REACH

SWE- 226- (Software Requirement Engineering Lab)

REACH

Software Requirement Specification Project
for
Software Requirement Engineering Lab (SWE 226)

Submitted By :

Team Undaunted

Members :

1. **Md Sadman Hafiz(Reg : 2018831057)**
2. **Bijon Saha (Reg : 2019831007)**
3. **Hamim Rahman (Reg : 2019831034)**
4. **Sanjana Afrin (Reg : 2019830154)**
5. **Ashrafur Rahman (Reg : 2019831070)**

Submitted To :

Partha Protim Paul
Lecturer , IICT,SUST

Submitted To :

Partha Protim Paul
Lecturer
Institute of Information and Communication Technology,
Shahjalal University of Science and Technology, Sylhet

Submition Date : 13 June , 2022

LETTER OF TRANSMITTAL

June 13, 2022

Partha Protim Paul

Lecturer

Institute of Information and Communication Technology

Shahjalal University of Science and Technology, Sylhet

Dear Sir,

We have prepared the attached report on the Software Requirements Specifications of the "REACH" application for your approval. This report describes the project requirements we've gathered so far.

The major objective of this report is to provide a synopsis of my results from our Software Project work. This report contains the specifics of each process used to gather the requirements.

Sincerely yours

Md Sadman Hafiz

Team Leader , Team Undaunted

On behalf of

Team Undaunted

Enclosure: SRS

Executive Summary

Introducing Reach—Hire as You Desire. Reach envisions a world in which individuals are able to explore their innate household and handyman skills, thereby eradicating classism. Our app, "Reach," intends to provide a virtual platform that connects people who are willing to provide odd-job or handyman assignments with others who are willing to accept them as part-time, full-time, or even contract work. Unique to our platform is the fact that we enable people to hire others for customized odd jobs, i.e., jobs that typically lack a label. By this they can earn some extra cash, regardless of their age, size, gender, educational background, experience, or skill set. As we provide positions that may require little to no experience (e.g., line-standers, workout assistants, etc.), anyone with a willingness to work can join. For instance, you enjoy gardening but require assistance with that task. So, you can post on our website in the customized odd-jobs section what you want one of our employees to perform for you. Whoever is willing to complete the task will react. Based on their profiles, you can choose any of the respondents who appeal to your preferences. In addition, connection fees, premium service fees, and Google AdSense would be significant sources of income. We propose to directly engage workers for premium services; district supervisors, divisional supervisors, state supervisors; technical teams, marketing teams, grooming teams, etc., using mass office personnel and on-field supervisors. We seek to safeguard the safety of both our employees and consumers by maintaining their authentic information. The concept of "**Reach**" adheres precisely to SDG-8 as it meets all three requirements.

AKNOWLEDGEMENT

We, the Team Undaunted , We've completed our second year first semester Software Requirement Analysis Lab(SWE 226) work , SRS, for an android app Project , REACH by the Grace of ALLAH. We are very grateful to our project supervisor Partha Protim Paul Sir for his supervision throughout in working time. He helped us during the whole worktime of this project and shared his experience and knowledge with us, instructed and guided us to complete the project. We want thank him form the core of our heart.

We worked as a team , and everyone's role was pre-defined and each of our teammate has given his/her best .We all have contributed on this equally and everyone's role has a significant impact on this project.

Table of Contents

Table of figures

Chapter 1

Introduction

This chapter describes the whole overview, objectives of the project “REACH” for the audience.

Purpose

The document is, basically, a Software Requirement Analysis for an android app “REACH”. It includes all the requirements (functional and non- functional), model needed for the app, diagrams for better understandings and other basic information which will work as a formal document for developing this project and the medium of communication between the developers, officials, and the stakeholders. This document was developed so that anyone can easily understand the gist of the concepts of “The REACH” easily. It will be evolved over time as there can be some modification after validating by developers and the stakeholders.

Intended Audience

This report is developed for the audience which includes client, project managers, designers, developers, and testers, debuggers.

- **The Client:** They will find what they will be getting, what features are there to be developed and they can also cross-check if their requirements are getting fulfilled or not.
- **The Project Manager:** He will find this document helpful to lead the developer team and set the deadline and other rules and make sure the quality of the project.
- **The Designers:** The designer will be helped as they are getting the basic system design and ensure that what they are fulling the client's requirement.
- **The Developers:** They will find the functionality of the app, what classes should be developed and what methods will be there and their interconnections as well.
- **The Testers:** They will develop the basic testing model for this app using this document.
- **The Debuggers:** They can use this document for checking all the features mentioned where are developed or not.

Chapter 2
Inception

This chapter describes the first step of Software Requirement analysis, Inception.

Introduction

What is inception?

According to Text book[1]

At project inception, you establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team. In general, most projects begin when a business need is identified or a potential new market or service is discovered. Stakeholders from the business community (e.g., business managers, marketing people, product managers) define a business case for the idea, try to identify the breadth and depth of the market, do a rough feasibility analysis, and identify a working description of the project's scope. All of this information is subject to change, but it is sufficient to precipitate discussions with the software engineering organization.

So, Basically, inception is identifying the basic concept of the project and what the actual product will be received by the client. There are some steps for this. We've followed these steps-

- Identifying Stakeholders
- Recognizing multiple view points
- Working towards collaboration
- Asking the first questions

[1] Roger S. Pressman_ Bruce R. Maxin - Software Engineering_ A Practitioner's Approach-McGraw-Hill Education (2014)

Identifying Stakeholders

Stakeholders are the list of people who will contribute the input as requirements. The initial list will grow as stakeholders are contacted because every stakeholder will be asked: "Whom else do you think I should talk to?". They have a positive or negative influence on the completion of this project.

For the app 'REACH", we've identified our stakeholders.

General Users: General Users are the heart of this project as this is turning out to be a public service app.

There will be two kinds of general users in this app.

- **The Customer:** They will be using this app for getting the solution of their problem by hiring workers or freelancers for part-time or fulltime job.

- **The Workers:** Those who are looking for part-time jobs for earning extra-cash will use the app

Service Provider: They will ensure the security and quality of the worker using the application, they will update workers info, their active status, order details, payment details, rating etc.

Recognizing Multiple Viewpoints

After going through a market survey analysis for general users, we've found their viewpoint of them

General user's viewpoints:

- Easy to use the map
- Make connection in minimum possible time
- Getting order from the nearest locality

- Jobs that don't require specific skills or experience

- Improvement of base asking price after good job and review
- Security of the both workers and customers
- Getting notification of job on their connection (of job)
- Getting permanently hired by the authority
- Getting payment receipt to customers

The Reach's View Points:

- Providing the solution of customer's any kind of day-to-day life problems keeping a section to describe their problem, having some specific job also.
- Ensuring the security of both workers and customers by authentic details on opening a new account
- Giving notification to both client and workers for job post or search or comment
- Chat through messaging system
- Tracking the workers to find the locality of them
- Updating the ratings and payment details
- No intention to hire the workers as they will individually connect with each other

- Update after successful service

Working Towards Collaboration

After analyzing both Reach and the general user's viewpoint, it seems there are some similarity and conflicts in these. We've used these steps to merge these-

- Identify the common and conflicting requirements
- Categorize the requirements
- Take priority points for each requirement from stakeholders and on the basis of this voting prioritize the requirements
- Make the final requirements

Common requirements:

- Getting order from the nearest locality
- Jobs that don't require specific skills or experience
- Improvement of base asking price after good job and review
- Security of the both workers and customers
- Getting notification of job on their connection (of job)

Conflicting requirements:

- Get permanently hired by the authority
- No intention to hire the workers as they will individually connect with each other

We've finalized the requirements in basis of priority

Final requirements

- Getting order from the nearest locality
- Easy to use the map
- Make connection in minimum possible time
- Jobs that don't require specific skills or experience
- Improvement of base asking price after good job and review
- Security of the both workers and customers
- Getting notification of job on their connection (of job)
- Update after successful service
- Give payment receipt to customers
- Chat through messaging system

Asking the First Question

After discussing about the requirements, we've set first set of question which basically focused on the overall description of the project, goals, benefits, result after implementation, possible alternatives and other context free discussion. The next set of question was for understanding the problem better with the participation of the client himself. The final set of question was set for the effectiveness of active communication between the stakeholders, developers and the authorities.

Conclusion

This Phase, Inception, is very important phase of any project. So, we've completed this phase very carefully and it helped us to understand the goal of the project better and by this, we become able to build a communication among all the members of this project, stakeholders. It also helped to finalize the requirement of the app "REACH" and we've found out what to do and who will use this.

Chapter 3

Elicitation

Introduction

According to our textbook[2]

> *Requirements elicitation (also called requirements gathering) combines elements of problem solving, elaboration, negotiation, and specification. In order to encourage a collaborative, team-oriented approach to requirements gathering, stakeholders work together to identify the problem, propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirements [Zah90][3].*

So, it is to define the boundary of the market or user of this project it is one of the most important phases of developing a project. So, we need to understand the scope of the problem. We need to find how it full-fil the business need of the client, how the product will be used in their daily usage, who will use as well. To do all these things there'll still be some problems, like problem of the scope, problem of understanding, problem of volatility. These problems were identified by Christel and Kang [Cri92]

[2] Roger S. Pressman_ Bruce R. Maxin - Software Engineering_ A Practitioner's Approach-McGraw-Hill Education (2014)

[3] This approach is sometimes called a facilitated application specification technique (FAST).

Eliciting the Requirements

In the 2nd chapter one we've found out the basic understanding of the whole project by inception method, asking the stakeholder a set of question and gathered some different point of view. The elicitation process basically combines with the other steps of requirement analysis, elaboration, negotiation, and specification. In order to elicit the requirement more effective way.

We've followed these step-

- Collaborative Requirements gathering
- Quality Function Deployment (QFD)
- Usage Scenarios
- Elicitation work product

Collaborative Requirements Gathering

We gathered more requirement by collaborating with the stake-holders. We've followed these steps-

- We've selected a 'Facilitator'[4] for our meeting,
- Set the rules for the participation and preparation for the meeting,
- Developed a 'definition mechanism' to use in the meeting.
- We've called some general users in the meeting and asked them if they were interested to use such a platform to get their daily life problem solved or make some extra money with a part-time job.

[4] A facilitator who has better knowledge of this project is chosen to conduct the meeting, can be from developer team or from the customer as well

- We've also asked why they want so and what is the current problem without the platform right now.

Quality function deployment

From our textbook-

> *Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software. QFD "concentrates on maximizing customer satisfaction from the software engineering process" [Zul92]. To accomplish this, QFD emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process. QFD identifies three types of requirements [Zul92]:*

Basically, QFD focuses on categorizing the requirements on priority basis. Such as-

- Normal requirements
- Expected requirements
- Exciting requirements

Normal Requirements

These requirements are basically stated on the meeting and the main objective is client's satisfaction.

These requirements were stated during our meeting –

- Active internet connection
- Allow System to check items for valid users.

- Allow valid users to login and logout
- Chat through messaging system
- Help and instruction features for the user to open an account
- Product manual
- Security of user's personal data

Expected Requirements

Basically, the client can't express exactly what they want and there are some functionalities which the customers don't know, but are necessary for the projects, these are expected requirements.

Such requirements for our project are –

- The App will run on an Android App
- User can easily view the map from the app
- Client can see who are active worker within client's locality
- Maintain an algorithm for finding the shortest path to decide the nearest person
- Clients location via GPS
- Notify for the post or job search

Exciting Requirements

These are those features that client would love to have and be quite satisfied if they experience these

In our Project, these types of requirements are-

- Save favorite worker's info
- Premium service option
- Getting updated the cumulative average rating of a worker automatically after client's review
- Giving some automatic search preferences to client

Chapter 4

Scenario Based Model

Introduction

Scenario based modeling is more understandable than anything. So, developing such prototype helps to involve all of the developers effectively. Our primary focus was to develop the model describing 'What?' not 'How?'. We've confined our models within the business domain of the customers and tried to keep the models as simple as possible.

For this purpose, we need to define two things as these will be used on a regular basis on the upcoming scenario-based models –

Primary Actor:

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor:

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

Use Case Diagram

Actually, it describes how the end users will interact with the software and is expressed with the primary and the secondary actors of the software. To begin developing a set of use cases, list the functions or activities performed by a specific actor.

Use case: REACH

Primary actor: Customers and Workers

Goal in context: To Connect the workers and the customers

Preconditions: App should be installed in android phone with active internet connection

Trigger: The Customer login into the app and search and connect a worker to do a job

Scenario:

For Customer:

1. login into the app
2. turn on GPS
3. Search for desired service
4. Selects work
5. Choose a worker
6. Contacts with him and hire him
7. Pay through the app
8. Give rating / review

For Workers:

1 login into the app

2 Turn on GPS.

3.wait for notification for work or search for post of work

4. Response to customer

Exceptions:

1. User forgot the password
2. Wrong account detail during opening a account
3. User don't turn on the location
4. Customer choose wrong payment gateway
5. Customer payment account is empty

Priority: Essential, must be implemented

When available: First increment

Frequency of use: Many times, per day

Channel to actor: via Android app and internet connection

Secondary actors: Support technician, Service provider

Channels to secondary actors:

Support technician: Internet and computer network server

Use Case Diagram

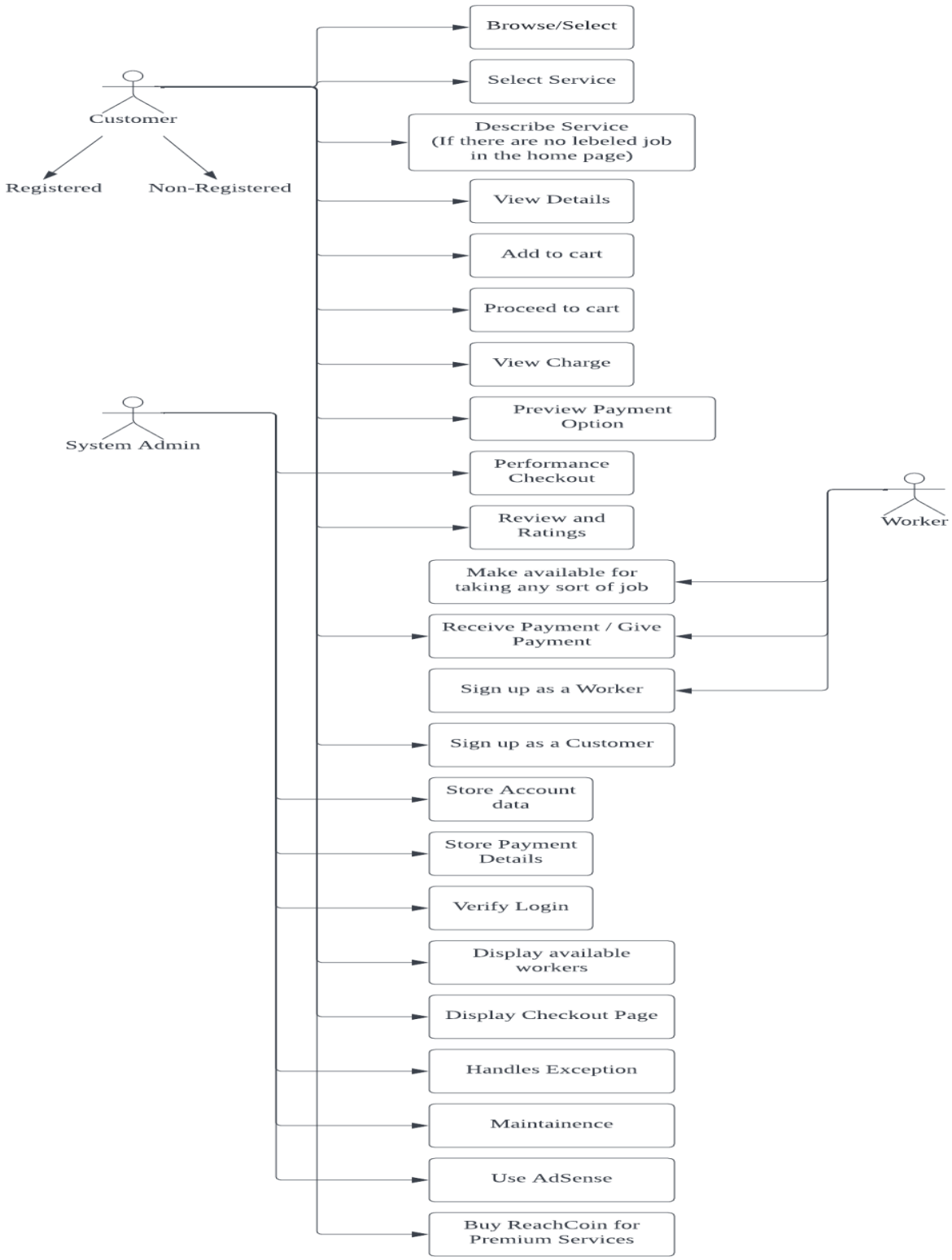

Figure 1 : Use Case Diagram of REACH

Activity diagram

Activity diagram is the control flow diagram which shows the total workflow through the full process and the flow of one activity to another activity. The activity can be described as an operation of the system.

Here is the activity diagram for “REACH”

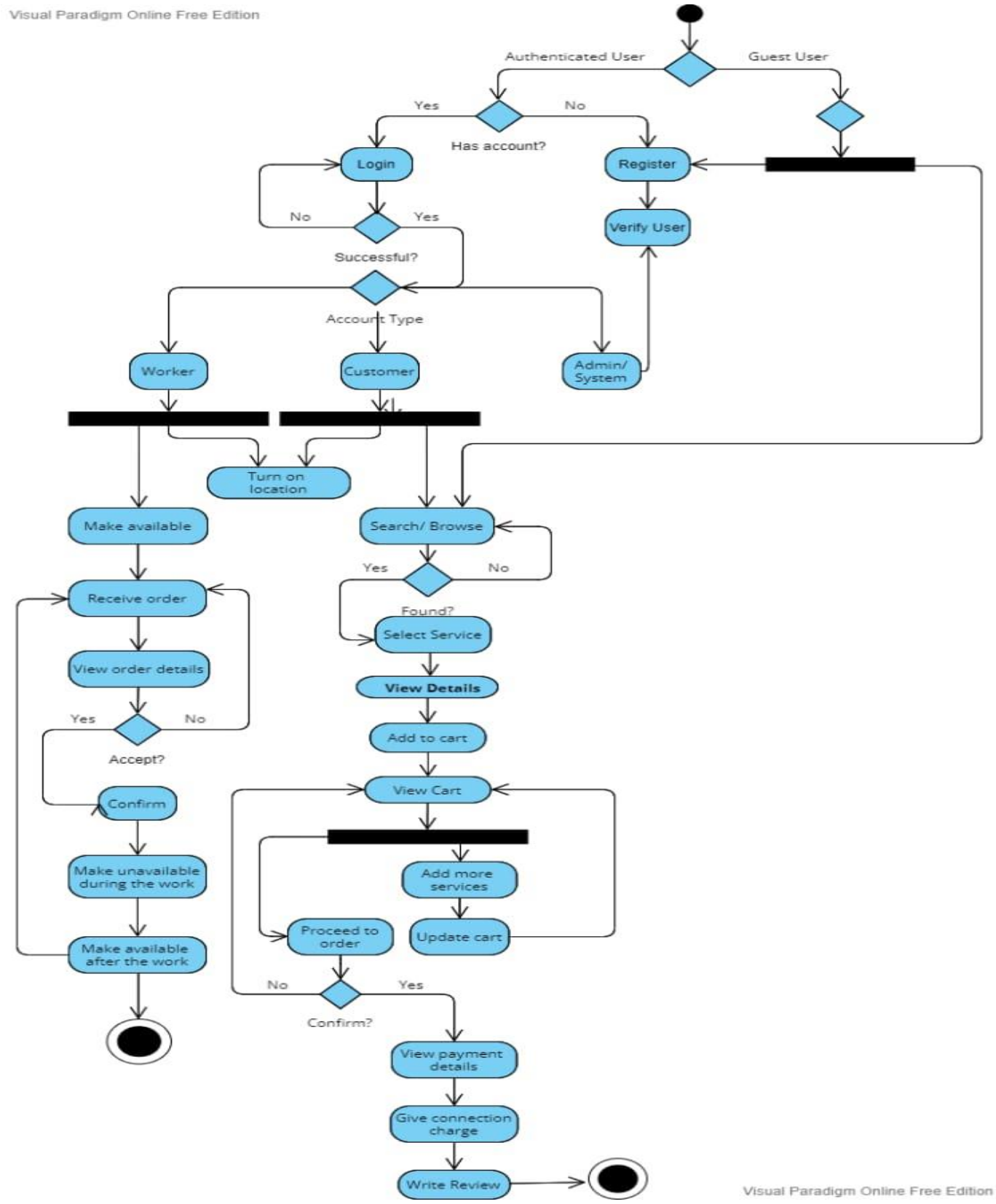

Figure 2 : Activity Diagram of REACH

Swim Lane Diagram

Swimlane diagram is more focus on actor of this project. We divide the activities basis of actors. A swim lane diagram makes responsibilities more clear than a regular flowchart. When looking to improve processes, knowing which department is responsible for what can help speed up the process of correcting inefficiencies and eliminating delays.

Here is the Swim lane diagram of “REACH”

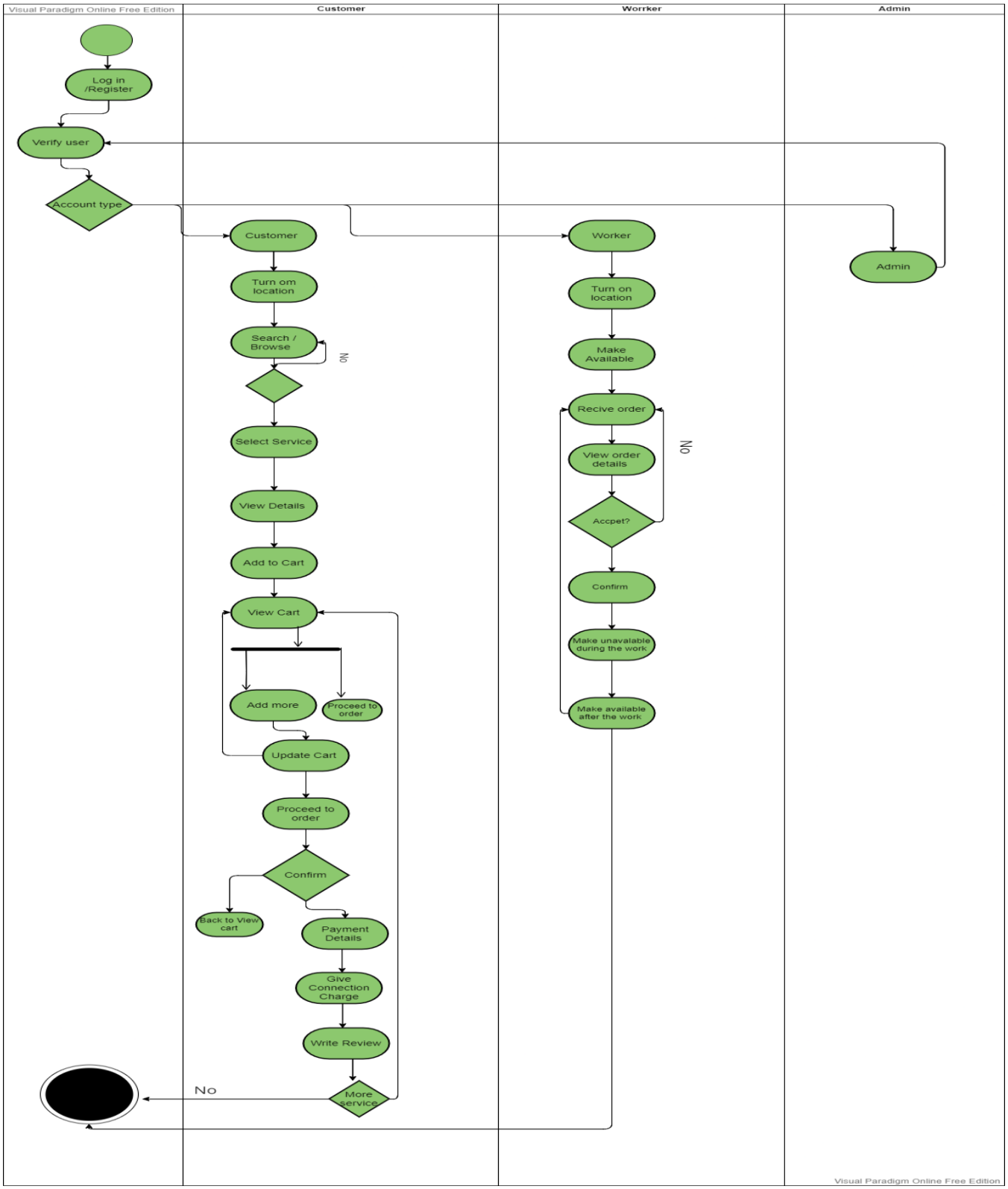

Figure 3 : Swim Lane diagram for the REACH

Chapter 5

Data Model

Introduction

Data models are very important for storing the user data and using the database. So, proper definition should be followed to define a data. Data Model helps us in this regard.

Data Objects

A data object is a representation of information which has different properties or attributes that
must be understood by software. I have found the following data objects in my Easy Bus Tracker

application:

Worker

Attributes:

- **Name**
- **Worker_ID**
- **Current_location**
- **NID**
- **Bank A/C**
- **Present_Address**
- **Skills**

Client

Attributes:

- **Name**
- **Client_ID**
- **Current_Location**
- **Contact_Number**
- **Address**
- **Preference-**

Order Details

Attributes:

- **Order_ID**
- **Worker_ID**
- **Worker's_Name**
- **Date**
- **Cost**
- **Client_Name**
- **Client_ID**

Post

Attributes:

- **Post_ID**
- **Post_Details**

Payment

Attributes:

- **ID**
- **Payment_Method**
- **Amount**

Order Cart

Attributes:

- **Cart_ID**
- **Worker_Name**
- **Worker_ID**
- **Date**

Availability info

Attributes:

- **Worker_ID**
- **Worker_Name**
- **Order_ID**
- **Start_Time**
- **Finish_Time**

Work Detail

Attributes:

- **Worker_ID**
- **Name**
- **Short_Detail**
- **Time**
- **Payment**

Review

Attributes:

- **Review_Title**
- **Review_Date**
- **Service_Pros**
- **Service_Cons**
- **Other_Comments**

Admin

Attributes:

- **User_Name**
- **Email_Id**

ER Diagram

After finding the Data Object we've drawn Entity – Relationship Diagram (ER diagram) for better understanding of data flow and storing to database-

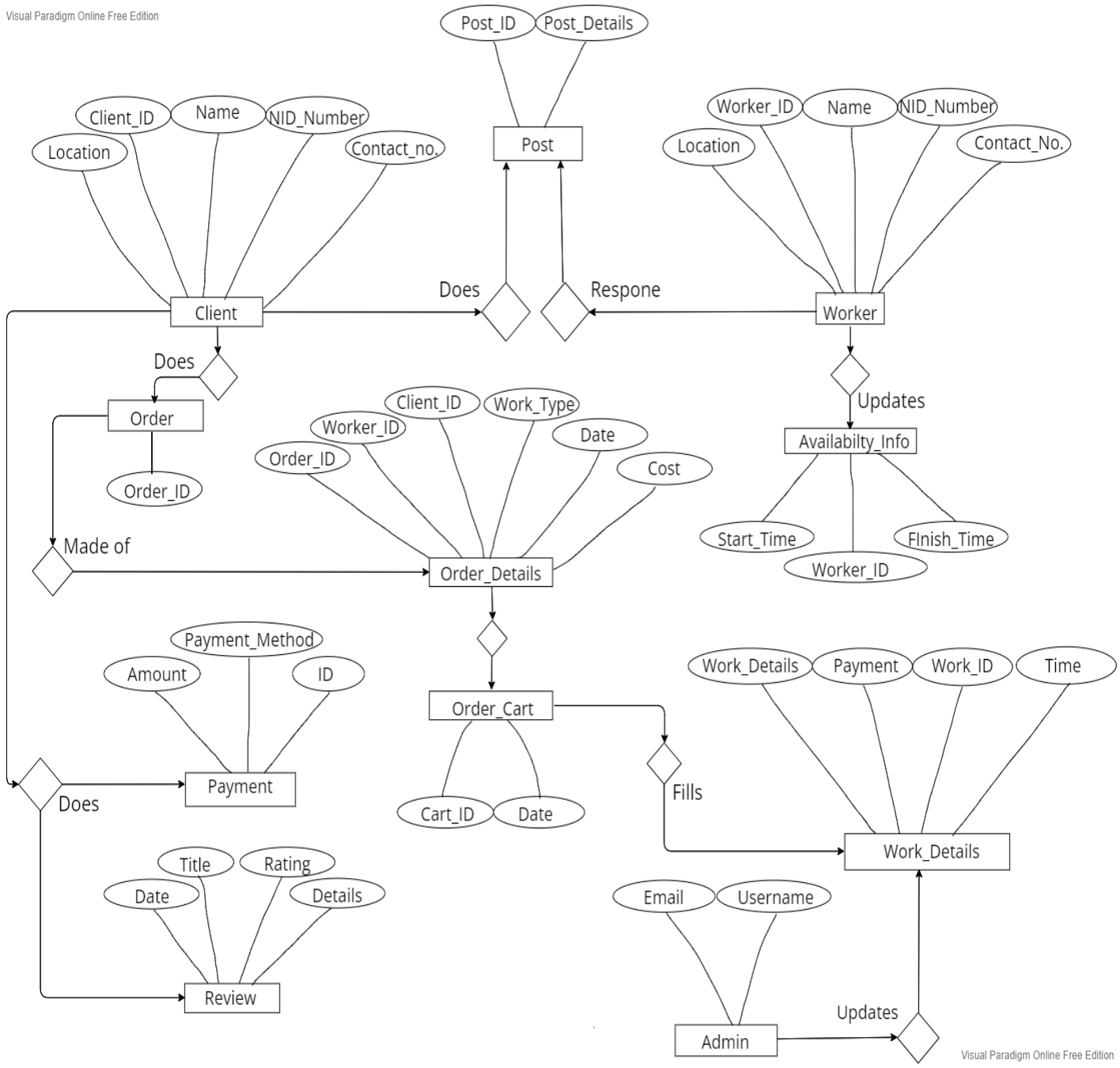

Figure 4 : ER diagram for REACH

Data Schema Table

Based on the data objects, the following data schema tables can be created:

Worker	
Attributes	Type(size)
Name	Var Char (20)
Worker_ID	Number (15)
Current_location	Double (20)
NID	Number (15)
Bank_A/C	Number (15)
Present_Address	Var Char (50)
Skills	Var Char (100)

Table 1 : Data Schema Table for Worker Object

Client	
Attributes	Type
Name	Var-Char (20)
Client_ID	Number (15)
Current_Location	Double (20)
Contact_Number	Number (15)
Address	Var Char (20)
Preference	Var Char (25)

Table 2: Data Schema Table for Client Object

Order	
Attributes	Type
Order_ID	Number (15)

Table 2: Data Schema Table for Order Object

Order_Details	
Attributes	Type
Order_ID	Number (15)
Worker_ID	Number (15)
Worker's_Name	Var Char (20)
Date	Var Char (8)
Cost	Number (10)
Client_Name	Var Char (20)
Client_ID	Number (15)

Table 4: Data Schema Table for Order Detail Object

Post	
Attributes	Type
Post_ID	Number (15)
Post_Details	Var Char (50)

Table 5 : Data Schema Table for Post Object

Payment	
Attributes	Type
ID	Number (15)
Payment_Method	Number (3)
Amount	Number (10)

Table 6 : Data Schema Table for Payment Object

Order_Cart	
Attributes	Type
Cart_ID	Number (15)
Worker_Name	Var Char (20)
Worker_ID	Number (15)
Date	Var Char (8)

Table 7 : Data Schema Table for Order Cart Object

Availability_info	
Attributes	Type
Worker_ID	Number (15)
Worker_Name	Var Char (20)
Order_ID	Number (15)
Start_Time	Var Char (15)
Finish_Time	Var Char (15)

Table 8: Data Schema Table for Availability info Object

Work_Detail	
Attributes	Type
Worker_ID	Number (15)
Name	Var Char (20)
Short_Detail	Var Char (50)
Time	Var Char (15)
Payment	Boolean

Table 9: Data Schema Table for Work_detail Object

Review	
Attributes	Type
Review_Title	Var Char (15)
Review_Date	Var Char (20)
Service_Pros	Var Char (50)
Service_Cons	Var Char (15)
Other_Comments	Var Char (20)

Table 10: Data Schema Table for Review Object

Admin	
Attributes	Type
User_Name	Var Char (20)
Email	Var Char (20)

Table 11: Data Schema Table for Admin Object

Relation Between Data objects

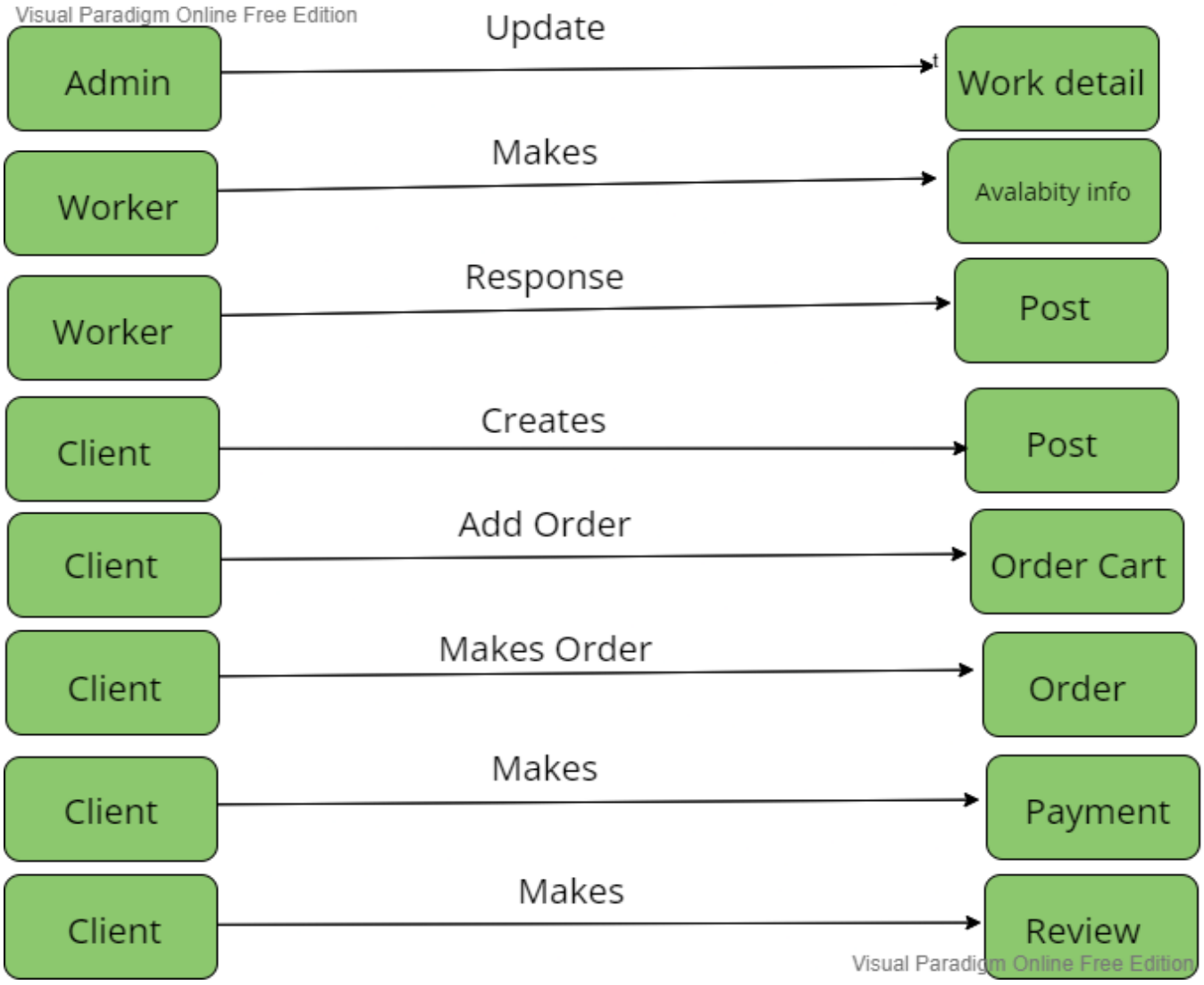

Figure 5 : Relation Between Data Objects

Chapter 6

Class Based Model

This Chapter is to describe class-based modeling of Easy Bus Tracker application.

Introduction

Class-based modeling represents the objects that the system will manipulate, the operations that will applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined. The elements of class-based model include classes and objects, attributes, operations, CRC models, collaboration diagram and packages.

UML Class Diagrams

We've identified nouns from the requirements and made them as attributes and identified the verbs and made them as method of functionality of the class objects

These classes are shown below-

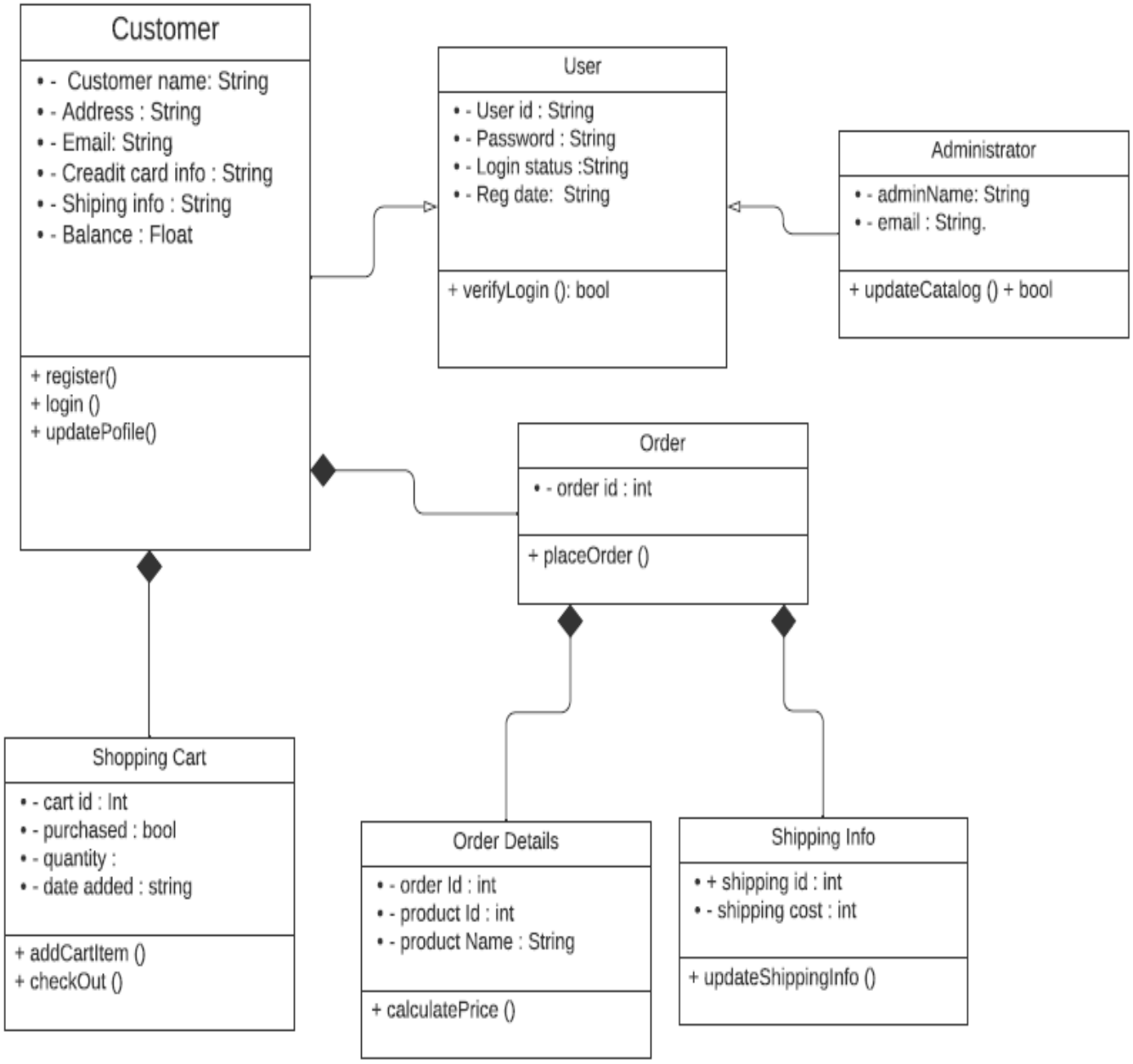

Figure 6 : UML class diagram of REACH

Class Responsibility Collaboration Diagram(CRC model)

A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

CRC Model helps us to understand how the classes will interact to each other and how they are connected with each other.

Here, we've defined CRC Model of "REACH" –

User	
Responsibilities	**Collaboration**
Browse	
Proceed	Client
Checkout	Admin
Turn on location	Worker
Order	
Store Data	

Table : CRC model of Object User

Admin	
Responsibilities	**Collaboration**
Store payment details Update work details Verify login Handle Exception	User Worker

Table : CRC model of Object Admin

Order	
Responsibilities	**Collaboration**
Place order Authentication Request Proceed Payment	Worker Client

Table : CRC model of Object Order

Worker	
Responsibilities	**Collaboration**
Search Select service View details Turn on location Make availability Confirmation	User Availability info

Table : CRC model of Object woker

Client	
Responsibilities	**Collaboration**
Browse Select service Add to cart Order Payment Write Review	Work details Order Review

Table 2 : CRC model of Object Admin

Chapter 7

Flow Oriented Model

Introduction

Although data flow-oriented modeling is perceived as an outdated technique by some software engineers, it continues to be one of the most widely used requirements analysis notation in use today.

Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as preliminary step create an overview of the system. The DFD takes an input-process-output view of a system. Data object flow into the software, are transformed by processing elements and resultant data objects flow out of the software. Data object are represented by labeled arrows and transformations are represented by circles. A DFD shows what kind of information will be input and output from the system, where the data will come from and go to, and where the data will be store.

DFD of Level: 0 of " REACH"

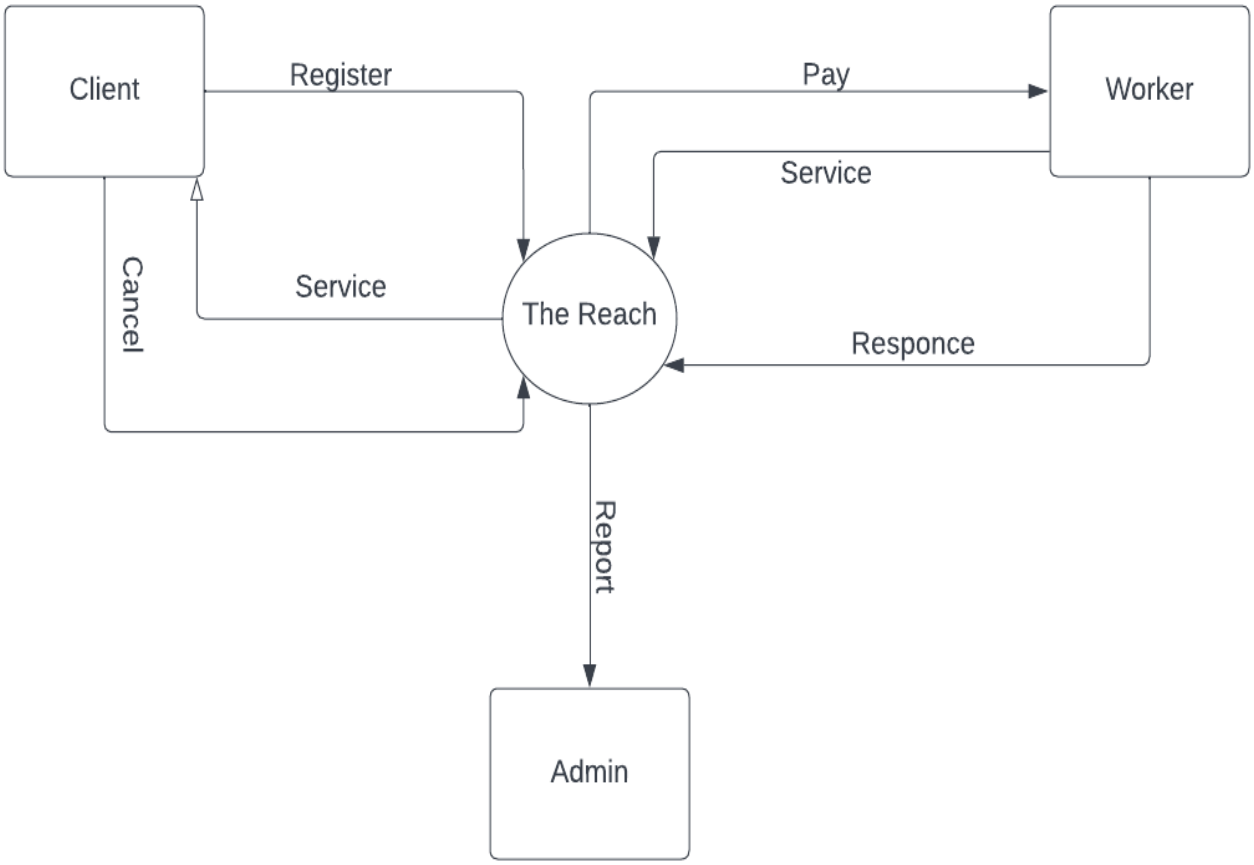

Figure 7 : Leve: 0 DFD of REACH

DFD of Level: 1 of "REACH"

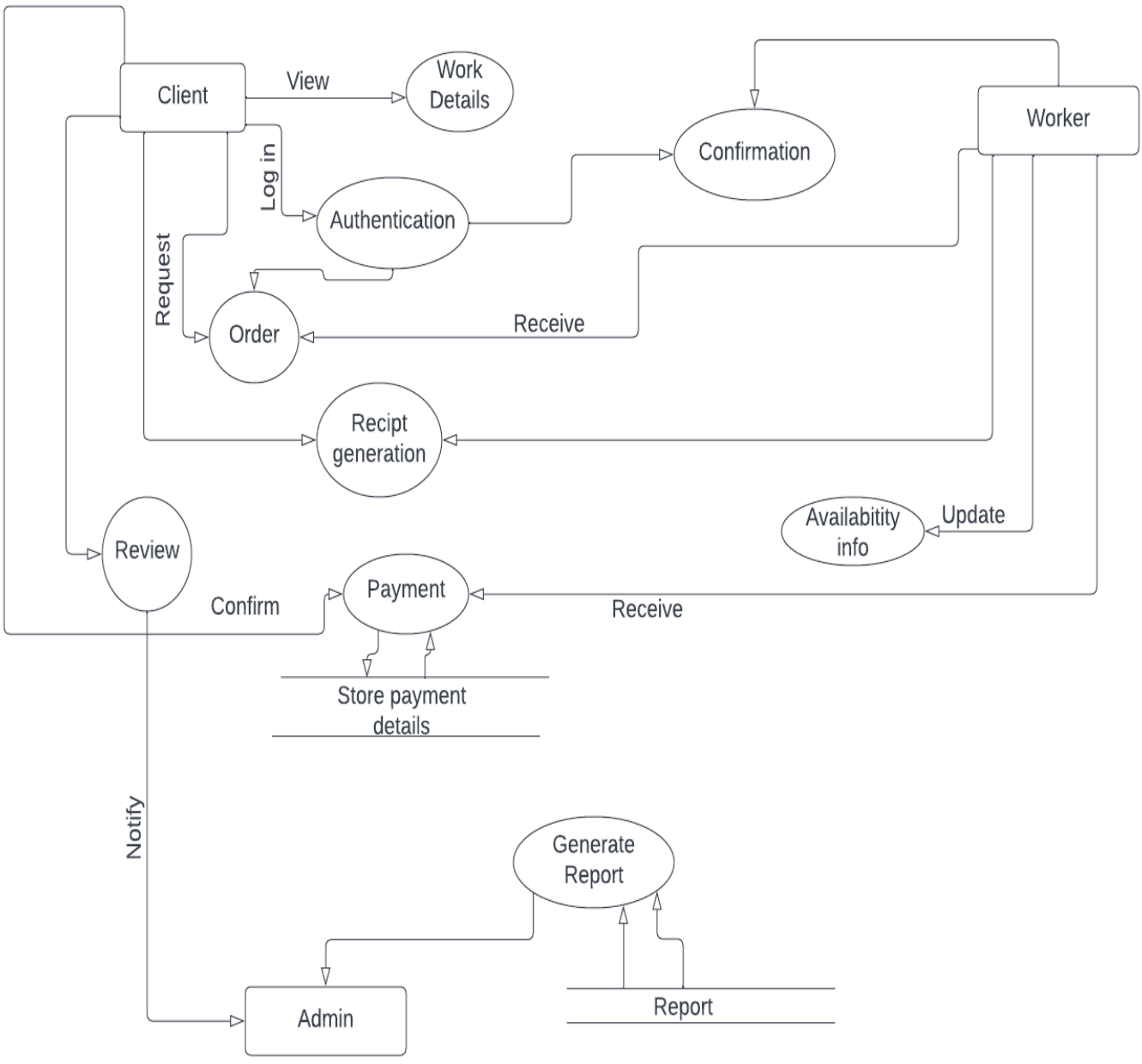

Figure 8 : Level: 1 DFD of REACH

DFD of Level: 2 of "REACH"

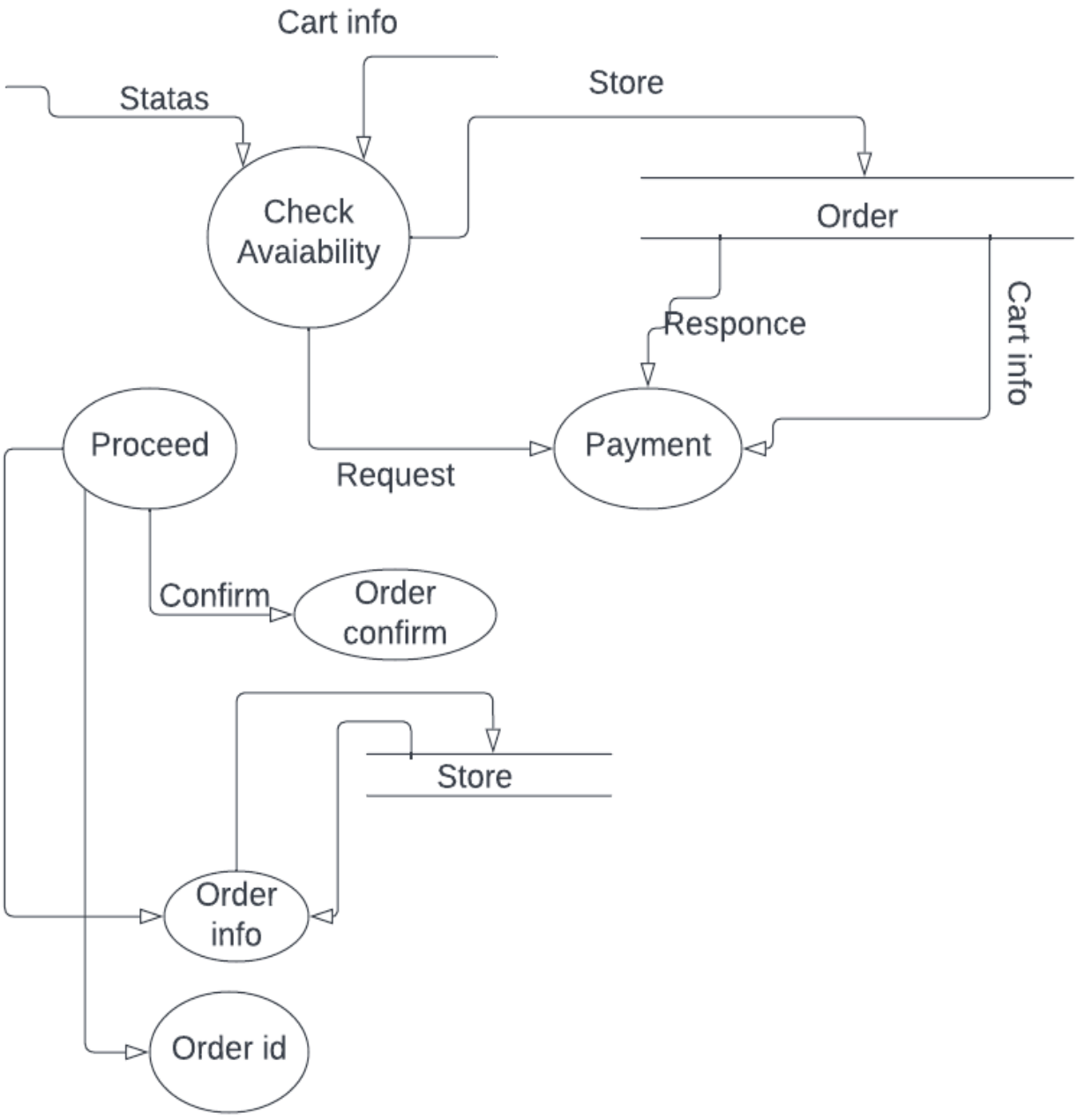

Figure 9 : Level 2: DFD diagram of REACH

Chapter 8

Behavioral Model

Introduction

Behavior modeling is also referred to as State modeling, State machines and State transition matrix. Behavior modeling is when one thinks of his ideas in terms of states and transitions. This requires both identifying all of the interesting states of being that software or its components are likely to be in. And also, at a high level, abstracting what events are likely to cause software or its components to change between states of being.

The behavioral model indicates how software will respond to external events or stimuli. To create the model, we have performed the following steps:

1. Evaluate all use cases to fully understand the sequence of interaction within

the system.

2. Identify events that drive the interaction sequence and understand how these

events relate to specific objects.

3. Create a sequence for each use case.

4. Build a state diagram for the system.

5. Review the behavioral model to verify accuracy and consistency.

Identifying Events

Here we have identified events from the **usage scenario** and listed their corresponding initiators and collaborators.

Events	Initiator	Collaborative Class
Login	All Users (Client, Worker, Admin)	System
Posts for jobs	Client	System
Updates availability	Worker	System
Updates work detail	Admin	System
Responses to the post	Worker	Client
Adds order	Client	System
Makes order	Client	System
Makes payment	Client	Worker
Writes review	Client	Worker

Table 12: Events Table of REACH

State Diagram

State diagram represents active states for each class the events (triggers). The following section will show state diagrams for each class in "REACH" app-

State Transition Diagram of User

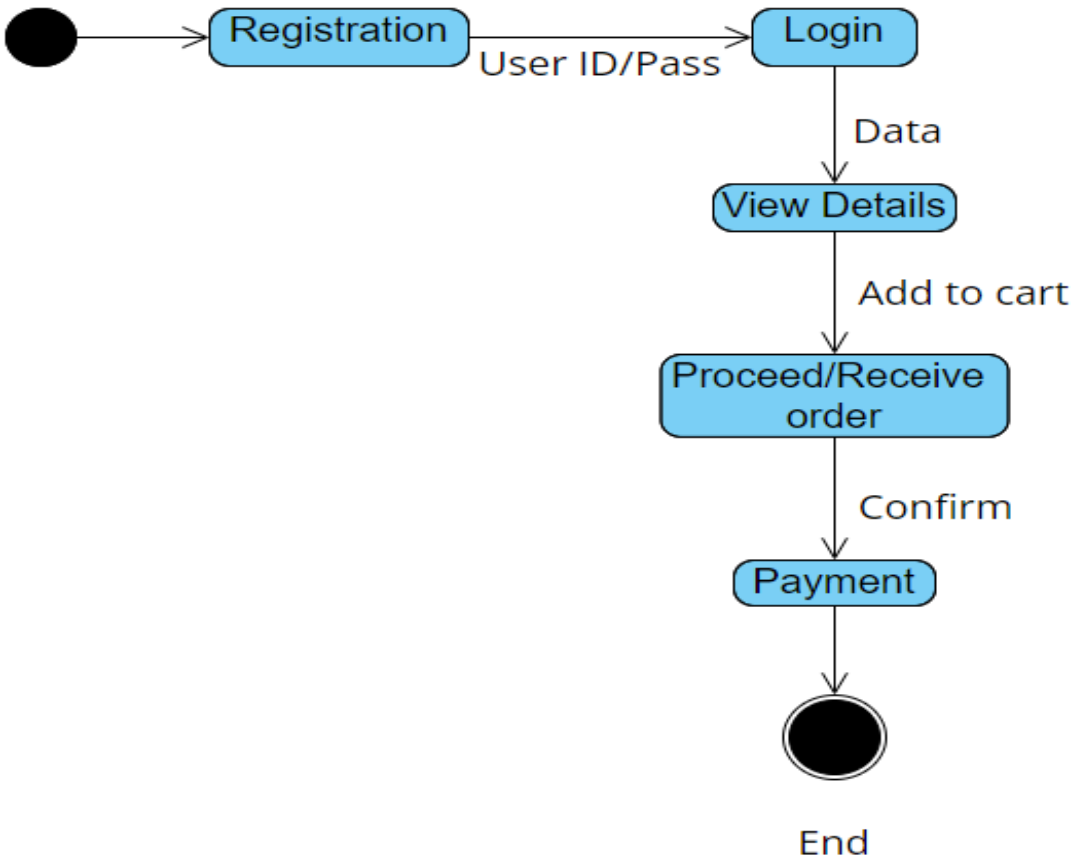

Figure 10 : State Diagram for User in "REACH"

State Transition Diagram of worker

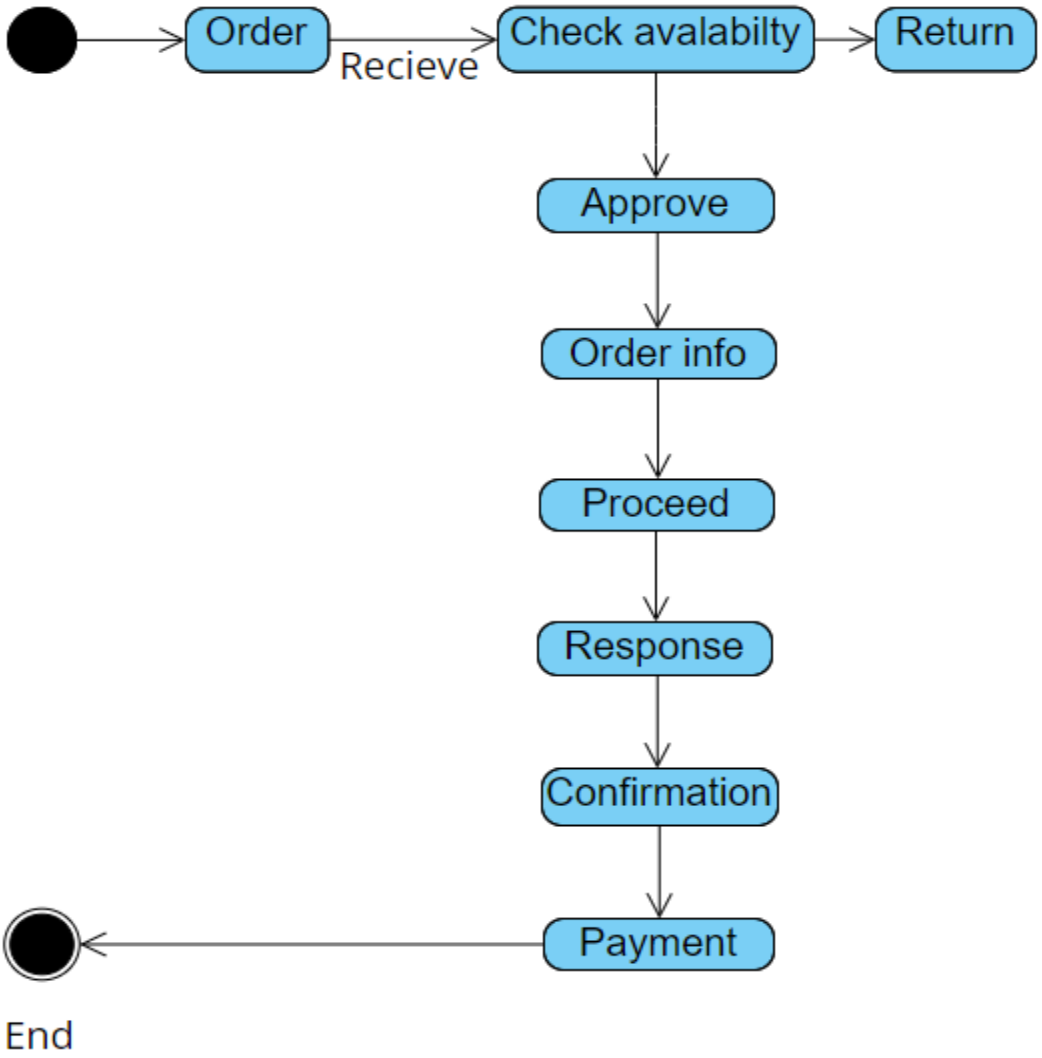

Figure 11 : State Diagram of the Worker in "REACH"

Chapter 9

Conclusion

It is extremely difficult to represent something real on paper. We are glad that we have been able to complete the requirements analysis of 'REACH' app and publish the SRS of the project. I believe that this report has been written as easy as possible manner and contains all the material necessary to have a complete understanding of the concept. I hope that any reader who pursues this document may quickly grasp the concept behind the 'REACH' application. I also hope that it will be an easy-to-follow document for implementing the application!

Appendix

References

1 Roger S. Pressman_ Bruce R. Maxin - Software Engineering_ A Practitioner's Approach-McGraw-Hill Education (2014)

2.Sommerville, I. Software Engineering, 7th ed. Harlow, UK: Addison Wesley, 2006
3. https://badr.co.id
4. Software requirements specification – Wikipedia - https://en.wikipedia.org
5. https://www.geeksforgeeks.org/software-requirement-specification-srs-format/

Diagram Design Tools:

1. **Visual Paradigm** - https://www.visual-paradigm.com
2. **Lucidchart** - https://www.lucidchart.com