NearWear: A Dress Sharing Platform

One semester long project report to be submitted in partial fulfillment of the requirements for the degree

Of

Bachelor of Science in Computer Science and Engineering

by

Tania Aktar Jhoma
182014055

Under the supervision of

Nafees Mansoor, PhD
Assistant Professor

Computer Science and Engineering
University of Liberal Arts Bangladesh

20th October, 2022

DECLARATION

This project report is submitted to the Computer Science and Engineering, University of Liberal Arts Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science. So, I hereby, declare that this project report is based on the surveys found by me. Materials of work found by other researchers are mentioned by reference. This project report, neither in whole, nor in part, has been previously submitted for any degree.

Tania Aktar Jhoma
ID: 182014055
Email Address: tania.aktar.cse@ulab.edu.bd

CERTIFICATE OF APPROVAL

The project report entitled "NearWear" is submitted to the **Computer Science and Engineering**, University of Liberal Arts Bangladesh, Dhaka in partial fulfillment of the requirements for the degree of Bachelor of Science**.**

Dated: 20th October 2022

Muhammad Golam Kibria, PhD
Associate Professor & Head (Acting)
Computer Science and Engineering
University of Liberal Arts Bangladesh

Signature & Date

Nafees Mansoor, PhD
Assistant Professor & Supervisor
Computer Science and Engineering
University of Liberal Arts Bangladesh

Signature & Date

ACKNOWLEDGEMENT

At the completion of this report, much support, guidance, and direction are needed and I am fortunate to achieve my goal with the help of a few people. I am indebted to my supervisor, Nafees Mansoor PhD, Assistant Professor in the Department of Computer Science and Engineering of ULAB, for his patience, guidance, and support.

The Department of Computer Science & Engineering of ULAB also remains one of the key players as it has provided me with the opportunity to nurture my skills and develop myself.

Finally, I must express my immense gratitude to my father, Md Abdul Jalil. I am grateful for his unconditional, unequivocal, and tremendous support. His constant love and support keep me motivated and confident. My accomplishments and success are because he believed in me.

ABSTRACT

Common natures of human beings are to upgrade their lifestyles by upgrading accommodation, food, vehicle, or dresses. But anything that is better costs a lot. Most of us can’t afford upgradation due to price factor. People with low incomes can't buy new dresses for different events. While reaching people buy a dress and lock it in the wardrobes. People don't wear at least 50 percent of their wardrobes. Prices of clothes have a 15-20% hike compared to the previous year. Minimizing the gap between these two classes and using better use of the money we have come up with “NearWear”. This will save time, money, and space for wardrobes. People with costly dresses will upload their dresses photo on the app “NearWear” and people who cannot buy them will take them on rent to fulfill their wishes to wear better dresses at Special Events. Thus, buying such a dress will not be a waste of money. Cause whenever someone will rent it the owner of the dress will get a payment. Also, people with low income will be able to wear the dresses they dream about by using this app to rent the dress. The service includes all the required features to be able to enjoy the dress-sharing service worldwide. As the service highly maintains the technical and non-technical requirements, it will be able to serve a huge number of users simultaneously.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	ACKNOWLEDGEMENT	**iv**
	ABSTRACT	**v**
	LIST OF TABLES	**viii**
	LIST OF FIGURES	**ix**
	LIST OF ABBREVIATIONS	**x**
1	**INTRODUCTION**	**1**
	1.1 Overview and Motivation	1
	1.2 Problem Statement	1
	1.3 Proposed Solution	2
	1.4 Objectives	2
	1.5 Expected Outcome	2
	1.6 Organization of the Report	3
2	**LITERATURE REVIEW**	**4**
	2.1 Introduction	4
	2.2 Existing Systems	6
	2.2.1 YourCloset (yourcloset.com.au)	4
	2.2.2 By Rotation (byrotation.com)	5
	2.2.3 Rent the Runway (renttnerunway.com)	5
	2.2 Comparative Analysis	6
3	**REQUIREMENTS ANALYSIS**	**8**
	3.1 Technical Requirements	8
	3.1.1 Target Platform(s)	8
	3.1.2 Hardware Requirements	8
	3.1.3 Software Requirements	9
	3.2 Development Requirements	9
	3.2.1 Hardware Requirements	9
	3.2.2 Software Requirements	10
	3.3 Development Stack	10

3.3.1 Frontend Stack 10
3.3.2 Backend Stack 10
3.4 Non-technical Requirements 11
3.4.1 Performance 11
3.4.2 Scalability 11
3.4.3 Security & Reliability 11
3.4.4 Availability 14
3.5 Business Model 12
3.6 Specifications 12
3.6.1 User Roles 12
3.6.2 User-Flow and Capabilities 13
3.6.2.1 Provider 13
3.6.2.2 Renter 13
3.6.2.3 Administrator 14

4 DESIGN AND IMPLEMENTATION 15
4.1 Introduction 15
4.2 Technical Diagrams 16
4.3 API (Backend) 19
4.3.1 The REST Architecture 19
4.3.1.1 HTTP Methods 19
4.3.1.2 Endpoint Architecture 20
4.3.2 Authentication System 20
4.3.3 Endpoints 21
4.4 Database Schema Diagrams 24
4.5 UI/UX Design 25
4.5.1 Auth UI 25
4.5.2 Provider UI 26
4.5.3 Renter UI 27
4.5.4 Admin UI 28

5 RESULT AND DISCUSSION 29
5.1 Introduction 29
5.2 Tests & Assessments 29
5.3 Discussion 30

6 CONCLUSION & FUTURE PLANS 31
6.1 Endwords 31
6.2 Future Plans 31

REFERENCES 33

LIST OF TABLES

NO	TITLE	PAGE
2.2	Comparison Among Various of Existing Systems	6
4.1	Authorization Endpoints	21
4.2	User Endpoints	22
4.3	Dress Endpoints	22
4.4	Message Endpoints	22
4.5	Order Endpoints	23
4.6	Review Endpoints	23

LIST OF FIGURES

NO	TITLE	PAGE
2.1	YourCloset Home Screen	4
2.2	By Rotation Home Screen	5
2.3	Rent the Runway Home Screen	6
4.1	Use Case Diagram	16
4.2	Data Flow Diagram	17
4.3	Entity Relationship Diagram	18
4.4	REST Architecture	19
4.5	Authentication System	21
4.6	Schema Diagram of the proposed System	24
4.7	(a) Landing Screen, (b) Login/SignUp Screen, (c) SignUp Screen, (d) SignUp Form, (e) OTP Screen, (f) Registration Success Screen	25
4.8	(a) Provider Dashboard Screen, (b) Terms & Condition Popup Screen, (c) Dress Add Screen, (d) Single Dress Screen, (e) History Screen	26
4.9	(a) Renter Dashboard Screen, (b) Order Summary Screen, (c) Order Details Screen, (d) Points Screen	27
4.10	Admin Dashboard Screen	28

LIST OF ABBREVIATIONS

ABBREVIATION	DEFINITION
1. API	Application Programming Interface
2. DBaaS	Database as a Service
3. UI	User Interface
4. UX	User Experience
5. REST	Representational State Transfer
6. HTTP	Hypertext Transfer Protocol
7. MVP	Minimum Viable Product
8. ERD	Entity Relationship Diagram
9. DFD	Data Flow Diagram

CHAPTER 1

INTRODUCTION

1.1 Overview And Motivation

Over the past century, the demands for proper clothes remained high, especially in occasional events. Events require proper dressing, depending on the type of event. This requires a lot of investment in clothes, which is not optimal. In recent years, ride-sharing services have become popular and serve people very well [1]. We think in the same way, but in a different domain – "dress".

A study shows that people don't wear at least 50 percent of their wardrobes [2]. Also, clothes have a 15-20% price uplift for the current year, compared to the previous [3]. There are many occasional events where people need to buy new dresses. But buying clothes every time for the events doesn't make sense as they'll probably use the dress once a year or so. The money, time, and space that is invested to have these dresses are more or less a waste. To encounter this issue and save time, money, and space - A new dress-sharing platform can be introduced where people can share their dresses for occasional events [4]. This way, the cloth owner will be able to earn extra money and the renter gets his/her job done with minimal spending of money and time.

1.2 Problem Statement

The problem relies on the financial dependency on dresses. To wear good dresses, a person needs to be financially capable. And as the financial restrictions really divide the society into pieces, good dresses should be available to most of the people and they should not be restricted by huge price tags.

1.3 Proposed Solution

Our proposed solution is a platform called “NearWear” – which acts as a middleman between the provider and renter. The goal is to make dresses available to most of the people, without really depending on financial requirements. This platform will make the most expensive dresses to be available at a cheap price, allowing the renter to give it a try. On the other hand, it will allow dress owners to earn some extra cash on the expensive dresses they own but use rarely.

It will help to solve the financial dependency on the dresses we wear, as the dresses are not restricted by the huge price tags, but instead - available for anyone to rent for a limited time, at a cheap rate.

1.4 Objectives

There are many objectives behind our service. The most important ones are the following –

- New passive income source
- Multiple payment methods
- Lessen waste of time, money, and space (No need to spend a big amount for occasional dresses, no need to go for shopping physically, no need to burden your collection of dresses)
- User own point system to award users
- A review system users can rely on (Cumulative average review systems on multiple factors)
- Earn money by referring

1.5 Expected Outcome

- A better world with less waste of time, money, and space
- Maximum usage of dresses
- A new way of passive earnings

1.6 Organization of the Report

The report includes figures, tables, and abbreviations for the discussion and explanation of the topics. Please refer to the “FIGURES”, “TABLES” and “ABBREVIATIONS” sections to get further details.

The first chapter – “Literature Review”, consists of 3 similar existing systems in the market with a comparative analysis of various aspects of the services. We briefly discuss the workings of each system and compare.

The next chapter, “Requirement Analysis” includes all the requirements analytics for the project. There is business analysis, specifications analysis, technical and non-technical requirement analysis

The “Design and Implementation” – chapter includes all the aspects of designing the service, for UI to the backend to the database schema. Also, it includes the implementation strategies of the backend and database. The backend is an API – which includes more than 25 endpoints. This report includes tables for each endpoint to showcase the intended use case.

Next, “Result and Discussion” includes the final project results and a discussion upon it. This includes the discussion on the testing and assertions, future improvement, and discussion about the current developed system.

Finally, the “Reference” chapter includes all the references to the resources that have been cited throughout the report.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

"NearWear" – a dress-sharing platform where providers can list their dresses and renter can rent a dress with minimal time and cost. Our goal is to build up the platform in a way where it will not only serve us profit but also serves the people very well throughout its lifetime.

2.2 Existing Systems

The existing system refers to the similar systems that are being served currently. We have gone through several services that exist and chosen the top 3 of them to compare.

2.2.1 YourCloset (yourcloset.com.au)

Briella Brown, at just 17, founded Australia's premier online luxury dress rental destination which is known as "Your Closet". It allows them to rent luxury Australian and International designer collections and they believe buying a beautiful product isn't the only way to experience it.

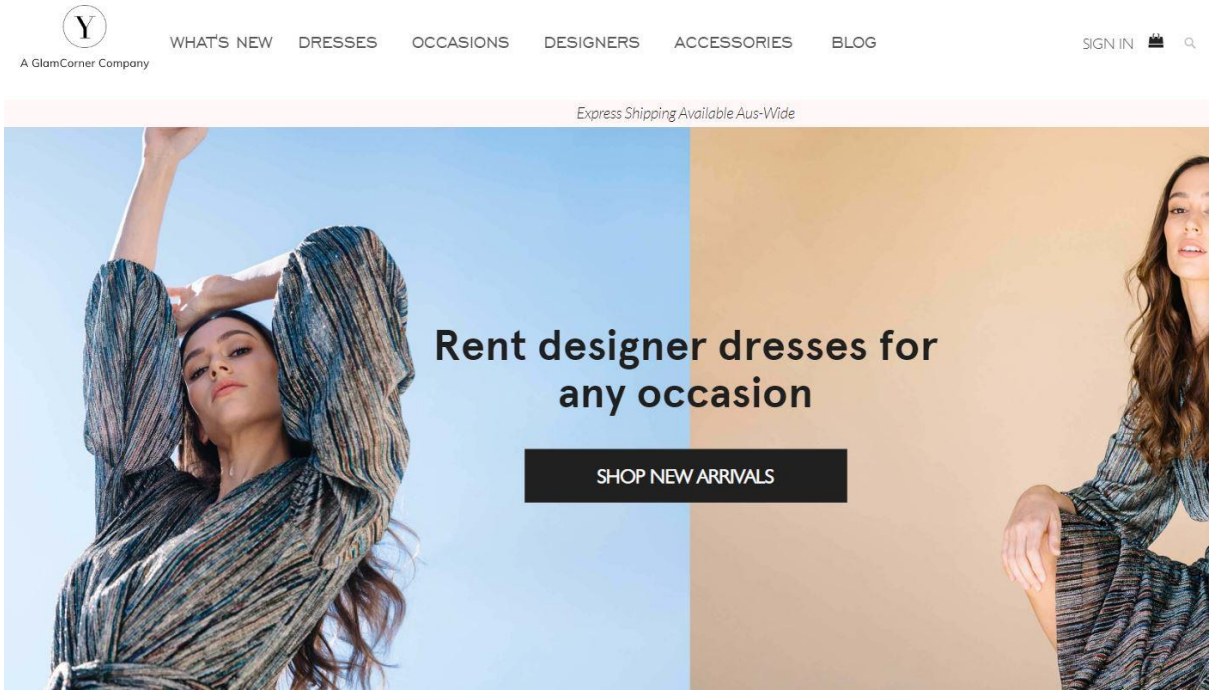


Fig 2.1: (a) YourCloset Home Screen

2.2.2 By Rotation (byrotation.com)

In October 2019, An Indian woman Eshita Kabra has launched the world's first social rental app, "By Rotation". Through community-powered platforms like By Rotation, they empower and connect like-minded individuals to transform the way they consume fashion.

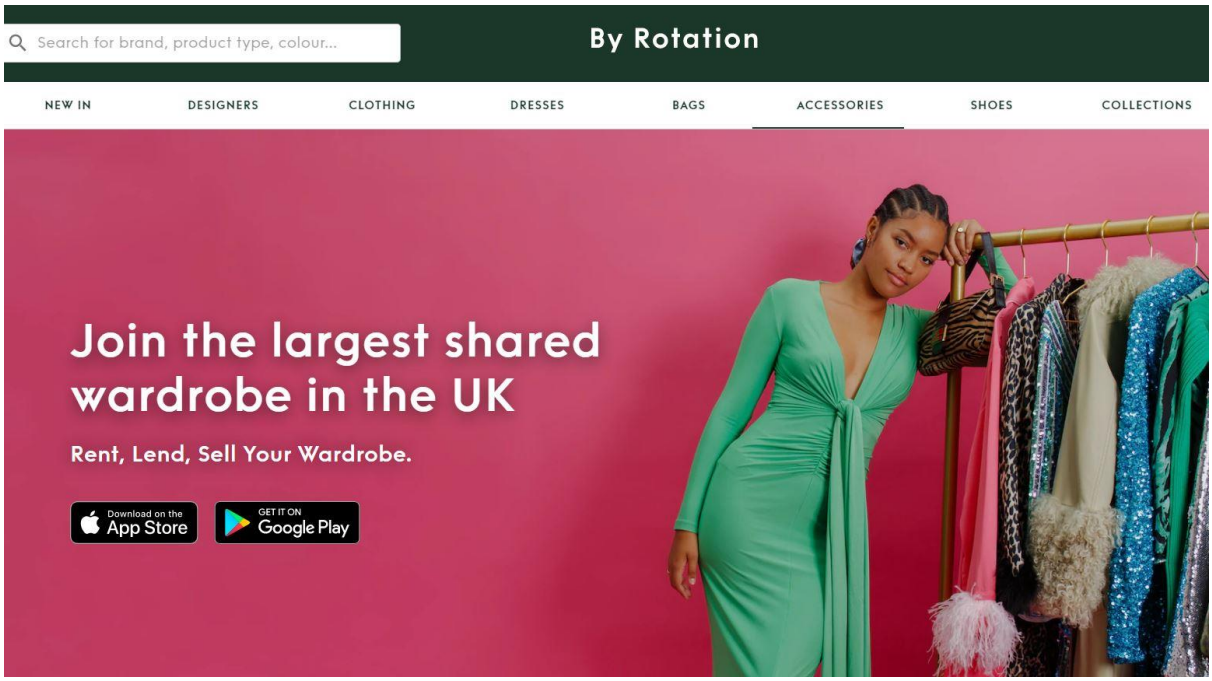


Fig 2.2: (b) By Rotation Home Screen

2.2.3 Rent the Runway (renttherunway.com)

In November 2009, Jennifer Hyman and Jennifer Fleiss founded the company "Rent the Runway". Users can rent, subscribe, or purchase designer apparel and accessories on Rent the Runway's e-commerce platform.

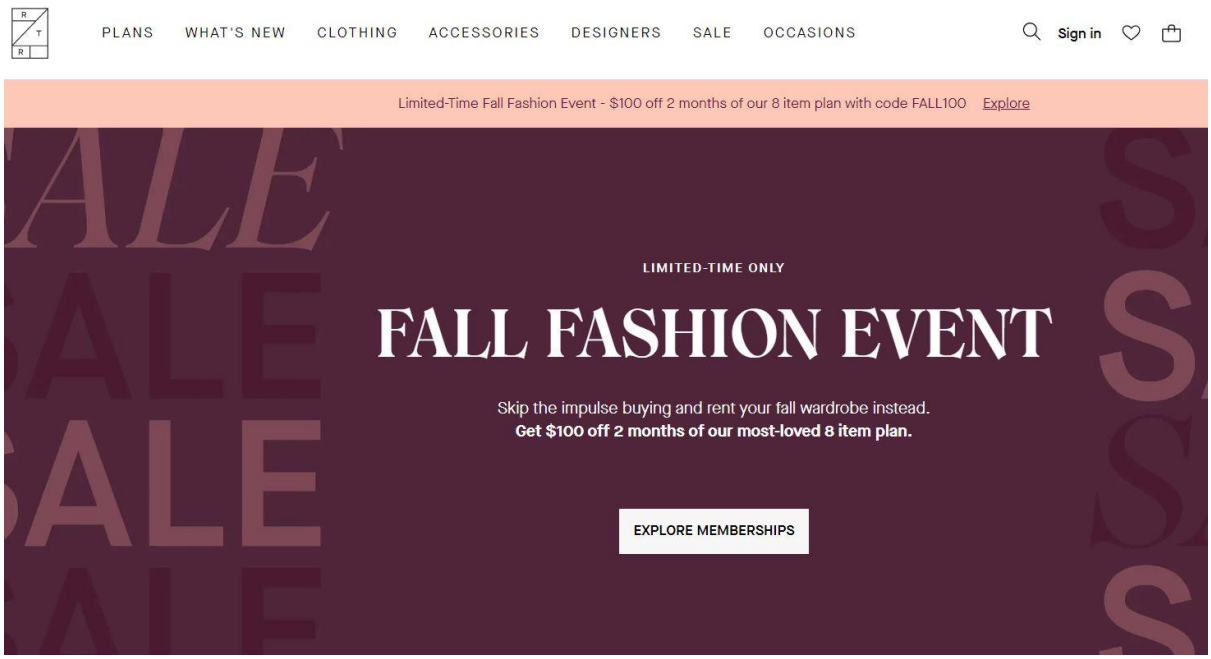


Fig 2.3: (c) Rent the Runway Home Screen

2.2 Comparative Analysis

Table 2.1: Comparison among various of the Existing Systems

	YourCloset	**By Rotation**	**Rent The Runway**	**NearWear**
Service Type	One-Time Rental	One-Time Rental	One-Time Rental / Monthly Subscription	One-Time Rental
Minimal Rental Time	4 days	3 days	4 days	1 day
Review System	No	No	Yes	Yes
Message / Chat	No	No	No	Yes
Offerings	In-house dresses	3rd party users	In-house dresses	Both in-house and 3rd party dresses
Platform	Web	Web	App & Web	App
Payment Method	Online	Online	Online	Cash on Delivery
Live Tracking	No	No	No	Yes

In the table above, we did a comparison among all the existing services and ours - a comparison between YourCloset, ByRotation, RentTheRunway and NearWear. “YourCloset” and “By Rotation” and “NearWear” offers a one-time rental service, where “Rent the Runway” offers both a one-time rental and monthly subscription. The minimal rental period is 4 days for “YourCloset” and “Rent the Runway”, 3 days for “By Rotation” and the lowest – 1 day for NearWear. Among all the compared services, only “Rent Runway” and “NearWear” includes a review/rating system. None of the other services includes messaging system except “NearWear”. “YourCloset” and “Rent the Runway” offers in-house dresses, whereas “By Rotation” offers 3rd party vendors and for “NearWear” – it offers both in-house dresses and 3rd party vendors. “YourCloset” and “By Rotation” and “Rent the Runway” don’t offer any live tracking systems, although these 3 platforms are following online payment methods. But “NearWear” offers both cash on delivery and live tracking systems.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 Technical Requirements

Technical requirements are one of the crucial parts of software development as it assumes the average requirements of both hardware and software to run the software or service. Also, it includes the development requirements – which are required to develop the system. Development Stack is the infrastructure - the set of technologies that are used to build the software/service.

3.1.1 Target Platform(s)

This service will be available as a mobile application initially. But it will expand its journey to all other platforms as well. The target platforms include:

- Android / iOS
- Web *
- Linux / Windows / MacOS *

[Note: asterisks (*) indicated platform's builds will be available in further development cycle]

3.1.2 Hardware Requirements

Android / iOS

The ***minimum*** hardware requirement includes:

CPU	Dual Core 1.0+ GHz
RAM	2GB

The ***recommended*** hardware requirement includes:

CPU	Hexa Core 1.8+ GHz
RAM	4GB

Web

The ***minimum*** hardware requirement includes:

CPU	Dual Core 1.0+ GHz
RAM	2GB

The ***recommended*** hardware requirement includes:

CPU	Quad Core 1.8+ GHz
RAM	4GB

Linux / Windows / MacOS

The ***minimum*** hardware requirement includes:

CPU	Dual Core 1.5+ GHz
GPU	DirectX9 supported graphics driver
RAM	2GB

The ***recommended*** hardware requirement includes:

CPU	Hexa Core 3.5+ GHz
GPU	DirectX10 / DirectX11 supported graphics driver
RAM	4GB or more

3.1.3 Software Requirements

Android / iOS

Android	Android 5.1 or better
iOS	iOS 7 or better

Web

Operating System	Any
Browser	Google Chrome v35.0 or later / Mozilla Firefox 24.0 or later / Safari 11 or later / JavaScript enabled browsers

Linux / Windows / MacOS

Linux	All Linux Distribution
Windows	Windows 7 or later
MacOS	MacOS 11 or later

3.2 Development Requirements

The development requirements include the hardware, and software requirements to develop a system/service.

3.2.1 Hardware Requirements

The ***minimum*** hardware requirement includes:

CPU	Quad Core 2.5+ GHz
GPU	DirectX9 supported graphics driver
RAM	8GB

The ***recommended*** hardware requirement includes:

CPU	Octa Core 3.5+ GHz

GPU	RTX 3050 or better / RX 6500 XT or better
RAM	16GB

3.2.2 Software Requirements

Operating System	Windows / Linux / MacOS
Android Studio / Emulator (Optional)	Arctic Fox (2020 3.1) or later
Visual Studio Code	Version 1.5 or later
Git	Version 2.17 or later

3.3 Development Stack

UI/UX Design	Figma
Frontend	React Native / Expo
Backend	Node.js with Express.js API
Database	MongoDB Atlas (NoSQL Cloud Database as A Service)

3.3.1 Frontend Stack

- Expo (React Native)
- Redux
- External Packages from npm (Axios, Axios-Hooks, DateTimePicker, etc.)

3.3.2 Backend Stack

- Node.js
- Express.js server for API
- Postman – API Testing Tool
- External Packages from npm (jsonwebtoken, validator, etc.)

3.4 Non-technical Requirements

Non-technical requirements include all other requirements aspects of a service or system except technical, such as performance, stability, scalability, security, etc.

3.4.1 Performance

The performance really depends on 3 things mainly:

1. Applications optimizations
2. The hardware capability
3. Internet connection speed

As the service provider, we cannot target the latter two but really can focus on the optimizations of the services. The service ensures better application optimizations by optimizing 2 main bottleneck points of the system:

Backend API Optimization - The performance of every function was tested using the Big-O notation of time and space complexity analysis.

Efficient Database Schema Design – Database Schema design and analysis are crucial as an optimal design can save minutes of time and a huge amount of space. The database schema for the application was designed in a way that can retrieve the most used queries in a fraction of a second and even the most complex queries within 1-2 seconds of time. The time depends on the query and size of the database/collections.

3.4.2 Scalability

API Backend - The backend service will be running inside a docker container. In need of scalability, our system can be integrated with Kubernetes Clusters, which automatically scale up / down the service based on the traffic load and offer a built-in load balancer.

Database – The service uses MongoDB Atlas as a DBaaS (Database as A Service). I need any scaling; it is a matter of clicks with competitive cost of service and maintenance

3.4.3 Security & Reliability

User Authentication – User authentication is done with the help of jsonwebtoken, which is a tool used for generating tokens for authenticated users. The tool uses cryptographically complex algorithms to generate the tokens – which is cannot be cracked.

Infrastructure - The backend infrastructure security will be given by the cloud service providers like AWS or Google Cloud. They ensure a rock-solid, their own infrastructure to host our backend services. For the database, MongoDB Atlas also provides rock-solid security for the data. As the infrastructure is built upon a could service provider – the service will be stable and reliable.

3.4.4 Availability

Our service targets the most common platforms, thus making it possible to use our service from anywhere, to almost any machine! The cloud service providers ensure a server uptime of more than 99%.

3.5 Business Model

Our service is a dress-sharing e-commerce platform, driven by the dress provider and renter with minimal hassle and cost. The business concept is very simple and straightforward – a dress-sharing platform where users can share dresses, allowing providers to earn extra money and renters to get their job done with minimal cost.

As the business profit margin truly depends on the number of sales done via the platform, the business logic should meet the following requirements:

- No direct interaction between provider and renter to get rid of off-platform deals
- Direct interaction between admin and provider / admin and renter for discussion / complain / reports
- Minimal profit per sale, to decrease the platform cost for users as much as possible and increase the total amount of sales instead.

3.6 Specifications

3.6.1 User Roles

This platform includes 3 types of users as follows:

1. Provider – The dress provider, who lists dresses for rent
2. Renter – The renter, who rents dresses for a certain period of time
3. Admin – The administrator, who manages and regulates the platform

3.6.2 User-Flow and Capabilities

3.6.2.1 Provider

A provider is a person who lists dresses and provides it for rent. The provider's capabilities include:

- SignUp
- Verify Account (Optional)
- Login
- Manage Profile
- Create New Listing
- Manage Listings
- Check Earnings
- Withdraw Funds
- Check Orders & Status
- Message Admin

3.6.2.2 Renter

A renter is a person who 'rents' dresses for a specific period of time. The renter's capabilities include:

- SignUp
- Verify Account (Optional)
- Login
- Manage Profile
- Search Dresses
- Buy / Rent Dresses

- Review Dresses
- Check Orders & Status
- Message Admin

3.6.2.3 Administrator

An administrator is a person who maintains and regulates the platform. Administrator capabilities include:

- Manage Users
- Manage Orders
- Review Transactions
- Review Reviews
- Manage Complains / Issues via Message

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Introduction

The design and implementation refer to the design and implementation of the system, both frontend UI and backend API with the database schema. This chapter also includes the diagrams such as use case diagram, entity relationship diagram, and data flow diagram. The API follows the REST architecture – which is the most popular API architecture available. The endpoint follows the conventional patterns for requesting resources with the GET, POST, PUT, PATCH, DELETE http methods. The endpoint includes more than 25 endpoints, which are described in tables later in this chapter.

The user interface is designed using the UI design tool “Figma”. The screens that are designed, follow the easy-to-use and intuitive components to support a smooth user experience. The application was built using the popular JavaScript framework, React Native. It supports cross-platform development, which helps to keep one single codebase to maintain, having the most identical application across the platforms. Expo, a wrapper of React Native allows for easily develop / run / deploy the application with extra added features like an image picker, sensor trackers, camera controls and many more. Expo also provides its own library of components to support a faster development cycle for time limited projects.

Moving toward the last piece of the puzzle, the database schema was designed to have all the necessary connections between the collections, which provides a balance between storage and efficiency. The database includes total of 7 collections including dresses, users, sessions, reviews, messages, orders, and groups. Each of the collections maintains the relation between the other collection were necessary to efficiently retrieve the most used information in need.

4.2 Technical Diagrams

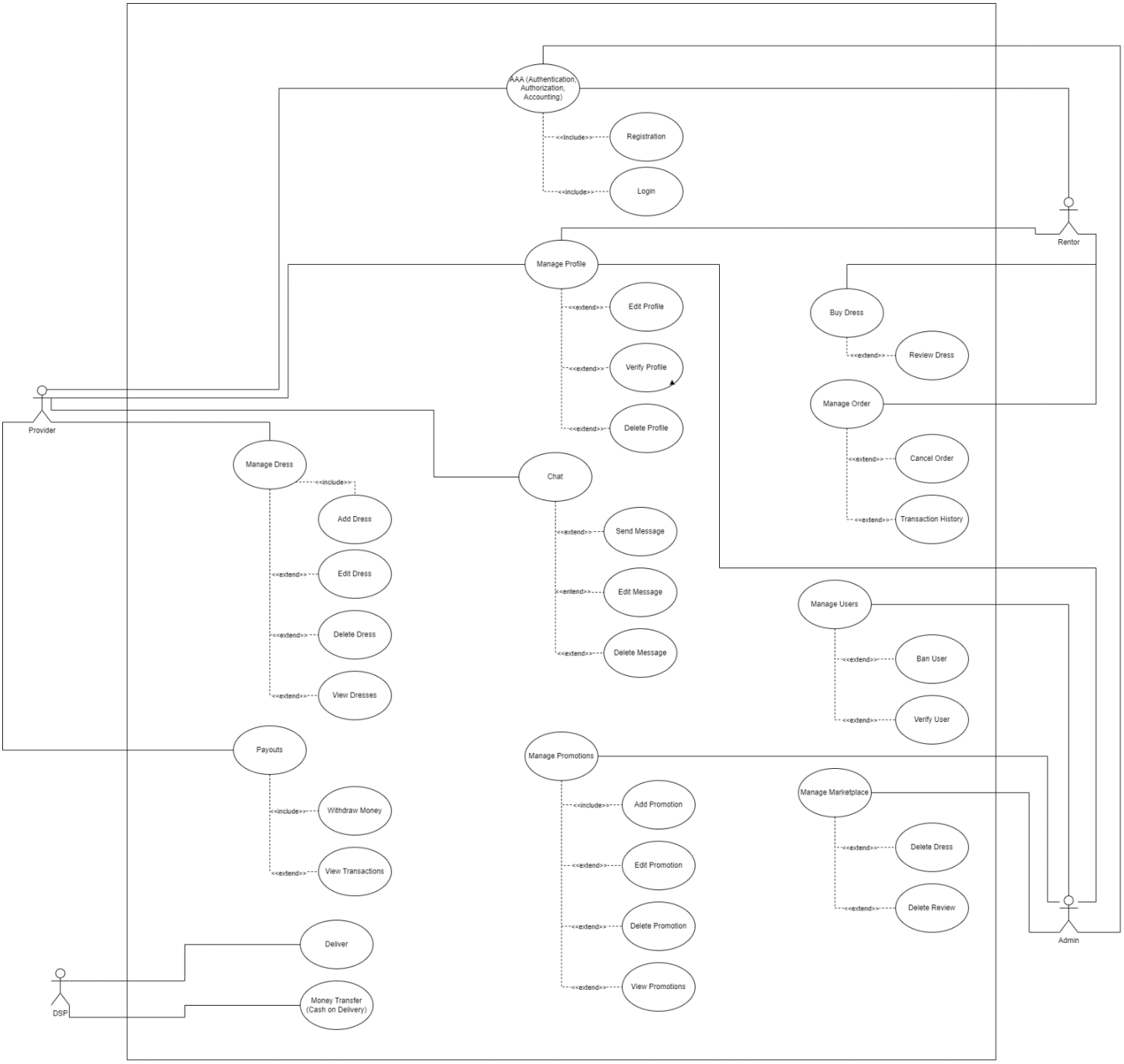


Fig 4.1: Use Case Diagram

In fig 4.1, the use case diagram of the proposed system has been depicted. The use case diagram includes 3 entities such as provider, renter, and admin. It demonstrates the use cases for each entity. The “extend” use cases are optional use cases, whereas the “include” use cases are the vital ones. The dotted line connectors indicate that the action can be taken multiple times by an entity.

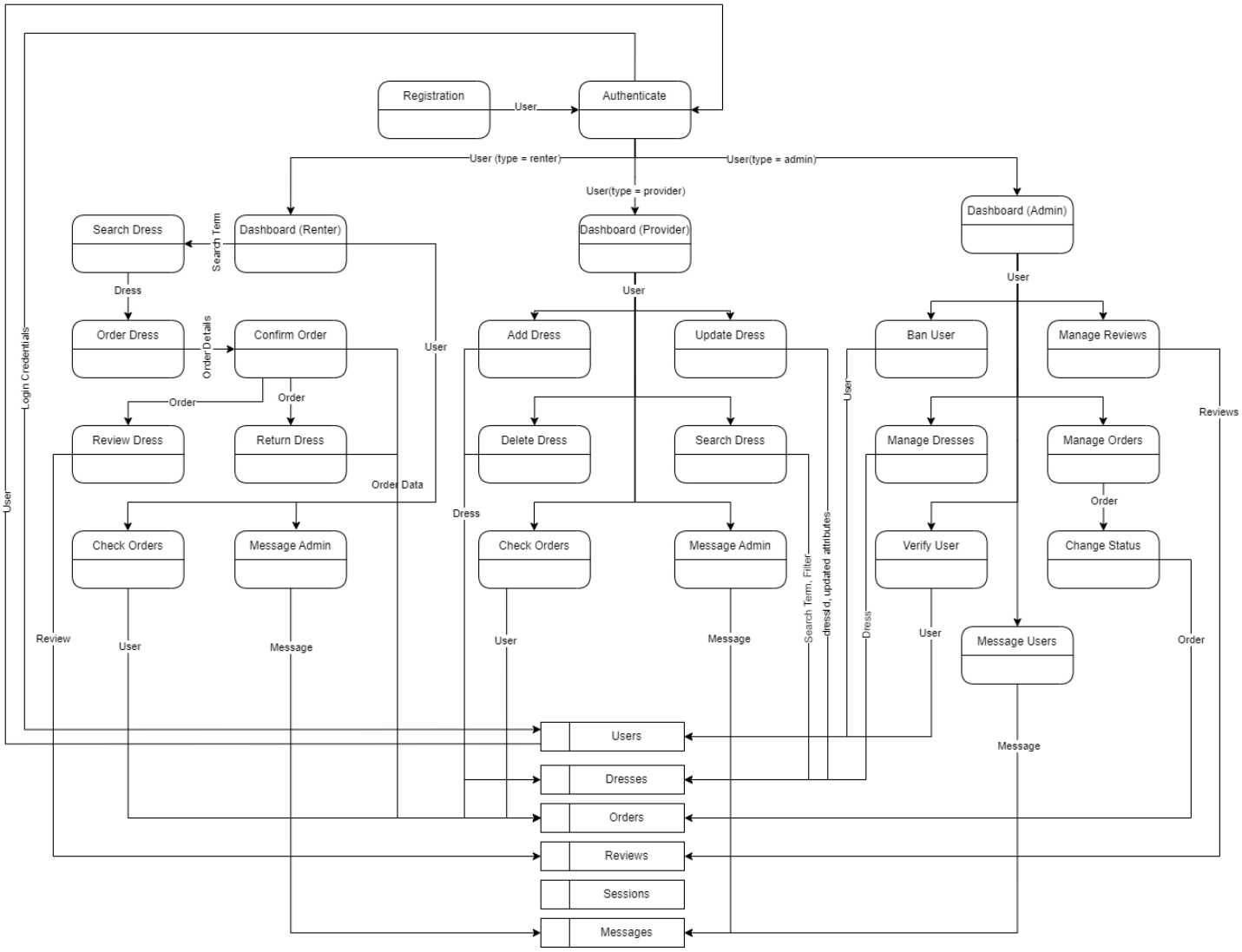


Fig 4.2: Data Flow Diagram

The dataflow diagram of the proposed system is presented in fig 4.2. The data flow diagram showcases the flow of data between several actions and the state of the application. The rounded rectangular shapes define the action/state of the application and the connecting lines

defines the direction of the flow. The text on the connecting lines defines the data which are being transmitted.

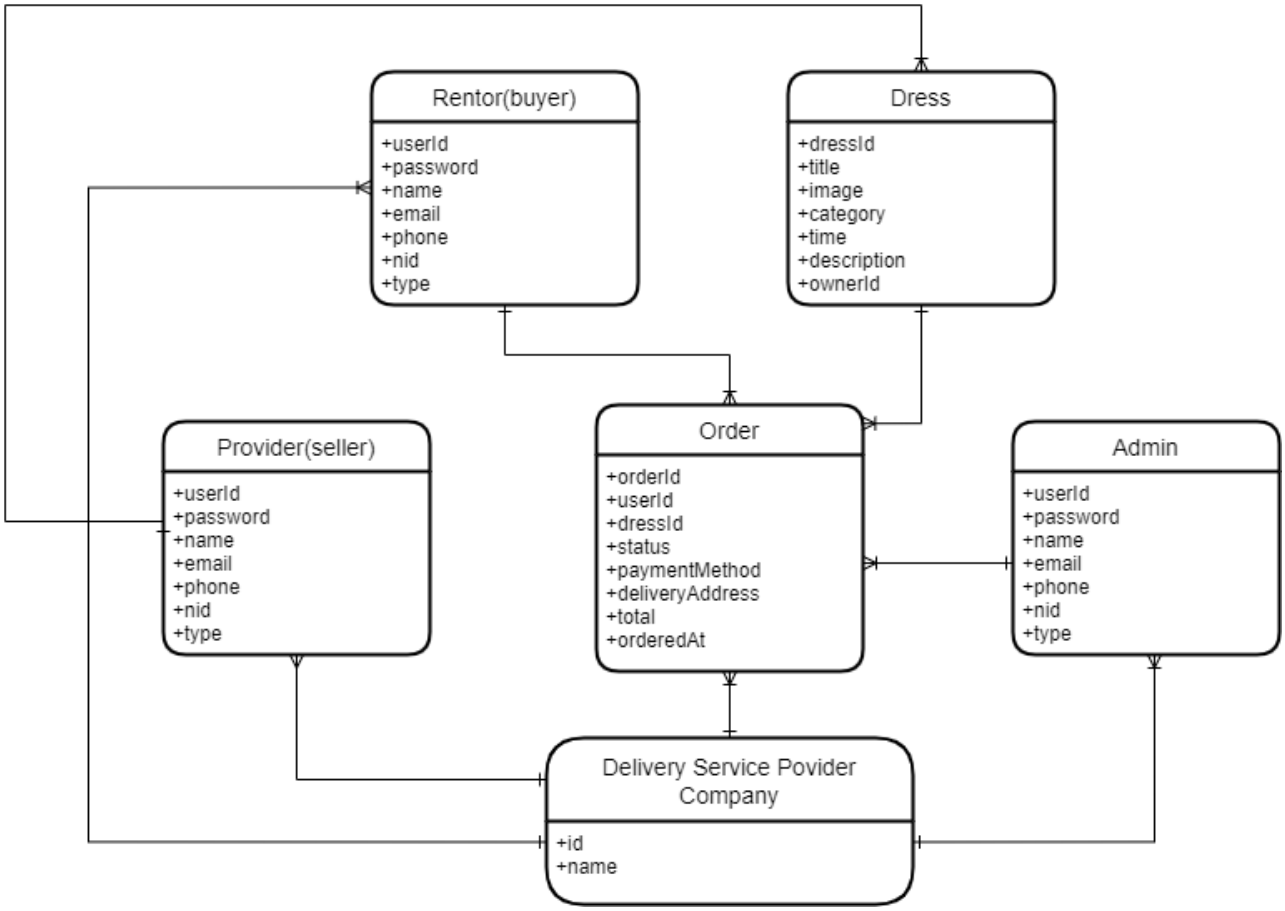


Fig 4.3: Entity Relationship Diagram

The entity relationship diagram of the proposed system has been depicted in fig 4.3. The entity relationship diagram shows all the relations between the entities and events. Here, we have 6 entities as Provider, Renter, Admin, Dress, Order, Delivery Service Provider Company. A provider can create multiple dresses, so there is a one-to-many relationship between Provider – Dress. A renter can do multiple orders, hence –also has a one-to-many relationship between the Renter – Order. Admin and Delivery Service Provider Company is treated as a single company, that why they have a one-to-many relationship with other entities and events.

4.3 API (Backend)

API stands for Application Programming Interface, where any outside service can request for resources, and the API processes it and responds. Our API will be based on node.js runtime, on top of the express.js server – which is a very popular server choice for lightweight API development.

4.3.1 The REST Architecture

REST stands for "Representational State Transfer" – which is an architecture in software development, used to define a uniform interface for all kinds of client sides.

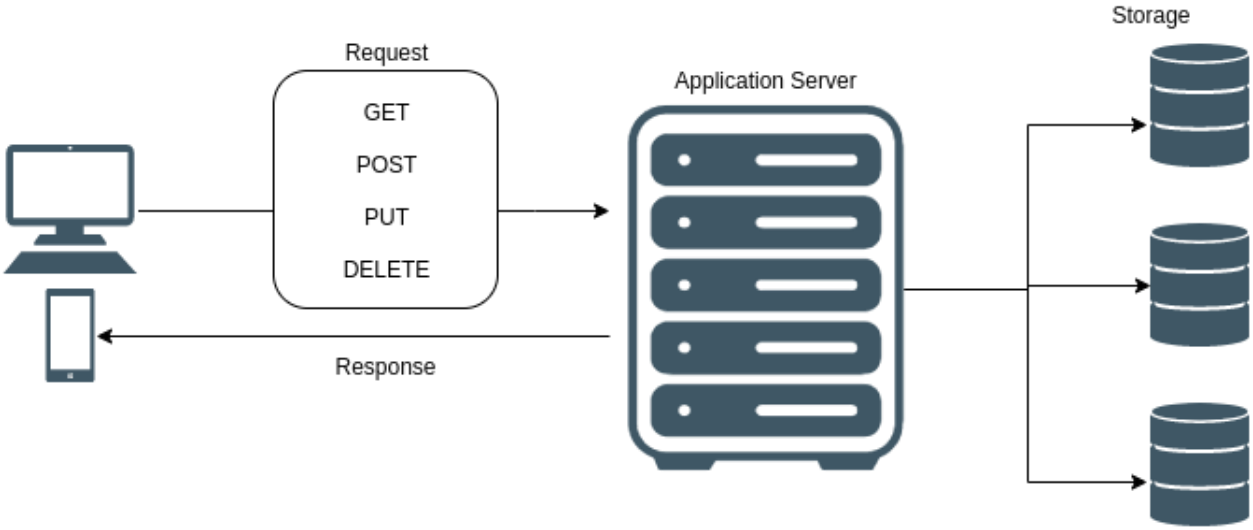


Fig 4.4: REST Architecture

4.3.1.1 HTTP Methods

REST architecture follows its own endpoint architecture and separate HTTP methods. REST support the adoption of GET, POST, PUT, PATCH, and DELETE methods.

GET – retrieve a piece of information from the server (without sending any confidential data to the server)

POST – used to request information from the server with confidential data sent while requesting

PUT – modifying resource where the client sends data that updates the entire resource.

PATCH – modifying resources where the client sends partial data that is to be updated without modifying the entire data.

DELETE – erase any kind of data from the server

4.3.1.2 Endpoint Architecture

REST also follows an architectural paradigm of defining endpoints. Here is an example below:

Purpose	**Suggested Method**	**Suggested Endpoint**	**Request Body**
Get all users data	GET	/users	N/A
Get a specific user data	GET	/users/:userId	N/A
Login with email & Password	POST	/login	{email, password}
Create a user	PUT	/users/	User Object {}
Update a user	PATCH	/users/	User Object {}
Delete a user	DELETE	/users/	{userId}

4.3.2 Authentication System

The authentication system is built upon the token system, widely known as web tokens. As the web data transmission vastly depends of JavaScript Object Notations (JSON) and our service uses JavaScript under the hood, we choose to use JSONWebTokens (JWT) to secure our website and authenticate users.

The authorization works like the following:

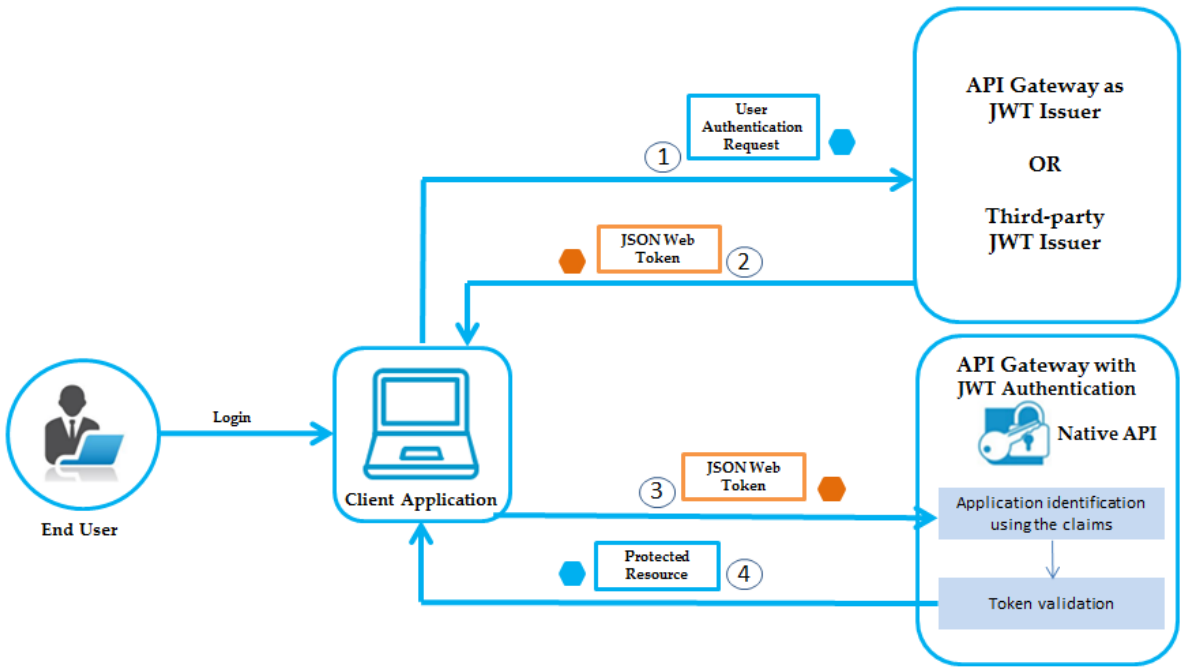


Fig 4.5: Authentication System

4.3.3 Endpoints

API endpoints are the specific URLs where the client-side (such as Mobile App) can make requests and get a response.

Table 4.1: Authorization Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/login	POST	{phone, password}	Not required	{token}	Generates OTP and token on successful login
/verify	GET	otp	Authorization	Session Object	Verify OTP and login successfully – generates a session for the user
/register	PUT	User Object	Not required	User Object	Register a user

Table 4.2 User Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/ban	PATCH	{userId, expiresAt}	Authorization	{User Object, expiresAt}	Ban a user
/user	GET	userId	Authorization	User Object	Get the profile information of a user
/user/verify	PATCH	{userId, status}	Authorization	User Object	Update the verification status of a user

Table 4.3 Dress Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/dress	PUT	Dress Object	Authorization	Dress Object	Create a dress
/dress	PATCH	Dress Object	Authorization	Dress Object	Update a dress
/dress	GET	dressId	Authorization	Dress Object	Get a dress information
/dress	DELETE	dressId	Authorization	Dress Object	Delete a dress
/dresses	GET	userId (optional) / category (optional)	Authorization	[Obj, Obj, Obj, Obj, …]	Get a list of dresses with filtering

Table 4.4 Message Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/message	PUT	Message Object	Authorization	Message Object	Create a message
/message	PATCH	Message Object	Authorization	Message Object	Update a message
/message	GET	messageId	Authorization	Message Object	Get a message
/message	DELETE	messageId	Authorization	Message Object	Delete a message
/messages	GET	userId (optional) / groupId (optional)	Authorization	[Obj, Obj, Obj, Obj, …]	Get a list of messages

Table 4.5: Order Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/order	PUT	Order Object	Authorization	Order Object	Create an order
/order	PATCH	Order Object	Authorization	Order Object	Update an order
/order	GET	orderId	Authorization	Order Object	Get an order
/orders	GET	userId / dressId	Authorization	[Obj, Obj, Obj, Obj, …]	Get a list of orders

Table 4.6: Review Endpoints

Endpoint	Method	Request Body / Param	Request Header	Response	Description
/review	PUT	Review Object	Authorization	Review Object	Create a review
/review	PATCH	Review Object	Authorization	Review Object	Update a review
/review	GET	reviewId	Authorization	Review Object	Get a review
/review	DELETE	reviewId	Authorization	Review Object	Delete a review
/reviews	GET	userId (optional) / dressId (optional)	Authorization	[Obj, Obj, Obj, Obj, …]	Get a list of reviews

4.4 Database Schema Diagrams

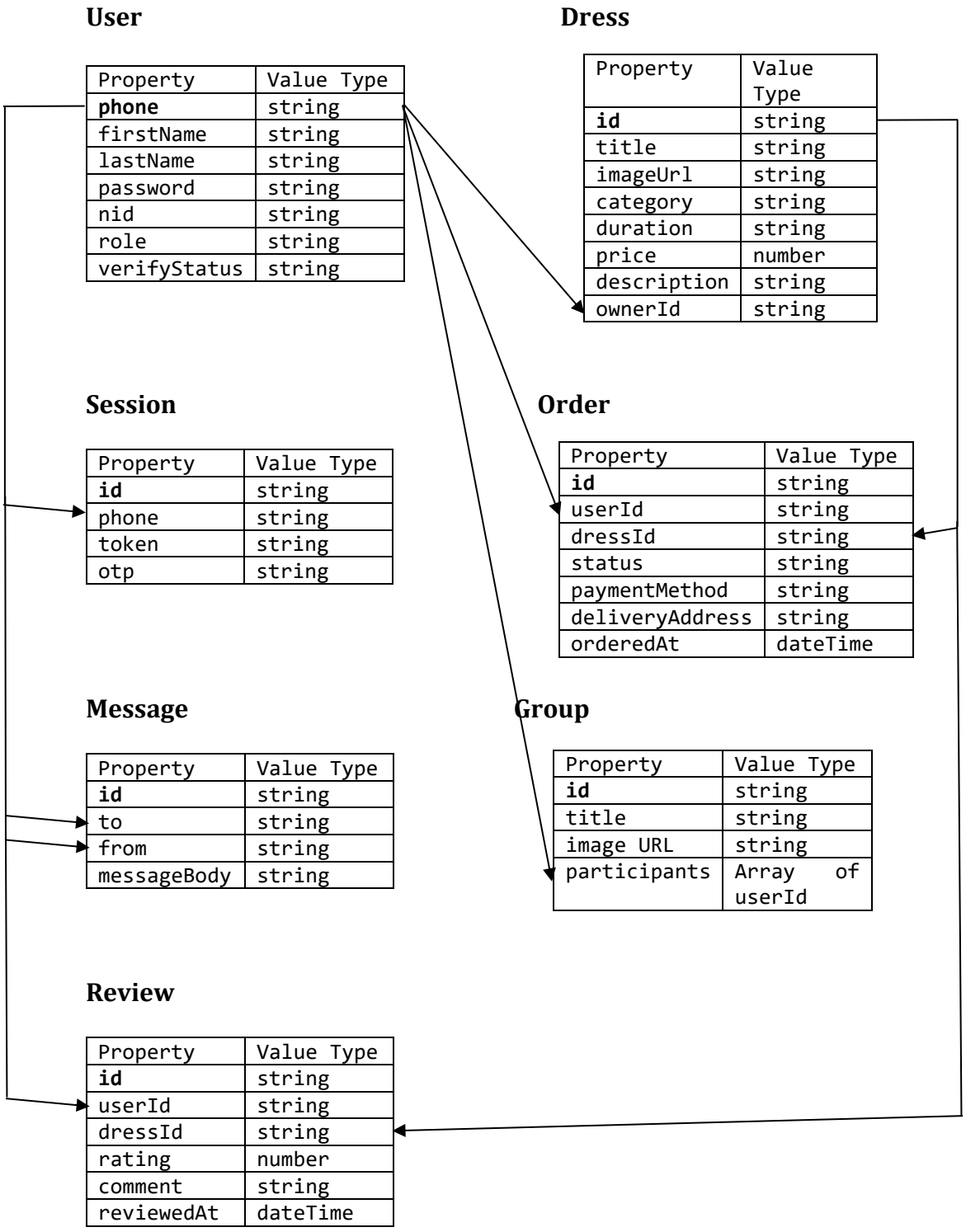
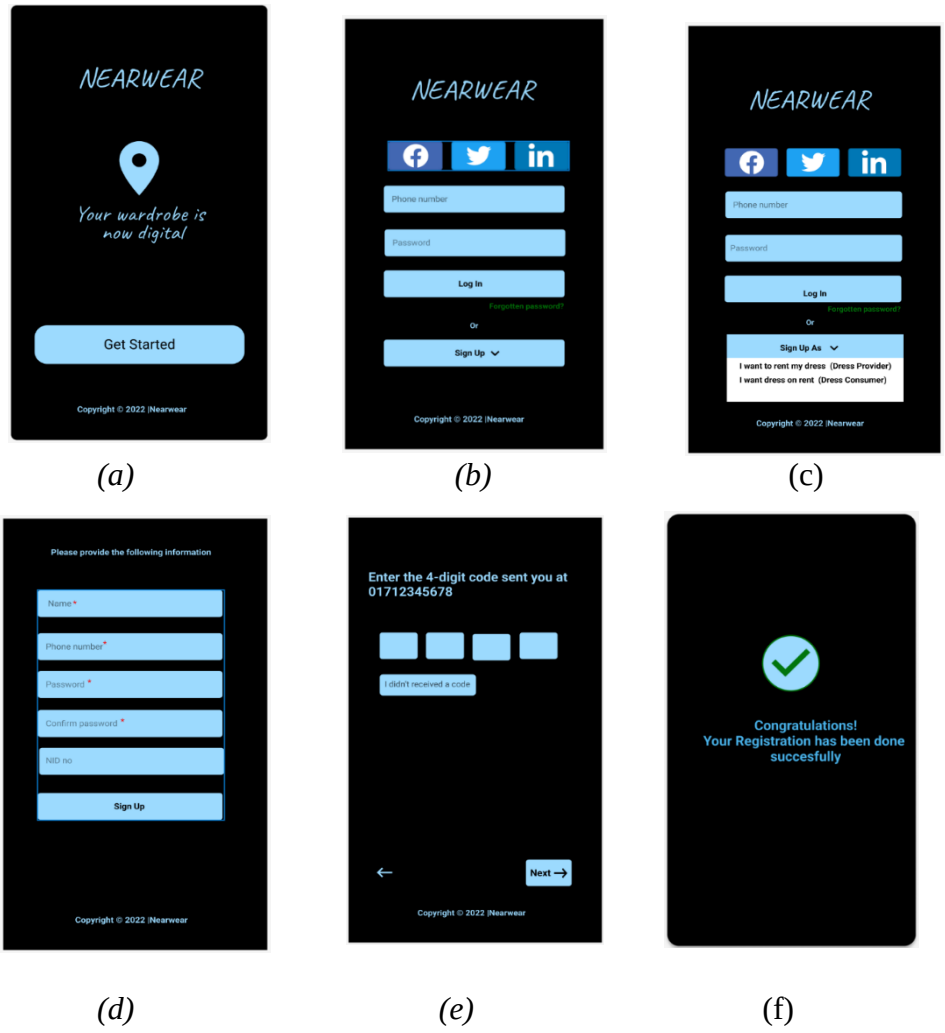


Fig 4.6: Schema Diagram of the proposed System

4.5 UI/UX Design

For UI/UX design and prototyping, Figma was used. Figma is a UI/UX tool, that exists for creating stunning UI with ease. Our application will include a good number of screens, and all the screens were designed and prototyped inside Figma.

4.5.1 Auth UI



(a) *(b)* (c)

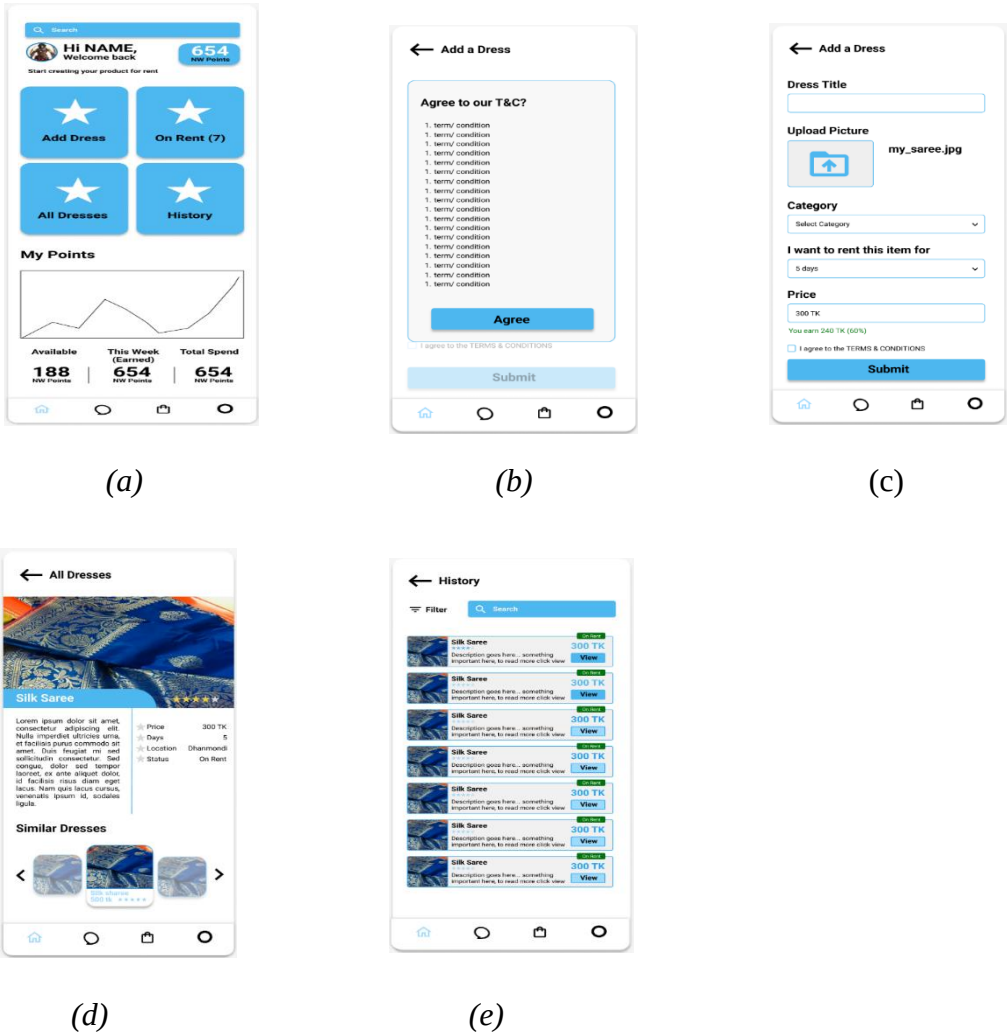
(d) *(e)* (f)

Fig 4.7: (a) Landing Screen, (b) Login/SignUp Screen, (c) SignUp Screen, (d) SignUp Form, (e) OTP Screen, (f) Registration Success Screen

In fig 4.7 (a) the landing screen of the app is present, which will be visible to the user once he/she opens the app. In fig 4.7 (b) the login/Signup screen is present. It will be the second screen, where the user can log in with a phone number and password. In fig 4.7 (c) is the

SignUp Screen for user registration and there are two types of users, a dress provider who will provide dresses for rent, and a dress renter who will rent dresses for events. Fig 4.7 (d) is present for the SignUp form Screen, new users can register through SignUp Form. Fig 4.7 (e) is the OTP screen for user verification. Verification will be done by sending the OTP code to the phone. And registration will be confirmed through Registration Success Screen which is in fig 4.7 (f).

4.5.2 Provider UI



(a) *(b)* (c)

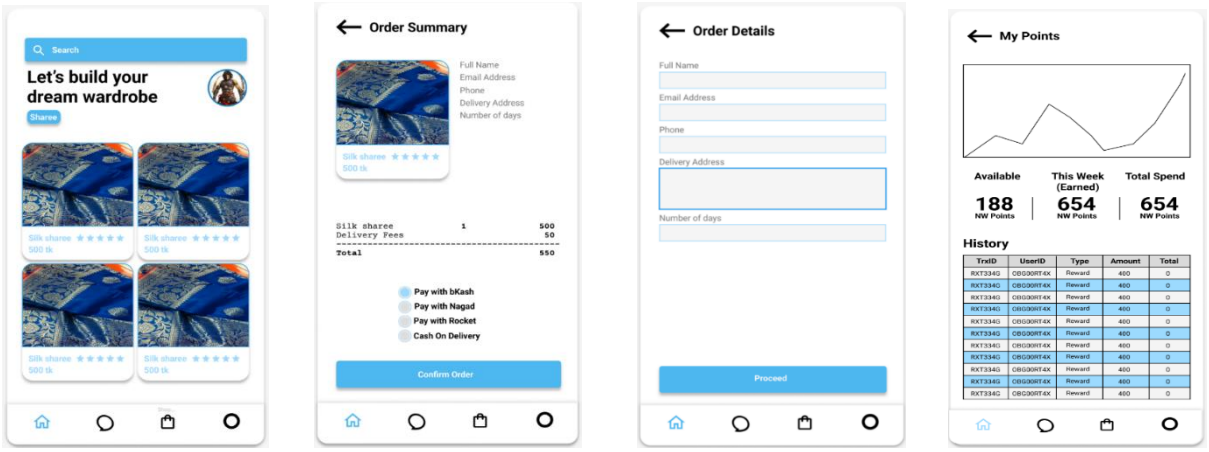
(d) *(e)*

Fig 4.8: (a) Provider Dashboard Screen, (b) Terms & Condition Popup Screen, (c) Dress Add Screen, (d) Single Dress Screen, (e) History Screen

In fig 4.8 (a), through the provider dashboard screen, the user can post dresses, see what dresses are on rent, and the user can see the list of all the dresses he/she has posted. Terms &

Conditions Popup Screen in fig 4.8 (b) is to help users to know the details about our app's rules and in fig 4.8 (c) through the dress add screen, the dress provider can post his/her dress picture and details about the dress for rent. The single dress Screen will show the details about the dress which is fig 4.8 (d) and in fig 4.8 (e) history screen will show the all records about a user's dresses.

4.5.3 Renter UI



(a) *(b)* (c) *(d)*

Fig 4.9: (a) Renter Dashboard Screen, (b) Order Summary Screen, (c) Order Details Screen, (d) Points Screen

Fig 4.9 (a) presents the renter dashboard screen, it will show the posts of dresses that are on rent. After ordering the clothes, the user will see the summary of that order in the Order Summary Screen which is fig 4.9 (b). Fig 4.9 (c) is the order details screen, which is to order with name, address, phone number, and for how many days users want to rent that dress. In fig 4.9 (d), the points screen shows the earning points of a user, which they can earn by the frequency of ordering the dress.

4.5.4 Admin UI

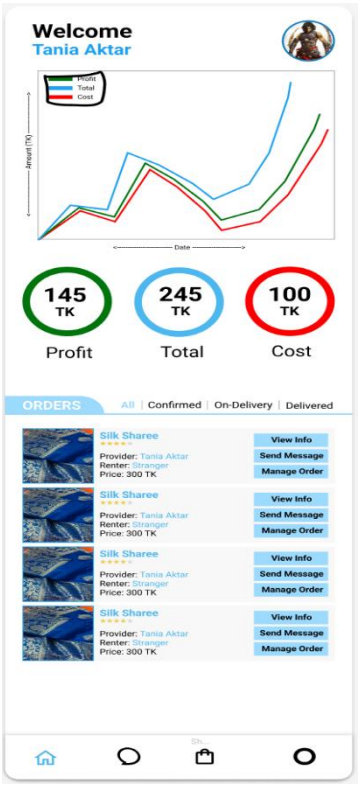


Fig 4.10: Admin Dashboard Screen

In fig 4.10 the Admin Dashboard Screen is present to manage, delete orders, and message users.

CHAPTER 5

RESULT AND DISCUSSION

5.1 Introduction

The result refers to the “MVP” – which includes the base features of the application that implements the main idea throughout the development.
The product/service included every vital / required feature and worked as expected. There were multiple tests and assessments done after completing the initial development phase. Features / Specifications that were included in the initial development phase include:

3 types of users,

1. Provider
2. Renter
3. Admin

The services included are:

1. Login / SignUp (Authentication)
2. Rent System
3. Review / Rating System
4. Messaging System
5. User Management

5.2 Tests & Assessments

App Testing – The application testing was carried out to ensure that the app runs smoothly without any shuttering or runtime errors. Several bugs/irregularities were fixed during the testing and fixing period – which made the app more stable than ever. We wish to continue the test and fix cycle to continue the development by fine-tuning the UI/UX.
The tests were done considering the –

1. Performance
2. Stability
3. User Experience

API Testing – API testing refers to the testing of the available endpoints. The following things were considered during the API testing –

1. Request & Response
2. Performance
3. Authorization

API testing was done using the "Postman" software – a tool that enables API testing with easy-to-use UI and customizations.

5.3 Discussion

The developed system has the full functionality to post a dress for the provider, rent a dress for the renter and manage orders for the admin. The system includes frontend, backend API, and database. In our tests and assessments, we carried out user testing on the UI, where a user tested the performance and reliability of the developed product. The results were good enough to deploy the service into a production environment. No major/minor flaw in the system was found during the user testing of the UI. The API testing phase had several mock tests with dummy data to assess the performance and success rate of the API. The API endpoint responds within seconds and the success rate is 100% (provided valid data).

End-to-end testing was done to conclude the assessment, which includes all the elements of the system to be tested together. This part is crucial for the product/service as it actually mimics the whole system orchestrations. The system passes every end-to-end test and is stable enough to be presented and deployed to a user base.

The result indicates that the frontend (UI) has a clean and performant touch into it. The API performance is exceptional as it uses node.js and express.js environment, which is lightweight and super-fast, especially widely used for API development. The fast response from the API also needs one mention, the efficient database schema design. The end result was up to the mark for a combination of best practices and evaluations.

CHAPTER 6

CONCLUSION & FUTURE PLANS

6.1 Endwords

"NearWear" – will serve as a primary platform for dress sharing where the provider can list dresses for rent and the renter can rent the dresses in a very cost-effective way. The service includes all the required features to be able to enjoy the dress-sharing service worldwide. As the service highly maintains the technical and non-technical requirements, it will be able to serve a huge number of users simultaneously.

The application was built on React Native, one of the most popular platforms for mobile app development. It also includes extended support when wrapped with Expo, a wrapper of React Native. Expo includes a lot more components and functionality built-in, ready to use for the developer. The UI was designed intelligently to have the best user experience while keeping the interface clean and gorgeous. The application communicates with the backend/database via the API. The API includes CRUD operation endpoints to have the full flexibility of data manipulation. The API follows the REST architecture – the most popular API architecture today. Multiple tests and assessments were done on the API endpoints to ensure that the request and response cycle behaves as expected. The database schema was designed in a way where the performance and storage come on balance. Database schema plays a vital role in the speed of the application, as the query time on the database really impacts the responsiveness and performance of the application.

Considering the discussion above, we expect to serve millions of users as platforms of their choice for dress sharing. "NearWear" will continue its development to add-up the future improvements and fix existing issues that may arise later.

6.2 Future Plans

The future improvements should include some of the features that were skipped in the initial development such as:

1. Gamification (Point Award System)

2. Promotion System
3. Active Status Tracking
4. Geographical Search
5. Order Live Tracking

Also, as the initial development can never be stable without proper evaluation and bug fixing – the future plan includes bug fixing and improving the performance through better implementation or design patterns.

REFERENCES

1. Report, T. (2020, February 18). *With 7.5m rides a month, ridesharing services take over Bangladesh*. The Business Standard. Retrieved October 10, 2022, from https://www.tbsnews.net/economy/75m-rides-month-ridesharing-services-take-over-bangladesh-45453
2. Elven, M. (2021, March 9). *People do not wear at least 50 percent of their wardrobes, says study*. FashionUnited. https://fashionunited.uk/news/fashion/people-do-not-wear-at-least-50-percent-of-their-wardrobes-according-to-study.
3. Krishna, T. (2022, January 18). *Clothes to get 15–20% costlier in 2022, probable GST rate hike may further add to the prices*. Financial Express.
4. Daily, S. C. (2019, June 11). *Why Sharing and Hiring Clothing Is a Great Idea In Today's Society*. Space Coast Daily. https://spacecoastdaily.com/2019/06/why-sharing-and-hiring-clothing-is-a-great-idea-in-todays-society
5. Lo, V. (2021, December 15). *A Deep Look into RESTful APIs - Analytics Vidhya*. Medium. Retrieved October 10, 2022, from https://medium.com/analytics-vidhya/a-deep-look-into-restful-apis-77b0af1dcdcf
6. Sohan, S. M., Maurer, F., Anslow, C., & Robillard, M. P. (2017, October). A study of the effectiveness of usage examples in REST API documentation. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 53-61). IEEE.
7. Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). *HTTP: the definitive guide*. " O'Reilly Media, Inc.".
8. *Designer Dress Hire Australia | Rent Dresses & Gowns*. (n.d.). Retrieved October 10, 2022, from https://yourcloset.com.au/?fbclid=IwAR29JbMVKYyhuQemxegzNRKzDSqMctbtEggwTJn_e9chD_rVwBgDuodlFpI
9. *Rent, Resell designer fashion dresses bags | By Rotation*. (n.d.). Retrieved October 10, 2022, from https://byrotation.com/

10. *Put Your Closet in the Cloud*. (n.d.). Rent the Runway. Retrieved October 10, 2022, from https://www.renttherunway.com/