> <input type="text"
41             value="<c:out value='${company.companyName}' />" class="form-control"
42             name="companyName" required>
43         </fieldset>
44
45         <fieldset class="form-group">
46           <label>Address</label> <input type="text"
47             value="<c:out value='${company.address}' />" class="form-control"
48             name="address" required>
49         </fieldset>
```

```

50
51  <fieldset class="form-group">
52    <label>Postcode</label> <input type="text"
53      value="<c:out value='${company.postcode}' />" class="form-control"
54      name="postcode" required>
55  </fieldset>
56
57  <fieldset class="form-group">
58    <label>Sector</label> <input type="text"
59      value="<c:out value='${company.sector}' />" class="form-control"
60      name="sector" required>
61  </fieldset>
62
63  <fieldset class="form-group">
64    <label>Details</label> <input type="text"
65      value="<c:out value='${company.details}' />" class="form-control"
66      name="details" required>
67  </fieldset>
68
69  <button type="submit" class="btn btn-success">Save</button>
70 </form>
71 </div>
72 </div>
73 </div>
74 </body>
75 <%@ include file="footer.jsp"%>
76 </html>

```

### 2.3.8 Accounts Register

Account register JSP file provides the connection with account servlet class. Using “/register” path we know this page should be displayed.

All fields are set to required, once save is clicked the account details will be added to the hosted database.

```

1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <meta charset="ISO-8859-1">
8 <title>Register Account</title>
9 <link href="css/bootstrap.css" rel="stylesheet" type="text/css">
10 <script defer src="password.js"></script>
11 </head>
12 <%@ include file="adminHeader.jsp"%>
13 <body>
14 <div>
15 <h1></h1>
16 <div id="error"></div>
17 <form id="form" action="<%=request.getContextPath() %>/register" method="post">
18 <div class="container col-md-5">
19 <div class="card">
20 <div class="card-body">
21 <fieldset class="form-group">
22 <label>First Name</label> <input type="text"
23   value="<c:out value='${account.firstName}' />" class="form-control"
24   name="firstName" required>
25 </fieldset>
26 <fieldset class="form-group">
27 <label>Last Name</label> <input type="text"
28   value="<c:out value='${account.lastName}' />" class="form-control"
29   name="lastName" required>
30 </fieldset>
31 <fieldset class="form-group">
32 <label>Username</label> <input type="text"
33   value="<c:out value='${account.username}' />" class="form-control"
34   name="username" required>
35 </fieldset>
36 <fieldset class="form-group">
37 <label>Password</label> <input id="password" type="text"
38   value="<c:out value='${account.password}' />" class="form-control"
39   name="password">
40 </fieldset>
41 <fieldset class="form-group">
42 <label>Phone</label> <input type="text"
43   value="<c:out value='${account.phone}' />" class="form-control"
44   name="phone" required>
45 </fieldset>
46 <fieldset class="form-group">
47 <label>Email</label> <input type="text"
48   value="<c:out value='${account.email}' />" class="form-control"
49   name="email" required>
50 </fieldset>
51 <button type="submit" class="btn btn-success">Save</button>
52 </div>
53 </div>
54 </div>
55 </form>
56 </div>
57 </body>
58 <%@ include file="footer.jsp"%>
59 </html>

```

### 2.3.9 Web XML

We set the welcome file to the companySearch.jsp file, this is how the home page is set up for end users to see the list and search if needed. Within the file we list our servlet name and URL patterns.

```

web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
3   <display-name>CompanyUnite</display-name>
4   <welcome-file-list>
5     <welcome-file>companySearch.jsp</welcome-file>
6   </welcome-file-list>
7
8   <servlet>
9     <description>
10    </description>
11    <display-name>AccountServlet</display-name>
12    <servlet-name>AccountServlet</servlet-name>
13    <servlet-class>org.project.companies.controller.AccountServlet</servlet-class>
14  </servlet>
15
16  <servlet-mapping>
17    <servlet-name>AccountServlet</servlet-name>
18    <url-pattern>/AccountServlet</url-pattern>
19  </servlet-mapping>
20
21  <servlet>
22    <description>
23    </description>
24    <display-name>CompanyListController</display-name>
25    <servlet-name>CompanyListController</servlet-name>
26    <servlet-class>org.project.companies.controller.CompanyListController</servlet-class>
27  </servlet>
28
29  <servlet-mapping>
30    <servlet-name>CompanyListController</servlet-name>
31    <url-pattern>/CompanyListController</url-pattern>
32  </servlet-mapping>
33  <servlet>
34    <description>
35    </description>
36    <display-name>LoginServlet</display-name>
37    <servlet-name>LoginServlet</servlet-name>
38    <servlet-class>org.project.companies.controller.LoginServlet</servlet-class>
39  </servlet>
40  <servlet-mapping>
41    <servlet-name>LoginServlet</servlet-name>
42    <url-pattern>/LoginServlet</url-pattern>
43  </servlet-mapping>
44  <servlet>
45    <description>
46    </description>
47    <display-name>UpdatedCompanyList</display-name>
48    <servlet-name>UpdatedCompanyList</servlet-name>
49    <servlet-class>org.project.companies.controller.UpdatedCompanyList</servlet-class>
50  </servlet>
51  <servlet-mapping>
52    <servlet-name>UpdatedCompanyList</servlet-name>
53    <url-pattern>/UpdatedCompanyList</url-pattern>
54  </servlet-mapping>
55 </web-app>

```

### 2.3.10 Company Model Class

Using the separation of concerns, we created a model package to store all our model classes.

Company model class we use annotations to define the @Entity, @Table, @Id and @Column. Allowing the class to connect and interact with the correct table. These classes are also known as POJO classes Plain Old Java Objects. Variables used to store



data listed in the bullet points below. Cosgrave, N., 2020. RESTful Web Services – Part III (JAX-RS in a nutshell). *Topic 7*, pp.21,22,23,24,25,26.

- `companyId` – This is used to identify business by a unique id.
- `companyName` – Variable is used to store the company name.
- `address` – Variable is used to store the address of the company.
- `postcode` – Here we will store the postcode where users will later be able to search businesses by postcode. Allowing them to see businesses close by
- `sector` – Variable is used to store the sector in which the business specialises in. Later users of the application will be able to query based on sector.
- `details` – Variable is used to store any additional information the business would like others to see.

```
Company.java ☒
10 @Entity(name="company_details")
11 @Table(name="company_details")
12 public class Company {
13     @Id
14     @Column(name="companyId")
15     int companyId;
16     String companyName;
17     String address;
18     String postcode;
19     String sector;
20     String details;
21
22     public Company() {
23     }
24
25
26     public Company(int companyId, String companyName, String address, String postcode, String sector, String details) {
27         super();
28         this.companyId = companyId;
29         this.companyName = companyName;
30         this.address = address;
31         this.postcode = postcode;
32         this.sector = sector;
33         this.details = details;
34     }
35     public Company( String companyName, String address, String postcode, String sector, String details) {
36         super();
37         this.companyName = companyName;
38         this.address = address;
39         this.postcode = postcode;
40         this.sector = sector;
41         this.details = details;
42     }
43
44     public int getCompanyId() {
45         return companyId;
46     }
47
48     public void setCompanyId(int companyId) {
49         this.companyId = companyId;
50     }
51
52     public String getCompanyName() {
53         return companyName;
54     }
55
56     public void setCompanyName(String companyName) {
57         this.companyName = companyName;
58     }
59 }
```

```

59
60 public String getAddress() {
61     return address;
62 }
63
64 public void setAddress(String address) {
65     this.address = address;
66 }
67
68 public String getPostcode() {
69     return postcode;
70 }
71
72 public void setPostcode(String postcode) {
73     this.postcode = postcode;
74 }
75
76 public String getSector() {
77     return sector;
78 }
79
80 public void setSector(String sector) {
81     this.sector = sector;
82 }
83
84 public String getDetails() {
85     return details;
86 }
87
88 public void setDetails(String details) {
89     this.details = details;
90 }
91
92 @Override
93 public String toString() {
94     return "Company [companyId=" + companyId + ", companyName=" + companyName + ", address=" + address + ", postcode=" + postcode
95         + ", sector=" + sector + ", details=" + details + "];"
96 }
97
98
99

```

### 2.3.11 Company Service

As we continue to follow the separation of concerns design, it was decided to create a service package. Here we will store our java service classes which will have the services we would like to use.

Service classes will later be called by our resource classes. This class was implemented to provide the ability to use the system through API requests.

```

7
8 public class CompanyService {
9
10     CompanyDAO DAO = new CompanyDAO();
11     public List<Company> getCompany() {
12         List<Company> list = DAO.getCompany();
13         return list;
14     }
15
16     public void addCompany(Company company) {
17         DAO.addCompany(company);
18
19     }
20
21     public void updateCompany(Company updatedCompany) {
22         DAO.updateCompany(updatedCompany);
23
24     }
25
26     public void deleteCompany(int companyId) {
27         DAO.deleteCompany(companyId);
28
29     }
30
31 }

```

### 2.3.12 Company Resource

Within our resource class we have annotations such as @GET @POST @PUT @DELETE @Path @Consumes @Produces.

@GET @POST @PUT @DELETE are developed to allow us to make HTTP requests

@Path used to specify specific paths needed for the request to work.

@Consumes @Produces developed to specify the type of request we would like to consume or produce. Cosgrave, N., 2020. RESTful Web Services – Part III (JAX-RS in a nutshell). *Topic 7*, pp.21,22,23,24,25,26.

We also create an object of our company service class so we can use the service logic which helps interact with the database.

```

20 @Path("/project/companies")
21 public class CompanyResource {
22     CompanyService service = new CompanyService();
23     AccountService ac = new AccountService();
24
25     @GET
26     @Produces(MediaType.APPLICATION_JSON)
27     public List<Company> getCompany() {
28         List<Company> list = service.getCompany();
29         return list;
30     }
31
32     @POST
33     @Consumes(MediaType.APPLICATION_JSON)
34     public void postCompany(Company company) {
35         service.addCompany(company);
36     }
37
38     @PUT
39     @Path("/{companyId}")
40     @Consumes(MediaType.APPLICATION_JSON)
41     public void putCompany(@PathParam("companyId") int companyId, Company updatedCompany) {
42         updatedCompany.setCompanyId(companyId);
43         service.updateCompany(updatedCompany);
44     }
45
46     @DELETE
47     @Path("/{companyId}")
48     public void deleteCompany(@PathParam("companyId") int companyId) {
49         service.deleteCompany(companyId);
50     }
51
52     @GET
53     @Path("/{companyId}/accounts")
54     @Produces(MediaType.APPLICATION_JSON)
55     public List<Account> getAccountsByCompany(@PathParam("companyId") int companyId) {
56         List<Account> accountlist = ac.getAccountsByCompany(companyId);
57         return accountlist;
58     }
59 }
60 }
61

```

### 2.3.13 Account Model

Account model class implementation is developed to allow us to create objects necessary to interact with the hosted database.

Available in the screenshot shown are annotations such as @Entity, @Table, @Id, @ManyToOne and @JoinColumn. These allow us to pass data to the correct database also making a many to one relationship between two tables. Class also contains variables, constructors, getters & setters.

```

9 @Entity(name="accounts_details")
10 @Table(name="accounts_details")
11 public class Account {
12     @Id
13     int accountId;
14     //hibernate feature
15     @ManyToOne(targetEntity=Company.class)
16     @JoinColumn(name="companyId")
17     Company company;
18
19     String firstName;
20     String lastName;
21     String username;
22     String password;
23     String phone;
24     String email;
25
26     public Account() {
27
28     }
29
30     public Account(String firstName, String lastName, String username, String password,
31                     String phone, String email) {
32         super();
33         this.firstName = firstName;
34         this.lastName = lastName;
35         this.username = username;
36         this.password = password;
37         this.phone = phone;
38         this.email = email;
39     }
40
41     public Account(int accountId, Company company, String firstName, String lastName, String username, String password,
42                     String phone, String email) {
43         super();
44         this.accountId = accountId;
45         this.company = company;
46         this.firstName = firstName;
47         this.lastName = lastName;
48         this.username = username;
49         this.password = password;
50         this.phone = phone;
51         this.email = email;
52     }
53
54     public int getAccountId() {
55         return accountId;
56     }
57

```

```
58 public void setAccountId(int accountId) {
59     this.accountId = accountId;
60 }
61
62 public Company getCompany() {
63     return company;
64 }
65
66 public void setCompany(Company company) {
67     this.company = company;
68 }
69
70 public String getFirstName() {
71     return firstName;
72 }
73
74 public void setFirstName(String firstName) {
75     this.firstName = firstName;
76 }
77
78 public String getLastName() {
79     return lastName;
80 }
81
82 public void setLastName(String lastName) {
83     this.lastName = lastName;
84 }
85
86 public String getUsername() {
87     return username;
88 }
89
90 public void setUsername(String username) {
91     this.username = username;
92 }
93
94 public String getPassword() {
95     return password;
96 }
97
98 public void setPassword(String password) {
99     this.password = password;
100 }
101
102 public String getPhone() {
103     return phone;
104 }
105
```

```
106 public void setPhone(String phone) {
107     this.phone = phone;
108 }
109
110 public String getEmail() {
111     return email;
112 }
113
114 public void setEmail(String email) {
115     this.email = email;
116 }
117
118
119 }
```

### 2.3.15 Database Class

Database connection class used in multiple DAO classes. Establishing a connection with the hosted database by stating the URL, Username and Password.

Code on line 19 & 20 shows how we establish that connection to the database using DriverManager.

Driver Manager provides a basic service for JDBC drivers, using the get method it provides all we need to do is pass in the url, username and password as parameters. If successful and no exceptions are caught, we then return that connection. (MySQL :: MySQL Connector/J 8.0 Developer Guide :: 7.1 Connecting to MySQL Using the JDBC DriverManager Interface, 2021)

Classes that import this connection

- Account Details DAO
- Company List DAO Impl
- Update Company DAO

```

10 public class DBConnection {
11
12     private static final String URL = "jdbc:mysql://eu-cdbr-west-01.cleardb.com/heroku_298fc475a99a62d";
13     private static final String USERNAME = "b0a60f8774be0e";
14     private static final String PASSWORD = "13f98876";
15
16     public static Connection getConnection() {
17         Connection connection = null;
18         try {
19             Class.forName("com.mysql.cj.jdbc.Driver");
20             connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
21         } catch (SQLException e) {
22             // TODO Auto-generated catch block
23             e.printStackTrace();
24         } catch (ClassNotFoundException e) {
25             // TODO Auto-generated catch block
26             e.printStackTrace();
27         }
28         return connection;
29     }
30
31     public Connection getConnection2() {
32         Connection connection = null;
33         try {
34             Class.forName("com.mysql.cj.jdbc.Driver");
35             connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
36         } catch (SQLException e) {
37             // TODO Auto-generated catch block
38             e.printStackTrace();
39         } catch (ClassNotFoundException e) {
40             // TODO Auto-generated catch block
41             e.printStackTrace();
42         }
43         return connection;
44     }
45 }

```

### 2.3.16 Update Company DAO

This Data Access Object Class contains multiple methods which interact with the hosted database.

List of interactions with the database

- Inserting company information
- Selecting a company by Id
- Selecting all companies in the database
- Deleting a company by identifying it using its unique id

At the top of the class, we declared the constants for each query, writing the SQL while adding in the placeholders “?” for the values that will be entered later through the application.

All methods must access the database before interacting with it. Described previously in section 2.3.15 the implementation of the database class is used by multiple classes.

We created an object of that class on line 24. Then in each method within the try we use that object to use method getConnection2() providing the connection needed.



The Insert company method uses the company object as a parameter, using the get methods through object "company".

We're able to assign the data to each field name listed in the SQL query then running the execute update. Allowing the insertion of the new data to the hosted database.

The remaining methods all take a very similar approach.

Delete method uses the Id of the company to identify the company details to be removed.

```
UpdateCompanyDAO.java
1 package org.project.companies.DAO;
2
3 import java.sql.Connection;
12
13 public class UpdateCompanyDAO {
14
15     private static final String INSERT_COMPANIES_SQL = "INSERT INTO heroku_298fc475a99a62d.company_details (companyName, address, postcode, sector, details) VALUES (?, ?, ?, ?, ?)";
16     private static final String SELECT_COMPANY_BY_ID = "select * from company_details where companyId = ?";
17     private static final String SELECT_ALL_COMPANIES = "select * from company_details";
18     private static final String DELETE_COMPANY_SQL = "delete from company_details where companyId = ?";
19     private static final String UPDATE_COMPANY_SQL = "update company_details set companyName = ?, address = ?, postcode = ?, sector = ?, details = ? where companyId = ?";
20
21     public UpdateCompanyDAO() {
22     }
23
24     DBConnection newDBConnection = new DBConnection();
25
26
27     //Inserting company information to the database
28     public void insertCompany(Company company) throws SQLException {
29         System.out.println(INSERT_COMPANIES_SQL);
30         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(INSERT_COMPANIES_SQL)){
31             preparedStatement.setString(1, company.getCompanyName());
32             preparedStatement.setString(2, company.getAddress());
33             preparedStatement.setString(3, company.getPostcode());
34             preparedStatement.setString(4, company.getSector());
35             preparedStatement.setString(5, company.getDetails());
36             System.out.println(preparedStatement);
37             preparedStatement.executeUpdate();
38         } catch (Exception e) {
39             e.printStackTrace();
40         }
41     }
42
43     //update existing company data to the database
44     public boolean updateCompany(Company company) throws SQLException {
45         boolean companyUpdated;
46         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(UPDATE_COMPANY_SQL)){
47             System.out.println(preparedStatement);
48             preparedStatement.setString(1, company.getCompanyName());
49             preparedStatement.setString(2, company.getAddress());
50             preparedStatement.setString(3, company.getPostcode());
51             preparedStatement.setString(4, company.getSector());
52             preparedStatement.setString(5, company.getDetails());
53             preparedStatement.setInt(6, company.getCompanyId());
54             companyUpdated = preparedStatement.executeUpdate() > 0;
55         }
56         return companyUpdated;
57     }
58
59     //Select company by id
60     public Company selectCompany(int companyId){
61         Company company = null;
62         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(SELECT_COMPANY_BY_ID)){
63             preparedStatement.setInt(1, companyId);
64             System.out.println(preparedStatement);
65
66             ResultSet rs = preparedStatement.executeQuery();
67
68             while (rs.next()) {
69                 String companyName = rs.getString("companyName");
70                 String address = rs.getString("address");
71                 String postcode = rs.getString("postcode");
72                 String sector = rs.getString("sector");
73                 String details = rs.getString("details");
74                 company = new Company(companyId, companyName, address, postcode, sector, details );
75             }
76         } catch (SQLException e) {
77             e.printStackTrace();
78         }
79         return company;
80     }
81
82
83     //select companies
84     public List<Company> selectAllCompanies(){
85         List<Company> companies = new ArrayList<>();
86         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_COMPANIES)){
87             System.out.println(preparedStatement);
88
89             ResultSet rs = preparedStatement.executeQuery();
90
91             while (rs.next()) {
92                 int companyId = rs.getInt("companyId");
93                 String companyName = rs.getString("companyName");
94                 String address = rs.getString("address");
95                 String postcode = rs.getString("postcode");
96                 String sector = rs.getString("sector");
97                 String details = rs.getString("details");
98                 companies.add(new Company(companyId, companyName, address, postcode, sector, details));
99             }
100         } catch (SQLException e) {
101             e.printStackTrace();
102         }
103         return companies;
104     }
105 }
```

```

107 //delete company
108 public boolean deleteCompany(int companyId) throws SQLException {
109     boolean rowDeleted;
110     try (Connection connection = newDBConnection.getConnection2(); PreparedStatement statement = connection.prepareStatement(DELETE_COMPANY_SQL);) {
111         statement.setInt(1, companyId);
112         rowDeleted = statement.executeUpdate() > 0;
113     }
114     return rowDeleted;
115 }
116 }

```

### 2.3.17 Account Details DAO

Account details DAO class provides the functionality of connecting to the hosted database.

Class contains the registerAccountDetails method which is the logic for the registering of administrator accounts.

Validate method provides the logic for when a user tries to log into the application. It checks the database to ensure username and passwords are valid and are already stored on the database.

```

11 public class AccountDetailsDAO {
12
13     //String constant containing an SQL query for inserting administrators details once a connection is established
14     public static final String INSERT_ACCOUNT_SQL = "INSERT INTO heroku_298fc475a99a62d.account_details "
15         + "(companyId, first_name, last_name, username, password, phone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
16
17     public static final String SELECT_USERNAME_PASSWORD = "select * from account_details where username = ? and password = ? ";
18
19     public AccountDetailsDAO() {
20     }
21
22     //Object used to connect to the hosted database
23     DBConnection newDBConnection = new DBConnection();
24
25
26     //Method for registering admin accounts first we need to establish a connection by using a method available by using newDBConnection object
27     //we then assign our sql query to the precompiled object, allowing us to set the values by using the get methods from Account class
28     //then execute the update allowing us to write to the database
29     public void registerAccountDetails(Account account) throws SQLException{
30         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(INSERT_ACCOUNT_SQL)){
31             preparedStatement.setInt(1, 5);
32             preparedStatement.setString(2, account.getFirstName());
33             preparedStatement.setString(3, account.getLastName());
34             preparedStatement.setString(4, account.getUsername());
35             preparedStatement.setString(5, account.getPassword());
36             preparedStatement.setString(6, account.getPhone());
37             preparedStatement.setString(7, account.getEmail());
38             System.out.println(preparedStatement);
39
40             preparedStatement.executeUpdate();
41         } catch (SQLException e) {
42             e.printStackTrace();
43         }
44     }
45
46     //validate method allows us to check if the username and password are stored on the database
47     public boolean validate(Account account) throws ClassNotFoundException {
48         boolean status = false;
49
50         try(Connection connection = newDBConnection.getConnection2(); PreparedStatement preparedStatement = connection.prepareStatement(SELECT_USERNAME_PASSWORD)){
51             preparedStatement.setString(1, account.getUsername());
52             preparedStatement.setString(2, account.getPassword());
53
54             System.out.println(preparedStatement);
55             ResultSet rs = preparedStatement.executeQuery();
56             status = rs.next();
57         } catch (SQLException e) {
58             printSQLException(e);
59         }
60         return status;
61     }

```

### 2.3.18 Hibernate.cfg.xml

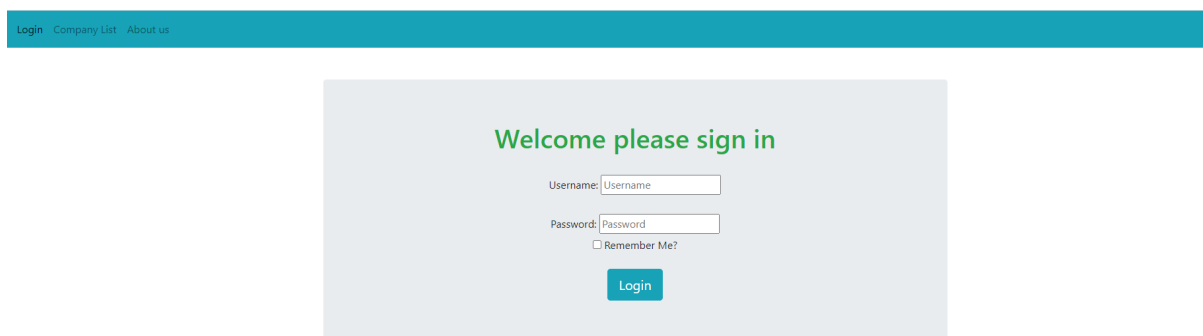
Provides the connection to our ClearDB MySQL for API functionality. We import the connection details within this file to our CompanyDAO class. Allowing us to add, update, retrieve and delete data on the host database through API requests.

```
hibernate.cfg.xml
1 <!DOCTYPE hibernate-configuration PUBLIC
2   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
3   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4
5 <hibernate-configuration>
6   <session-factory>
7     <!-- Connection settings -->
8     <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
9     <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
10    <!-- Sample MySQL URL provided -->
11    <property name="connection.url">jdbc:mysql://eu-cdbr-west-01.cleardb.com/heroku_298fc475a99a62d</property>
12    <property name="connection.username">b0a60f8774be0e</property>
13    <property name="connection.password">13f98876</property>
14
15    <!-- Show SQL on console -->
16    <property name="show_sql">>true</property>
17
18    <!--Setting Session context model -->
19    <property name="current_session_context_class">thread</property>
20
21  </session-factory>
22 </hibernate-configuration>
```

## 2.4. Graphical User Interface (GUI)

### 2.4.1 Login Page

Administrators can only log into the system if their details are stored in the database, gaining access to CRUD functionalities.



### 2.4.2 Register Company

Given administrators log in successfully, they are brought to the page below. Allowing them to add, edit or delete companies from the platform. They have the ability to navigate to Registration to add new administrators to the database.

## Register Company

[Add New Company](#)

ID	Name	Address	Postcode	Sector	Additional Information	Modify/Delete
1	DeliveryService2	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
5	MobileCatering	Dublin city centre	d4	Catering	Mobile catering service, looking to provide catering service on business grounds. Please contact MobileCatering@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
6	CompanyB	Crumlin Dublin	D2	Delivery	Looking to provide delivery service for companies in the south Dublin City Centre area. Please contact CompanyB@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
8	Tom's Wings Food Truck	Dublin City Centre	d2	Hospitality	Food Truck looking for open space, in Dublin City Centre. Please contact Tomwings@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
9	Cafe Truck	glasnevin	D11	Hospitality	Need to rent space, in a busy area. Please contact CafeTruck@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
10	GASTROFAKE	London	SW100AD	Hospitality	Need to link up with a delivery service to deliver our products to our customers. Please contact GASTROFAKE@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
11	DeliverServicefake	London	SW110AD	Delivery	Looking to provide a delivery service in the London area. Please contact DeliverServiceFake@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
12	Best Cafe On Wheels	Howth	D13	Hospitality	Need space with high footfall to rent. Please contact cafewheels@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
13	Greek Food Truck	Grafton Street	D2	Hospitality	Looking to rent space in the city centre. Please contact GrEEKTrUck@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>

### 2.4.3 Add New Company

Clicking add new company shown in the previous screenshot, the administrator will be presented with this form. Filling all form fields the administrator will successfully insert the company data to the hosted database.

Register Company Registration Logout

### Add New Company

Company Name

Address

Postcode

Sector

Details

[Save](#)

### 2.4.4 Edit New Company

If the administrator is tasked to update some business details, they are presented with this page. Please note previous details are displayed in the text boxes. Administrators have the

privilege to update any of these fields. Once save is clicked the application will show new updates immediately.

### Edit New Company

Company Name

Address

Postcode

Sector

Details

#### 2.4.4 Register New Administrator Account

Administrators can be added to the system through the application. Once details have been entered to all available fields and save is clicked then the new account details will be stored in the hosted database. Allowing the new administrator to access the system.

### Register new administrator account

First Name

Last Name

Username

Password

Phone

Email

## 2.4.5 Search for Businesses

This page is available to the public, businesses or people can visit the application without the need to login and search for businesses in an area that suits them or by a sector entering in the search box.

Login Company List About us

Delivery

Name	Address	Postcode	Sector	Additional Information
EasyDrinks	Main Street	D1	Hospitality	Looking to team with a delivery service who can manage a high number of deliveries throughout the evening
Bakery Truck	Phisboro	D1	Hospitality	Need to rent space from 8am - 2pm
Greek Food Truck	Grafton Street	D2	Hospitality	Looking to rent space in the city centre
Best Cafe On Wheels	Howth	D13	Hospitality	Need space with high footfall to rent
DeliverServicefake	London	SW110AD	Delivery	Looking to provide a delivery service in the London area
GASTROFAKE	London	SW100AD	Hospitality	Need to link up with a delivery service to deliver our products to our customers
Cafe Truck	glasnevin	D11	Hospitality	Need to rent space, in a busy area
Tom's Wings Food Truck	Dublin City Centre	d2	Hospitality	Food Truck looking for open space, in Dublin City Centre
Company8	Crumlin Dublin	D2	Delivery	Looking to provide delivery service for companies
MobileCatering	Dublin city centre	d4	Catering	Mobile catering service, looking to provide catering service on business grounds
DeliveryService2	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area

## 2.4.6 About Page

About us page provides a description of the goal CompaniesUnite have set out. Also explaining what happens when a customer sends data through the form if they wish to display their information on the application.

### Who we are

We're CompaniesUnite a platform to help businesses and entrepreneurs to work together, adapting to the restrictions introduced by the pandemic.

### What matters to us

What matters to us is that businesses and people looking to venture new areas or looking to adapt under these circumstances are kept in business.

We want to help you adapt and create relationships outside of your organisation providing better chances of survival.

### What we do

We provide a platform that allows anyone interested on the internet to browse the list of businesses interested or looking to help others.

We display the details YOU send to us, as we have administrators working around the clock to get your information out there.

We update customers data if they wish to make enhancements to the initial data they provided.

We also remove the data immediately once requested by that business.

### How we do it

Once the form below is received, one of our administrators will pick up the task and write the corresponding information to the system.

Business information will be displayed immediately on the system once the administrators enters it into the database.

Company name  Address  Postcode

Company Name  Address  Postcode

Additional Information  Sector

Additional Information  Sector

## 2.5. Testing

Testing is a very important phase when developing an application. Ensuring there are no major issues blocking customers using the application provide great comfortability. We executed on three different types of testing, unit, integration, and end user testing.

### Unit Tests

Unit tests were developed using Eclipse IDE, creating a new unit test case for each of the model classes was simple enough.

In the project explorer you can find JUnit Test Case option. Providing us with the dialog box to add test packages and test classes, allowing us to link test classes with the java classes keeping naming convention in sync.

As the model java classes are used a lot for creating objects while inserting, manipulating, retrieving, and deleting data from our hosted database. It made sense to get unit test coverage on company and account model classes.

### Integration Tests

Postman is a tool widely used across the industry; many companies use the tool for integration testing.

Showing through API requests the interactions between java classes model (POJO), Service, Resource, DAO, and database connections.

We can confirm through all this interaction, data is returning as expected along with adding, updating, and deleting through all the classes mentioned.

Sending these requests and receiving their responses we can confirm the integration tests are being covered.

### End User Tests

End user test proved to be very important on our initial run we seen one or two features weren't working as expected. By executing the tests over and over after each update we ensured that all tests are now passing. The goal here is to mimic the customers behaviour to catch any issues they might come across.

#### 2.5.1 Unit Tests

Unit tests below confirm the code for our accounts class is working as expected. This class is important for when the administrators want to create new admin accounts. We are testing the getter methods from the accounts class.

Comparing the expected result with the actual result, by doing an `assertEquals()` we can determine if the test is a pass or fail.

### Account Unit Test Class

```
AccountTest.java
25
26 @Test
27 public void testGetFirstName() {
28     String expectFirstName = "Rachel";
29     String actualFirstName = account.getFirstName();
30     assertEquals(expectFirstName,actualFirstName);
31 }
32
33 @Test
34 public void testGetLastName() {
35     String expectLastName = "Johnson";
36     String actualLastName = account.getLastName();
37     assertEquals(expectLastName,actualLastName);
38 }
39
40 @Test
41 public void testGetUsername() {
42     String expectUsername = "RachJohnson567";
43     String actualUsername = account.getUsername();
44     assertEquals(expectUsername,actualUsername);
45 }
46
47 @Test
48 public void testGetPassword() {
49     String expectPassword = "hhfotyf369";
50     String actualPassword = account.getPassword();
51     assertEquals(expectPassword,actualPassword);
52 }
53
54 @Test
55 public void testGetPhone() {
56     String expectPhone = "08924372";
57     String actualPhone = account.getPhone();
58     assertEquals(expectPhone,actualPhone);
59 }
60
```



```

60
61 @Test
62 public void testGetEmail() {
63     String expectEmail = "RachJohnson@fakegmail.com";
64     String actualEmail = account.getEmail();
65     assertEquals(expectEmail,actualEmail);
66 }
67
68 }

```

### Account Unit Test Results

Results shown in the screenshot below, all tests passed as expected.

The screenshot displays the Eclipse IDE interface during a JUnit test run. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons. The Package Explorer on the left shows the project structure, with the test results for 'org.project.companies.model.AccountTest' expanded. The test results show 6/6 runs, 0 errors, and 0 failures. The test results list includes testGetPassword (0.001 s), testGetEmail (0.000 s), testGetPhone (0.000 s), testGetLastName (0.000 s), testGetUsername (0.000 s), and testGetFirstName (0.000 s). The code editor on the right shows the testGetEmail method in AccountTest.java, which is highlighted in blue. The code in the editor is as follows:

```

42     String
43     String
44     assertI
45 }
46
47 @Test
48 public voi
49     String
50     String
51     assertI
52 }
53
54 @Test
55 public voi
56     String
57     String
58     assertI
59 }
60
61 @Test
62 public voi
63     String

```

(JUnit - Plug with Eclipse - Tutorialspoint, 2021)

## Company Unit Test Class

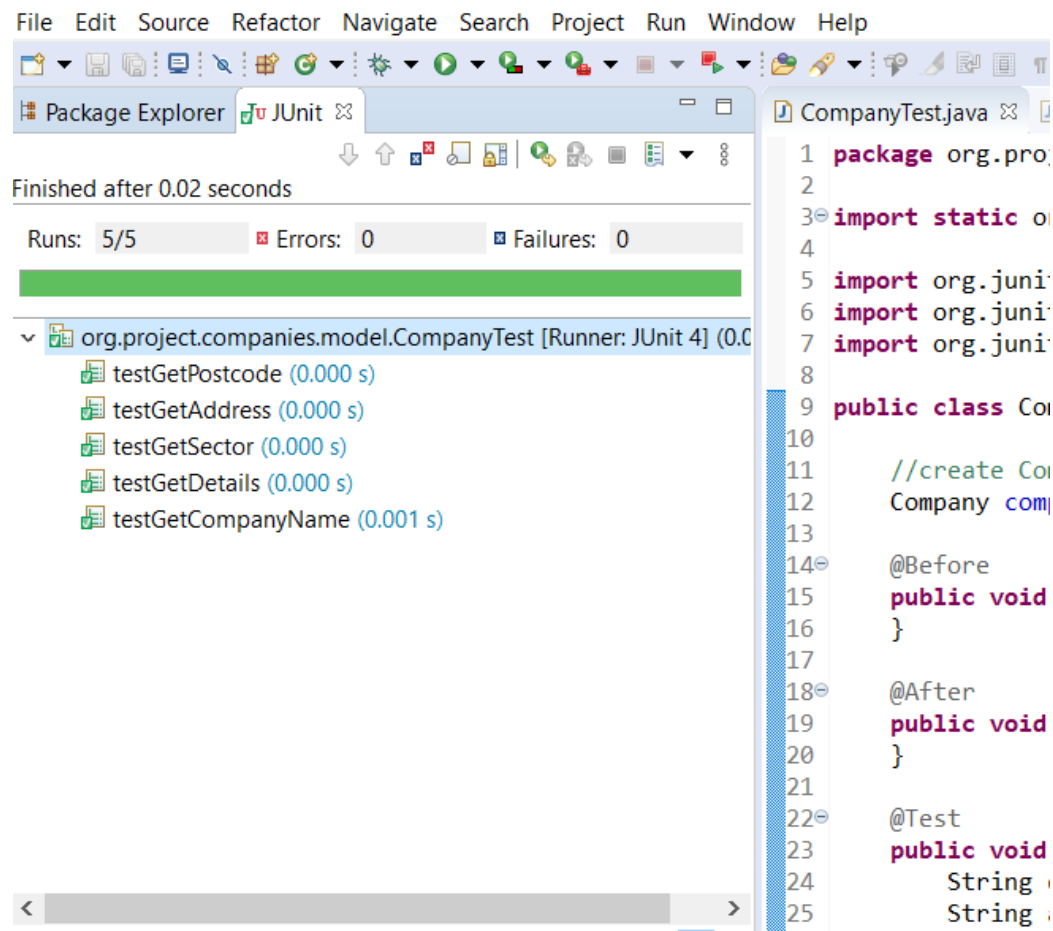
Unit tests below confirm the code for our company class is working as expected. This class is important for when the administrators want to add new or update existing companies to the database. We're testing the getter methods from the company class.

Comparing the expected result with the actual result, by doing an `assertEquals()` we can determine if the test is a pass or fail.

```
CompanyTest.java
22 @Test
23 public void testGetCompanyName() {
24     String expectCompanName = "MobileCafe";
25     String actualCompanyName = company.getCompanyName();
26     assertEquals(expectCompanName, actualCompanyName);
27 }
28
29 @Test
30 public void testGetAddress() {
31     String expectAddress = "23 oldtown road, whitehall";
32     String actualAddress = company.getAddress();
33     assertTrue(expectAddress.contentEquals(actualAddress));
34 }
35
36 @Test
37 public void testGetPostcode() {
38     String expectPostcode = "D11";
39     String actualPostcode = company.getPostcode();
40     assertEquals(expectPostcode, actualPostcode);
41 }
42
43 @Test
44 public void testGetSector() {
45     String expectSector = "Hospitality";
46     String actualSector = company.getSector();
47     assertEquals(expectSector, actualSector);
48 }
49
50 @Test
51 public void testGetDetails() {
52     String expectDetails = "Looking for open space to serve the public";
53     String actualDetails = company.getDetails();
54     assertEquals(expectDetails, actualDetails);
55 }
56 }
57
```

## Company Unit Test Results

Results shown in the screenshot below, all tests passed as expected.



```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit CompanyTest.java
Finished after 0.02 seconds
Runs: 5/5 Errors: 0 Failures: 0
org.project.companies.model.CompanyTest [Runner: JUnit 4] (0.0
  testGetPostcode (0.000 s)
  testGetAddress (0.000 s)
  testGetSector (0.000 s)
  testGetDetails (0.000 s)
  testGetCompanyName (0.001 s)
1 package org.pro
2
3 import static o
4
5 import org.juni
6 import org.juni
7 import org.juni
8
9 public class Co
10
11 //create Co
12 Company com
13
14 @Before
15 public void
16 }
17
18 @After
19 public void
20 }
21
22 @Test
23 public void
24     String
25     String
```

### 2.5.2 Integration

Integration testing were executed using a tool called Postman. Confirming the java classes and business logic are working as expected while integrating with the database.

Ensuring our company model class along with other java classes such as our data access object classes, which are used to create SQL queries and send the prepared statement to the database when a connection is secured covering the full integration.

We covered the major integration tests using Postman.

Integration tests executed are:

- Retrieve all Companies
- Add New Company
- Update Existing Company
- Delete Company by ID

### 2.5.2.1 Get all Companies

Get all Companies test covers the fact the user can retrieve data from the hosted database.

URL	Method	Header	Value
http://localhost:8080/CompanyUnite/JAXRS/project/companies	GET	Content-Type	application/json

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:8080/CompanyUnite/JAXRS/project/companies
- Method:** GET
- Content-Type:** application/json
- Status:** 200
- Time:** 5312 ms
- Body:** A JSON array of 11 company objects. The first few objects are:
 

```

      [
        {
          "address": "santry",
          "companyID": 1,
          "companyName": "DeliveryService3",
          "details": "Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com",
          "postcode": "d11",
          "sector": "Delivery"
        },
        {
          "address": "Dublin city centre",
          "companyID": 5,
          "companyName": "MobileCatering",
          "details": "Mobile catering service, looking to provide catering service on business grounds. Please contact MobileCatering@fakemail.com",
          "postcode": "d4",
          "sector": "Catering"
        },
        {
          "address": "Crumlin Dublin",
          "companyID": 6,
          "companyName": "CompanyB",
          "details": "Looking to provide delivery service for companies in the south Dublin City Centre area. Please contact CompanyB@fakemail.com",
          "postcode": "D2",
          "sector": "Delivery"
        },
        {
          "address": "Dublin City Centre",
          "companyID": 8,
          "companyName": "Tom's Wings Food Truck",
          "details": "Food Truck looking for open space, in Dublin City Centre. Please contact Tomwings@fakemail.com",
          "postcode": "d2",
          "sector": "Hospitality"
        },
        {
          "address": "glasnevin",
          "companyID": 9,
          "companyName": "Cafe Truck",
          "details": "Need to rent space, in a busy area. Please contact CafeTruck@fakemail.com",
          "postcode": "D11",
          "sector": "Hospitality"
        },
        {
          "address": "London",
          "companyID": 10,
          "companyName": "GASTROFAKE",
          "details": "Need to link up with a delivery service to deliver our products to our customers. Please contact GASTROFAKE@fakemail.com",
          "postcode": "SW100AD",
          "sector": "Hospitality"
        },
        {
          "address": "London",
          "companyID": 11,
          "companyName": "DeliverServiceFake",
          "details": "Looking to provide a delivery service in the London area. Please contact DeliverServiceFake@fakemail.com",
          "postcode": "SW110AD",
          "sector": "Delivery"
        }
      ]
      
```

```

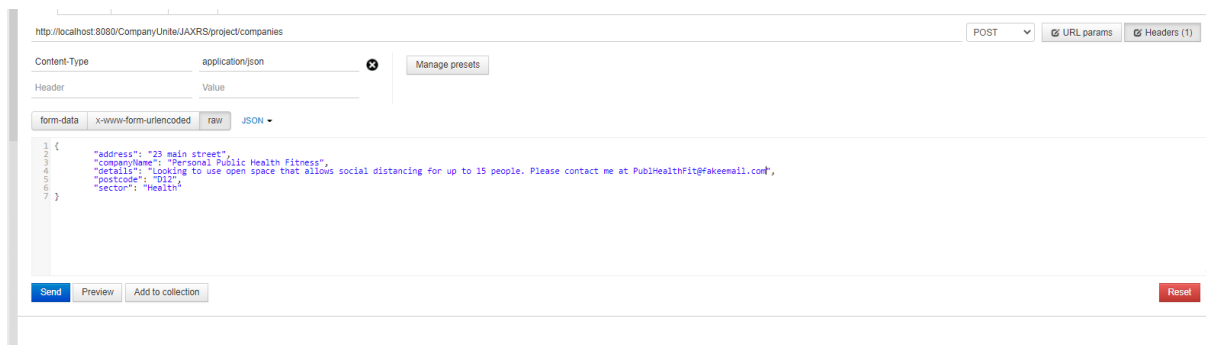
58 {
59   "address": "Houth",
60   "companyId": 12,
61   "companyName": "Best Cafe On Wheels",
62   "details": "Need space with high footfall to rent. Please contact cafewheels@fakemail.com",
63   "postcode": "D13",
64   "sector": "Hospitality"
65 }
66 }
67 {
68   "address": "Grafton Street",
69   "companyId": 13,
70   "companyName": "Greek Food Truck",
71   "details": "Looking to rent space in the city centre. Please contact GrEEKTrUck@fakemail.com",
72   "postcode": "D2",
73   "sector": "Hospitality"
74 }
75 }
76 {
77   "address": "Phisboro",
78   "companyId": 14,
79   "companyName": "Bakery Truck",
80   "details": "Need to rent space from 8am - 2pm. Please contact bakery@fakemail.com",
81   "postcode": "D1",
82   "sector": "Hospitality"
83 }
84 }
85 {
86   "address": "Main Street",
87   "companyId": 15,
88   "companyName": "EasyDrinks",
89   "details": "Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakemail.com",
90   "postcode": "D1",
91   "sector": "Hospitality"
92 }
93 }
94 {
95   "address": "testing ",
96   "companyId": 55,
97   "companyName": "Test Company ",
98   "details": "testing",
99   "postcode": "testing",
100  "sector": "testing"
101 }
102 ]

```

### 2.5.2.1 Add New Company

Add New Company integration test covers the overall integration of java classes and the database while inserting new company data. Using details below, along with a json request body shown in the screenshot, we confirmed company data is added to the system.

URL	Method	Header	Value
http://localhost:8080/CompanyUnite/JAXRS/project/companies	POST	Content-Type	application/json



```

76 {
77   "companyId": 14,
78   "companyName": "Bakery Truck",
79   "details": "Need to rent space from 8am - 2pm. Please contact bakery@fakemail.com",
80   "postcode": "D1",
81   "sector": "Hospitality"
82 }
83 }
84 {
85   "address": "Main Street",
86   "companyId": 15,
87   "companyName": "EasyDrinks",
88   "details": "Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakemail.com",
89   "postcode": "D1",
90   "sector": "Hospitality"
91 }
92 }
93 {
94   "address": "testing ",
95   "companyId": 55,
96   "companyName": "Test Company ",
97   "details": "testing",
98   "postcode": "testing",
99   "sector": "testing"
100 }
101 }
102 {
103   "address": "23 main street",
104   "companyId": 85,
105   "companyName": "Personal Public Health Fitness",
106   "details": "Looking to use open space that allows social distancing for up to 15 people. Please contact me at PubHealthFit@fakeemail.com",
107   "postcode": "D12",
108   "sector": "Health"
109 }
110 ]

```

### 2.5.2.1 Update Existing Company

Update Existing Company integration test covers the overall integration of java classes and the database while updating existing company data. Using details below, along with a json request body shown in the screenshot, we confirmed company data is updated on the system.

URL	Method	Header	Value
http://localhost:8080/CompanyUnite/JAXRS/project/companies/{companyId}	PUT	Content-Type	application/json

```

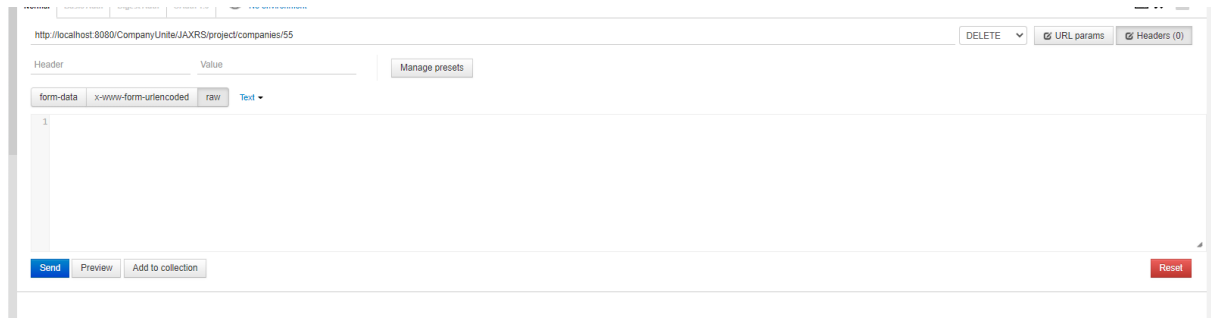
73     },
74   },
75   "address": "Phisboro",
76   "companyId": 14,
77   "companyName": "Bakery Truck",
78   "details": "Need to rent space from 8am - 2pm. Please contact bakery@fakeemail.com",
79   "postcode": "D1",
80   "sector": "Hospitality"
81 }
82 }
83 "address": "Main Street",
84 "companyId": 15,
85 "companyName": "EasyDrinks",
86 "details": "Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakeemail.com",
87 "postcode": "D1",
88 "sector": "Hospitality"
89 }
90 }
91 "address": "78 church street",
92 "companyId": 55,
93 "companyName": "Mobile Hairdressers",
94 "details": "Need to rent space outside to cut hair. Please contact me at MobileHairdressing@fakeemail.com",
95 "postcode": "D5",
96 "sector": "Hairdressing"
97 }
98 }
99 "address": "23 main street",
100 "companyId": 85,
101 "companyName": "Personal Public Health Fitness",
102 "details": "Looking to use open space that allows social distancing for up to 15 people. Please contact me at PubHealthFit@fakeemail.com",
103 "postcode": "D12",
104 "sector": "Health"
105 }
106 ]

```

### 2.5.2.3 Delete Company by ID

Delete Company by ID integration test covers the overall integration of java classes and the database while allowing for customer data to be removed when requested. As seen no request body is required for this test, just the details provided below.

URL	Method	Header	Value
http://localhost:8080/CompanyUnite/JAXRS/project/companies/{companyId}	DELETE	Content-Type	application/json



```

66  {
67    "address": "Grafton Street",
68    "companyId": 13,
69    "companyName": "Greek Food Truck",
70    "details": "Looking to rent space in the city centre. Please contact GrEEKTruck@fakeemail.com",
71    "postcode": "D2",
72    "sector": "Hospitality"
73  },
74  {
75    "address": "Phisboro",
76    "companyId": 14,
77    "companyName": "Bakery Truck",
78    "details": "Need to rent space from 8am - 2pm. Please contact bakery@fakeemail.com",
79    "postcode": "D1",
80    "sector": "Hospitality"
81  },
82  {
83    "address": "Main Street",
84    "companyId": 15,
85    "companyName": "EasyDrinks",
86    "details": "Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakeemail.com",
87    "postcode": "D1",
88    "sector": "Hospitality"
89  },
90  {
91    "address": "23 main street",
92    "companyId": 85,
93    "companyName": "Personal Public Health Fitness",
94    "details": "Looking to use open space that allows social distancing for up to 15 people. Please contact me at PublHealthFit@fakeemail.com",
95    "postcode": "D12",
96    "sector": "Health"
97  }
98 ]

```

### 2.5.3 End User Testing

As mentioned in the brief description, end user is extremely important stage to sign off. As it shows the application is working as expected.

It provides us with what the customer experiences, not only is it an important stage, but it's also very important we understand how the customer uses the application.

If we don't know how the end user will interact with the application, our test cases are not valid. After reviewing how 5 different people use the application. The test cases reflect how the end user will interact with the application.

Test Case ID	Test Case Objective	Steps	Expected Result	Actual Result	Status
TC_01	Login Successfully	<ol style="list-style-type: none"> <li>Go to url: <a href="https://testprojectunite.herokuapp.com/">https://testprojectunite.herokuapp.com/</a></li> <li>Add username - johnsmith</li> <li>Add password - johnsmith123</li> <li>Click login</li> </ol>	Login Successfully	Logged in Successfully	Pass
TC_02	All fields Required for Registration	<ol style="list-style-type: none"> <li>Go to url: <a href="https://testprojectunite.herokuapp.com/register">https://testprojectunite.herokuapp.com/register</a></li> <li>Leave each field blank at a given time <ul style="list-style-type: none"> <li>First Name</li> <li>Last Name</li> <li>Username</li> <li>Password</li> <li>Phone</li> </ul> </li> </ol>	Fields display warning if left blank	Top blank field displays warning	Pass

		<ul style="list-style-type: none"> <li>Email</li> </ul> 8. Click Save			
TC_03	All fields Required for entering company details	1. Go to url: <a href="https://testprojectunite.herokuapp.com/new">https://testprojectunite.herokuapp.com/new</a> 2. Leave each field blank at a given time <ul style="list-style-type: none"> <li>Company Name</li> <li>Address</li> <li>Postcode</li> <li>Sector</li> <li>Details</li> </ul> 7. Click Save	Fields display warning if left blank	Top blank field displays warning	Pass
TC_04	Edit Company	1. Go to url: <a href="https://testprojectunite.herokuapp.com/edit?companyId=13">https://testprojectunite.herokuapp.com/edit?companyId=13</a> 2. Edit Company Name 3. Edit Address 4. Edit Postcode 5. Edit Sector 6. Edit Details 7. Click Save	Company details display updated information when saved	Company details displayed the new edited information	Pass
TC_05	Delete Company	1. Go to url: <a href="https://testprojectunite.herokuapp.com/list">https://testprojectunite.herokuapp.com/list</a> 2. Go to the last Company on the list 3. Click Delete	Company is deleted from the platform and the database	Company removed from the system completely	Pass
TC_06	Search Postcode	1. Go to url: <a href="https://testprojectunite.herokuapp.com/companySearch.jsp">https://testprojectunite.herokuapp.com/companySearch.jsp</a> 2. Search for an available postcode on the list of companies	Companies within matching postcode should display on the application	Expected businesses returned when matching postcode was entered	Pass
TC_07	Search Sector	1. Go to url: <a href="https://testprojectunite.herokuapp.com/companySearch.jsp">https://testprojectunite.herokuapp.com/companySearch.jsp</a> 2. Search for an available sector on the list of companies	Companies within the same sector should display on the application	Expected businesses returned when matching sector was entered	Pass

### TC\_01 Results

End user logged in successfully as we are brought to the list of businesses. We can also see the features such as Add New Company, Edit and Delete which are provided to only admins after a successful login.



## Register Company

Add New Company

ID	Name	Address	Postcode	Sector	Additional Information	Modify/Delete
1	DeliveryService2	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
5	MobileCatering	Dublin city centre	d4	Catering	Mobile catering service, looking to provide catering service on business grounds. Please contact MobileCatering@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
6	Company8	Crumlin Dublin	D2	Delivery	Looking to provide delivery service for companies in the south Dublin City Centre area. Please contact Company8@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
8	Tom's Wings Food Truck	Dublin City Centre	d2	Hospitality	Food Truck looking for open space, in Dublin City Centre. Please contact Tomwings@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
9	Cafe Truck	glasnevin	D11	Hospitality	Need to rent space, in a busy area. Please contact CafeTruck@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
10	GASTROFAKE	London	SW100AD	Hospitality	Need to link up with a delivery service to deliver our products to our customers. Please contact GASTROFAKE@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>
11	DeliverServiceFake	London	SW110AD	Delivery	Looking to provide a delivery service in the London area. Please contact DeliverServiceFake@fakemail.com	<a href="#">Edit</a> / <a href="#">Delete</a>

### TC\_02 Results

Shown in the screenshot below we can see the warning “Please fill out this field” where we left Last Name blank and tried to save all the other details. We ran this test on each field to ensure that all fields are required when writing to the database.

## Register new administrator account

First Name


Last Name

Username

Password

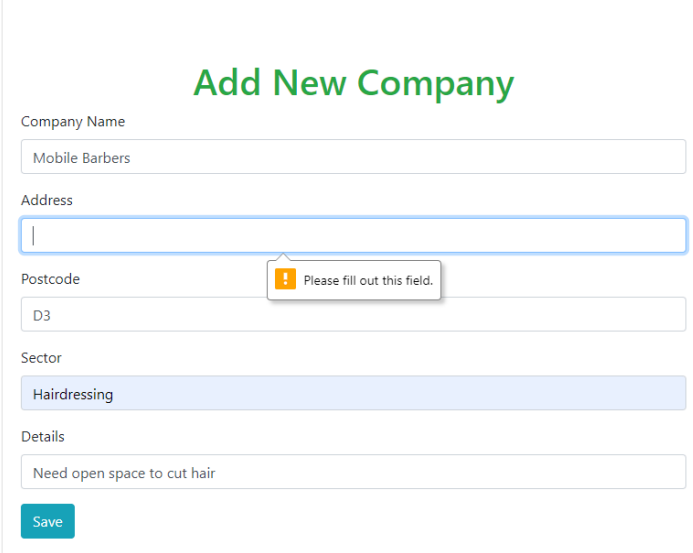
Phone

Email

 Please fill out this field.

### TC\_03 Results

We ran this test on all company fields, leaving one field blank at a time. Shown below in the screenshot we can see the warning “Please fill out this field” when Address field is left blank as the admin tries to save all other details.



The screenshot displays a web form titled "Add New Company" in green text. The form contains several input fields: "Company Name" with the value "Mobile Barbers", "Address" which is empty and highlighted with a blue border, "Postcode" with the value "D3", "Sector" with a dropdown menu showing "Hairdressing", and "Details" with the value "Need open space to cut hair". A blue "Save" button is located at the bottom left. A validation error message, "Please fill out this field.", is shown in a grey box with an exclamation mark icon, pointing to the empty Address field.

#### TC\_04 Results

If you look at the screenshot shown in TC\_01, we update the company with ID 1 which has a Company Name of “DeliveryService2”. Presented in the screenshot below, we update the Company Name to be “DeliveryService3”.

In the second screenshot below we now see the name of the company with ID1 has a Company Name of “DeliveryService3”.

This shows a successful update, we ran this edit test on all fields available, each field updated as expected.

## Edit New Company

Company Name

Address

Postcode

Sector

Details

Registration Logout

## Register Company

ID	Name	Address	Postcode	Sector	Additional Information	Modify/Delete
1	DeliveryService3	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com	<a href="#">Edit / Delete</a>
5	MobileCatering	Dublin city centre	d4	Catering	Mobile catering service, looking to provide catering service on business grounds. Please contact MobileCatering@fakemail.com	<a href="#">Edit / Delete</a>
6	CompanyB	Crumlin Dublin	D2	Delivery	Looking to provide delivery service for companies in the south Dublin City Centre area. Please contact CompanyB@fakemail.com	<a href="#">Edit / Delete</a>
8	Tom's Wings Food Truck	Dublin City Centre	d2	Hospitality	Food Truck looking for open space, in Dublin City Centre. Please contact Tomwings@fakemail.com	<a href="#">Edit / Delete</a>

### TC\_05 Results

This test is very important, if the customer no longer wants their details on the system, they should be removed immediately.

Shown in the screenshot below we delete company with ID number 45.

- Company Name – Pizza Food Truck

- Address - Meath
- Postcode – M11
- Sector – Food
- Additional Information – Need to rent open space

ID	Business Name	Address	Postcode	Sector	Description	Actions
14	Bakery Truck	Phisboro	D1	Hospitality	Need to rent space from 8am - 2pm. Please contact bakery@fakemail.com	<a href="#">Edit / Delete</a>
15	EasyDrinks	Main Street	D1	Hospitality	Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakemail.com	<a href="#">Edit / Delete</a>
45	Pizza Food Truck	Meath	M11	Food	Need to rent open space	<a href="#">Edit / Delete</a>

© 2021 Copyright:

Available in the screenshot below is an updated list after the delete is made.

We can see there is no longer a company with an ID of 45 or any of the other details provided in the bullet points.

This shows the test case successfully passed the end user testing.

ID	Business Name	Address	Postcode	Sector	Description	Actions
13	Greek Food Truck	Grafton Street	D2	Hospitality	Looking to rent space in the city centre. Please contact GrEEKTrUck@fakemail.com	<a href="#">Edit / Delete</a>
14	Bakery Truck	Phisboro	D1	Hospitality	Need to rent space from 8am - 2pm. Please contact bakery@fakemail.com	<a href="#">Edit / Delete</a>
15	EasyDrinks	Main Street	D1	Hospitality	Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakemail.com	<a href="#">Edit / Delete</a>

© 2021 Copyright:

## TC\_06 Results

Searching will be used a lot by the end users, allowing them to get the details they're looking for a lot quicker.

Some businesses may not want to deal with others outside of their town or city. Providing the end users to search by surrounding postcodes.

Seen in the screenshot, we entered in postcode D1, all the businesses displayed that are in D1, along with D11 and D13.

Name	Address	Postcode	Sector	Additional Information
DeliveryService3	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com
Cafe Truck	glasnevin	D11	Hospitality	Need to rent space, in a busy area. Please contact CafeTruck@fakemail.com
Best Cafe On Wheels	Howth	D13	Hospitality	Need space with high footfall to rent. Please contact cafewheels@fakemail.com
Bakery Truck	Phisboro	D1	Hospitality	Need to rent space from 8am - 2pm. Please contact bakery@fakemail.com
EasyDrinks	Main Street	D1	Hospitality	Looking to team with a delivery service who can manage a high number of deliveries throughout the evening. Please contact easyDrinks@fakemail.com

## TC\_07 Results

Another search option is the sector the customer is interested in. Seen in the screenshot below Delivery is entered, showing businesses in this sector only.

Name	Address	Postcode	Sector	Additional Information
DeliveryService3	santry	d11	Delivery	Interested in providing delivery service for other businesses in the dublin area. Please contact DeliveryService2@fakemail.com
CompanyB	Crumlin Dublin	D2	Delivery	Looking to provide delivery service for companies in the south Dublin City Centre area. Please contact CompanyB@fakemail.com
DeliverServiceFake	London	SW110AD	Delivery	Looking to provide a delivery service in the London area. Please contact DeliverServiceFake@fakemail.com

## 2.6. Evaluation

Evaluating the system was reached by executing different techniques of testing.

Executing unit, integration, and end user tests, prove to be a wide range of coverage when evaluating the system. Shown in the screenshots above all unit tests were executed successfully, covering isolated parts of the code.

Integration tests provided us with more comfortability in our evaluation by covering interaction over multiple Java classes, along with generating SQL queries and connecting to the database. Postman was very useful to test all of the integration as a whole ensuring the functionality is working as expected.

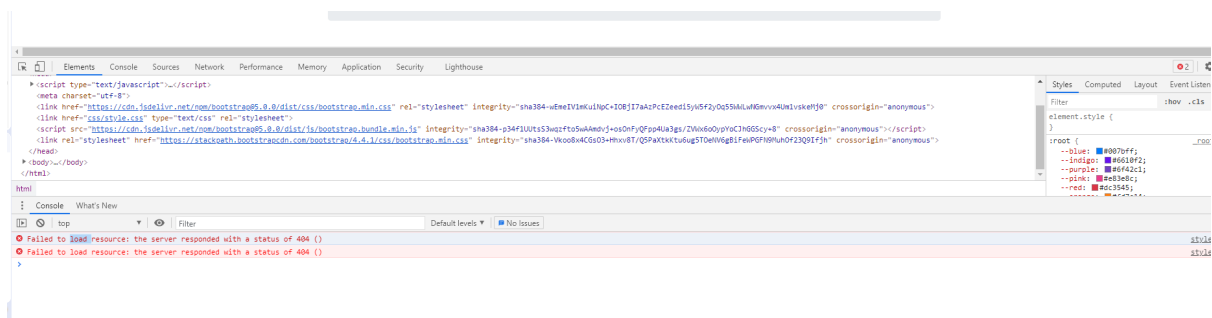
End user tests were extremely useful, by replicating how the customer will interact with the application. We could be sure the application behaves as expected. Customer will see no major issues as end user tests covered all major functionalities.

### 3.0 Conclusions

We cover advantages and disadvantages, along with strengths and limitations to the project.

Advantages of companiesUnite project is providing a platform for people who wish to adapt and survive the current restrictions. Encouraging building new business relationships outside of your organisation. Easy to use web application, with the ability to use API for faster interaction with the database. Some advantages personally were the skills I gained. For example, getting stronger at Java. Completing difficult tasks using JSP files. Deploying a full web application on Heroku while using a deployed version of MySQL. Implementing college curriculum into my project. Getting exposure to how full end to end applications are integrated. Using Bootstrap for designing the GUI. Creating a project that could actually help people in the future. Getting familiar with new tools such as Postman and putting them into practice. Providing potential customers with the opportunity to adapt to the issues forced on us due to the pandemic.

Disadvantages of the companiesUnite project little or no resources, hosted on a free dyno type on Heroku which sleeps when their inactivity for 30 minutes. Unable to import JavaScript or CSS file like I wanted into the JSP file. This was the biggest disadvantage as I found it out later in the project life cycle. Seen in the screenshot below there is a bug which I was unable to resolve. Forced to use HTML required tags to have some sort of form validation. I really wanted to integrate JavaScript into this project. Issue I faced trying to design the application using CSS and Javascript, I would receive this Failed to load resource: the server responded with a status of 404 () by trying a simple import like `<link rel="stylesheet" href="css/style.css" type="text/css"/>`



Strengths of the companiesUnite project, real time updates reflected on the application. JSP files work well with Servlets allowing for smooth communication. Server-side code was very useful for the search feature. Project is hosted on Heroku, providing availability and

security. Project was created to help people, which might entice others to get behind and support.

Limitations are the fact we have a cheap set up when it comes to instances on the cloud as there is no money invested. Minimum security set up, just the basics provided by Heroku.

## 4.0 Further Development or Research

Given the opportunity arose and the benefits of additional time for the project. I'd may have considered a newer framework such as Angular or REACT for the frontend.

MySQL for the database done the job it needed, integrating the database into the cloud, wasn't as tough as I thought it would be originally.

MongoDB is being used a lot in the modern applications, a lot of the big companies uses it. Given I decide to continue the project, I'll consider using different frameworks.

At the beginning of the project, I wasn't familiar with a lot of frameworks and what would best suit me. Deciding to go with what I knew some companies use and languages thought by the college.

For example, using the whole API option, is widely used in the insurance industry. This was covered in the Web Services API module. My goal was to use what I learnt from the college.

Provided there were additional resources, I would look to host the applications on more secure machines. Ensure better performance once the funding was available.

Direction I would like to take the project would be including a lot of new features to enhance the application. Get feedback from other people, see if they have any interesting ideas.

I would go with the new frameworks out there; I'd have to spend some time getting familiar with them.

Ensure I choose the right cloud platform for this project, get the best security, performance, and availability my resources allow.

## 5.0 References

ClearDB.com. 2021. *ClearDB Developer Center - Welcome*. [online] Available at: <<https://www.cleardb.com/developers/platform/overview>> [Accessed 27 April 2021].

Tutorialspoint.com. 2021. *JSP Tutorial - Tutorialspoint*. [online] Available at: <<https://www.tutorialspoint.com/jsp/index.htm>> [Accessed 12 February 2021].

Tyson, M., 2019. *What is JSP? Introduction to Java Server Pages*. [online] Info World. Available at: <<https://www.infoworld.com/article/3336161/what-is-jsp-introduction-to-javascript-pages.htm>> [Accessed 10 February 2021].

Tutorialspoint.com. 2021. *JSP - Standard Tag Library (JSTL) Tutorial - Tutorialspoint*. [online] Available at: <[https://www.tutorialspoint.com/jsp/jsp\\_standard\\_tag\\_library.htm](https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm)> [Accessed 18 February 2021].

JournalDev. 2013. *JSTL Tutorial, JSTL Tags Example - JournalDev*. [online] Available at: <<https://www.journaldev.com/2090/jstl-tutorial-jstl-tags-example>> [Accessed 22 February 2021].

Dev.mysql.com. 2021. *MySQL :: MySQL Connector/J 8.0 Developer Guide :: 7.1 Connecting to MySQL Using the JDBC DriverManager Interface*. [online] Available at: <<https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-connect-drivermanager.htm>> [Accessed 17 January 2021].

[https://mymoodle.ncirl.ie/pluginfile.php/127662/mod\\_resource/content/0/Week5.RESTful%20Web%20Services%20III%20JAX-RS%20in%20a%20nutshell.pdf](https://mymoodle.ncirl.ie/pluginfile.php/127662/mod_resource/content/0/Week5.RESTful%20Web%20Services%20III%20JAX-RS%20in%20a%20nutshell.pdf) [Accessed 8 December 2020].

Tutorialspoint.com. 2021. *JUnit - Plug with Eclipse - Tutorialspoint*. [online] Available at: <[https://www.tutorialspoint.com/junit/junit\\_plug\\_with\\_eclipse.htm](https://www.tutorialspoint.com/junit/junit_plug_with_eclipse.htm)> [Accessed 2 April 2021].

Youtube.com. 2017. *JSP 6 CRUD - Search Data*. [online] Available at: <<https://www.youtube.com/watch?v=W4D1TVGd7UA>> [Accessed 03 March 2021].

## 6.0 Appendices

This section should contain information that is supplementary to the main body of the report.

### 6.1. Project Plan





# National College of Ireland

## Project Proposal Companies Unite 01/11/2020

BSHCE

Software Development

2020/2021

Aaron Reilly

x17124719

x17124719@student.ncirl.ie

### Contents

1.0	Objectives.....	73
2.0	Background .....	73
3.0	Technical Approach.....	73
4.0	Project Plan .....	74

5.0	Technical Details .....	76
6.0	Evaluation .....	76

## 7.0 Objectives

The goal for this project is to build an API system that helps companies stay open throughout the pandemic. The application will be built in java while stored in the cloud. This app will provide companies the visibility to other interested companies. On this app they can find companies that best match their needs and that are more suitable.

1. Provide companies a platform where they can unite
2. Increase company interaction by providing a stable system
3. Enhance other companies understanding on the help available
4. Implement a way for companies to quickly find help they need
5. Encourage company to company engagement
6. Create a teamwork environment amongst competitors
7. Platform to be stored in the cloud

## 8.0 Background




















During the pandemic, many companies had to shut down. In some cases, they had to shut down completely as they could not afford to fund through the pandemic. Many companies had the ability to serve outside and keep social distancing. We have seen cases where if a place did not serve food they had to completely shut down. This happened on more than one occasion also, seeing many people lose jobs and family businesses. Companies Unite is an application which will help those companies, allowing them to react dynamically to the situation. There are companies out there that can help others and provide a service, whether that is providing food to a local pub or even an open space to a restaurant if the only option is to serve outside. On this application companies can provide information on services they provide along with help they need. Some companies might want to just help maybe they have a big open area they could rent to other companies.

## 9.0 Technical Approach

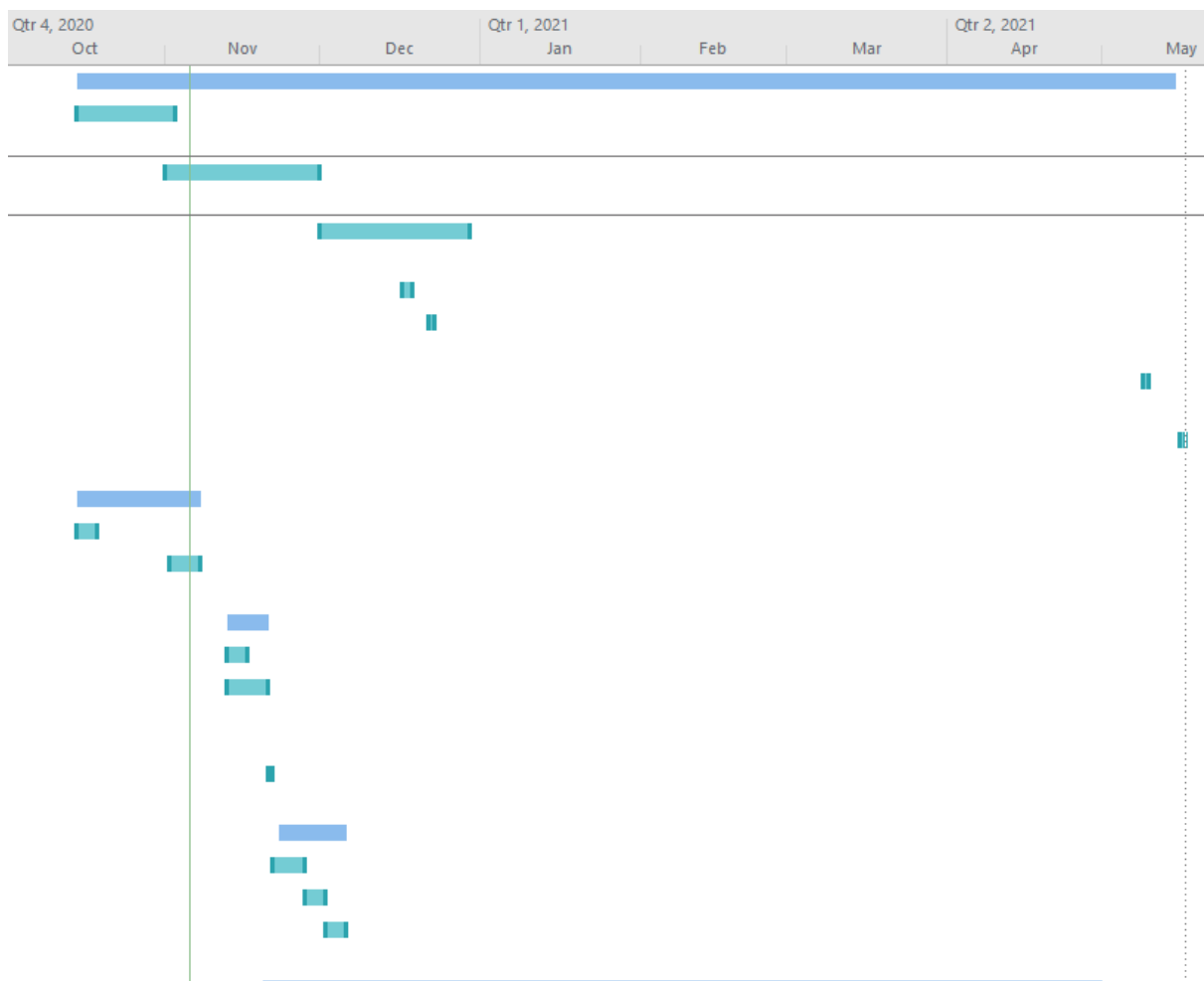
To provide customers with a stable and effective platform the approach I am looking to take for this project is an API system to transfer data. This will be developed in the programming

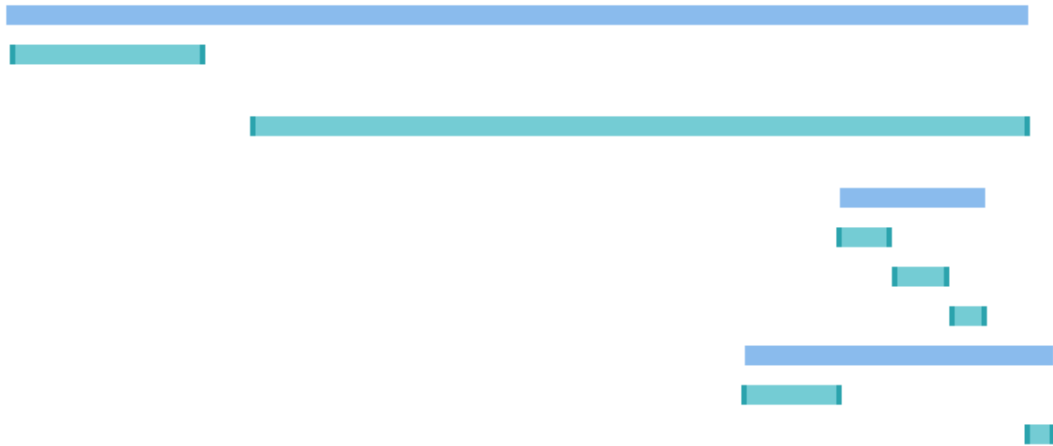
language java, requests and responses used for the transferring of data will be in JSON format. Data to be stored in MySQL database where the application will query that stored data. For the frontend I am planning on outputting to a form where I will use html and JavaScript. This platform will be stored in the cloud as this module is in semester 2.

## 10.0 Project Plan

	Task Mode ▾	Task Name ▾	Duration ▾	Start ▾	Finish ▾
		<b>Milestones</b>	<b>152 days</b>	<b>Thu 15/10/20</b>	<b>Fri 14/05/21</b>
		Reflective Journal - October	13 days	Thu 15/10/20	Mon 02/11/20
		Reflective Journal - November	22 days	Sun 01/11/20	Mon 30/11/20
		Reflective Journal - December	21 days	Tue 01/12/20	Tue 29/12/20
		Project Prototype	2 days	Thu 17/12/20	Fri 18/12/20
		Mid Point Presentation	1 day	Tue 22/12/20	Tue 22/12/20
		Final Project Documentation	1 day	Sun 09/05/21	Sun 09/05/21
		Final Project Presentation	1 day	Sun 16/05/21	Sun 16/05/21
		<b>Planning</b>	<b>17 days</b>	<b>Thu 15/10/20</b>	<b>Sat 07/11/20</b>
		Project Pitch	3 days	Thu 15/10/20	Sun 18/10/20
		Project Proposal Document	6 days	Mon 02/11/20	Sat 07/11/20
		<b>Analysis</b>	<b>6 days</b>	<b>Fri 13/11/20</b>	<b>Fri 20/11/20</b>
		Research Requireme	2 days	Fri 13/11/20	Mon 16/11/20
		Implement Requirement Documents	6 days	Fri 13/11/20	Fri 20/11/20
		Review Requirements	1 day	Sat 21/11/20	Sat 21/11/20
		<b>Design</b>	<b>10 days?</b>	<b>Mon 23/11/20</b>	<b>Sat 05/12/20</b>
		API Design	6 days	Sun 22/11/20	Fri 27/11/20
		Database Design	3 days	Sat 28/11/20	Tue 01/12/20
		User Interface Design	4 days	Wed 02/12/20	Sat 05/12/20

	★	Prototype Implementation	21 days	Fri 20/11/20	Sun 20/12/20
	★	Final Project Implementation	89 days	Tue 29/12/20	Fri 30/04/21
📅	➡	<b>Testing</b>	<b>17 days</b>	<b>Thu 01/04/21</b>	<b>Fri 23/04/21</b>
	★	Unit Testing	6 days	Thu 01/04/21	Thu 08/04/21
	★	Integration Testing	7 days	Sat 10/04/21	Sat 17/04/21
	★	Functional Testing	5 days	Mon 19/04/21	Fri 23/04/21
📅	➡	<b>Deployment</b>	<b>35 days</b>	<b>Wed 17/03/21</b>	<b>Tue 04/05/21</b>
	★	Consider Platform (A	11 days	Wed 17/03/21	Wed 31/03/21
	★	Deploy to suitable of	3 days	Sat 01/05/21	Tue 04/05/21





## 11.0 Technical Details

Approach for the frontend is to use the following

- JavaScript – provide feedback to the customer after an action has been taken.
- HTML – Structure the UI here we will create the form and login page
- CSS – Design the UI using CSS to improve on the UI

Backend programming language

- Java - Core programming language

Database Management System

- MySQL – Using this we can store the data and make modifications to the data

Storage

- Cloud Services – Heroku is the provider used to host the project.

## 12.0 Evaluation

This project will use mock data, where customers can be replicated with fake data. This data will contain information about help they need. Junit will be used for unit testing running tests against the java code. Integration testing, I would like to follow the BDD (Behaviour Driven Development) process using cucumber here I will create feature files explaining the behaviour with the integration code running in the background. Functional testing will happen when the UI is available, this may be a manual process, if time allows, I would like to automate those tests with selenium.

12.1. Ethics Approval Application (only if required)

12.2. Reflective Journals

#### Reflective Journal (October)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: October 2020

#### *My Achievements*

For the month of October my achievements were successfully submitting my project proposal. Also, I have made a good start on my CA's for Mobile Applications, Strategic Management & AI modules.

#### *The main issues I faced*

The main issue I had this month was getting familiar to Teams and doing everything online. This is my first semester online, its very new to me. At the beginning, I was missing the start of some classes just because I was not sure how to get there. Lecturers use Teams differently, setting up groups and working within a team on Teams.

#### *My Reflection*

As I have little or no time at all, I decided to log into my account early before work and play around with Teams just to become more comfortable with it. Now I have a really good understanding on how to use teams, its no longer an issue. Only issue I have now is my mic won't work which means I can talk to others I need to type everything.

#### *Intended Changes*

Now that I have been logging on earlier than before, I'm using that time on CA's to take pressure off later. Coming into November I plan to focus more on Data Application Development & API modules. I plan to resolve my mic issue this week.

#### Reflective Journal (November)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: November 2020

#### *My Achievements*

For the month of November my achievements were successfully submitting CA's for Mobile Applications, Strategic Management, API and Data Application Development.

### *The main issues I faced*

The main issue I had this month was on my CA1 for Data Application Development. I found it very difficult to find datasets online that I could use for the CA. The delay put on a lot of pressure right before submission. I never had deal with data in that way, which left me unsure if the data would be enough for the CA. Other difficulties were the fact I never used R before. Also, the documentation was a lot. I never had used IEEE format before and its referencing. It would have been good to go over some examples.

### *My Reflection*

I could have managed my time better and been more decisive in choosing what datasets to work with for the Data Application Development CA. Then I could have spent more time on the more interesting parts of the CA, rather than rushing at the end.

### *Intended Changes*

CA2 is similar but with python, I plan to decide on the dataset a lot earlier and get an early start on the CA. Hopefully this will take off more pressure around the time of submission.

### Reflective Journal (December)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: December 2020

### *My Achievements*

Achievements for December were successfully submitting CA's for AI, Mobile Applications, Data Application Development. My main achievement for this month was delivering the mid-point presentation. As a lot was needed to get done for this, given the deliverables for other modules.

### *The main issues I faced*

The main issues I had this month was delivering my CA for Data Application Development. The main issue I had with Data Application Development was writing python programs, having no practical experience with that programming language.

### *My Reflection*

I feel we could have covered more practical work during the semester. Maybe I could have spent more time on Python courses in my own time before starting the semester.

### *Intended Changes*

Be more prepared before starting a semester, get a good understanding on what will be delivered in each module.

### Reflective Journal (January)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: January 2020

### *My Achievements*

For the month of January my achievement was successfully migrating my codebase from NetBeans to Eclipse. Being able to connect the application to a MySQL database. The program can now ADD, UPDATE, DELETE and RETRIEVE data from a MySQL database.

### *The main issues I faced.*

The main issue I faced this month was trying to connect my project using NetBeans to a MySQL database. Having searched for examples on the web, looking on stack overflow and other sites. Unfortunately, I couldn't get it to work. Given I have more experience with Eclipse I decided to migrate my project to Eclipse.

### *My Reflection*

Making the decision to move to Eclipse proved to be the right decision as I can now interact with a MySQL database. Maybe I shouldn't have waited to migrate at the end of the month. I believe I spent too much time trying to stay with NetBeans rather than using Eclipse.

### *Intended Changes*

Spend more time at the beginning of the project researching and deciding on what tools I should use throughout the project.

### Reflective Journal (February)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: February 2020

### *My Achievements*

For the month of February my achievement was getting more familiar with eclipse. Have some UI available for my project using JSP and Servlets. My main achievement was being able to read data



from the database and displaying it on the UI. Setting this up took a lot of time and effort which is why I have this as my main achievement. I would have liked to spend more time on the code or designing the UI.

#### *The main issues I faced.*

The main issue faced for the month of February was trying find a way to use both my machine-to-machine code along with UI implementation while interacting with MySQL database all in the same project.

#### *My Reflection*

Looking at how long I spent trying to get everything to work together, probably should have picked easier technologies. JAX-RS we covered in our modules which was fine but assuming we would be covering the UI and how they all link together was what I spent a lot of time on.

#### *Intended Changes*

Know more about the technologies I decide to use for projects, I knew I would be interested in these but was not aware of how difficult I'd find setting up the project or finding the necessary jar files in order to make the project work.

#### Reflective Journal (March)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: March 2020

#### *My Achievements*

For the month of March my achievements were successfully submitted Usability Design and Distributed Systems CA's. Distributed System RPC CA was difficult but, in the end, I was happy with what I got done. Even though it was difficult I found the assignment very interesting.

#### *The main issues I faced.*

The main issue faced for the month of March was not getting to spend as much time on my project as I would have liked. We had a lot of CA's due for this month, which took up most of my time.

#### *My Reflection*

Overall, I'm happy with my performance this month. Completing all assignments in time, could have managed my time better and worked on my project more.

### *Intended Changes*

Better time management, even though I'm happy with what I got done this month. I feel with better time management I could have spent more time on my project.

### Reflective Journal (April)

- Student name: Aaron Reilly
- Programme: BSHCSD4 – Software Development Stream
- Month: April 2020

### *My Achievements*

For the month of April my achievements were successfully submitting Cloud Computing CA's. Working on a ruby on rails project was interesting and new to me, never used it before. Deploying the project on Heroku proved to be challenging. Other achievements were adding search and CRUD functionalities to my project.

### *The main issues I faced.*

The main issue faced for the month of April was creating my own gem for my ruby on rails cloud computing CA. Unfortunately, I was unable to get the functionality working when it was deployed to Heroku. I did present the functionality on my local environment.

### *My Reflection*

Would have liked to spend more time on my project. Happy, with what I achieved given the time spent on the project this month.

### *Intended Changes*

Spend all my time on the project now that all the modules are completed.

## 12.3. Other materials used

### 12.3.1 Think Aloud

Think aloud is a very popular technique it's one of the techniques we covered in this semester for our usability design module.

Through this technique we can get a full understanding on the end users' interaction with our web application.

Task Name / Number	Edit Company Information / TC_04
Task Goal	Edit existing company details
Start & End Times	15:34 – 15:37

Expected / Ideal Behaviour	The end user will successfully edit a company's detail
Actual Behaviour	Company details updated successfully
Notes / Comments	End user took a bit of time to decide what they should update
Anything additional	Should have suggested an update before executing the technique