Class: 1819\_GRLA08002\_01

Assignment: Software Engineering Assignment 1

Student Name: Grant Martin

Student Number: B00340010

University of the West of Scotland

HastEaters – Software Architecture Document

A context-aware eatery recommendation system

Purpose of document: This document should be readable at a technical level for software developers to build the HastEaters mobile app, communicating information in visual diagrams.

This document should also detail the scope and functional and non-functional scope of the app and be understandable to read by the customer product owner.

**Contents of Software Architecture Document for HastEaters mobile App**

[Introduction 4](#_Toc530159941)

[Purpose 4](#_Toc530159942)

[Introduction to the mobile Application 4](#_Toc530159943)

[Scope of HastEaters Mobile App 5](#_Toc530159944)

[Detailed HastEaters Mobile App Description 5](#_Toc530159945)

[HastEaters Mobile App Functionalities 5](#_Toc530159946)

[User Profile 5](#_Toc530159947)

[What the Product Contains 5](#_Toc530159948)

[Architectural Representation 6](#_Toc530159949)

[Logical View 6](#_Toc530159950)

[Process View 6](#_Toc530159951)

[Physical View 6](#_Toc530159952)

[Development View 6](#_Toc530159953)

[Architectural Goals and Constraints 7](#_Toc530159954)

[Main User Cases for the HastEaters Mobile Application 8](#_Toc530159955)

[Inputting Eating Preferences 8](#_Toc530159956)

[Recognize User Preferences 9](#_Toc530159957)

[Recommend A Restaurant 9](#_Toc530159958)

[15 Minute Restaurant Notification 10](#_Toc530159959)

[Calculate the top 3 Recommendations for the user 10](#_Toc530159960)

[Main Interface Requirements 10](#_Toc530159961)

[UI Designs for the mobile application and How the user will interact with the app 11](#_Toc530159962)

[Login and Registration Designs 11](#_Toc530159963)

[My Account and Homepage Designs 13](#_Toc530159964)

[Inputting Eating Preferences Designs 14](#_Toc530159965)

[Customers Stored Eating Preferences 15](#_Toc530159966)

[VISITED RESTAURANT DESIGNS 17](#_Toc530159967)

[Recommend A Restaurant Designs 18](#_Toc530159968)

[15 Minute Notification 20](#_Toc530159969)

[Logical Views 21](#_Toc530159970)

[Use Case View 21](#_Toc530159971)

[Actors 21](#_Toc530159972)

[Mobile Application Use Case 22](#_Toc530159973)

[Login and Registration Use Cases 23](#_Toc530159974)

[Inputting Eating Preference Use Case 23](#_Toc530159975)

[Recommend A Restaurant Use Case 24](#_Toc530159976)

[Package Diagram 25](#_Toc530159977)

[Process View 26](#_Toc530159978)

[Interaction Diagram: Create a New Account 26](#_Toc530159979)

[26](#_Toc530159980)

[Interaction Diagram: Login 27](#_Toc530159981)

[27](#_Toc530159982)

[Interaction Diagram: Inputting Eating Preference 28](#_Toc530159983)

[28](#_Toc530159984)

[Interaction Diagram: Recommend A Restaurant 29](#_Toc530159985)

[Development View 30](#_Toc530159986)

[Physical View 32](#_Toc530159987)

[Deployment Diagram 32](#_Toc530159988)

[3 Tier Architecture 33](#_Toc530159989)

[Reasons for using a 3-tier architecture system 33](#_Toc530159990)

[3 Tier Architecture Diagram 34](#_Toc530159991)

[34](#_Toc530159992)

[References 35](#_Toc530159993)

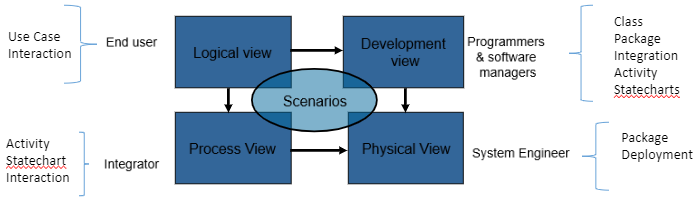
# Introduction

The software architecture document (SAD) will address how the HastEaters App high-level structure will be designed and implemented. [1] The app will be constructed by creating a certain number of architectural elements in practical ways to fulfil the crucial functionality and performance criteria of the system, furthermore the design and implementation will address non-functional requirements like reliability and scalability.

## Purpose

The software architecture document (SAD) will provide high level understanding and knowledge of the app to all members of the project. The document will display a series of diagrams and app designs to depict different elements of the system.

To convey this information about the proposed mobile application, the structure of this document is based on Philippe Kruchten’s “4+1” model view of software architecture. [1]



The “4+1” View Model allows different types of stakeholders with different degrees of technical ad user knowledge to find out what they need to know about the software architecture.

## Introduction to the mobile Application

HastEaters is a mobile app which will offer the user recommendations of where to eat lunch using the users live information such as location, to give the user the information of which eatery will result in the user having lunch and returning to work in the fastest amount of time.

# Scope of HastEaters Mobile App

## Detailed HastEaters Mobile App Description

The user interface for the HastEaters product will be a mobile app that will be able to run on the user’s mobile phone, both IOS and Android mobile devices. The application will access the user’s location on the phone and schedule of the user from the user’s calendar application (Google Calendar or Microsoft Outlook).

How will the mobile app work? Before the user’s lunch break, the app will notify the user with their lunch recommendations and the estimated roundtrip time for the restaurants displayed to the user.

What is the main purpose of the mobile app? The main purpose of the mobile app is to optimise the users working time and reduce the users time spend waiting in restaurants for lunch close to the user’s workplace.

## HastEaters Mobile App Functionalities

The mobile app has three main components:

* Functionalities for the mobile application:
* Register an account and log in
* Input eating preferences (User will only be able to have 3 preferences at any one time)
* Provide the user with a recommendation
* Notify user before lunchtime
* Non-invasive advertisement for product revenue
* Identify the location of the user
* Functionalities for the platform:
* Calculate the top three recommendations for the user
* Integrate with contextual providers
* Integrate with map providers
* Integrate with advertisement providers
* Connect to an external calendar application
* A basic user interface for maintenance and monitoring must be provided
* Secure access: web access to this UI must be provided

## User Profile

HastEaters is a mobile App aimed at hardworking mid to high level executive, who need to maximise their working day. The target market is profiled as fast paced, short attention span and require a service which is easy to navigate and allows them to optimise their time. The target user for this app will be busy mid to high level executives in the age range of 30 plus.

## What the Product Contains

* Non-invasiveness. The UI for the mobile app must be as minimal as possible.
* All calculations for routeing algorithms must be performed server-side
* The UI must provide non-intrusive yet targeted adds, as they are currently the sole revenue channel for the company.

# Architectural Representation

The views used to detail this document are shown and explained below, these views are taken from the “4+1” view model of software architecture:

## Logical View

**Typical Stakeholder**: End Users (all stakeholders of the app)

**Considers**: Describes the main requirements of the mobile app by representing the central functionality of the app. Introduces the actors to key functionality and how they interact with certain features of the mobile app.

**Supporting Diagram**: Use Case Models and Use Case documents

## Process View

**Typical Stakeholder**: Integrations

**Considers**: Non-functional requirements, the process of grouping tasks which form an executable unit.

**Supporting Diagram**: Interaction diagram

## Physical View

**Typical Stakeholder**: System Engineer

**Considers**: The mapping(s) of the software onto the hardware and reflects its distributed aspect

**Supporting Diagram**: Package diagrams, Deployment

## Development View

**Typical Stakeholder**: Software Developers and Managers

**Considers**: How the code will be developed and organization of the software code.

**Supporting Diagram**: Class diagram, package diagrams, integration diagrams

# Architectural Goals and Constraints

This part of the document highlights the main goal of the mobile application and the constraints impacting the software architecture of the mobile application:

1. The system of the mobile application will be a proof of concept to aid a fully completed system to be built.
2. The system will communicate and interact with multiple third-party APIs: Google Maps API, Native phone calendar, Outlook/Gmail (to retrieve users work schedule), Google Awareness API. Furthermore, the document will define how the system interacts with these third-party systems/API’s, this will be one of the primary concerns of the entire software architecture.
3. The code base for the mobile application and system architecture will be designed to be clean and structured to allow system changes, to minimize the amount of code rework and backend service rework. The software architecture will achieve this by modularization and information hiding to isolate components which are more likely to change and maintenance from other parts of the system and code base.
4. It is expected that the user base will grow to 1000 users in the first three months and then at a 5% monthly rate.

1. Native applications will be implemented to increase performance in the user devices (naïve designs and applications will need to be considered for both IOS and Android)
2. No technological constraints on the server side are expected.
3. The app will be able to use google maps/google API for getting the user location.
4. The app will be able to use the mobile devices calendar application where the user stores their work schedule (google calendar/Microsoft Outlook)

# Main User Cases for the HastEaters Mobile Application

## Inputting Eating Preferences

**Description**: The user can complete their eating preferences in the app by completing a form and the forms will be stored within the app, as the user will be able to save their preferences. This form will consist of fields labelled: Dietary restrictions (text field), Allergy information (text field), type of restaurant (sit in or take away), Cuisine (dropdown list with options: Italian, Asian, Chinese, American, etc), type of foods (dropdown list with options: burgers, pizza, sandwiches, etc). Once the preferences have been submitted the app will display the user top preferences for lunch.

**Pre-requisites**:

* The user must have the app installed on their phone and the user must have set up an account on the app.
* The user must navigate to the preferences screen

**Conversational use case:**

1. The HastEaters app presents the user with the current preferences of the user (if the user has already submitted preferences in the past, the maximum number of preferences a customer can have is 3 at any one time).
2. The user selects (turn on or turn off) preferences
3. Periodically the mobile app requests the server to serve the current state of the preferences (will need to determine how often the app asks the server if the user has updated their preferences)

**Related use cases:**

* List the best matching restaurants (The app will return the top 3 restaurants for each user preferences)
* Recognize user preferences

## Recognize User Preferences

**Description**: The mobile app will have the ability to recognize when the user is in a restaurant and the app will update the user’s preferences to reflect this information. The mobile app will use google maps to recognize when the user is in a restaurant.

**Pre-requisites:**

* The user must have the app installed on their phone and the user must have set up an account on the app.
* The user must enable permissions to access GPS data on their mobile device.

**Requirements notes:**

Once the mobile app recognizes that the user has visited a restaurant, the app will tell the server and the server will store the restaurant the user visited and the date the user visited the restaurants.

Restaurants visited by the user will be taken and inputted into the recommendation algorithm but with lower weight than the user-driven inputted data.

**Related use cases:**

* The mobile app will list the best matching restaurants according to the users visited restaurants
* Input eating preferences

## Recommend A Restaurant

**Description**: The mobile app will recommend restaurants based on the user’s preferences, the users available time for lunch according to the user’s calendar and the total time for the round trip (including sit in or take away restaurants). User will be able to select 1 or many preferences and the app will return the top 3 restaurants for each selected preference.

**Pre-requisites:**

* The user must have the app installed on their phone and the user must have set up an account on the app.
* The user must navigate to the Recommendations screen

**Conversational use case**

1. The mobile app will present recommended restaurants sorted by lowest round-trip time.
2. Each recommendation must include the restaurant name, distance, round trip time and the current number of people at the restaurant.
3. If the user chooses a recommended restaurant, the app must provide the option to integrate the destination to the navigational application on the user’s mobile phone (this will be done using Google Maps)

**Related use cases**

* Light notification restaurant recommendation.

## 15 Minute Restaurant Notification

**Description**:

* 15 minutes before lunchtime, the system recommends a restaurant to the user through a notification in the mobile phone operating system (The mobile app will access the users calendar application).
* The recommended restaurants must fulfil the users inputted preferences
* From the native notification interface, the user will be able to swipe for a new recommendation, accept or discard the notification.

**Pre-conditions:**

* The user must have the app installed on their phone and the user must have set up an account on the app.
* The mobile app must have permissions to show notifications on the user’s mobile device

## Calculate the top 3 Recommendations for the user

**Description:**

* The algorithm for calculating the recommendation must take into consideration both live data and data stored in the system.
* Data stored in the system includes: user preferences, restaurants last visited, etc.
* Live date includes contextual data of the nearby restaurants including: average people at the time of calculation and a current number of people.
* Live data also includes GPS data to calculate the distance from the current location to the available restaurants.

**Restrictions**

* Calculations are intensive parts of the algorithm and must be performed in the server(s).

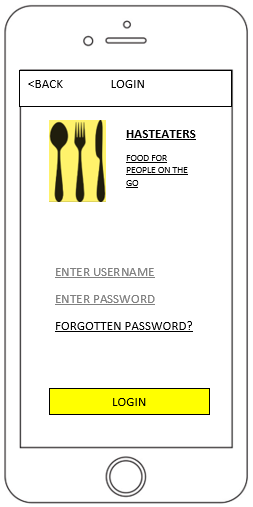
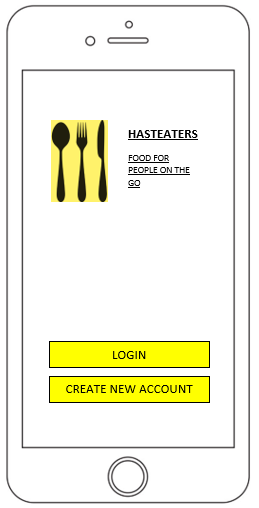
## Main Interface Requirements

The application must interact with the following external services.

* Google Maps
* Google Awareness API
* Native Calendar
* Native Maps
* Database

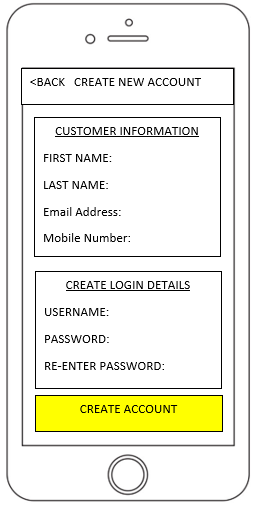
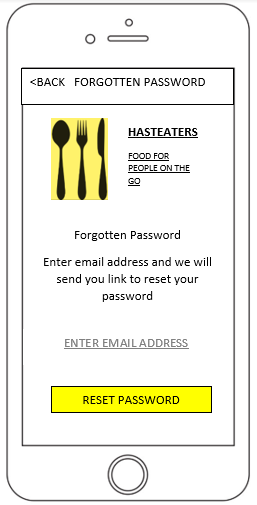
# UI Designs for the mobile application and How the user will interact with the app

## Login and Registration Designs



The login page of the mobile app, the user must complete the mandatory fields username and password, and these must match the login details stored in the database.

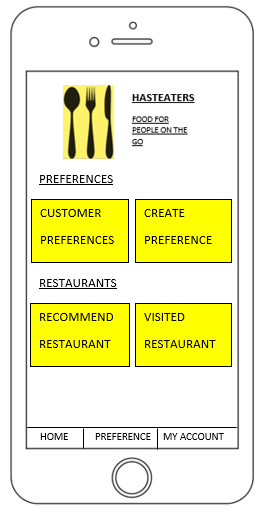
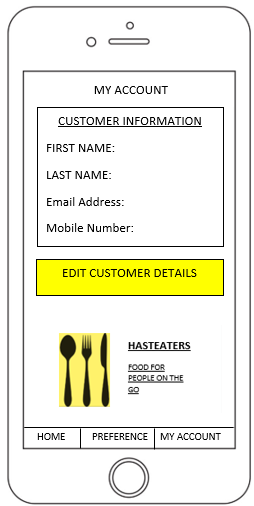
The first page of the app where existing users choose to login and new users can select to register for the mobile application.



Create new account page, all the fields are mandatory, and the details will be stored in the database.

The forgotten password page where users can reset their password by sending reset email to the user email address.

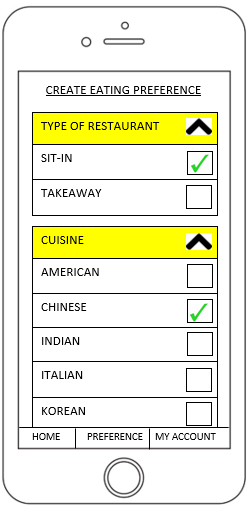
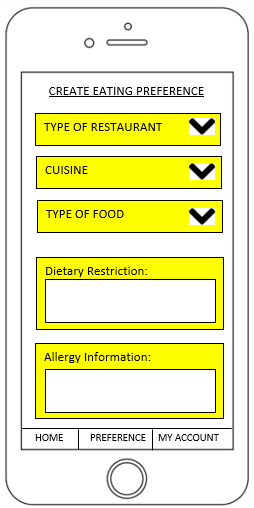
## My Account and Homepage Designs



The homepage of the mobile app, this will be the landing page of the app after the user successfully logs in. This page allows the user to navigate to all the features of the app. The bottom navigation page is on all pages contained within the app to aid usability and navigation.

My account page will allow the user to update their customers information at any time when using the mobile app.

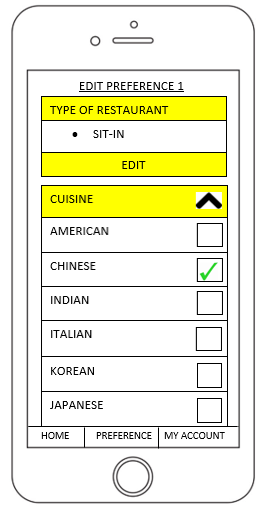
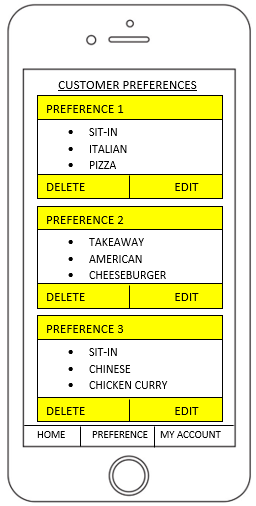
## Inputting Eating Preferences Designs



This image shows how the drop-down lists will operate. Type of restaurant/cuisine and type of food will have stored options for the user to select from.

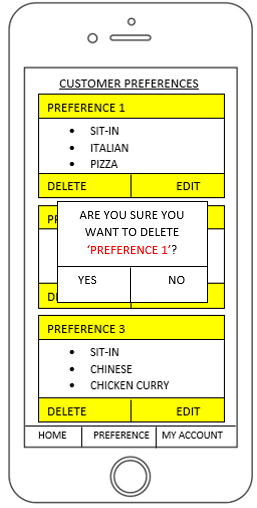
The user will be able to store 3 preferences at any one time in the mobile app. This image shows the fields to be completed for a user preference to be completed. The dietary restrictions and allergy information text fields are not mandatory, the rest of the fields are.

## Customers Stored Eating Preferences



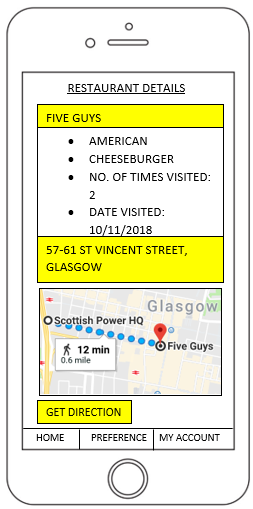
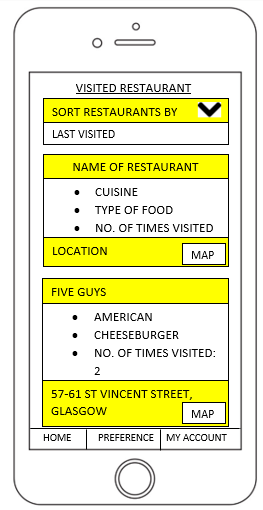
The edit user eating preference page.

The user will only be able to have a total of 3 eating preferences at any one time, so this page allows the user to edit or delete the stored eating preferences by the user.



The delete eating preference action form that will be presented if the user chooses to delete an eating preference.

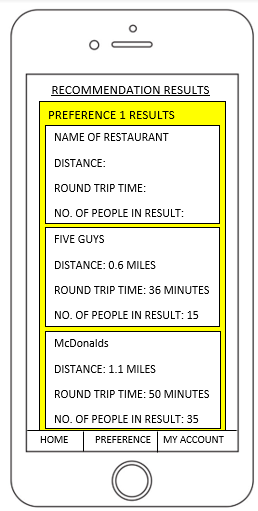
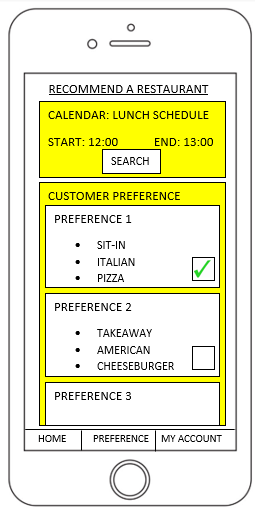
## Visited Restaurant Designs



The user can click on ‘Map’ to view the restaurant location using google maps and the user can then decide to get direction to the restaurant from their current location.

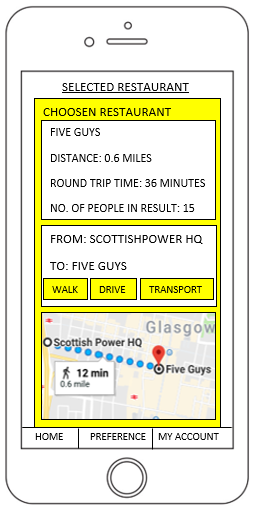
The app will store the users visited restaurants and details about the eating preference of that restaurant. The user will be able to view their visited restaurants and sort them by: last visited/A-Z/Z-A.

## Recommend A Restaurant Designs



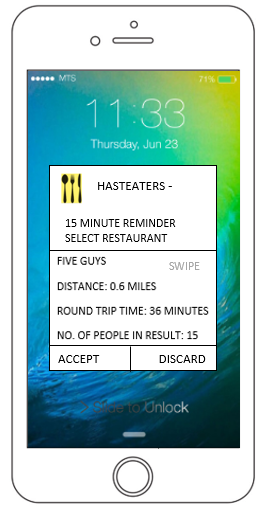
The results of the recommended restaurants will be presented on this page, the top 3 recommendations will be returned for the selected eating preferences.

On the recommend a restaurant page the user can manually decide to look for a restaurant for their lunch, by selecting a preference or multiply preferences and clicking search.



When the user selects a restaurant, they will be presented with this page.

## 15 Minute Notification



This is the 15-minute notification, the user can swipe to see a new recommendation and when they see a restaurant they want to go to, they will click accept and directions to the restaurant will be presented to the user.

# Logical Views

## Use Case View

The use case diagrams will give details of the features contained within the mobile application and how they interact with each other and which functionality they contain. The actors in the diagrams show how the internal systems and third-party API’s interact with the mobile application functional features.

## Actors

The users will drive the operations of the software. The App user and Server actors are primary actors and the Native Map app, Native schedule app/calendar, Google map, Google Awareness API actors are secondary actors.

App user: The app user will interact with the mobile application and kick start all application actions and systems. The user will login, input eating preferences and request recommendations based on the user’s location and schedule.

Native Map app: The native map application on the user’s phone will provide the users current location to will provide accurate restaurant recommendations.

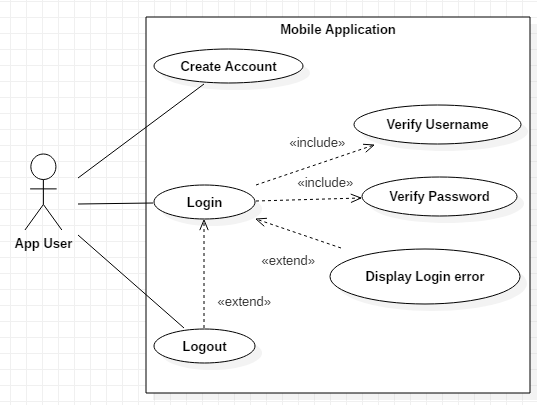
Native schedule app/calendar: The native calendar application on the user’s device will get the users lunch time schedule to help generate appropriate restaurant recommendations.

Google map API: Will be used to get the locations of restaurants in the user’s area that fulfil the users eating preferences, and once the API has the user’s current location and restaurant location the app can provide directions to the selected restaurant.

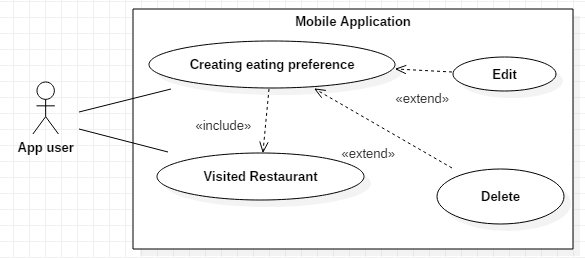
Google Awareness API: Will collect all the relevant information about visited restaurants and recommended restaurants, for example: type of restaurant, cuisine, type of food, current number of people in restaurant, average number of people in restaurant when the user is there. The information will be used in the calculation for recommending restaurants.

## Mobile Application Use Case

## Login and Registration Use Cases

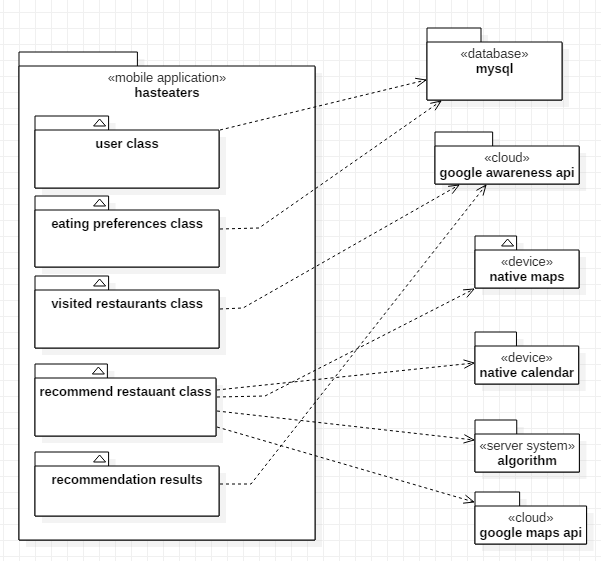


## Inputting Eating Preference Use Case



## Recommend A Restaurant Use Case

## Package Diagram



This package diagram shows the relationship of all the components used to operate the mobile app. The ‘mobile application’ package contains the classes which make up the working code of the mobile app. These classes contained within the ‘mobile application’ package make ‘get’ requests to the external packages: ‘google awareness api’, ‘google maps’, ‘native calendar’, ‘native maps’ and the information from these package are then returned to the appropriate class within the mobile application. Finally, once this information has been retrieved the ‘server system’ will calculate the recommend a restaurant algorithm to generate recommendation results.

# Process View

The process views will show the non-functional requirements of the mobile application, these diagrams and views will display the process of grouping tasks which will form the executable unit for the features contained within the mobile application. Furthermore, the process view diagrams will show the order of the component interactions and how the components interact with each other.

The following interaction diagrams will display how the components of the feature of the app interact with each other and in which order.

## Interaction Diagram: Create a New Account

## 

## Interaction Diagram: Login

## 

Else: invalid login

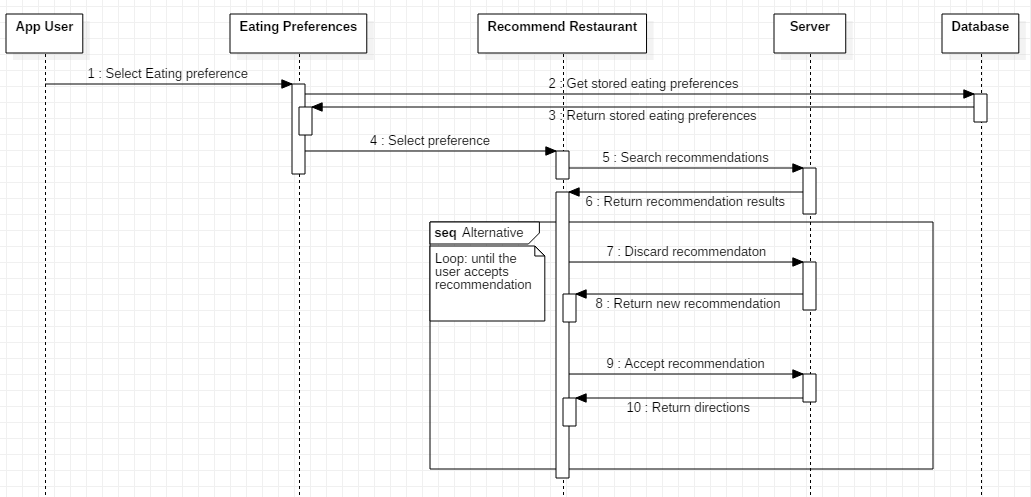
Valid Login

## Interaction Diagram: Inputting Eating Preference

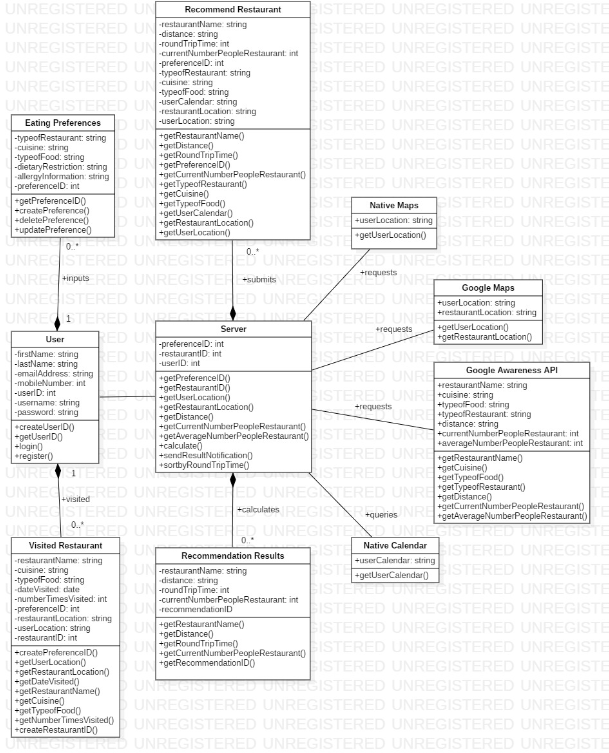


## 

## Interaction Diagram: Recommend A Restaurant



# Development View

The development view will explain visually how the code will be developed and the organization of the software code. The class diagram will depict the name of each individual class and detail the attributes and operations of the class, the class diagram will also show how the classes will interact with each other and their relationships.

Relationships Explained:

* The user ‘inputs’ their ‘eating preferences’ and the user can have 1 or many eating preferences (the user can input a maximum of 3 preferences). The ‘eating preference’ is dependent on the user creating an eating preference.
* The user must physically visit a restaurant for the app to recognize the user is in a restaurant and for the app the store details about the visited restaurant. The ‘visited restaurant’ class is dependent on the user class and the user can visit none or many restaurants.
* The ‘recommendation result’ is dependent on the ‘server’ class performing the algorithm calculation to generate the recommendation results.
* The ‘recommend restaurant’ is associated with the ‘server’ class as the ‘recommend restaurant’ class submits the preference form for restaurants to searched as part of the server calculation for recommending a restaurant.
* The ‘native maps’, ‘native calendar’, ‘google maps’, ‘google awareness API’ classes are all linked to the server class as the ‘server’ class collects information for these classes to perform the calculation.

# Physical View

The physical view depicts through the deployment diagram below the mapping(s) of the software onto the hardware and reflects its distributed aspect. This gives an overall view of the software and hardware used to build this app and how they interact with each other.

## Deployment Diagram

* Mobile Application: Contained within this package are the mobile application code base classes.
* Execution Environment: Refers to the operating system of the mobile devices the app will be built to support. The app will be developed for IOS and Android devices.
* Apache Web Server: Will deliver app content using Hypertext Transfer Protocol (HTTP) and will use many other modules to add functions to the software, also Apache will provide proxy services, digital certificates and much more.
* Application Server: will contain the calculation algorithm to recommend restaurants and the application server will contain the database system, which contains stored app information for example customer details and saved eating preferences.
* Google Maps API, Google Awareness API, Native Maps, Native calendar are not contained with the application server. They are external services that the application must interact with to retrieve information.

## 3 Tier Architecture

The mobile application will be built using a 3-tier architecture system. [According to Techopedia](https://www.techopedia.com/definition/24649/three-tier-architecture), [2] “3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.” [2]

The 3-tier architecture system contains 3 unique tier which address different areas and functional requirements of the mobile application. These 3-tiers are: Presentation, Application and Data.

1. The Presentation tier will deliver app content to the user interface in the form of HTML/JavaScript/Cascade Styling Sheets.
2. The Application tier will use the application server (Apache) to process the logic of the mobile application. The code for the mobile application will be written in Java.
3. The Data tier will control the database management system which will store the user details, eating preferences and user visited restaurants details. The database management system to be used will be MySQL.

## Reasons for using a 3-tier architecture system

1. [3] It will allow the developers to update features and technology stacks in one tier without effecting different tiers or areas of the app.
2. It will allow different development teams to work on different parts of the app at the same time without impacting each other, for example the front-end developer can work independently from the back-end developer.
3. This type of system construction will increase app reliability and provide increased independence of the main servers and services.
4. This system will provide easier maintenance of the code. [3]

## 3 Tier Architecture Diagram

## 

Third Tier

Database Layer

Second Tier

Application Layer

First Tier

Presentation Layer

# References

[1] Cs.ubc.ca. (2018). [online] Available at: https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf [Accessed 8 Nov. 2018]. [1]

[2] Izenda. (2018). *5 Benefits of a 3-Tier Architecture - Izenda*. [online] Available at: https://www.izenda.com/blog/5-benefits-3-tier-architecture/ [Accessed 8 Nov. 2018]. [2]

[3] Techopedia.com. (2018). *What is Three-Tier Architecture? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/24649/three-tier-architecture [Accessed 8 Nov. 2018]. [3]