University of Prince Mugrin

College of Computer and Cyber Sciences

Department of Software Engineering

SE323- Software Process and Modeling

Project – Semester I (Fall 2022)

RecycleMe- Recycling portal for unwanted items

Team Members:

Fatima Aljalmoud 4010096

Jana Aldubai 4010372

Salwa Shama 4010405

Samah Shama 4010403

Sana Shama 4010404

Instructor:

Dr. Khaled Khankan

18 December 2022

Chapter 1: Introduction

1.1 System Overview……….……..………………………..……...........................……..3

1.2 System Objective...…………………..…...……………………….....................………3

1.3 System Scope……………………………………..……...........................…...….3

1.4 Proposed Solution (System Functions and Features)……………………..…………4

Chapter 2: Software Requirements

2.1 Functional Requirements…………………………………………………………..4

2.2 Non-functional Requirements……………………………….……………………..4

Chapter 3: Software Analysis and Design Models

3.1 Use Case Diagram………………………………………………………………… 5

3.1.1 Use case description……………………………………………………………..7

3.2 Sequence Diagram………………………………………………………………… 8

3.3 Overall Architecture……………………………………………………………… 8

3.4 Component Diagram………………………………………………………….... 9

3.5 Initial Set of Objects……………………………………………………………... 9

3.6 Class Diagram………………………………………………………………………10

Chapter 4: Evaluating the Quality of Models

4.1 Validation of Models………………………………………………………………….. 11

Chapter 5: Conclusion and References

5.1 Conclusion……………………………………………………………………………. 14

5.2 References………………………...………………………………………………...15

Chapter 1: Introduction

1.1 System Overview

This system will be in the form of a mobile app that will be available to those who live in Medina. Using this system, materials can be easily recycled from reputable sources. The customer can place recycling orders and earn points for their transactions. Later, these points will be checked in the system and exchanged for offers. Besides, the customers will be able to see their levels on the leaderboard, in addition to their positive influence on the environment, which will raise people's social awareness and make it easier for them to achieve a high-quality environment. Further, the admin will be able to add the offers that will be exchanged with points by customers and generate a report about the app's statistics. For the application's delivery representative, the drivers will be able to pick up customers' orders to start the recycling life of the materials added by customers.

1.2 System Objective

The app aims to change people's attitudes and behaviors towards the concept of recycling, making it part of their lifestyle for a more sustainable future.

The app facilitates the recycling and utilization of waste rather than the process of disposing of it and opens the door for users to contribute to reducing the negative impact on the environment. The app provides users with financial rewards in the form of points and recycling discounts, in addition to providing leader boards to make them more competitive. This idea creates a competitive environment and encourages them to recycle their waste.

1.3 System Scope

All features of the system, including login, downloading a report, adding offers, the ability of customers to create orders and exchange points, alongside the system functionality of displaying orders to users, a leaderboard, and customer impact, are represented by the use case diagram, the software architecture diagram, the sequence diagram, and the component diagram.

In terms of the class diagram, it is limited to visualizing functionalities that are related to order processing, system users with authentication, and offering functionality.

1.4 Proposed Solution

To assist in the process of recycling the items, the application provides five main functionalities. First, deliver the items that need recycling by sending a driver to gather them. Second, receive points for each order created to deliver the items. Third, exchange the points with the offers provided in the application by requesting an offer. Fourth, display the positive impact on the environment after recycling the items. Finally, display a leaderboard that shows the top 50 customers who’s the most positive impact on the environment.

Chapter 2: Software Requirements

2.1 Functional Requirements

- The user shall be able to view orders with their details.
- The admin shall be able to download a report showing the statistics about the user and recycled item up to that day.
- The admin shall be able to add offers in the app.
- The customer shall be able to create an order.
- The customer shall be able to view the leader board that depends on the customer's impact on the environment. (User impact is measured by the amount of reduced CO2 per recycled item gram)
- The customer shall be able to view his/her own positive impact on the environment.
- The customer shall be able to view his/her points in the app.
- The customer shall be able to request offers by his/her collected points.
- The customer shall be able to add details in his/her order (Item's name, type, location, time, date, picture, statuses (sent, in progress, received), weight)
- The driver shall be able to select the customer's order.

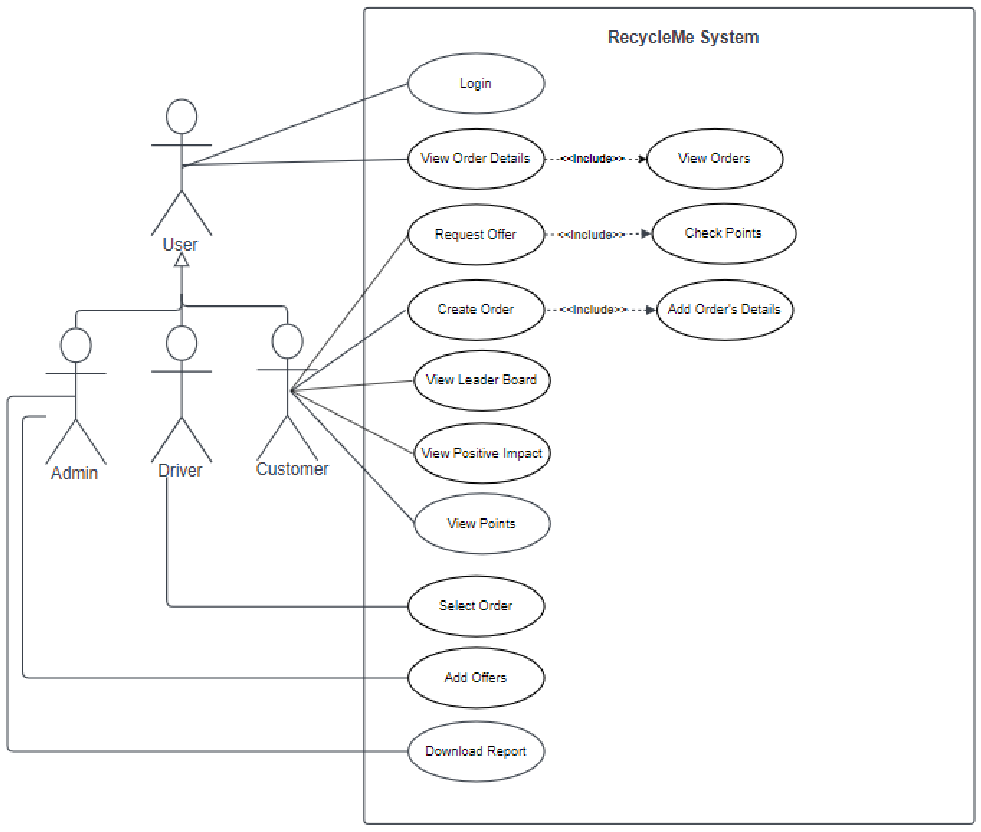
2.2 Non-functional Requirements

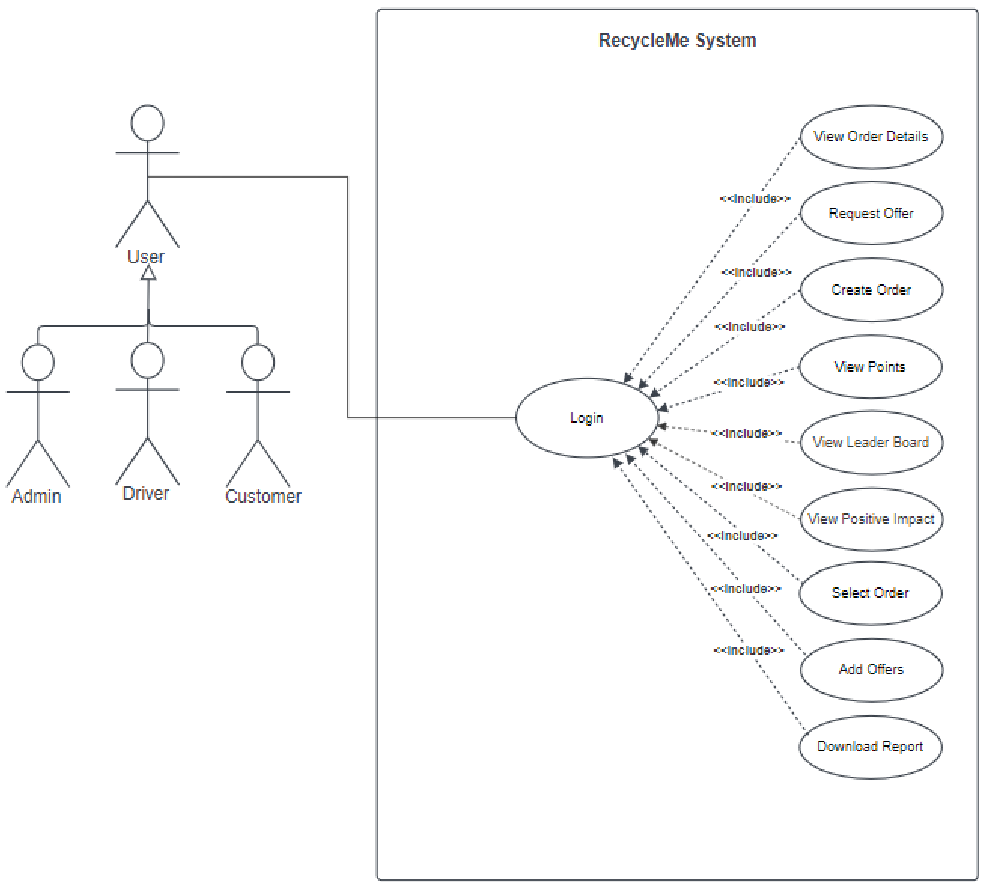
- The user shall be able to login after matching username and password. (security)
- The system shall be available 24/7. (availability)
- The order wight shall be more than or equal to 5kg. (Constrain)

- The system shall be able to respond quickly to any transaction in less than 10 seconds. (Performance)
- The user shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Usability)
- The system shall be able to reject any exchanging points process with not enough points. (Constrain)
- The customer shall not be able to see other customers' orders. (Privacy)

Chapter 3: Software Analysis and Design Models

3.1 Use Case Diagram

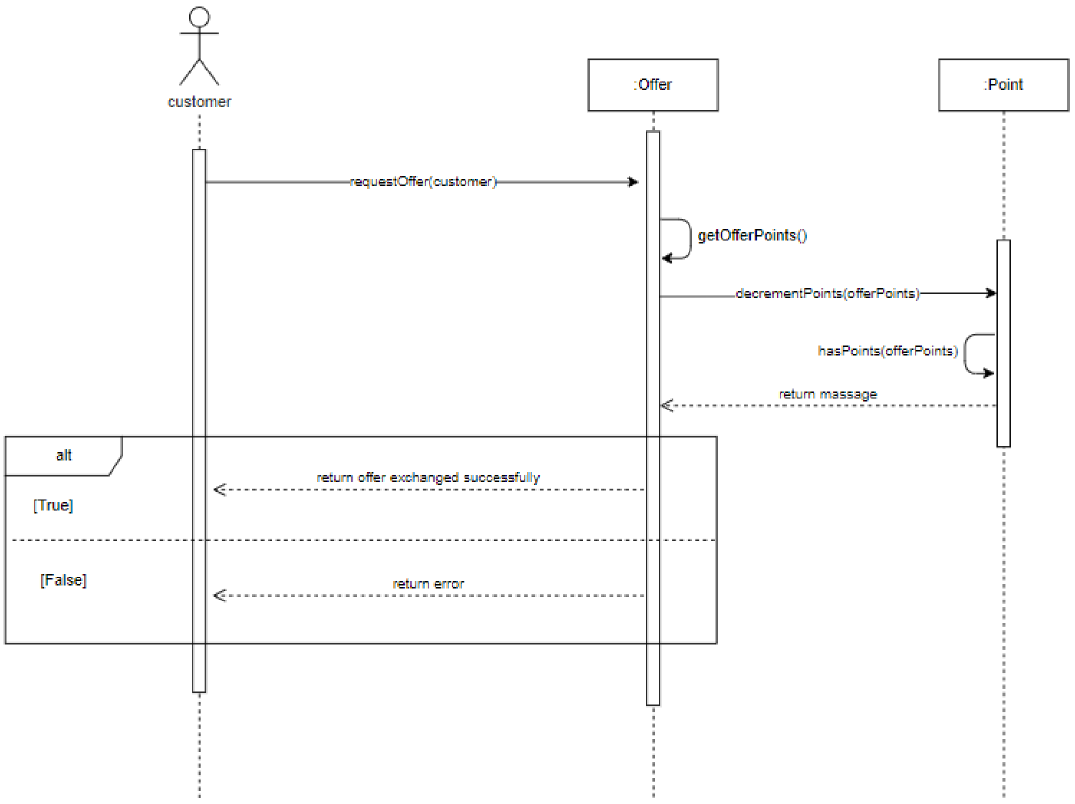



RecycleMe System
User
Admin
Driver
Customer
Login
View Order Details
Request Offer
Create Order
View Points
View Leader Board
View Positive Impact
Select Order
Add Offers
Download Report
<<include>>

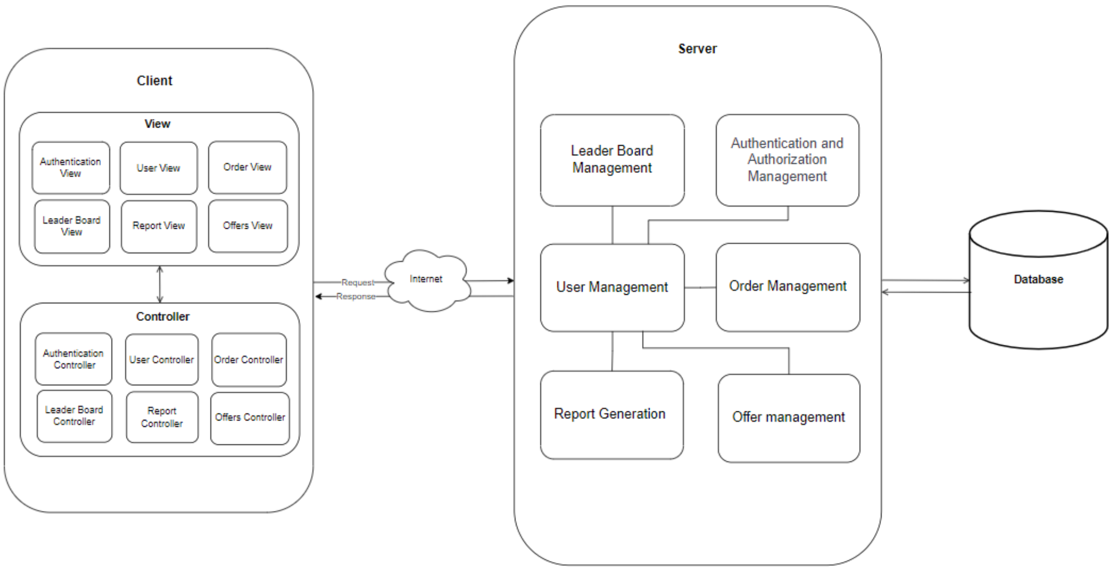
3.1.1 Use case description

Use Case Name: RequestOffer	
Primary Actor: Customer	**Use Case Type:** Essential (include checkpoints use case)
Brief Description: This use case allows the customer to exchange his/her points with any offer/promo according to his/her points calculated and provided by the system. Points are gathered according to the number of orders s/he made in the system.	
Pre-Conditions: 1. The customer must be logged on to the system successfully.	
Basic Flow of Events: 1. The system reads the offer points. 2. The system checks if the customer has enough points. 3. The system will decrement the customer points. 4. The system displays succussed massage.	
Exceptional Flows: 2.1 The customer doesn't have enough points. 2.1.1 Return failure massage. 3.1 The system will not be able to decrement customer points. 3.1.1 Return failure massage.	
Post-Conditions: The system automatically notifies the customer with the response.	

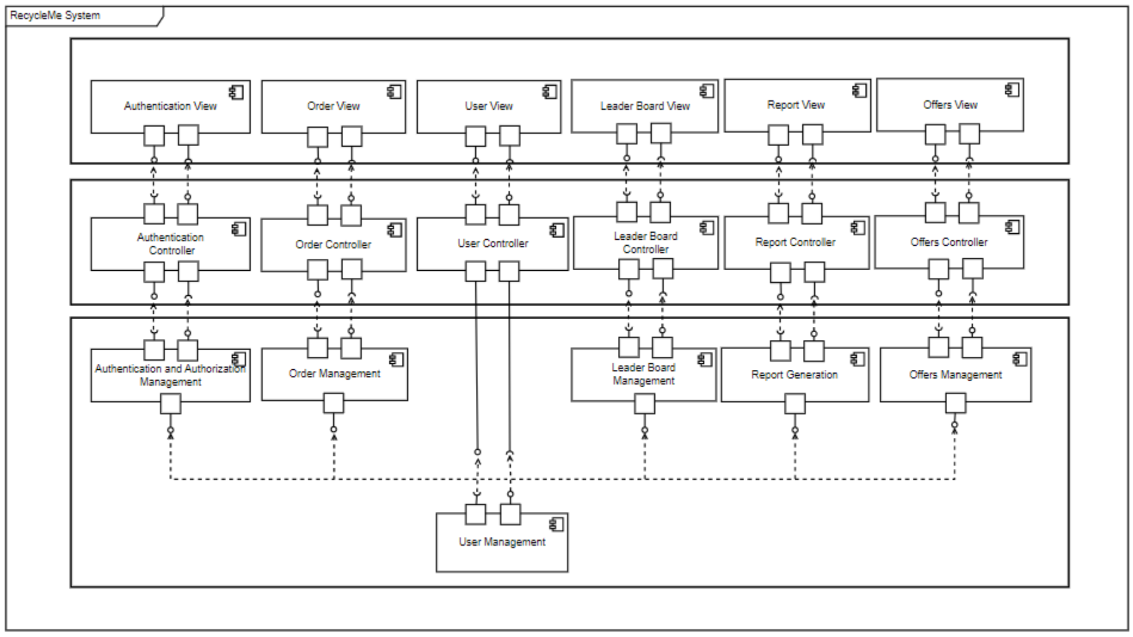
3.2 Sequence Diagram



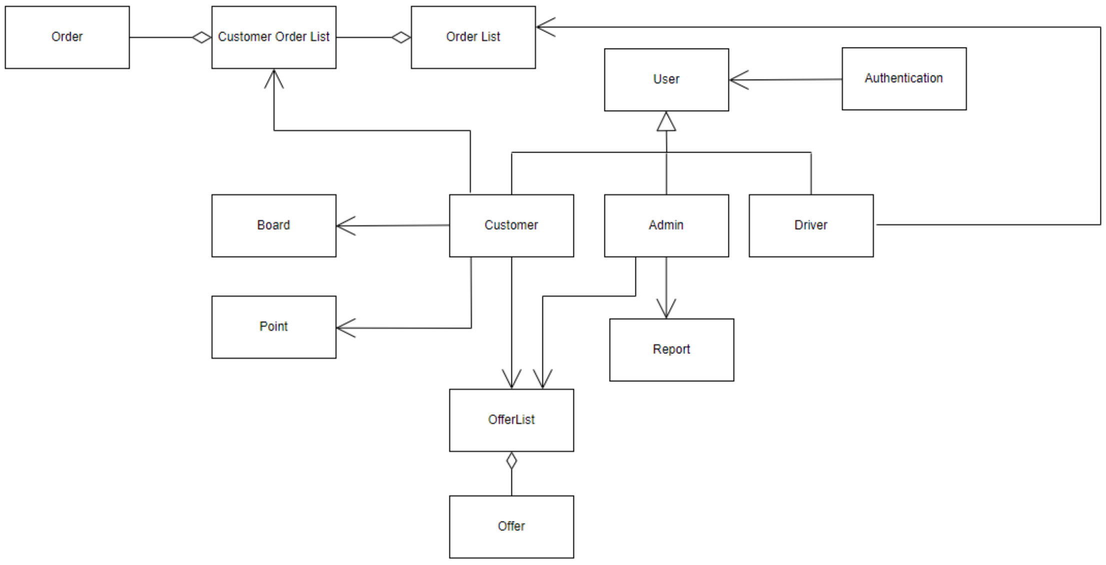
3.3 Overall Architecture



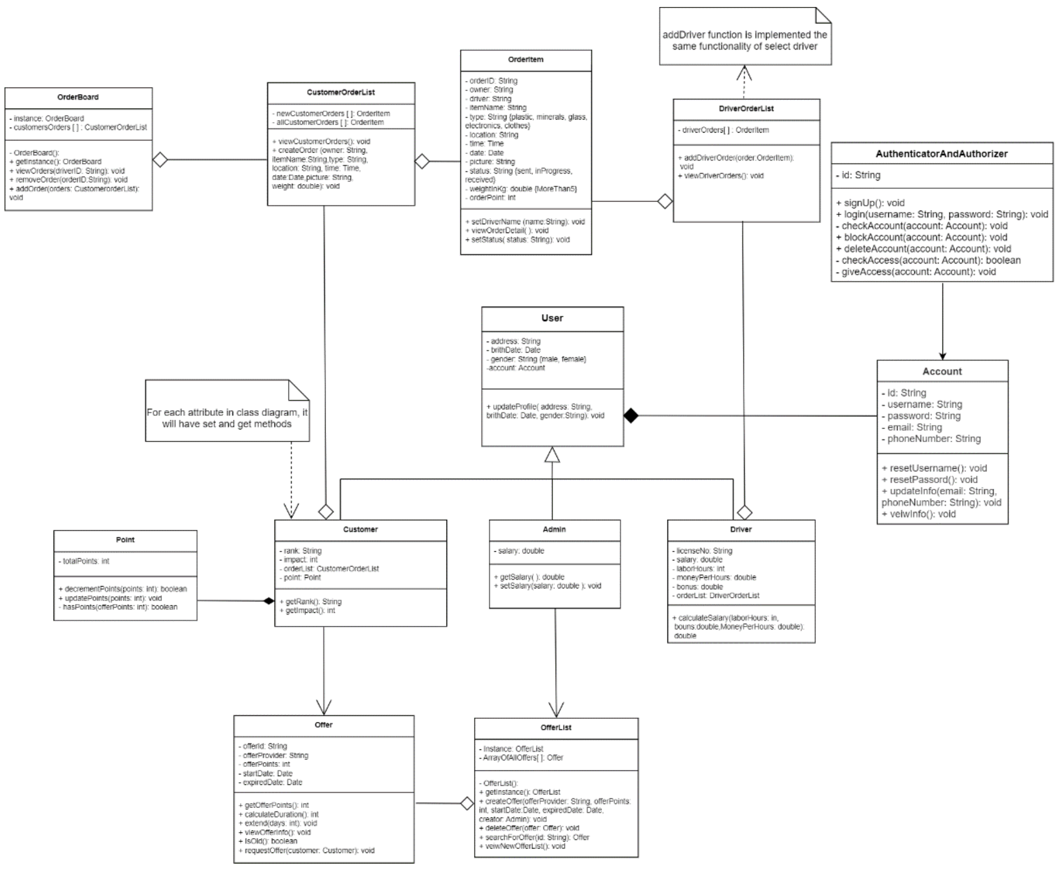
3.4 Component Diagram



3.5 Initial Set of Objects



3.6 Class Diagram


addDriver function is implemented the same functionality of select driver
OrderBoard
- instance: OrderBoard
- customersOrders [ ] : CustomerOrderList
- OrderBoard():
+ getInstance(): OrderBoard
+ viewOrders(driverID: String): void
+ removeOrder(orderID:String): void
+ addOrder(orders: CustomerorderList): void
CustomerOrderList
- newCustomerOrders [ ]: OrderItem
- allCustomerOrders [ ]: OrderItem
+ viewCustomerOrders(): void
+ createOrder (owner: String, itemName:String,type: String, location: String, time: Time, date:Date,picture: String, weight: double): void
OrderItem
- orderID: String
- owner: String
- driver: String
- itemName: String
- type: String {plastic, minerals, glass, electronics, clothes}
- location: String
- time: Time
- date: Date
- picture: String
- status: String {sent, inProgress, received}
- weightInKg: double {MoreThan5}
- orderPoint: int
+ setDriverName (name:String): void
+ viewOrderDetail( ): void
+ setStatus(status: String): void
DriverOrderList
- driverOrders[ ] : OrderItem
+ addDriverOrder(order:OrderItem): void
+ viewDriverOrders(): void
AuthenticatorAndAuthorizer
- id: String
+ signUp(): void
+ login(username: String, password: String): void
- checkAccount(account: Account): void
+ blockAccount(account: Account): void
+ deleteAccount(account: Account): void
- checkAccess(account: Account): boolean
- giveAccess(account: Account): void
User
- address: String
- brithDate: Date
- gender: String {male, female}
-account: Account
+ updateProfile(address: String, brithDate: Date, gender:String): void
Account
- id: String
- username: String
- password: String
- email: String
- phoneNumber: String
+ resetUsername(): void
+ resetPassord(): void
+ updateInfo(email: String, phoneNumber: String): void
+ veiwInfo(): void
For each attribute in class diagram, it will have set and get methods
Point
- totalPoints: int
+ decrementPoints(points: int): boolean
+ updatePoints(points: int): void
- hasPoints(offerPoints: int): boolean
Customer
- rank: String
- impact: int
- orderList: CustomerOrderList
- point: Point
+ getRank(): String
+ getImpact(): int
Admin
- salary: double
+ getSalary( ): double
+ setSalary(salary: double): void
Driver
- licenseNo: String
- salary: double
- laborHours: int
- moneyPerHours: double
- bonus: double
- orderList: DriverOrderList
+ calculateSalary(laborHours: in, bouns:double,MoneyPerHours: double): double
Offer
- offerId: String
- offerProvider: String
- offerPoints: int
- startDate: Date
- expiredDate: Date
+ getOfferPoints(): int
+ calculateDuration(): int
+ extend(days: int): void
+ viewOfferInfo(): void
+ IsOld(): boolean
+ requestOffer(customer: Customer): void
OfferList
- Instance: OfferList
- ArrayOfAllOffers[ ]: Offer
- OfferList():
+ getInstance(): OfferList
+ createOffer(offerProvider: String, offerPoints: int, startDate:Date, expiredDate: Date, creator: Admin): void
+ deleteOffer(offer: Offer): void
+ searchForOffer(id: String): Offer
+ veiwNewOfferList(): void

Note: The scope of the class digram is order management, user management, offers management and authentication and authorization components.

Chapter 4: Evaluating the Quality of Models

4.1 Validation of Models

We validate our work in three stages: in the first stage, we check that every model addresses all the requirements (except the class diagram, which has its own scope as mentioned earlier); in the second stage, we validate every model by itself in terms of syntactic, semantic, and aesthetic concepts. The following is more detailed information about this stage:

Validation of the Use Case Diagram:

A. Syntax Checks:

- We check the notation for elements, where actors are represented as stickman notation and use cases are represented by ellipses.
- We check the existing of the system boundary in the use case diagram.
- We check that the actor's name must be singular, noun and representing a role.
- We check that use cases name has a verb-like fashion.
- We check that there is the relationship between actors and all related use cases with an association line.
- We check the correct direction of generations and include arrows.

B. Semantic Checks:

- We check the semantics meanings of actors, use cases name.
- We check the meaning name of the entire use case diagram.
- We check the name of use case is written in the actors point view.

C. Aesthetic Checks:

- We check the use case diagram must have a reasonable number of relationships, as a result the second use diagram is drown to address this issue.

Validation of the Sequence Diagram:

A. Syntax Checks:

- We check the massages are represented by lines.
- We check that the objects are donated by rectangles.
- We check there is actor who starts this use case.

B. Semantics Checks:

- We check the meaning of the overall sequence diagram.

C. Aesthetic Checks:

- We check that there is only one object from every class that appears in the sequence diagram.
- We check that there is suitable number of the steps in sequence.

Validation of Software Architecture:

We check the components in the architecture service for the overall developed system, and we also check that the components are distributed logically and that the relationships between components are sufficient.

Validation of the Component Diagram:

A. Syntax Checks:

- We check component names as nouns.
- We checked the correct arrows symbol of the provider and consumer component.

B. Semantics Checks:

- We check the meaning of the component name.

C. Aesthetic Checks:

- We check there is reasonable number of components in the diagram.
- We check that there is reasonable number of relationships between components.

Validation of the Class Diagram:

A. Syntax Checks:

- We check that the class name is a single noun with a capital first letter.
- We check the class attributes type and visibility
- We check that the relationship notation between classes (aggression, composition, inheritance, association) are correct.

B. Semantic Checks:

- We check that the meaning of the class name is clear.
- We check that the opportunities for generation, so the class user is generated as a result of this tips.
- We check the semantics of inheritance.

C. Aesthetic Checks:

- We check class balance, as a result we split "point" attribute in User class and make it as a class.
- We check there is suitable numbers of attributes and methods in every class.

The last stage is cross-validation among all models.

Components Diagram:

We compared the component diagram with the software architecture and noticed a missing relationship that was added later, which appears in the architecture but not in the components that were added later.

Also, there is a tiny link between components and the class diagram, where the relationship between components is proven in the class diagram.

Sequence Diagram:

We checked the sequence diagram represents only one-use case, and it has the same flow of events of the tabular description, and all objects come from class in class diagram that contains the used methods by these objects.

Use Case Diagram:

We checked that every use case is implemented by a class diagram as a method with the right actors' access; we checked it also with a sequence diagram, and we noticed there is a mismatch between the names of use cases and the tabular descriptions as a result of changing use case names later.

Class Diagram:

Class diagram represents the objects in sequence with methods, also methods represent functionality of use cases.

Chapter 5: Conclusion and References

5.1 Conclusion

All of the models were based on requirements that were elicited from by the course instructor during explaining the suggested ideas, plus some requirements techniques were followed which are brainstorming with team members and looking at similar analogy apps such as RecycleSmart, Recycling Coach, and so on.

Moreover, this report provides some analysis and design models which are a use case diagram, sequence diagram, and initial set of objects in terms of the analysis stage, while component diagram and class diagram with singleton design pattern was provided in the design stage. In addition to analysis and design models, overall architecture was designed as a bridge between these two stages. All of these artifacts were implemented to design software solutions to solve real-life problems in terms of facilitating the recycling of unwanted items through the RecyclingMe mobile application to be our project in the SE323 course.

In addition, the report provides main points that are related to the objective of the project and which is achieved by following: to meet our main goal, which is encourage people to adopt the recycling attribute, one feature has been considered which is give them some discount codes, that will provide from our partners, to exchange the points that are collected from the previous recycling orders with these codes.

If given the opportunity, we will continue working on the class diagram, which due to time limitations in the first version of the system eliminated some functionality from our scope. The features include a leader board that will be displayed alongside the point's features to encourage people to recycle; additionally, more features will be available to the admin account, such as a downloaded report with statistics about the user and their orders .In terms of new ideas that we have in mind, we would like to add some notifications and calendar features to our app to enhance the user experience in recycling management.

References

Fakhroutdinov, K. (2013, November 27). *Online shopping*. UML graphical notation overview, examples, and reference. Retrieved December 18, 2022, from https://www.uml-diagrams.org/examples/online-shopping-uml-component-diagram-example.html?context=cmp-ex

Fakhroutdinov, K. (2018, April 21). *UML component diagrams*. UML graphical notation overview, examples, and reference. Retrieved December 18, 2022, from https://www.uml-diagrams.org/component-diagrams.html

Fowler, M. (2005, February 3). *Bliki: Ballandsocket*. martinfowler.com. Retrieved December 18, 2022, from https://martinfowler.com/bliki/BallAndSocket.html

StuartReid. (2015, July 22). *Algorithmic trading system architecture - Stuart Gordon Reid*. Turing Finance. Retrieved December 18, 2022, from http://www.turingfinance.com/algorithmic-trading-system-architecture-post/#prettyPhoto

Unhelkar, B. (2005). Verification and validation for quality of Uml 2.0 models. Wiley-Interscience.

YouTube. (2017). *Singleton Pattern – Design Patterns (ep 6)*. *YouTube*. Retrieved December 18, 2022, from https://www.youtube.com/watch?v=hUE_j6q0LTQ