DESIGN DOCUMENT

for

FOOD AGGREGATOR AND DELIVERY WEBSITE

Submitted by :
Amit Kumar (190001002)
Anikeit Sethi (190001003)
Jyotishna Baishya (190001021)
Yash Bontala (19000167)

May 12, 2021

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the implementation of the Foodkart(Food Aggregrator and Delivery Website). The Foodkart is designed to digitalize one stop food services and keep things smooth running. Foodkart allows customers to get take-out from any restaurant of their choice with just one single account and a few clicks. The site also brings small scale businesses to the forefront. Hence the application serves in mordernising the food services.

1.2 Scope

This document describes the implementation details of the Foodkart . The software will consist of a three major functions. First to implement ordering for Users based on their preferences, and the second is to take the orders and show on their order table of restaurant's,third is to perform the delivery from restaurant to user .

1.3 Overview of the Document

This document will contain the specifics of our application. It is a high level explanation of how the project will function and the various components developed to build the application. This document will also serve to explain the technologies used to develop the website as well as the user interaction with the various features.

2 Design Overview

2.1 Description of Problem

The aim of the project is to build a responsive online application that connects restaurants with customers across the internet. Along with this, the application also takes care of delivery of the orders from restaurant to customers. Hence, the application aims to tackle the entire process of ordering from a restaurant of choice, processing the order and delivering the order to customer's doorstep.

2.2 Technologies Used

The application is designed as a website. The implementation will use Django Framework with SQLite database as the back-end server. Firebase will be used as real-time database and for caching purposes. On the front-end HTML, CSS, Bootstrap and Javascript is used to create an interactive and responsive interface. The application also makes use of the Geolocation API to get the location of users and Mapbox API to show places on the interface.

2.3 System Architecture

Figure 2.1 shows a use-case diagram of Foodkart use case-model

In the below use-case there are three actors named User,Restaurant Management and Delivery Executive.Each actor interacts with a particular use case.A User can login,logout ,search for dishes or restaurant,order food,Keep the track of the order like if the order is confirm or in mid preparation or it is ready for delivery,they can also review orders,and keep track of the order's location.A Restaurant Manager can add dish,delete dish,update order status accept order.A Delivery Executive can view Deliveries and select deliveries and can also view their current position respect to user.It is not necessary that each actor should interact with all the use cases, but it can happen.

Further in the document, it explains the data models along with the associated functions which will give more understanding of this high-level explanation.

Another important part of a website is its interface with users. Users make requests to server through get, post, put, delete operations and using forms. Hence the document also explains the different forms use din the website to carry out these requests.

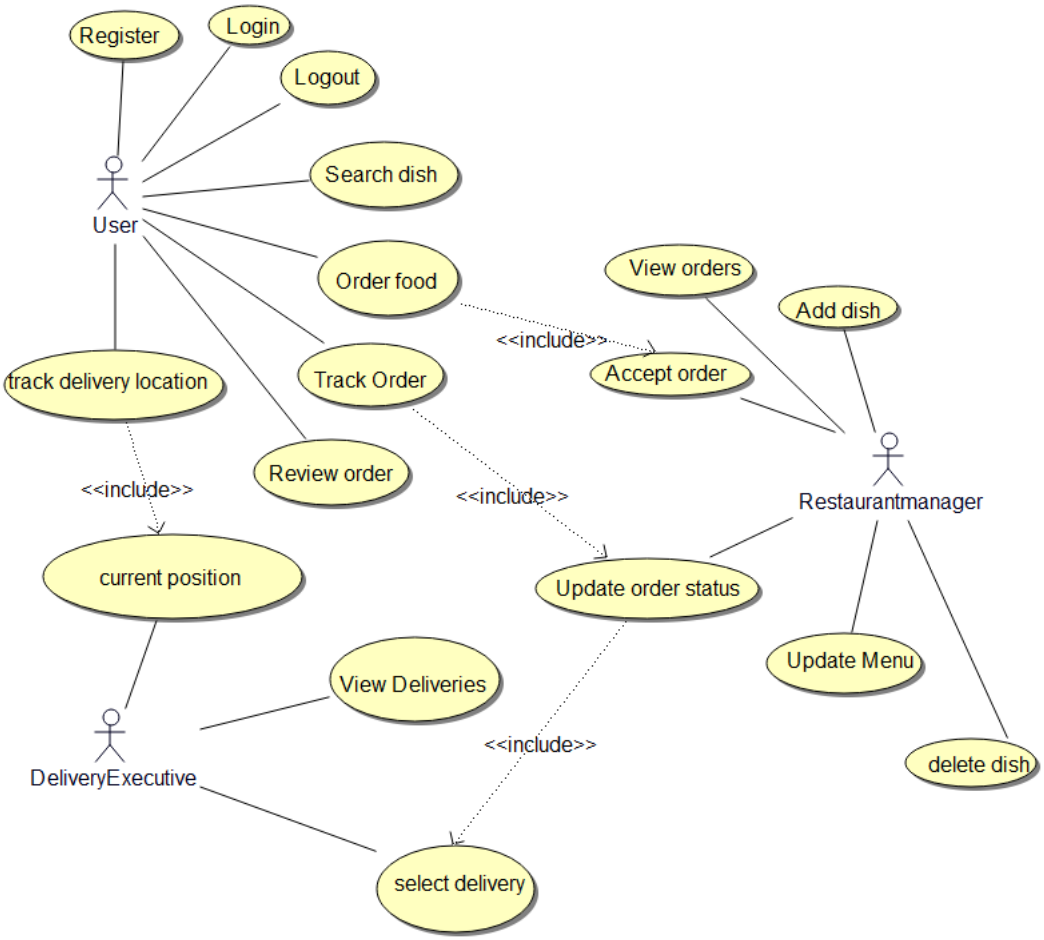

Figure 2.1: Usecase diagram

3 System Operation

Following figure demonstrates the typical flow of operations in the system.

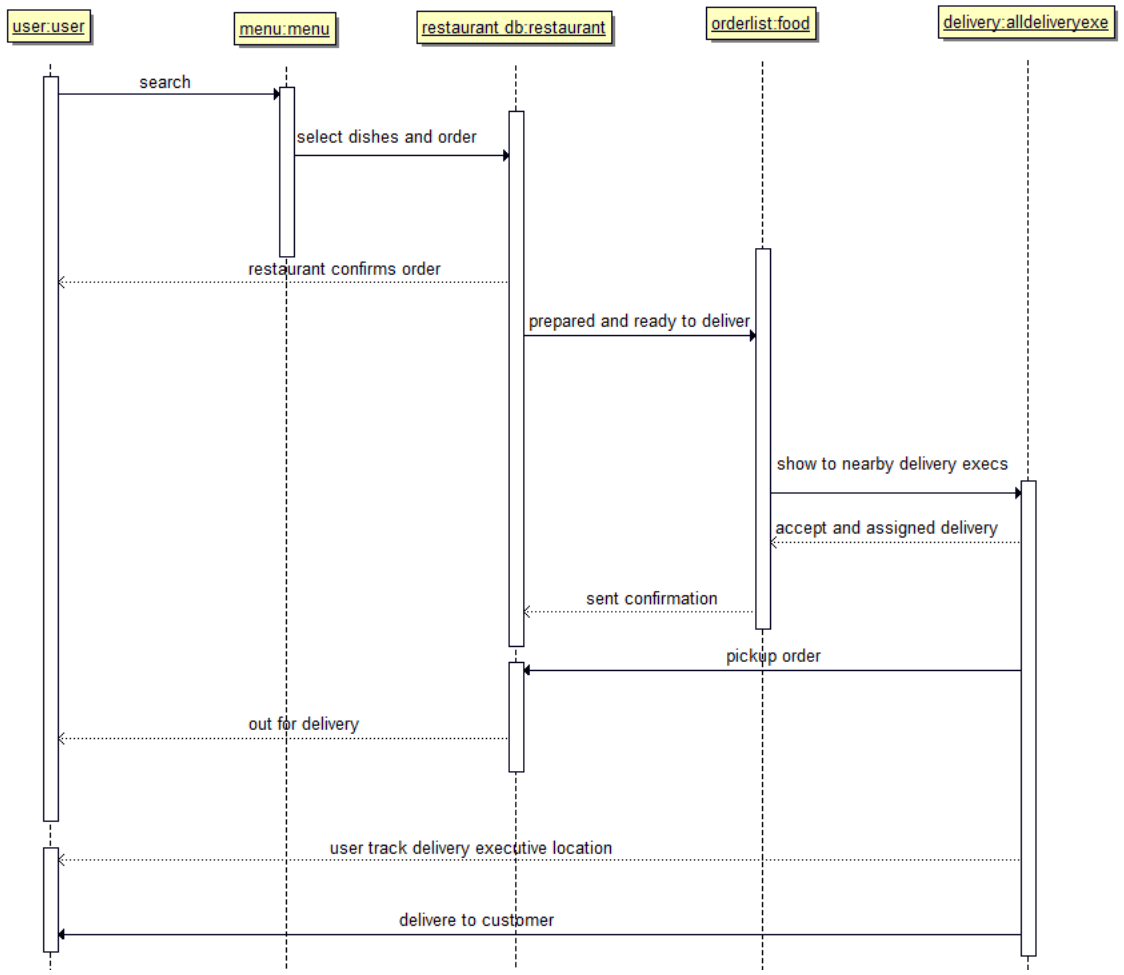

Figure 3.1: System Sequence Diagram

4 User Interface Design

4.1 Login form

Label title:label=Enter a valid user name. label1:label= username label2:label=password submit:button login_user:	inputbox1:inputbox inputbox2:inputbox

4.2 Register form

- **Customer**

label1:label= email	inputbox1:inputbox
label2:label= address	inputbox2:inputbox
label3:label= latitude	inputbox3:inputbox
label4:label= longitude	inputbox4:inputbox
label5:label= customerid	inputbox5:inputbox
label6:label= name	inputbox6:inputbox
submit:button	register_customer:bool

- **Restaurant**

label1:label= restaurant_name	inputbox1:inputbox
label2:label= owner	inputbox2:inputbox
label3:label= restaurant	inputbox3:inputbox
label4:label= mobile	inputbox4:inputbox
label5:label= email	inputbox5:inputbox
label6:label= address	inputbox6:inputbox
label7:label= latitude	inputbox7:inputbox
label8:label= longitude	inputbox8:inputbox
label9:label= registration_number	inputbox9:inputbox
submit:button	register_restaurant:bool

- **Deliveryexec**

label1:label= deliveryexecid	inputbox1:inputbox
label2:label= name	inputbox2:inputbox
label3:label= mobile	inputbox3:inputbox
submit:button	register_deliveryexec:bool

4.3 Add To Menu

label1:label= item	inputbox1:inputbox
label2:label=cuisine	inputbox2:inputbox
label3:label= restaurantid	inputbox3:inputbox
label4:label= vegeterian	inputbox4:inputbox
label5:label= price	inputbox5:inputbox
submit:button	add_to_menu:void

4.4 Delete From Menu

label1:label= item delete_from_menu:void	inputbox1:inputbox submit:button

4.5 Edit Menu

label1:label=item	inputbox1:inputbox
label2:label=cuisine	inputbox2:inputbox
label3:label= restaurantid	inputbox3:inputbox
label4:label= vegeterian	inputbox4:inputbox
label5:label= updateprice	inputbox5:inputbox
submit:button	edit_menu: void

4.6 Search

label1:label=restaurant or food submit:button	inputbox1:inputbox search:bool

5 Food Ordering Data Model and Storage

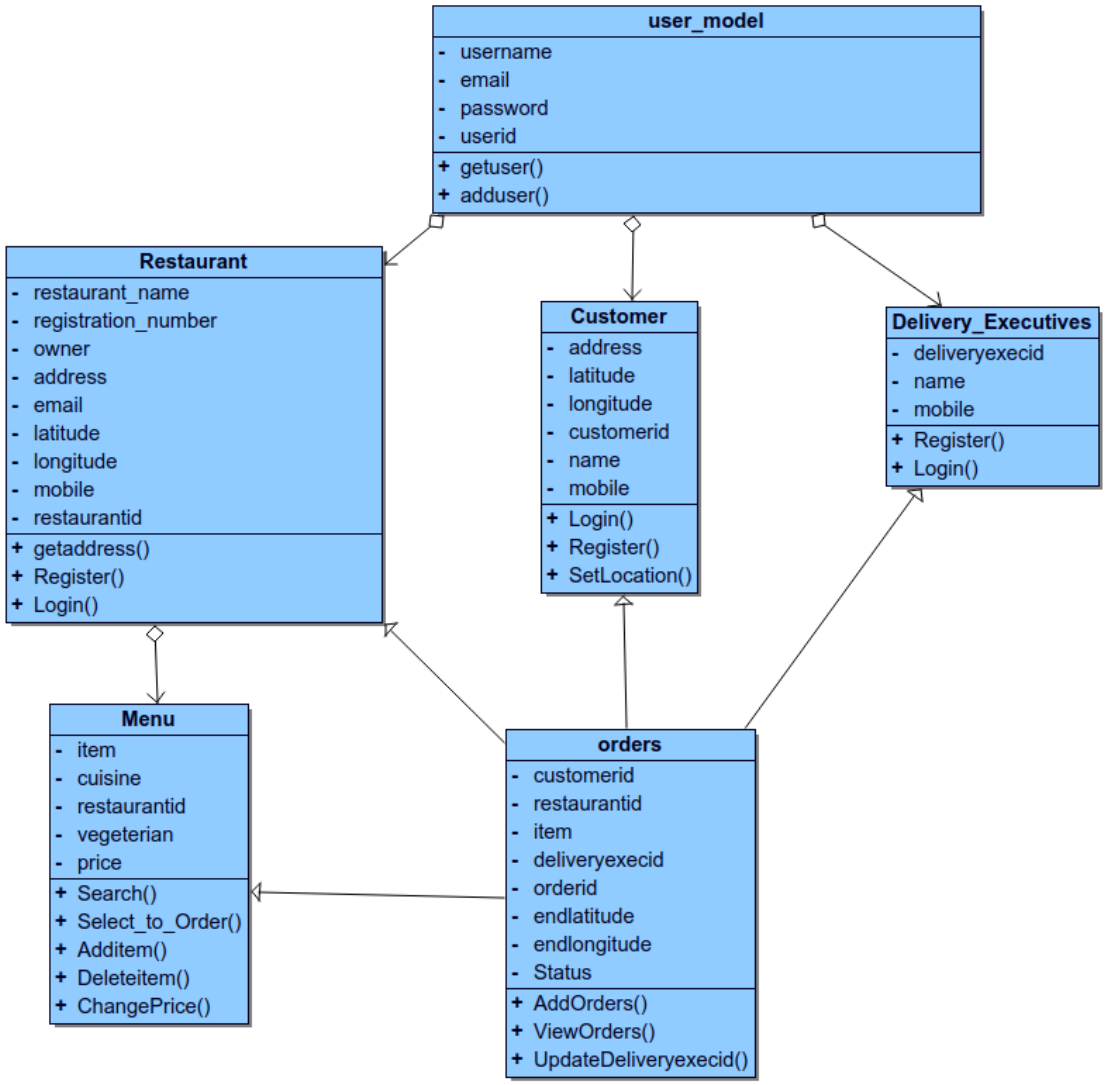

Figure 5.1: Food Ordering Data Model

5.1 user_model

5.1.1 Attributes

Name	Type	Description
username	string	stores the username of a user, can be null
email	string	stores the email of the user
password	string	stores the user password in encrypted form
userid	integer	an auto-incremented field that assigns an id to every user registered in database

5.1.2 Methods

getuser()	
Input	email
Return type	user instance matching the input email
Description	used for authentication purposes

adduser()	
Input	all user details
Return type	void
Description	used for registering

5.2 Restaurant

5.2.1 Attributes

Name	Type	Description
restaurant_name	string	stores the name of the restaurant
registration_number	string	stores the restaurant registration number
owner	string	stores owner name
address	string	address of the restaurant
email	string	official email
latitude	decimal number	latitude of the place
longitude	decimal number	longitude of the place
mobile	integer	contact
restaurantid	integer	foreign key referring to user_model

5.2.2 Methods

getaddress()	
Input	void
Return type	latitude and longitude of the restaurant
Description	used to locate restaurant

Login()	
Input	email
Return type	restaurant instance matching the input email
Description	used for authentication purposes

Register()	
Input	all restaurant details
Return type	void
Description	used for registering

5.3 Customer

5.3.1 Attributes

Name	Type	Description
address	string	address of the restaurant
latitude	decimal number	latitude of the place, can be null
longitude	decimal number	longitude of the place, can be null
name	string	stores name of customer
mobile	integer	contact
customerid	integer	foreign key referring to user_model

5.3.2 Methods

Login()	
Input	email
Return type	customer instance matching the input email
Description	used for authentication purposes

Register()	
Input	all customer details
Return type	void
Description	used for registering

setLocation()	
Input	latitude, longitude
Return type	void
Description	used to set the location of customer

5.4 Delivery_Executives

5.4.1 Attributes

Name	Type	Description
name	string	stores name of customer
mobile	integer	contact
deliveryexecid	integer	foreign key referring to user_model

5.4.2 Methods

Login()	
Input	email
Return type	delivery executive instance matching the input email
Description	used for authentication purposes

Register()	
Input	all delivery executive details
Return type	void
Description	used for registering

5.5 Menu

5.5.1 Attributes

Name	Type	Description
item	string	stores name of dish
cuisine	string	category of dish
restaurantid	integer	foreign key referring to user_model
vegetarian	bool	whether dish is vegetarian
price	integer	price of the item

5.5.2 Methods

Search()	
Input	dish name or cuisine
Return type	list of matching instances
Description	used to search for a dish by restaurant, name or cuisine

Select_to_Order()	
Input	Select items to order
Return type	all ordered items added to cart if confirmed
Description	used to slect items to order

Additem()	
Input	all details of the item to be added
Return type	void
Description	used to add new items to menu

Deleteitem()	
Input	item and restaurantid
Return type	delete the specified instance
Description	used to delete items from menu

ChangePrice()	
Input	item and restaurantid
Return type	void
Description	update the menu to set item to required price

5.6 Orders

5.6.1 Attributes

Name	Type	Description
customerid	integer	foreign key referring to Customer
restaurantid	integer	foreign key referring to Restaurant
deliveryexecid	integer	foreign key refereing to Delivery_Exectuives
item	vector¡string¿	vector of foreign keys referring to Menu
latitude	decimal number	latitude of customer place
orderid	integer	unique order number
longitude	decimal number	longitude of customer place
Status	string	current order status about procession and delivery

5.6.2 Methods

AddOrders()	
Input	all details of order except delivery statuses
Return type	void
Description	used to add new orders

ViewOrders()	
Input	any or all id
Return type	all order instances matching parameter
Description	used to view the orders by all users accordingly

updateDeliveryexecid()	
Input	delivery executive id who delivered the order, orderid
Return type	void
Description	update the delevery executive in the order delivered by them

6 Food Delivery Data Model and Storage

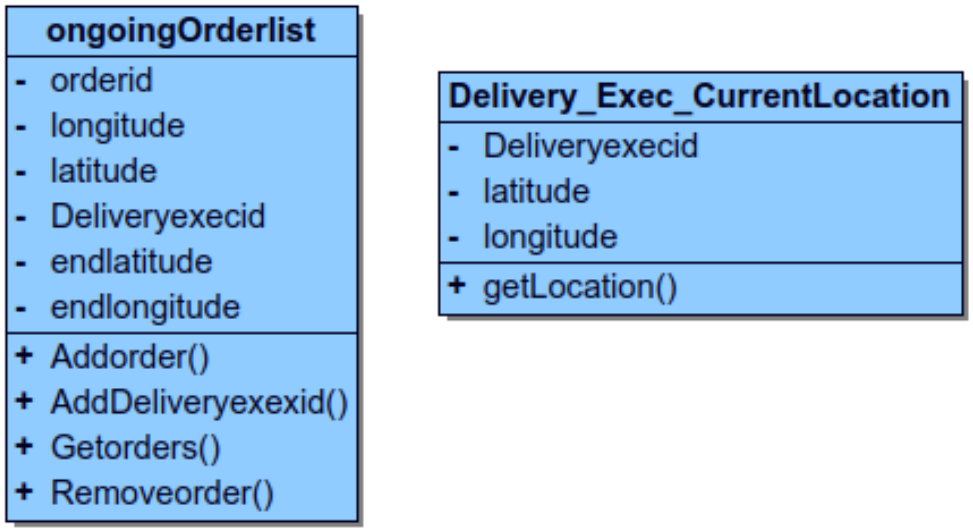

Figure 6.1: Food Delivery Data Model

6.1 ongoingOrderList

This part of system uses the firebase real-time database in order to update information of orders quicker.

6.1.1 Attributes

Name	Type	Description
orderid	integer	unique id alloted to all orders
longitude	decimal number	restaurant latitude
latitude	decimal number	restaurant longitude
Deliveryexecid	integer	delivery executive unique id
endlatitude	decimal number	customer latitude
endlongitude	decimal number	customer longitude

6.1.2 Methods

Addorder()	
Input	orderid, latitudes and longitudes
Return type	void
Description	adds an order to list to find a driver

AddDeliveryexecid()	
Input	delivery executive id who delivered the order, orderid
Return type	void
Description	update the delevery executive in the order delivered by them

Getorders()	
Input	get all the orders to be delivered
Return type	list of orders
Description	to read currently not delivered orders

Removeorder()	
Input	orderid
Return type	void
Description	delete from the list the specified orderid once delivered

6.2 Delivery_Exec_CurrentLocation

6.2.1 Attributes

Name	Type	Description
Deliveryexecid	integer	delivery executive unique id
latitude	decimal number	delivery executive latitude
longitude	decimal number	delivery executive longitude

6.2.2 Methods

getLocation()	
Input	delivery executive id
Return type	Location
Description	Used to get current location of driver