

Review Eat

HND Graded Unit Project

Planning

Yogesh Parajuli

Table of Contents

[Project Brief and analysis 2](#_Toc36752216)

[Introduction 2](#_Toc36752217)

[Brief and overview of the project 2](#_Toc36752218)

[Future Vision 2](#_Toc36752219)

[Background Research 3](#_Toc36752220)

[How does this project unique from similar services in the market? 4](#_Toc36752221)

[Target Users 4](#_Toc36752222)

[Project Plan 5](#_Toc36752223)

[SDLC Model 5](#_Toc36752224)

[Identification of Resources 6](#_Toc36752225)

[Cost IdentificaTIon 6](#_Toc36752226)

[Identification of Information Source 6](#_Toc36752227)

[Project Tasks Identification 7](#_Toc36752228)

[Functional and Non-functional Requirements 9](#_Toc36752229)

[Design 10](#_Toc36752230)

[Programming Paradigm 10](#_Toc36752231)

[Programming Architecture 10](#_Toc36752232)

[User Interface Design 11](#_Toc36752233)

[Use Case Diagram 15](#_Toc36752234)

[Activity Diagram 21](#_Toc36752235)

[Classes Identificaiton 25](#_Toc36752236)

[Class Diagram 26](#_Toc36752237)

[Design Pattern Identification 27](#_Toc36752238)

[Strategy Pattern 27](#_Toc36752239)

[Iterator Pattern 28](#_Toc36752240)

[Singleton Pattern 28](#_Toc36752241)

[Data Dictionary 29](#_Toc36752242)

[Sequence Diagram 34](#_Toc36752243)

[Database Design 37](#_Toc36752244)

[Normalisation 37](#_Toc36752245)

[Identifying Data Types 39](#_Toc36752246)

[References 41](#_Toc36752247)

# Project Brief and analysis

## Introduction

A software is being developed as part of the HND Software Development Graded Unit Project, which allows user to review restaurants they have visited. The software will focus on restaurants featuring a relatively small number of restaurants; however, it will be developed with a vision to expand it further. The software will be written in Java and will make use of various other techniques for graphical user experience and to handle data.

## Brief and overview of the project

The program bases on the idea of helping local businesses to find a customer and also to help a customer find hidden gems in their area. The proposed platform aims to provide the user with an overview about restaurants available in an area, list them based on review ranks and allow users to write and share their dining experience with other people via the platform.

The software will be a menu-driven desktop application and will feature graphical user experience. The users will have to log in with their details to access the software. There will be two different logins, one for regular users and another one for the admin. The admin will have certain privileges that users would not have. Admin will be responsible for adding restaurant details that can be then viewed by a user. They will also have the right to delete a user from the system. If a user wants to delete their account, they will have an option to do so as well. The user can view all the restaurants added by the admin. They will have an opportunity to view the list based on different categories of the restaurant and also see the list based on review ranking. A database would store all the details of users, restaurants and reviews. The platform will feature different types of restaurants based on cuisine which will be added by the admin.

The project is relatively small-scaled; hence it would take less time to complete, will have minimum cost and a high-quality product could be delivered within the timeframe.

## Future Vision

Review Eat as it stands a small-scale project, but this platform has prospects. We can use internet and APIs to add restaurants automatically. Each restaurant can have their page where they could feature their top reviews and share photographs of the restaurant. We can cover a large area and feature restaurants from around the city or the country. However, these could be future expansion plans, but they are not part of the current scope of the project.

## Background Research

A research was conducted to find applications or websites that focus on reviewing restaurants, but, no services were found that would solely review restaurants. There were various online platforms that either had restaurant reviews as part of their system, or they also included other types of business in their platform. They were all online-based. Here are some online platforms that have restaurant review as part of their system.

1. **TripAdvisor[[1]](#footnote-1)**

TripAdvisor is a global online platform that provides price comparisons and online reservation for different experiences. Their website feature 730 million reviews out of which approximately 4.9 million reviews are for restaurants. (trip-10k\_20181231.htm, 2018)

TripAdvisor has an admirably beautiful website for restaurants. Since TripAdvisor is an online travel company, it features restaurants from all around the world. When narrowed down to a smaller area, the website featured filters that could be used to view restaurants based on cuisine, price, meals and others. Restaurants were ranked based on the best reviews and top rating. A user could only add reviews when logged into the system, but reviews were visible to everyone, even those who were not logged in. The website ran advertisement and also displayed sponsored restaurant on top of the page (Website visited: (TripAdvisor, 2020))

1. **Yelp[[2]](#footnote-2)**

Similar to TripAdvisor, Yelp is an online platform that runs a review forum. There are many businesses listed there, and the restaurant business is one of them. The website has 148 million reviews as of 2017. (Wikipedia, 2020)

The search was narrowed down to a small area for research purpose. The website displays a list of all restaurants on the left half of the screen, and on the right half, it has a map of the small area with numbers (based on ranks of restaurants) that show their location. Details of only two restaurants cold fit the screen at a time. (Website Visited: (Yelp, 2020)) Also, Yelp did not feature as many restaurants as TripAdvisor.

Website such as **The Guardian**[[3]](#footnote-3) and **Independent**[[4]](#footnote-4) also does reviews on the restaurant, but they are news services and require a member of their team to visit the restaurant to review them. Since they do not allow everyone to publish their opinion, they were not researched further.

## How does this project unique from similar services in the market?

1. The project bases on the principle of specialization and focuses on providing a simple functionality to a high standard. When software offers too many services, it complicates the system.
2. It will not feature any annoying advertisement.
3. It is free. Since there is no paid functionality, this encourages unbiased and genuine reviews of the restaurant.

## Target Users

The targeted users of the software are as follows:

* People who want to find a decent place for dining.
* The people who want to share their dining experience.

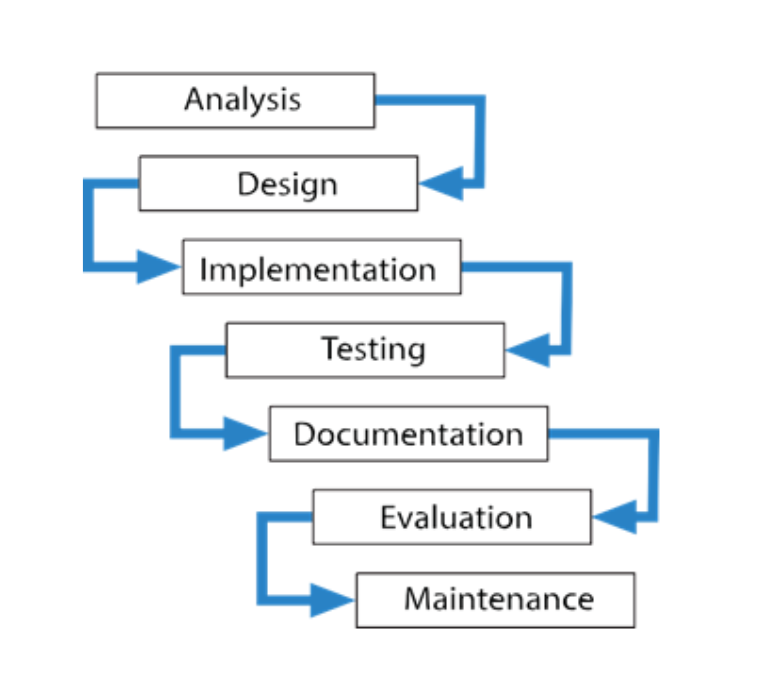
Businesses will have the opportunity to reach new people, and hence the software can contribute to the growth of local hospitality businesses. It could also aid in quality control of restaurants as their reviews are public and would have the business’s reputation at stake.

# Project Plan

## SDLC Model

Software Development Life Cycle is a set of steps that are carried out during the development of software. SDLC includes stages such as Analysis, Design, Implementation and Testing, Documentation, Evaluation and Maintenance. Various SDLC models are used to control the project and to ensure that the project is delivered on time.

Review Eat as a project will be completed using the **Waterfall** Model. Waterfall model is a beginner-friendly and widely accepted Software Development Life Cycle model. As the name suggests, the Waterfall method is a structural model where stages are hierarchically completed one after another only when the current phase is completed. Each step involves time constrains and deliverables. Waterfall method has established stages. It is easy to understand and arrange tasks. Likewise, documenting the process is uncomplicated.

[[5]](#footnote-5)

Since the current project is short and has relatively simple requirements, the Waterfall Model would be the right choice for the project.

## Identification of Resources

**Java[[6]](#footnote-6)** will be the primary programming language used to build the project. Java is an object-oriented high-level programming language. It is fast, secure and reliable. **Java FX[[7]](#footnote-7)** will also be used for the graphical interface of the program. Java FX is an application framework that works with Java and allows us to create a desktop application with a Graphical User Interface. **SQLite[[8]](#footnote-8)** is a Relational Database Management System. SQLite uses SQL commands to perform queries in the database. SQLite will be used to store information in the database. Likewise, **Eclipse (Version: 2018-09 (4.9.0))[[9]](#footnote-9),** a software development kit, will be the development environment that will be used for this project. **Project Manager (2016)[[10]](#footnote-10)** is used for formal planning and scheduling of tasks. (Microsoft Azure account was used to access the download portal). For design purposes, **Draw.io[[11]](#footnote-11)**, an online platform, will be used. Furthermore, **Microsoft Visio[[12]](#footnote-12)** will also be used for design purposes.

The machine that will be used is an **HP Pavilion Laptop** 14-ce0xxx(Intel Core i7, 8GB RAM, 64-bit Operating System, Windows 10).

## Cost IdentificaTIon

Most of the resources are available free of cost. Project Manager requires Microsoft Subscription. However, the college provides a free subscription for the students. Hence, it could be used free of charge. Likewise, the machine that is being used is a personal computer already owned by the developer (i.e. student developing the project). Therefore, the cost of this project is virtually zero.

## Identification of Information Source

As the project requires a significant amount of research, the information will be gathered from various internet sources. Here is a list of websites will be accessed during the execution of the project:

* <https://www.sqlitetutorial.net/sqlite-java/>
* <https://www.tutorialspoint.com/javafx/index.htm>
* <https://support.office.com/en-gb/article/basic-tasks-in-project-8fdbf020-a9e1-45e4-bf15-23a8d2b6797d>
* <https://stackoverflow.com/>
* <https://www.youtube.com/>

Any further resources used are cited along the way.

## Project Tasks Identification

The work-breakdown structure has been identified and also a Gantt Chart has been created alongside work-breakdown structure.

Figure 0‑1 Work Breakdown Structure

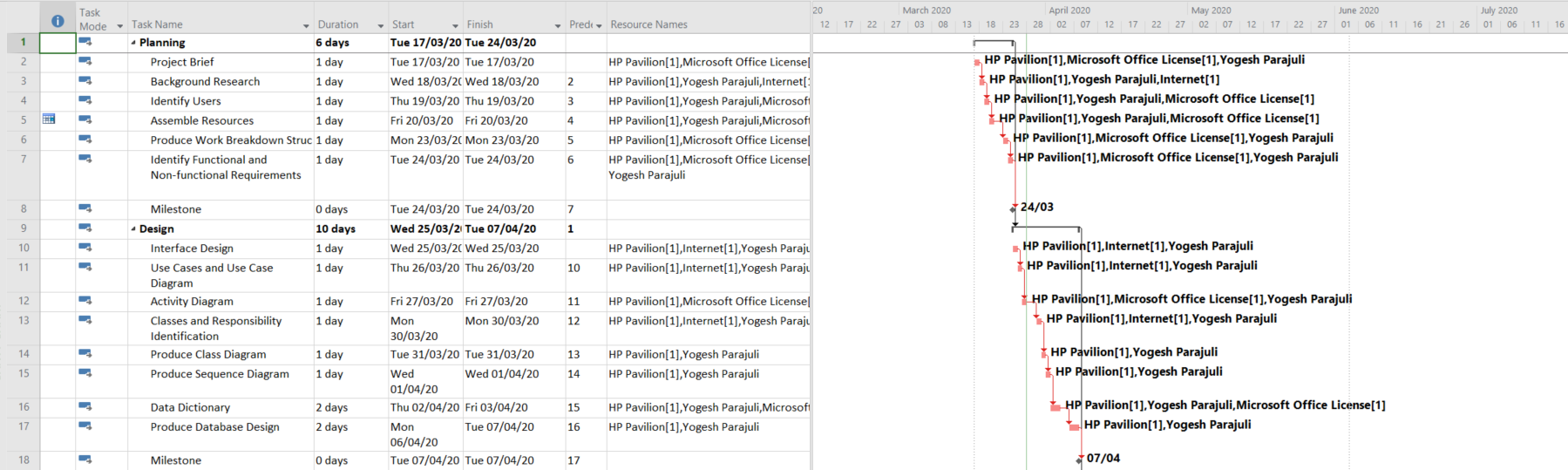


Figure 0‑2 Gantt Chart (1)

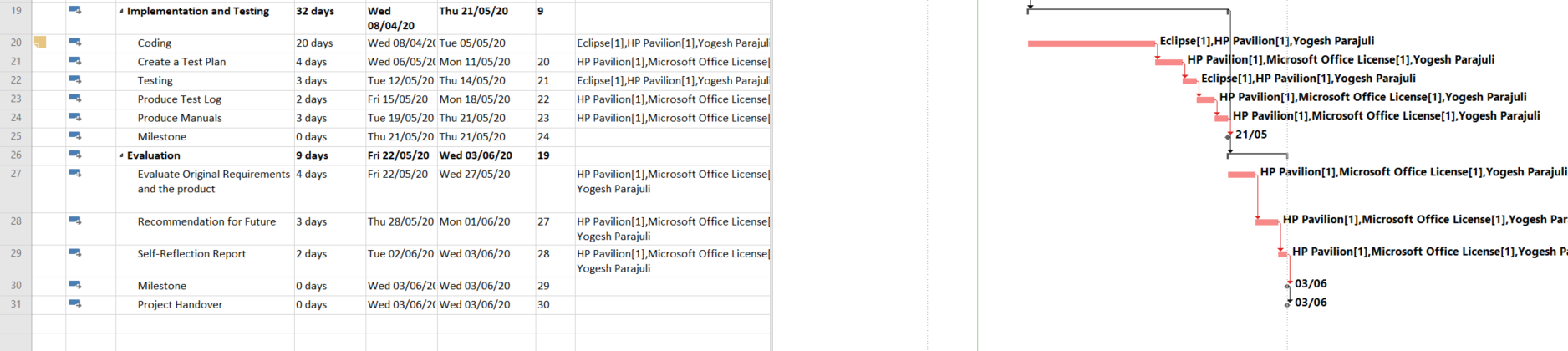


Figure 0‑3 Gantt Chart (2)

## Functional and Non-functional Requirements

Functional Requirements:

* Allow user to create an account with username and password. User can choose their Username, but it must be unique, and the password will have specific criteria.
* Allow user to reset the password if necessary.
* The software should store the user’s information in a database.
* User can update their information and delete their account if they want.
* Allow access to the software with valid id and password.
* Admin will have predefined Username and password.
* Admin will be able to add restaurant details in the system.
* Admin can update restaurant details or delete them.
* Admin also has the privilege to delete a user’s account.
* User logged in the system will see all the restaurant added by the admin. The restaurant can be ranked according to their review rating or by alphabetical order.
* Allow user to add a review for a restaurant. The review is displayed to everyone, but it can be edited or deleted by the person who added the review. There will be a five-star rating system that a user can use to rate the restaurant.
* A review can only be added if the user has been to the restaurant in the past thirty days.
* User should be able to search for a restaurant using a search bar. User can filter results based on categories of restaurants.
* User should be able to log out of the system.

It is likely that the final product may or may not incorporate all the functions mentioned above. They will be evaluated at the later stage.

Non-Functional Requirements:

* **Data Integrity**. The data provided by the users will be used binding by the regulation of data use. Storing and processing of data will comply with the Data Protection Act 2018. (Legislation.gov.uk, n.d.)
* **Security**. The login must be secure. Implement session management. Protect the database from threats such as SQL injection. Unsuccessful login tries should have audit trailing.
* **Usability**. The software should be easy to use.
* **Support**. The product should come with a manual. Support functionality should be added to the software itself as well. Preferably you could build a website for assistance.
* **Robust**. The software should be sturdy and should run without any bugs or glitches.
* Set permission for various activities. For example, a user should not be able to add a restaurant or update restaurant information.

# Design

## Programming Paradigm

The software bases on **Object-Oriented Paradigm**. Object-oriented Programming (OOP) is a methodology to design and develop a program using the concept of classes and objects. Objects are real-world entities and classes are a blueprint of the objects. Object-oriented programming provides several benefits such as modularity which aids in unit testing, reuse of code, flexibility and other.

Review Eat uses real world object such as restaurants and user, and classes will be introduced that dictates their functionality. Likewise, in OOP problems are broken down into small solvable chunks called modules. For example, searching restaurant will be handled by one module while writing a review is handled by another. This brings benefit of easy troubleshooting and testing. Similarly, OOP allows concept of encapsulation where internal representation is hidden from outer access. This enhances security. On similar note, concepts such as Inheritance and Polymorphism provides flexibility and allows code to be reused.

Hence, OOP is justifiable and suitable for the current project.

## Programming Architecture

Review Eat will use **Multi -Tier Architecture (3 tier Architecture).** A 3- tier architecture is made up of three different layers. The layers are Presentation Layer, Application Layer and Data Layer.

The Presentation Layer is the front-end layer in the 3-tier architecture. It consists of graphical user interface. For current software, JavaFX will be used to create the presentation layer. Likewise, Application layer is the core layer that drives the program. The main capabilities of the software are determined by the application layer. In current case, Java will be used to write the application layer. Finally, the Data Layer is made up of the database and data access system. SQLite will be used for the data layer along with a dedicated class that accesses the database.

3-tier Architecture provides various benefits. The breaking down of software into three different layer makes it easier to develop. It allows flexibility and scalability. Likewise, modularity also allows easier troubleshooting without having much impact on other layers. (JReport, 2020)

Therefore, this architecture would be feasible for the current project.

## User Interface Design

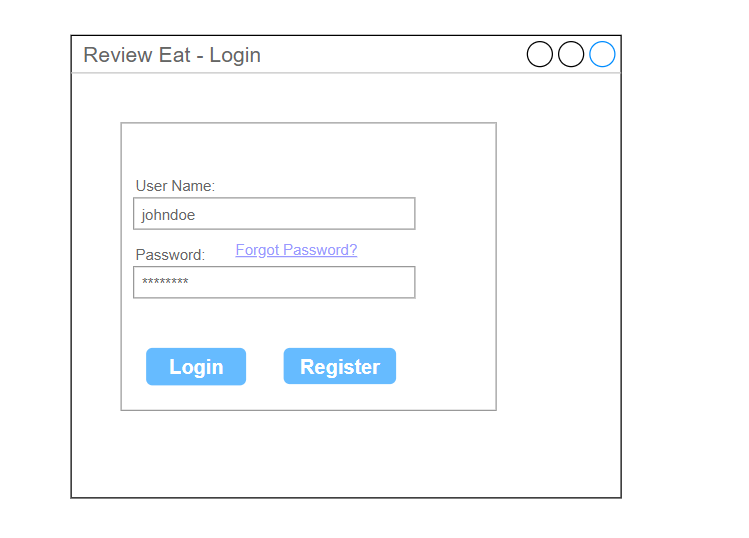


Figure 0‑1 Login Design

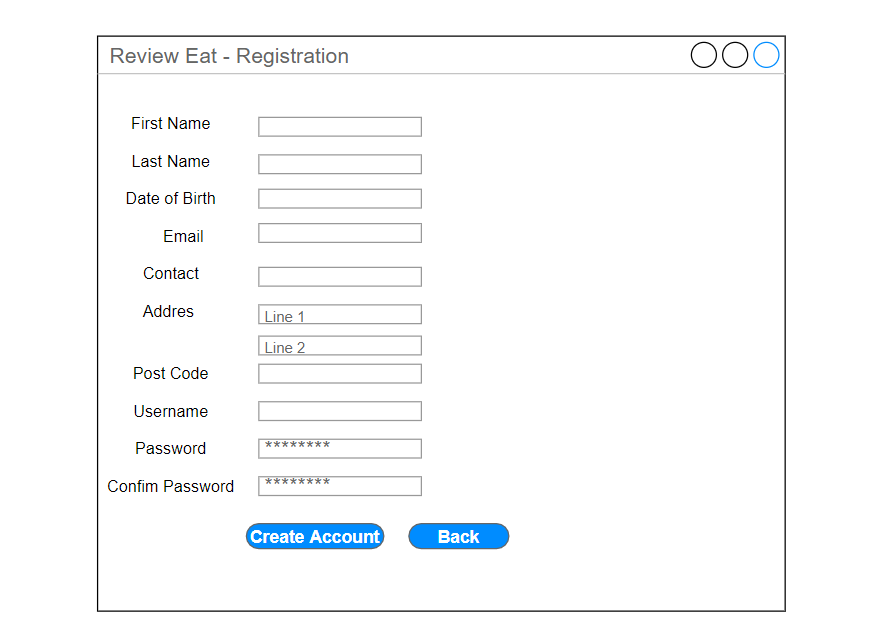


Figure 0‑2 Registration Form Design

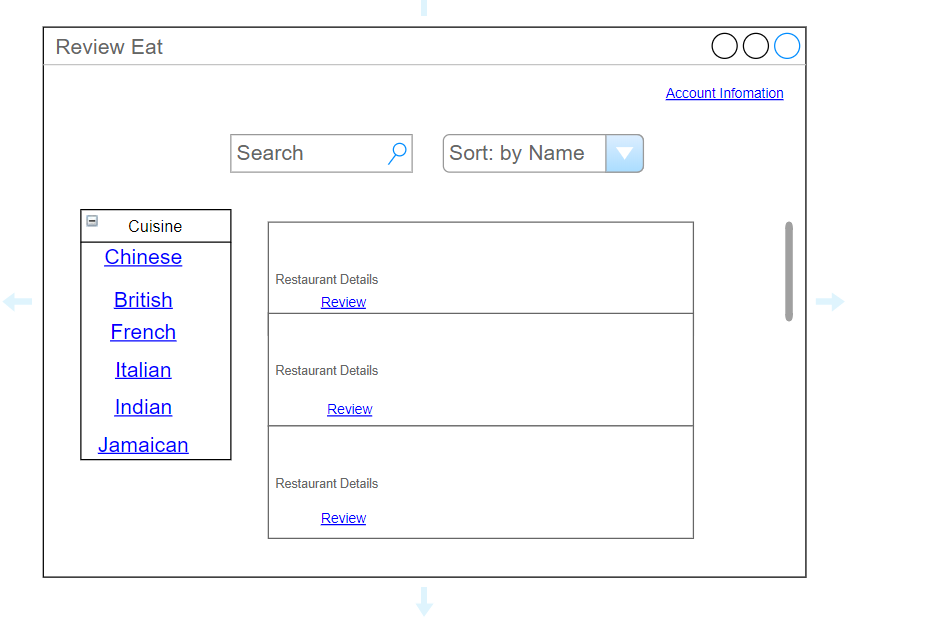


Figure 0‑3 Main Page Design

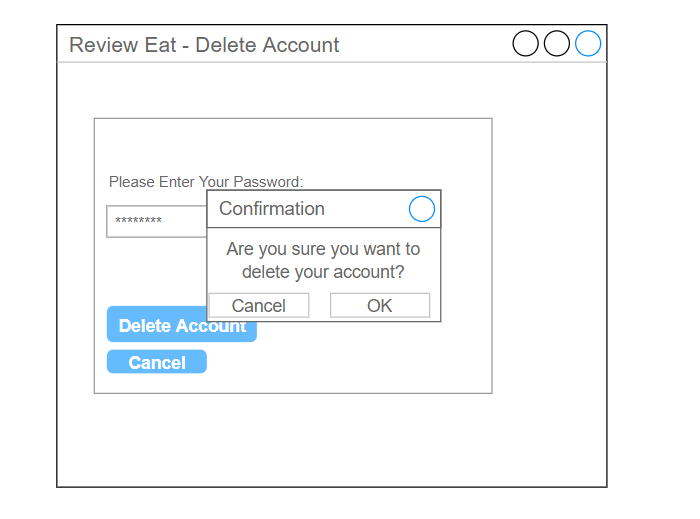


Figure 0‑4 Delete Account Design

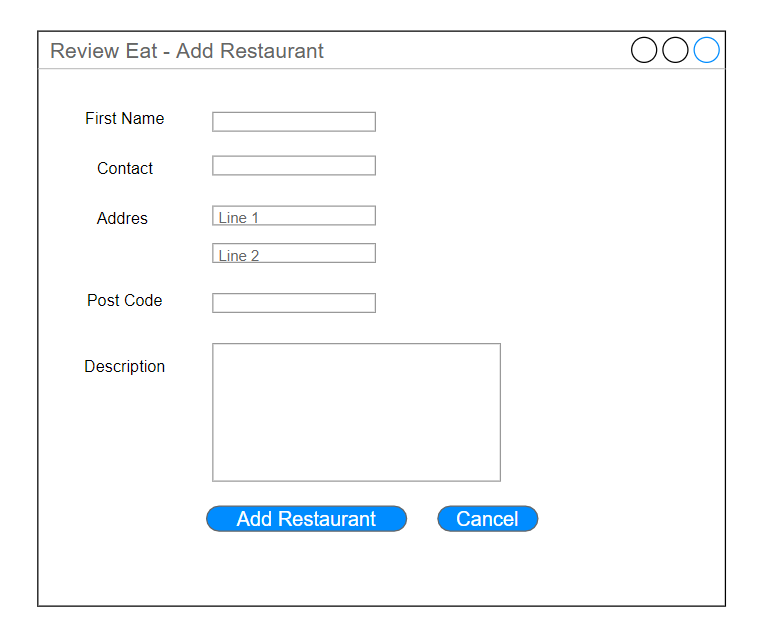


Figure 0‑5 Add Restaurant Design

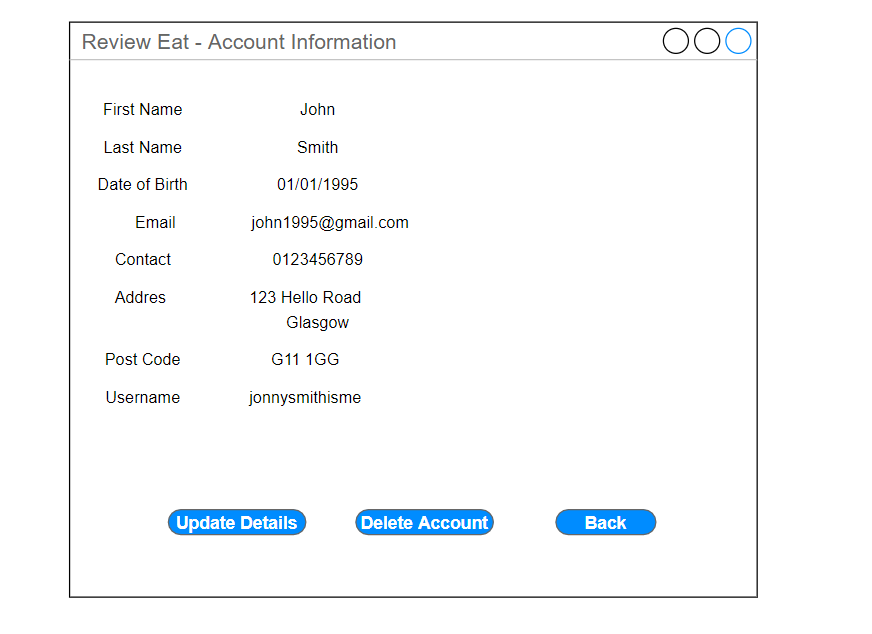


Figure 0‑6 Account Information Design

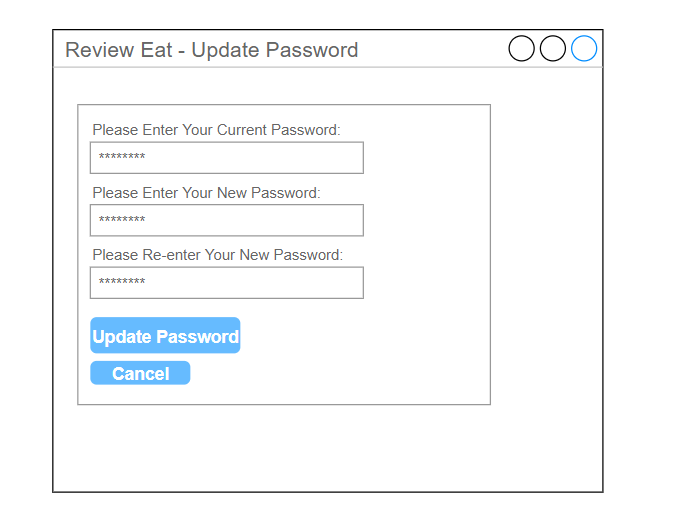


Figure 0‑7 Update Password Design

Unified Modelling Language (UML) provides a standard way to visualize the object-oriented design of the system. It uses diagrams to represent software components. Review Eat uses different UML diagrams during the design stage.

## Use Case Diagram

Use Case Diagram is used to represent the high-level function of the software. It visualizes the function, person or system that uses or interacts with that function and relationship between them. The functional requirements are called use cases and a person or system that uses the function are called actors.

**Determining the actors**: The software interacts with three central bodies. It includes user, admin and the database.

**The Use Case Diagrams:**

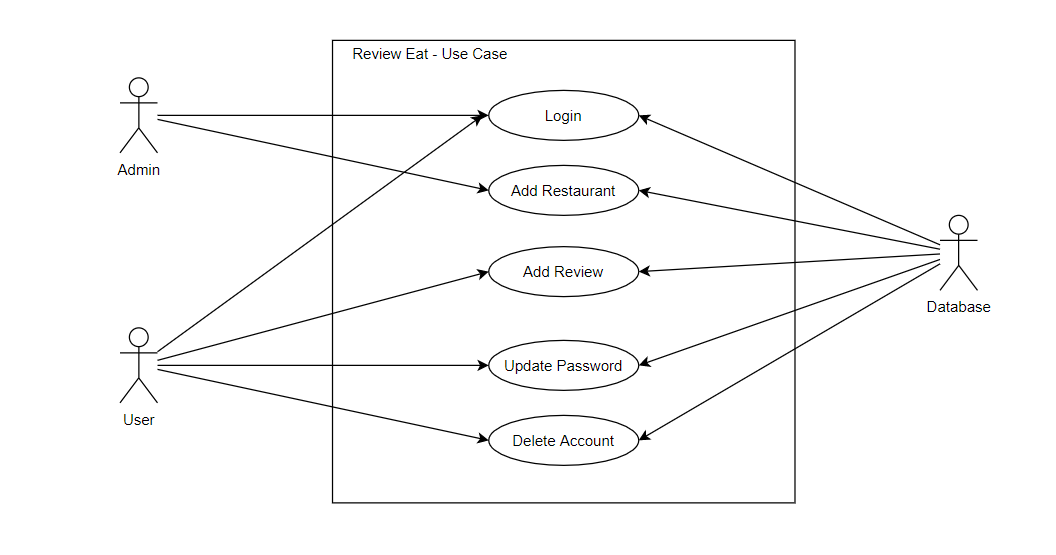


Figure 1‑0‑8 High Level Use Case Diagram

The initial Use Case Diagram is further broken down into smaller use case diagrams. Use case description is included with the diagrams.

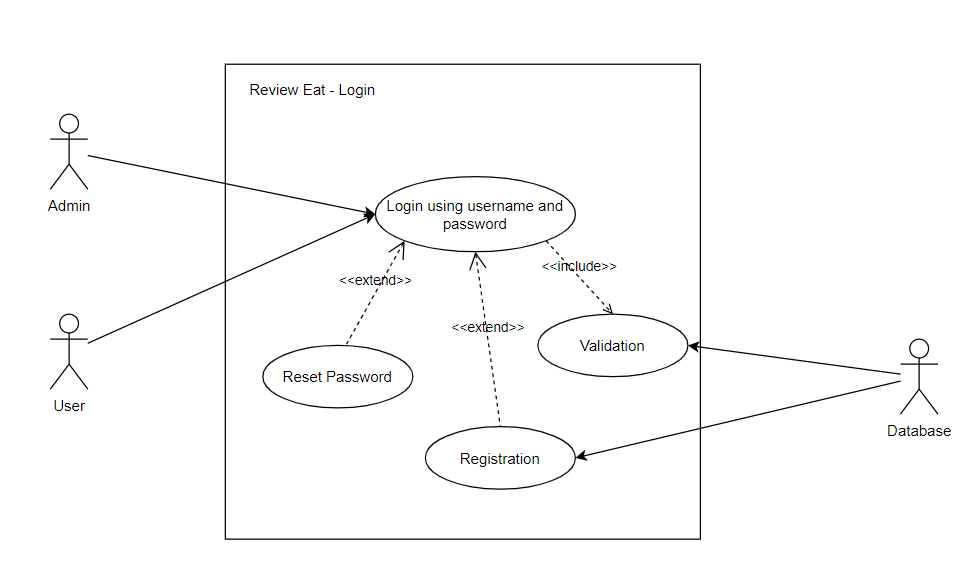


Figure 1‑0‑9 Login - Use Case

|  |  |
| --- | --- |
| **Use Case Name**: Login | |
| **Initiating Actor**: User/Admin | **Receiving Actor**: Database |
| **Trigger:** The User/Admin wants to log onto the system | |
| **Scenario 1**   1. User/Admin enters Username. 2. The system validates the Username. 3. User/Admin enters the password. 4. The system validates the password. 5. User/Admin is logged onto the system. | |
| **Alternative Scenarios**  2.a Username is not valid.  2.b Prompt user to register or re-enter the Username.  3.a Password is not valid.  3.b Prompt user to re-enter or reset the password. | |
| **Assumptions:** None | |
| **Post-Condition**: User/Admin is logged onto the system | |

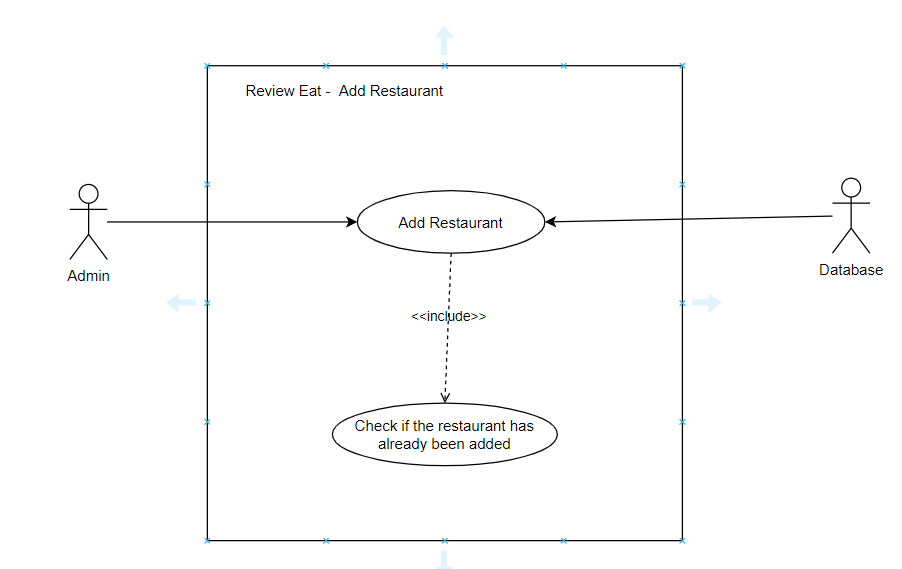


Figure 1‑0‑10 Add Restaurant - Use Case

|  |  |
| --- | --- |
| **Use Case Name**: Add Restaurant | |
| **Initiating Actor**: Admin | **Receiving Actor**: Database |
| **Trigger:** The Admin wants to add a restaurant. | |
| **Scenario 1**   1. Admin enters restaurant details. 2. The system confirms that the same restaurant has not been added to the system yet. 3. The restaurant is added to the system. | |
| **Alternative Scenarios**  2.a The restaurant has already been added.  2.b Notify user that they cannot add the same restaurant twice.  2.c Prompt user if they want to add another restaurant | |
| **Assumptions:** The admin is logged onto the system. | |
| **Post-Condition**: New restaurant has been added to the database. | |

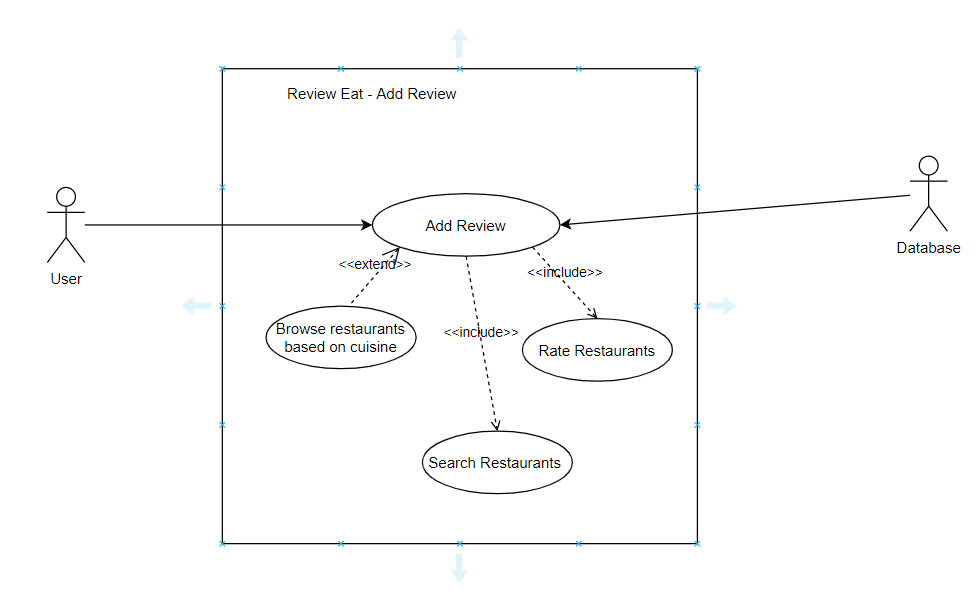


Figure 1‑0‑11 Add Review - Use Case

|  |  |
| --- | --- |
| **Use Case Name**: Add Review | |
| **Initiating Actor**: User | **Receiving Actor**: Database |
| **Trigger:** The User wants to add a review. | |
| **Scenario 1**   1. User searches for the restaurant. 2. The system gives back the search result. 3. The user writes a review. 4. User rates the restaurant. 5. A review is added. | |
| **Alternative Scenarios**  2.a The system cannot find the restaurant.  2.b Request admin to add the restaurant. | |
| **Assumptions:** User has logged on to the system. | |
| **Post-Condition**: Review is added. | |

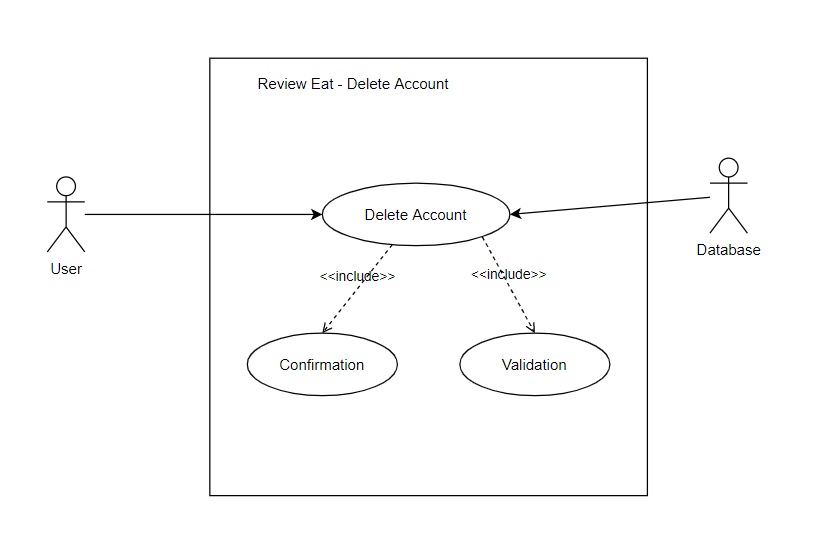


Figure 1‑0‑12 Delete Account - Use Case

|  |  |
| --- | --- |
| **Use Case Name**: Delete Account | |
| **Initiating Actor**: User | **Receiving Actor**: Database |
| **Trigger:** The User wants to delete their account. | |
| **Scenario 1**   1. The user selects delete account option. 2. The system asks for a password. 3. The system validates password. 4. The system prompts a message to confirm the user’s decision. 5. User’s account is deleted. | |
| **Alternative Scenarios**  2.a Password is not valid.  2.b Display message saying their account cannot be deleted.  4.a The user decides not to delete their account.  4.b The account is not deleted. | |
| **Assumptions:** User is already logged on to the system | |
| **Post-Condition**: User’s account is deleted. | |

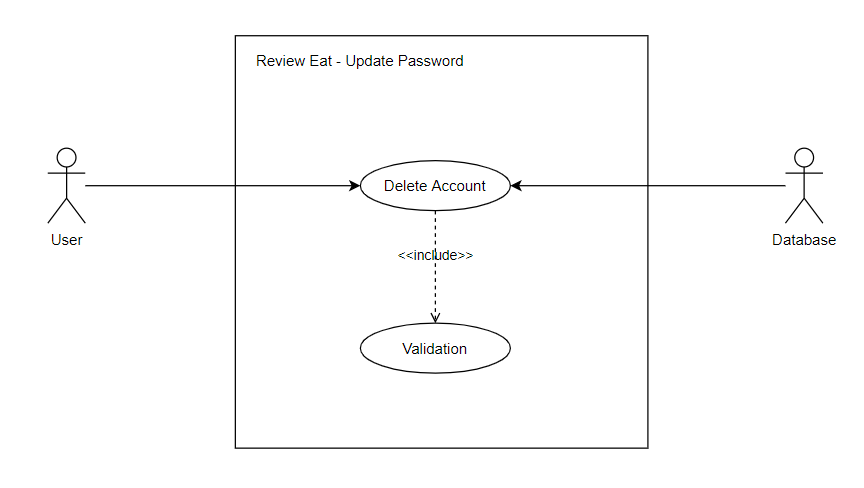


Figure 1‑0‑13 Update Password -Use Case

|  |  |
| --- | --- |
| **Use Case Name**: Update password | |
| **Initiating Actor**: User | **Receiving Actor**: Database |
| **Trigger:** The User wants to update their password. | |
| **Scenario 1**   1. The user selects the update password option. 2. The user enters their current password. 3. The system validates the password. 4. The user enters a new password. 5. User re-enters new password. 6. The system validates both password match. 7. The user selects update password. 8. Password is updated. | |
| **Alternative Scenarios**  2.a Password is not valid.  2.b Display message saying password cannot be changed.  6.a The password entered do not match  4.b Ask the user to enter passwords again. | |
| **Assumptions:** User is already logged on to the system | |
| **Post-Condition**: User’s password is updated. | |

## Activity Diagram

Activity Diagram is similar to flow charts, and it shows the flow of processes. It is used to expand use case scenarios and is based on the use case description. Unlike flowcharts, the concurrent flow of operations can be demonstrated in the activity diagram.

Activity diagram has been used to expand use case scenarios in this project.

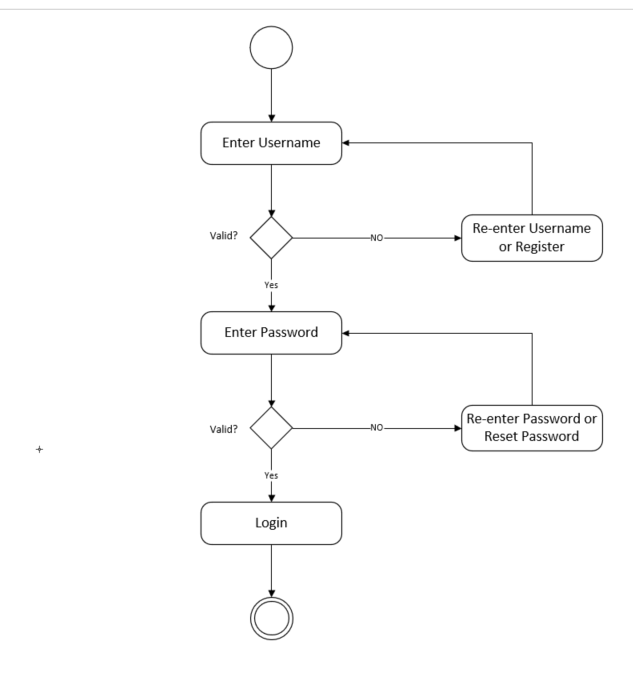


Figure 2‑0‑14 Login - Activity Diagram



Figure 2‑0‑15 Add Restaurant - Activity Diagram



Figure 2‑0‑16 Add Review - Activity Diagram



Figure 2‑0‑17 Delete Account



Figure 2‑0‑18 Update Password - Activity Diagram

## Classes Identificaiton

|  |  |
| --- | --- |
| **Classes** | **Description** |
| User | This class would contain all the attributes of the user. User will get username which are unique (Either they can choose their own or it will be generated by the system). A method to show details will be implemented. |
| Admin | A class will be created for the admin. It would contain admin username and password. |
| Restaurant | This class would contain the information about the restaurants. Restaurants will be given a unique ID. A restaurant would also contain a collection of reviews. Likewise, this class would also include a method to display all the information of the restaurant. A method to add reviews will also be implemented. |
| Login | This class will be used to verify the login. It would contain methods such as verification and registration. Verification would verify user’s username and password and allow them to login. Registration would allow new user to register to the system. |
| Review | Review class will contain attributes such as Review Description and Rating. It will also have a method to display the review. |
| Review Eat | Review Eat would contain collection of Restaurants. Admin will be able to add restaurant to Review Eat. |
| Main | This could contain the main method. |
| Address | Since User and Restaurants will have address as one of their attribute and address by itself would be very complex attribute. So, we can create a class Address which will have attributes such as Street Name, Post Code etc. |

## Class Diagram



Figure 3‑0‑19 Class Diagram

## Design Pattern Identification

Design Pattern are used to ensure that the code is highly cohesive and loosely coupled. Here are some of the design patterns that could be relevant to the software and are planned to be used.

### Strategy Pattern

In Strategy pattern, objects are created which represents various strategies and a context object whose behaviour varies just per its strategy object. (Tutorialspoint, 2020) This design pattern allows algorithm to vary without being dependent to the client.



The admin and user, both use the software. Hence, they are extended from a parent class. Likewise, Admin and User will have privileges but different privileges. Hence Strategy Pattern is introduced. This allows us to implement different algorithms (Figuring out if someone has an admin privilege) being independent to the client (In our case the clients are Admin and User).

### Iterator Pattern

In simple words, Iterator Pattern provides an iterator for collections which lets it iterate through its object but does not expose its implementation. (GeeksforGeeks, n.d.)



Here, we have Restaurant class that contains a collection of Reviews object. Likewise, ReviewEat class contains collection of Restaurants object. We want to iterator through the collection without exposing their implementation. Iterator pattern allows us to achieve that.

### Singleton Pattern

Singleton Pattern is a design pattern that is used to ensure that a class only has a single instance.

Here, DataSQLite is a class that controls the database. We would not want more than one instance of the database. Singleton Pattern allows only one instance of the class. Hence, singleton pattern would be a good choice here.

## Sequence Diagram

Sequence Diagram is used to show connection and interaction between internal objects of the system. It allows visualisation of internal working of the system.

Following Sequence Diagrams have been prepared based on good day scenario of various activities that are carried out by the system.



Figure 4‑20 Login - Sequence Diagram



Figure 4‑21 Add Restaurant - Sequence Diagram

Figure 4‑22 Add Review - Sequence Diagram



Figure 4‑23 Delete Account - Sequence Diagram



Figure 4‑24 Update Password – Sequence Diagram

## Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| SoftwareUsers | username: String | Stores username of the user | User1 |
|  | password: String | Store password of user | password1 |
|  | privileges: Privilege | Store usage privilege | notPrivileged |
|  | **Methods** | **Purpose** |  |
|  | areYouAdmin: Boolean | Checks if the user has admin privilege or not. |  |
|  | Getters and setters for username and password. | To achieve encapsulation |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| Admin (Child Class of SoftwareUsers) | adminUsername: String | Stores admin username | admin |
|  | adminPassword: String | Store admin password | nimda |
|  | **Methods** | **Purpose** |  |
|  | Override setters from SoftwareUsers | Admin will have unique username and password unlike regular user. The setters are overwritten for this purpose |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| User (Child Class of SoftwareUsers) | fName: String | Stores First Name of a user | Jack |
|  | sName: String | Store Second Name of a user | Sparrow |
|  | dob: LocalDate | Stores date of birth | 04/19/1998 |
|  | email: String | Stores email address of a user | helloyou@gmail.com |
|  | contact: int | Stores Contact Number | 0987654321 |
|  | address: Address | Stores address | 2/1  123  Hello Road  G11 1GG |
|  | **Methods** | **Purpose** |  |
|  | Setters and Getters | Encapsulation |  |
|  | displayDetails | Display all the details of the user |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Attributes** | | | **Purpose** | **Example** | |
| Address | flat: String | | | Stores flat detail | 1/1 | |
|  | houseNumber: int | | | Store house numebr | 12 | |
|  | streetName: String | | | Stores street name | Hello Road | |
|  | postcode: String | | | Stores Postcode | G11 1GG | |
|  | **Methods** | | | **Purpose** |  | |
|  | Setters and Getters | | | Encapsulation |  | |
|  | | displayDetails | Display all the details of the address | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| loginRegister | loginUsername: String | Stores username of current user | User1 |
|  | loginPassword: String | Store password of current user | password |
|  | **Methods** | **Purpose** |  |
|  | login | It validates the username and password of current user for login. |  |
|  | register | It allows new user to register to the system. |  |
|  | deleteAccount | It removes the current user from the database. |  |
|  | updatePassword | Allows user to update their password. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| DataSQLite | instance: DataSQLite | This is an instance variable of DataSQlite class. |  |
|  | **Methods** | **Purpose** |  |
|  | getInstance: DataDSQLite | This method makes sure that there is only one instance of DataSQLite. |  |
| **NOTE: This class would incorporate CRUD (Create, Retrieve, Update and Delete) functionality and will be responsible for communicating with the database.** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| Review | description: String | Stores Review description. | This restaurant is an excellent place to dine. |
|  | rating: int | Stores rating (On the scale of 10) | 9 |
|  | **Methods** | **Purpose** |  |
|  | Getters and Setters | Encapsulation |  |
|  | displayDetail | Display Details of the reveiw |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| Restaurant | restID: String | Stores an ID for a restaurant | Rest1001 |
|  | Name: String | Stores Name of the restaurant | The best Restaurant |
|  | restaurantDescriptioin: String | Store restaurant’s detail | This is a small sized restaurant that can accommodate 40 people at a time. |
|  | cuisine: String | Store the type of cuisine | Indian |
|  | contact: int | Stores Contact Number | 0987654321 |
|  | address: Address | Stores address | 2/1  123  Hello Road  G11 1GG |
|  | reviews: Hashmap<User, Review> | Store collection of reviews as key value pair using users as key and Review as value |  |
|  | **Methods** | **Purpose** |  |
|  | Setters and Getters | Encapsulation |  |
|  | addReview | Allows you to add review to the restaurant |  |
|  | averageRating | Finds the average rating of the restaurant. |  |
|  | createIterator: Iterator | Creates an iterator of so we can iterate through the reviews. |  |
|  | displayDetails | Display all the details of the user |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| reviewEat | restaurants: TreeSet<Restaurant> | Stores collection of restaurants in a binary tree. |  |
|  | **Methods** | **Purpose** |  |
|  | addRestaurant (Restaurant rest) | Allows to add restaurant to the collection restuarants |  |
|  | removeRestaurant (Restaurant rest) | Allows to delete restaurant from the collection restaurants |  |
|  | searchRestaurant (Restaurants rest) | Allow to search restaurant from the collection |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Purpose** | **Example** |
| InputValidationException {extends Exception} |  |  |  |

## Database Design

The data from the software needs organization and we use database for that purpose. But before we go out and construct a database, we need to come up with a model to organize those data.

Identified requirements are as follows:

* The database should store information about users.
* The database should store information about restaurants.
* The database should store reviews added by the users on restaurant.

Complying with requirements, following initial entities have been identified:



Figure 5‑25 Database Design Figure 1

The attributes have been also identified for each entity. All the attributes are compulsory.

### Normalisation

Normalisation is the process of breaking down the entity diagram to normal form to reduce redundant data. There are three degree of normalisation that should be carried out to come up with a design that reduces redundancy, duplication and dependency of data.

The current entity diagram contains attributes that contain multiple attributes of their own. Likewise, if we were to show a relationship between User and Restaurant, they will have a Many-to-Many relationship which is not acceptable.



Figure 5‑26 Database Design Figure 2

This is the final ERD of the database. Here, Address has been broken down into further attributes all of which are dependant on the primary key of the entity. All attributes are compulsory on both User and Restaurant apart from Flat (**i.e. Flat is an optional attribute**). Likewise, separate table has been created for reviews. Reviews is an intersection entity and has barred relationship. The UID of Reviews are derived from UID of User and Restaurant (i.e. username and restID). The attributes of Reviews entities are compulsory. When barred relationship is created, the initial problem of having a Many-to-Many relationship has also been solved. Similarly, the final ERD also shows the optionality and cardinality between relationships.

This is the fully normalised form of proposed database design. However, following assumption have been made when designing the database:

* One user can add review on one restaurant only one time. Same user cannot rate the restaurant more than once.

### Identifying Data Types

|  |  |  |  |
| --- | --- | --- | --- |
| **User** | | | |
| **Attribute** | **Datatype** | **Description** | **Null Value** |
| username | varchar | (Primary Key) Store unique username of user. It could be mixture of numbers and texts. | Not Accepted |
| Password | varchar | Stores password of the user. It could be mixture of numbers and texts. | Not Accepted |
| fName | Text | Stores First name | Not Accepted |
| sName | Text | Stores Second Name | Not Accepted |
| Contact | Int (size 10) | Stores Contact Number. The size is limited to 10 characters | Not Accepted |
| Email | Text | Stores email address of the user | Not Accepted |
| Flat | char | Stores Flat if the user has a flat in their address. | Accepted |
| Street Number | Int | Stores Street Number | Not Accepted |
| Street Name | Text | Stores Street Name | Not Accepted |
| Postcode | Char | Stores Post Code | Not Accepted |

|  |  |  |  |
| --- | --- | --- | --- |
| **Restaurant** | | | |
| **Attribute** | **Datatype** | **Description** | **Null Value** |
| restID | varchar | (Primary Key) Stores unique restaurant id. It could be mixture of numbers and texts. | Not Accepted |
| Name | Text | Stores name of the restaurant | Not Accepted |
| Cuisine | Text | Stores type of cuisine that the restaurant serves | Not Accepted |
| Contact | Int (size 10) | Stores Contact Number. The size is limited to 10 characters | Not Accepted |
| Description | Text | Stores description of the restaurant | Not Accepted |
| Flat | char | Stores Flat if the restaurant has a flat in their address. | Accepted |
| Street Number | Int | Stores Street Number | Not Accepted |
| Street Name | Text | Stores Street Name | Not Accepted |
| Postcode | Char | Stores Post Code | Not Accepted |

|  |  |  |  |
| --- | --- | --- | --- |
| **Review** | | | |
| **Attribute** | **Datatype** | **Description** | **Null Value** |
| Rating | int | Stores rating for the restaurant (On the scale of 10). | Not Accepted |
| Description | Text | Stores description of the review | Not Accepted |

This is the end of Planning Stage Documentation.

# References

GeeksforGeeks. (n.d.). *Iterator Pattern - GeeksforGeeks*. Retrieved from https://www.geeksforgeeks.org/iterator-pattern/

JReport. (2020). *3-Tier Architecture - A Complete Overview*. Retrieved from jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/

Legislation.gov.uk. (n.d.). *Data Protection Act 2018*. Retrieved from Data Protection Act 2018: http://www.legislation.gov.uk/ukpga/2018/12/contents/enacted

trip-10k\_20181231.htm. (2018, 12 31). *US Security and Exchange Commision, Tripadvisor, Inc.* Retrieved from https://www.sec.gov/Archives/edgar/data/1526520/000156459019003889/trip-10k\_20181231.htm

TripAdvisor. (2020). *The 10 Best Restaurants and Place to Eat in Glasgow 2020*. Retrieved from https://www.tripadvisor.co.uk/Restaurants-g186534-Glasgow\_Scotland.html

Tutorialspoint. (2020). *Design Patterns - Strategy Pattern - Tutorialspoint*. Retrieved from https://www.tutorialspoint.com/design\_pattern/strategy\_pattern.htm

Wikipedia. (2020, 03 17). *Yelp-Wikipedia*. Retrieved from Wikipedia: https://www.tripadvisor.co.uk/Restaurants-g186534-Glasgow\_Scotland.html

Yelp. (2020, 04). *The Best 10 Restaurant in Glasgow* . Retrieved from https://www.yelp.co.uk/search?cflt=restaurants&find\_loc=Glasgow

1. <https://www.tripadvisor.co.uk/Restaurants-g186534-Glasgow_Scotland.html> [↑](#footnote-ref-1)
2. <https://www.yelp.com/search?find_desc=fine+dining&find_loc=Glasgow%2C+United+Kingdom> [↑](#footnote-ref-2)
3. <https://www.theguardian.com/food/restaurants+tone/reviews> [↑](#footnote-ref-3)
4. <https://www.independent.co.uk/topic/restaurant-review> [↑](#footnote-ref-4)
5. <https://slideplayer.com/slide/10657813/> [↑](#footnote-ref-5)
6. <https://www.java.com/en/download/> [↑](#footnote-ref-6)
7. <https://gluonhq.com/products/javafx/> [↑](#footnote-ref-7)
8. <https://www.sqlite.org/index.html> [↑](#footnote-ref-8)
9. <https://archive.eclipse.org/eclipse/downloads/drops4/R-4.9-201809060745/#EclipseSDK> [↑](#footnote-ref-9)
10. <https://portal.azure.com/#blade/Microsoft_Azure_Education/EducationMenuBlade/software> [↑](#footnote-ref-10)
11. <https://app.diagrams.net/> [↑](#footnote-ref-11)
12. <https://products.office.com/en-gb/visio/flowchart-software> [↑](#footnote-ref-12)