Takeaway.com x Saxion

PRODUCT PROPOSAL – GROUP 4

Jane Nguyên
DHI2V.SQ

Table of Contents

Project Description

Since last year, Takeaway has its own couriers delivering food for restaurants that do not have in-house delivery service, e.g. Burger King, McDonalds, etc. The client would like to narrow the communication gap between the drivers and the consumers by mean of an application.

For this project, Takeaway is expecting a text-chat based for Android platform, which will allow the courier to directly message the customers to receive additional directions, and also for the customer to communicate with the driver as well.

Deliverables

- An application:
    - That support the communication between couriers and customers
    - That show the status of the users
    - That allows couriers to mark orders as delivered
    - That cache messages when network is unavailable, and send the cached messages when connection becomes available again
    - That supports multi-languages
    - That allows customers to create an account directly from the app
- A database:
    - That stores the information of customer's accounts
    - That stores the information of courier's accounts
    - That stores the information of order's details

User Stories

Code	Title	User story	Acceptance criteria	Priority
US1	Single order	As a customer, I want to place a single order via the app, so that I know the app is functioning and good for use	• The order is successfully placed after payment • The order is delivered • Delivery person marked the order as completed	Must
US2	Multiple orders	As a customer, I would like to place multiple order from different restaurants at the same time, So that it is less time-consuming waiting for one order to be finished before placing another one	• New order can be created after right after placing an order • Orders are delivered as planned • The deliverer marked order as delivered	Could
US3	Send and receive messages	As a courier or customer, I want to be able to send and receive messages, So that I can communicate when it is needed	• Messages sent and is received by the recipient	Must
US4	Send and receive photo	As a courier or customer, I want to be able to send and receive photos, So that I can provide the courier a description of where the delivery destination is, or I can provide the customer of the received condition of the delivery	• When photos are sent and received by the recipient • If network is not available, photos will be cached and sent when it became available again	Could
US5	Sign up via application	As a customer, I would like to be able to create and account directly from the app, So that the experience of using the app goes smoothly, without the need to change between a web browser and the app	• Account is successfully created. • `Customer` can successfully login to his/her account afterward	Should

Code	Title	User story	Acceptance criteria	Priority
US6	Visible account status	As a customer or courier, I would like to be able to see the status of the courier or customer, So that I am aware if the messages is getting to the recipient or not	• Online or Offline status is indicated in the chat interface	Must
US7	Set account status as needed	As a courier, I would like to freely set my status to online or offline, So that I (courier) can choose when to receive order or to take a minor break from delivering	• After setting status to online/offline, status should be indicated as that	Should
US8	Support multi languages	As a user, I want the app to support multiple languages, So that I can easily understand what I am looking at	• The overall language of the app should be changed accordingly to the language chosen • The overall language should be changed accordingly to the language of the phone	Should
US9	Mark order as completed	As a courier, I need to be able to mark an order as completed after I delivered, So that I can take the next order	• After marking the order as completed, order status should change to delivered • Completed order should be add to courier's list of orders	Must
US10	Add tips to order	As a customer, I would like to have the ability to add tips to the order, So that I can add tips if I am satisfied with the service	• After `courier` marked order as complete, `customer` can choose whether to add tips to the order or not • Tips should be added to the total of the order	Could

Code	Title	User story	Acceptance criteria	Priority
US11	Leave ratings for restaurants	As a customer, I would like to leave a rating for restaurants, So that I can express my opinion on the service	• After order is completed, `customer` can choose to rate restaurant on a 1-5 scale • Rating is added to the restaurant data	Could
US12	Leave ratings for couriers	As a customer, I would like to have the option of leaving a rating for the courier, So that I can indicate whether I am satisfied with the service or not	• After delivery is completed, `customer` can rate `courier` on a scale from 1-5	Could
US13	Leave ratings for customers	As a courier, I would like to be able to leave a rating for customer, So that I can indicate the attitude of the customer and I can avoid taking order from low ratings customers	• After finishing the order, `courier` can rate `customer` on a 1-5 scale	Could
US14	Courier's account stay logged in	As a courier, I would like to stay logged in with the app, So that I do not have to miss any orders queuing up	• When closing and reopening the app, account of `courier` should stay logged in	Must
US15	Filter profanity out of chat	As a user, I want the app to filter out profanity, So that I can avoid unnecessary toxicity	• When profanity is use, the word will be replaced with *	Could

Project Design

Database Design

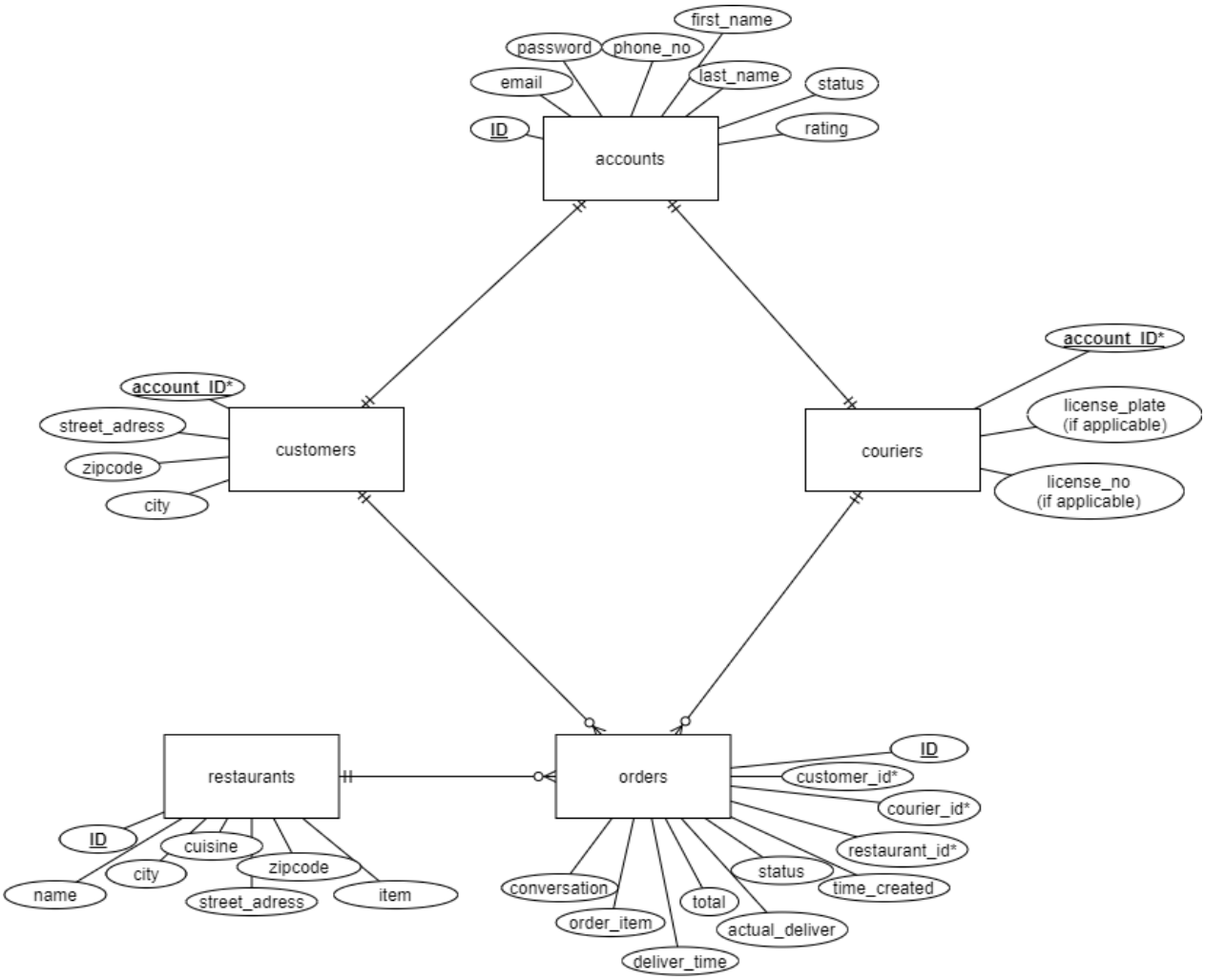


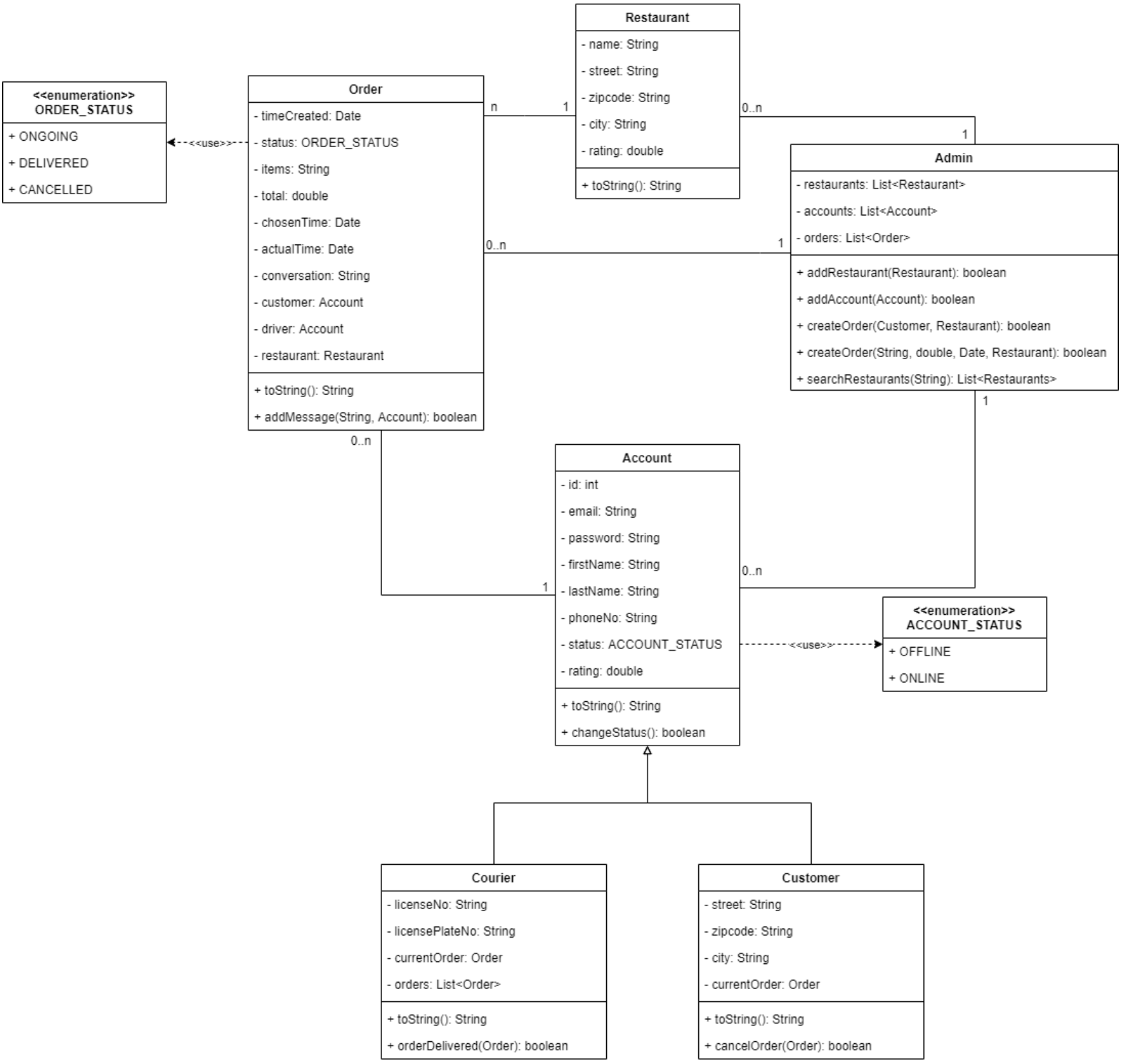
Figure 1. ERD of the system

Table	Attributes	Explanation
Accounts	- **<u>"ID": INTEGER</u>** - "email": STRING - "password": STRING - "phone_no": STRING - "first_name": STRING - "last_name": STRING - "status": STRING - "rating": DOUBLE	- As account is an important attribute for this project, it is to be expected to be able to store data on those accounts. - Each entity of this table will have a unique "ID". - Having an *account* table is necessary for users to be able to create an account for themselves.

Table	Attributes	Explanation
Couriers	- **“account_id”*: INTEGER** - “license_plate”: STRING - “license_no”: STRING	- As we created a plan for the project, we decided to make *courier* and *customer* an inheritance of *account.* - *Courier* will connect to *account* using “account_id” as a foreign key and a primary key.
Customers	- **“account_id”*: INTEGER** - “street_address”: STRING - “zipcode”: STRING - “city”: STRING	*Same as courier*
Orders	- **“ID”: INTEGER** - “customer_id”*: INTEGER - “courier_id”*: INTEGER - “restaurant_id”*: INTEGER - “time_created”: DATETIME - “status”: STRING - “total”: DOUBLE - “deliver_time”: DATETIME - “actual_deliver”: DATETIME - “order_item”: STRING - “conversation”: STRING	- The way we planned on establishing communication between courier and customer is through order. - When a customer placed an order, and when an order is assigned to a courier, they will automatically allow to send each other messages directly. - For this table, it will contain “customer_id” from *customer*, “courier_id” from *courier*, “restaurant_id” from *restaurant,* they will be used to establish connection between customer and courier.
Restaurants	- **“ID”: INTEGER** - “name”: STRING - “cuisine”: STRING - “street_address”: STRING - “zipcode: STRING - “city”: STRING - “item”: STRING - “rating”: DOUBLE	- As for this project, we are not focusing on the process of how the order is placed with the restaurants, we have simplified the models of *restaurants.* - Initially we had the table *items* and *order line* to connect the *restaurants* and *orders.* But as mentioned before, we decided to remove those tables, as we are mostly focusing on the communication of customer and courier. - “item” of the restaurant will be saved and a formatted String in the application as a demonstrative tool.

Application design

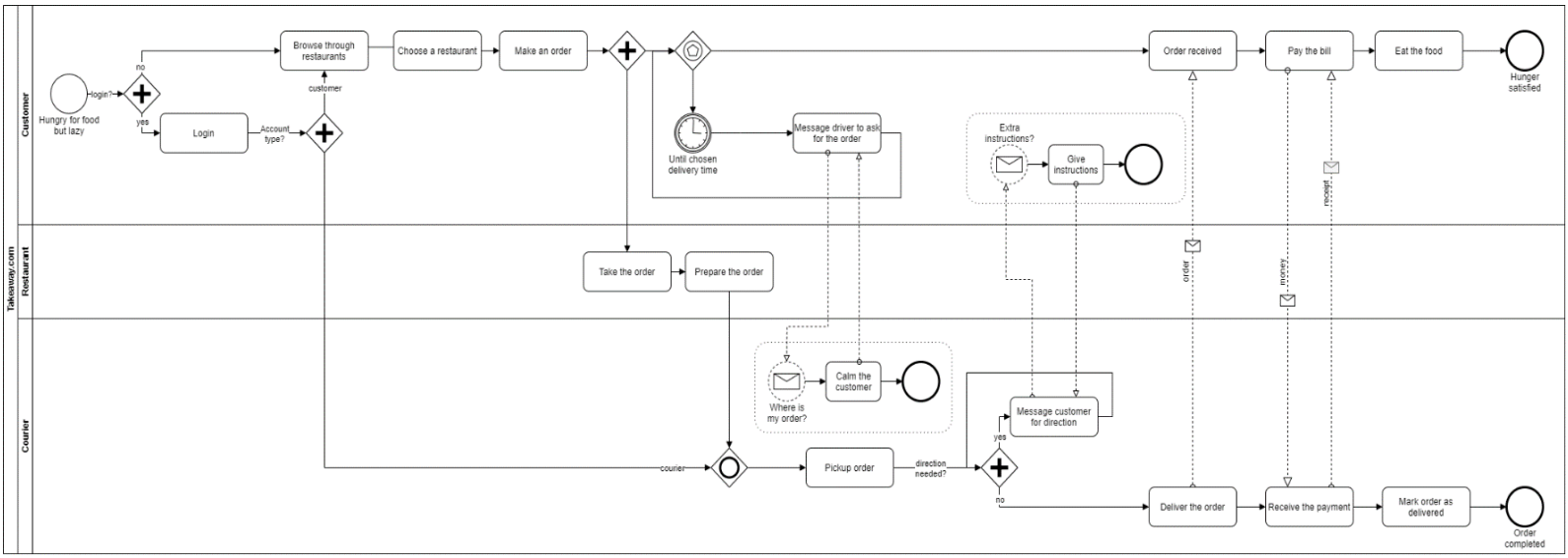
UML


Restaurant
- name: String
- street: String
- zipcode: String
- city: String
- rating: double
+ toString(): String
<<enumeration>>
ORDER_STATUS
+ ONGOING
+ DELIVERED
+ CANCELLED
<<use>>
Order
- timeCreated: Date
- status: ORDER_STATUS
- items: String
- total: double
- chosenTime: Date
- actualTime: Date
- conversation: String
- customer: Account
- driver: Account
- restaurant: Restaurant
+ toString(): String
+ addMessage(String, Account): boolean
n
1
0..n
1
Admin
- restaurants: List<Restaurant>
- accounts: List<Account>
- orders: List<Order>
+ addRestaurant(Restaurant): boolean
+ addAccount(Account): boolean
+ createOrder(Customer, Restaurant): boolean
+ createOrder(String, double, Date, Restaurant): boolean
+ searchRestaurants(String): List<Restaurants>
0..n
1
1
Account
- id: int
- email: String
- password: String
- firstName: String
- lastName: String
- phoneNo: String
- status: ACCOUNT_STATUS
- rating: double
+ toString(): String
+ changeStatus(): boolean
0..n
<<use>>
<<enumeration>>
ACCOUNT_STATUS
+ OFFLINE
+ ONLINE
Courier
- licenseNo: String
- licensePlateNo: String
- currentOrder: Order
- orders: List<Order>
+ toString(): String
+ orderDelivered(Order): boolean
Customer
- street: String
- zipcode: String
- city: String
- currentOrder: Order
+ toString(): String
+ cancelOrder(Order): boolean

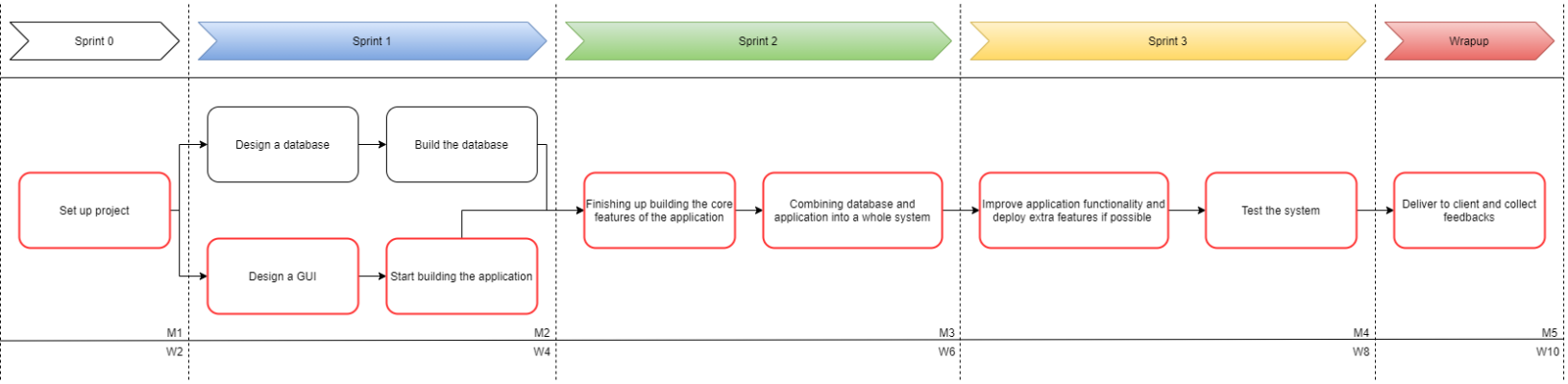
Class	Description	Methods explanation
Account	The Account class is the parent class to Courier and Customer.	+ toString(String): show general information of the account as a String + changeStatus(): change the status from ONLINE to OFFLINE, and vice versa
Courier	The Courier class is a child of Account, which means it will inherit all the attributes included in Account and its own attributes. This class will hold the attributes that only Courier's accounts would need.	+ toString(String): show general information of the account as a String. + orderDelivered(Order): After finishing delivering an order, couriers are expected to mark it as delivered, so that the order can be closed
Customer	Same as Courier, Customer is a child of Account. The class will hold the attributes only Customer's accounts would hold.	+ toString(String): show general information of the account as a String + cancelOrder(Order): customer can choose to cancel an ongoing order (e.g. order created by mistakes etc.)
Restaurant	The Restaurant class will hold general information of a restaurant.	+ toString(String): show general information of the account as a String
Order	For order, general information of the order (e.g. time created, customer, driver, etc.) will be stored.	+ toString(String): show general information of the account as a String + addMessage(String, Account): for this method, the parameter includes a message (String), and the sender (Account). We will store messages as a String to simplify the process of reading and writing the data
Admin	This class is for the overall function of the application (e.g. adding new accounts, open an order, order for customer without an account, etc.)	+ toString(String): show general information of the account as a String + addAccount(Account): this method will create a new account from the information passed over by the user. Also, only Customers can create an account directly from the application. Couriers will the to be given an account (to which they can change the password later) by the management

		+ searchRestaurants(String): when given a keyword (String), this method will return a list of Restaurants contains the keyword provided
ACCOUNT_STATUS	An enumeration to represent the account status (ONLINE/OFFLINE)	
ORDER_STATUS	An enumeration to represent the order's status (ONGOING/DELIVERED/CANCELLED)	

BPMN



Project Phasing



Sprint	User Stories to Achieve
0	NA
1	US1, US5, US6, US9
2	US3, US7, US14
3	US4, US10, US11, US12, US13
4	NA

Sprint 0: Initiation Phase

The initiation phase has only one activity, 'Set up project'.

<u>Activity: Set up project (2 weeks)</u>

1. Get critical information of the project (incl., but not limited to, asking questions about requirements, specify the scope of project, code of conduct, etc.)
2. Devise a plan
3. Set up meeting with the client to ask questions prepared
4. Create a rough draft to visualize the application
5. Create project plan
6. Create project proposal for client

Estimated duration for the first sprint of the project is 2 weeks.

Deliverables for **Sprint 0** are:

- Rough draft of Project Plan
- Code of conduct between the team members
- Visual of the application

M1 is to prepare a general plan for the project.

Sprint 1: Design and Implement

This phase is made up of 5 activities: design a database, an application, start implementing the basics of the database, the application, and finishing the first version of the project plan. For each activity, the tasks are as described below.

Activity: Design a database (4 days)

1. Draft up an ERD of the database required for the system
2. Research and decide which database will be used for the Android application
3. Finalize the design accordingly to the database chosen (Horizontal or Vertical DBMS)

Activity: Design the application (5 days)

1. Design the GUI
2. Discuss functionalities
3. Read through brand's identity, provided by the client, to visualize the theme
4. Finalize the design

Activity: Build the database (1 week)

1. Setup Firebase for Android application
2. Learn about different services will be implementing in the application
3. Read up on cloud messaging service and how to integrate into the app

Activity: Start building the app (1 week)

1. Install required software, if have not
2. Implement models
3. Design layouts for application

Activity: Finish the first version of Project Plan (3 days)

1. Put all the newly acquired information from the client and mentor to finish the first version of Project Plan
2. Draft out Technical Document

Estimated duration for sprint 1 is 2 weeks.

Deliverables for **Sprint 1** are:

- Project Plan for mentor
- Project Proposal (incl. UML, project phasing, ERD, etc.) for client

M2 is to review the Project Plan and draft up Technical Document.

Sprint 2: Implementation and Integration

For this sprint, there are 3 activities: finishing the application main features, and combining the database with the application, finishing documents.

Activity: Finish building main features of the application (10 days)

1. Implements necessary methods that were overlooked during the previous sprint
2. Make sure activities are well connected and not crashing in between
3. Check the User stories to see if all issues with Must haves' and Should haves' labels are closed
4. Runtime tests
5. Fix bugs
6. Update documents

Activity: Integrate application with database (4 days)

1. Combining the application and the database
2. Check for errors

Activity: Finishing documents (3 days)

1. Update the document accordingly
2. Go through the checklist to make sure everything is done.

Estimated duration for **Sprint 2** is 2 weeks.

Sprint 3: Review and Testing

The following phase has 3 activities: Implement selected extra features, test the system, make process report

Activity: Implement selected extra features

1. Assess the current situation of the project
2. Select features that can be integrate into the project without causing conflicts
3. Implement extra features

Activity: Test the system

1. Test the application's functionalities
2. Write tests for the system
3. Fixing system's vulnerabilities (if applicable) to enhance security
4. Test application with some users and get feedbacks

Activity: Make process report

1. Put minutes of meetings during the project together
2. Describe clearly work division
3. Finish individual reflections
4. Finish team reflections

Estimated duration for this sprint is 2 weeks

Deliverables for **Sprint 3** are:

- The fully functioning application
- The application with checked and resolved issues
- Final revision of the process report

M4 is for pre final demonstration of the application to the client and reviewing everything with the mentor.

Sprint 4: Wrap Up

This phase only has one activity, deliver the product to the client, and collect feedbacks

Activity: Deliver the system to the client, and collect feedbacks

1. Hand in required deliverables to the client
2. Instruct client of the functionalities, and giving demonstration on the usage of the application
3. Collect feedbacks from the client
4. If client's needs are not met, the team will use the remaining time to attend to the specified problems

Estimated duration is 2 weeks in total.

Deliverables for **Sprint 4** are:

- A working application

M5 is when the final version of the product is sent to the client.