Daily service delivery management

Submitted To,

Faculty of Technology,
Department of Computer Engineering,
Dharmsinh Desai University.

DEVELOPED BY: Dhruv Bhimani, Keyur jivani

Name: *Dhruv Bhimani [CE-021] [20CEUOS023]*

Keyur Jivani [CE-049] [20CEUOG027]

Semester: *B.Tech 6th*

Subject: System Design practice

Project Title: Daily service delivery management

Guided By: *Prof. Brijesh S. Bhatt*

Faculty of Technology,

Department of Computer Engineering,

Dharmsinh Desai University.

Dharmsinh Desai University, Nadiad
Department of Computer Engineering
Faculty of Technology

CERTIFICATE

This is to certify that the project work carried out
in the subject of
System Design practice
is the bonafide work of

Dhruv Bhimani (CE021) [20CEUOS023]
Keyur Jivani (CE049) [20CEUOG027]

of B. Tech. Semester-VI Computer Engineering during the academic year **2022-2023**.

Prof. Brijesh S. Bhatt

Computer Engg. Department
Faculty of Technology
D.D.U., Nadiad

Dr. C. K. Bhensdadia
Professor and HoD,
Computer Engg. Department
Faculty of Technology
D.D.U., Nadiad

INDEX

Abstract:

In today's hectic world of work-loaded life, there is no time for an individual to give a dedicated schedule for small things like check milk man's bill paper boy's order etc. You just want to relax after returning from work , and don't want to spend time on such small things. So , we build a website which can accurately do all these things.

Introduction:

Our website will provide you facility of selecting milkman and paperboy. Also we can book appointment to take service from launderer , housekeeper , plumber , electrician , R.O. servitor .

This website also provide some more features like generating final bill of the month , choosing date when we don't want particular service etc.

Users of the System:-

- 7 Different providers
    - Milkman
    - Paperboy
    - Launderer
    - Housekeeper
    - Plumber
    - Electrician
    - R.O. Servitor
- Customers of Providers
- Admin

❖ Technology and Tools Used:

✦ Technology:

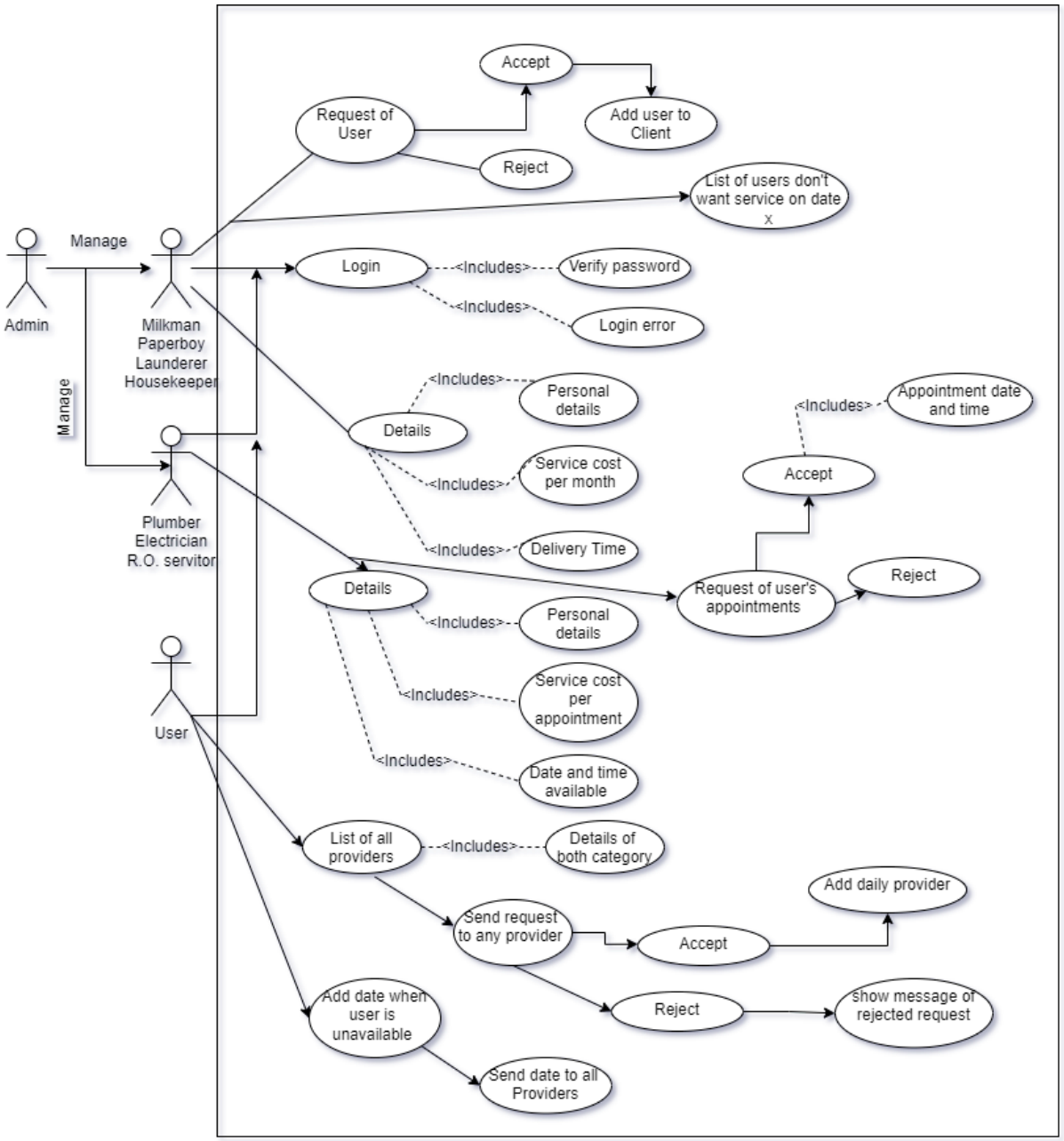
- React Js
- Node Js
- Express Js
- JavaScript
- HTML
- CSS

✦ Tools:

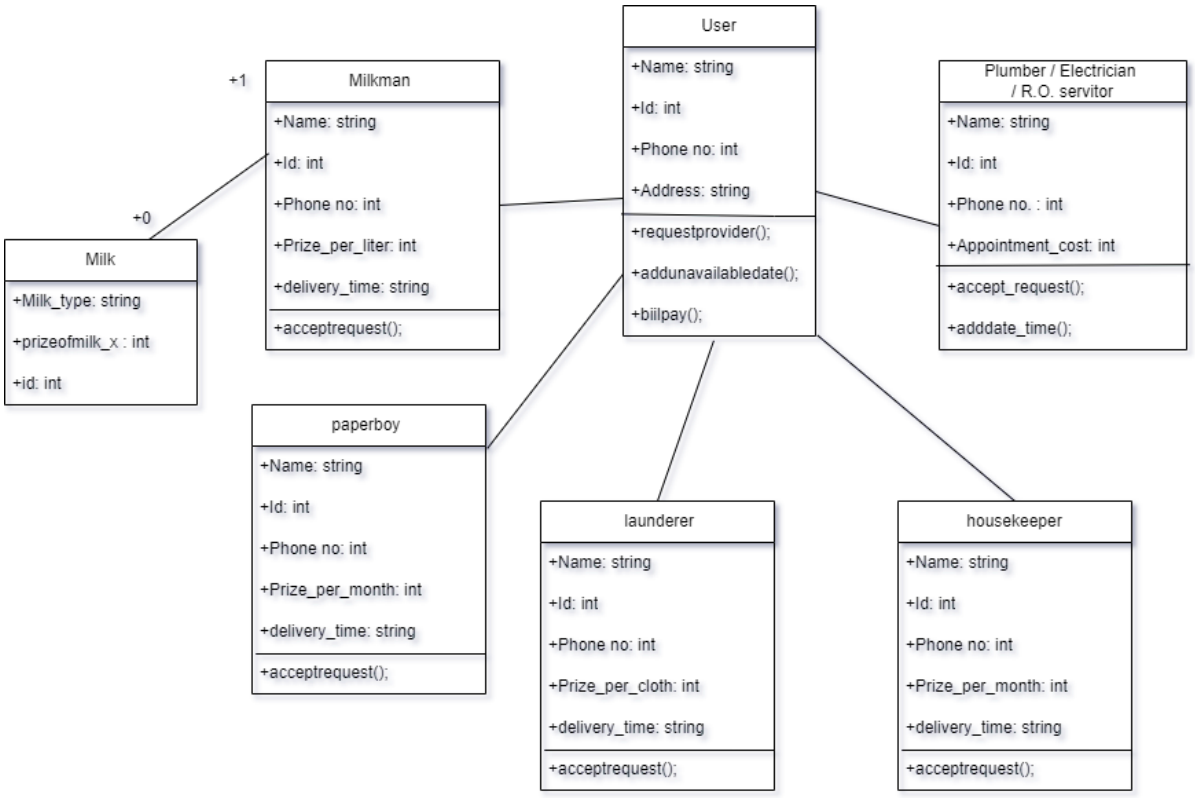
- Git
- Visual Studio
- Mongo DB

Software Requirement Specifications:

- **Use case diagram**



- **Class diagram**



Functional Requirements:-

1. Manage users:

R.1.1 Sign Up:-

Description: new user can sign up in website by giving basic information like email id, name and passwords;

Input: - Basic details, ID and password

Output: - Confirmation of sign up

R.1.2.1 Sign in:

Description: Once user sign up by email id then for next time user can sign in by using ID and password.

Input: ID, Password.

Output: User redirected to relative home page.

R.1.2.2 Forgot Password:

Description: If user forgot password then he will give mail id and system will send link via mail for reset password.

Input: - username

Output: - set password functionally

R.1.3.1 **Update user profile:**

Description: Whenever registered user wants to change details then they can change it and it will be updated

in database.

Input: New details.

Output: Confirmation of updating.

R.1.3.2 **View user profile:-**

Description: - user can see how his/her profile looks like.

Input: - user select view profile

Output: - User Profile overview

2. Manage Providers

NOTE: - All providers can be added by admin only. Admin will add them after validating all the details.

R.2.1 **Milk provider:**

Description: When user wants to add milk provider he/she can find details about milk provider's prize per liter and arrival timing then accordingly they can send request to particular milkman.

Input: Send request to milkman

Output: request sent.

R.2.2 Newspaper provider:

Description: When user wants to add newspaper provider he/she can find details about new provider's prize per month and arrival timing then accordingly they can send request to particular paperboy.

Input: Send request to paperboy.

Output: request sent.

R.2.3 laundryman:

Description: When user wants to add launderer he/she can find details about launderer's prize per cloths and arrival timing and time of delivering cloths then accordingly they can send request to particular laundryman.

Input: Send request to laundryman

Output: request sent.

R.2.4 Housekeepers:

Description: When user wants to add housekeepers he/she can find details about launderer's prize per month and arrival timing then accordingly they can send request to particular housekeepers.

Input: Send request to laundryman

Output: request sent.

R.2.5 plumber / electrician / R.O repairer:

Description: when user has any problem when they need plumber / electrician / r.o repairer then they can see registered providers and request for appointment after adding appointment date and time.

Input: Send request to plumber / electrician / r.o repairer

Output: request sent successfully.

Common functionality for all providers

Description: when user is not available for some day they can specify that date and select unavailable on particular day so when provider log in he/she will be able to see that their which customers aren't available on particular days.

Input: Date when user don't want service

Output: Add date to providers dashboard

Non-Functional Requirements:

1: Database:

A database management system that is available free of cost in the public domain should be used.

2: Platform:

Both Windows and UNIX versions of the software need to be developed.

3: Web-support:

It should be possible to access website from any place by using a web browser.

4: Secure access to confidential information (user's details).

5: 24*7 Availability.

6: Better component design to get better performance.

7: It should be platform independent. So we can run it in windows, mac and Linux

8: Security requirements: User credentials should be encrypted so as to ensure confidentiality, integrity and availability and the project ideas should be protected so as to avoid being stolen by other parties

9: Usability requirements- This system should be user-friendly

and easy to use so that users can perform their tasks nicely.

10: Performance should be fast and error free.

11: User friendly: System should be easily used by

the customer.

12: Flexible service based on system architecture so it can be useful in future extensions.

Database Design

We use mongo db as database in this database we have one collection named as user this collection contains data of all user in document format. Each document have some specific fields like

- _id
- Firstname
- Lastname
- Emailed
- Password
- Money
- boughtStock
- soldStock.

Apart from this we also have some derived property like invested money and current money.

Customer Collection:-

```
const userschema = new mongoose.Schema({
    user_id: {
        type: String,
        required: true,
        unique: true
    },
    username: {
        type: String,
        required: true,
        unique: true
    },
    fname: {
        type: String,
        required: true,
    },
    lname: {
        type: String,
        required: true,
    },
```

```
    email: {
        type: String,
        required: true,
        unique: true
    },
    PhoneNumber: {
        type: Number,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    address: {
        type: String,
        required: true
    },
    milkprovider_id: {
        type: String,
    },
    newsprovider_id: {
        type: String,
    },
})
```

Milkman Collection:-

```
const milkprovider = new mongoose.Schema({
    milk_provider_id: {
        type: String,
        required: true,
        unique: true
    },
    username: {
        type: String,
        required: true,
        unique: true
    },
    fname: {
        type: String,
        required: true,
    },
    lname: {
        type: String,
        required: true,
```

```
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    PhoneNumber: {
        type: Number,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    address: {
        type: String,
        required: true
    },
    prize: {
        type: Number,
        required: true
    },
    morning: {
        type: Boolean,
        required: true,
    },
    morning_start: {
        type: String,
    },
    morning_end: {
        type: String,
    },
    evening: {
        type: Boolean,
        required: true,
    },
    evening_start: {
        type: String,
    },
    evening_end: {
        type: String,
    }
})
```

Newsboy Collection :-

```
const newspaper = new mongoose.Schema({
    news_provider_id: {
        type: String,
        required: true,
        unique: true
    },
    username: {
        type: String,
        required: true,
        unique: true
    },
    fname: {
        type: String,
        required: true,
    },
    lname: {
        type: String,
        required: true,
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    PhoneNumber: {
        type: Number,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    address: {
        type: String,
        required: true
    },
    prize: {
        type: Number,
        required: true
    }
})
```

Customer and milkprovider Connection Collection :-

```
const customerprovider = new mongoose.Schema({
    provider_id: {
        type: String,
        required: true,
    },
    customer_id: {
        type: String,
        required: true,
    },
    milkperday: {
        type: Number,
    },
    request: {
        type: String,
        default: false,
    },
    dates: {
        type: Array,
    },
    month: {
        type: Number,
    },
    year: {
        type: Number,
    }
})
```

Customer and Newsprovider Connection Collection :-

```
const newsprovider = new mongoose.Schema({
    provider_id: {
        type: String,
        required: true,
    },
    customer_id: {
        type: String,
        required: true,
    },
    request: {
        type: String,
        default: false,
```

```
    },
    dates: {
        type: Array,
    },
    month: {
        type: Number,
    },
    year: {
        type: Number,
    }
})
```

Testing:

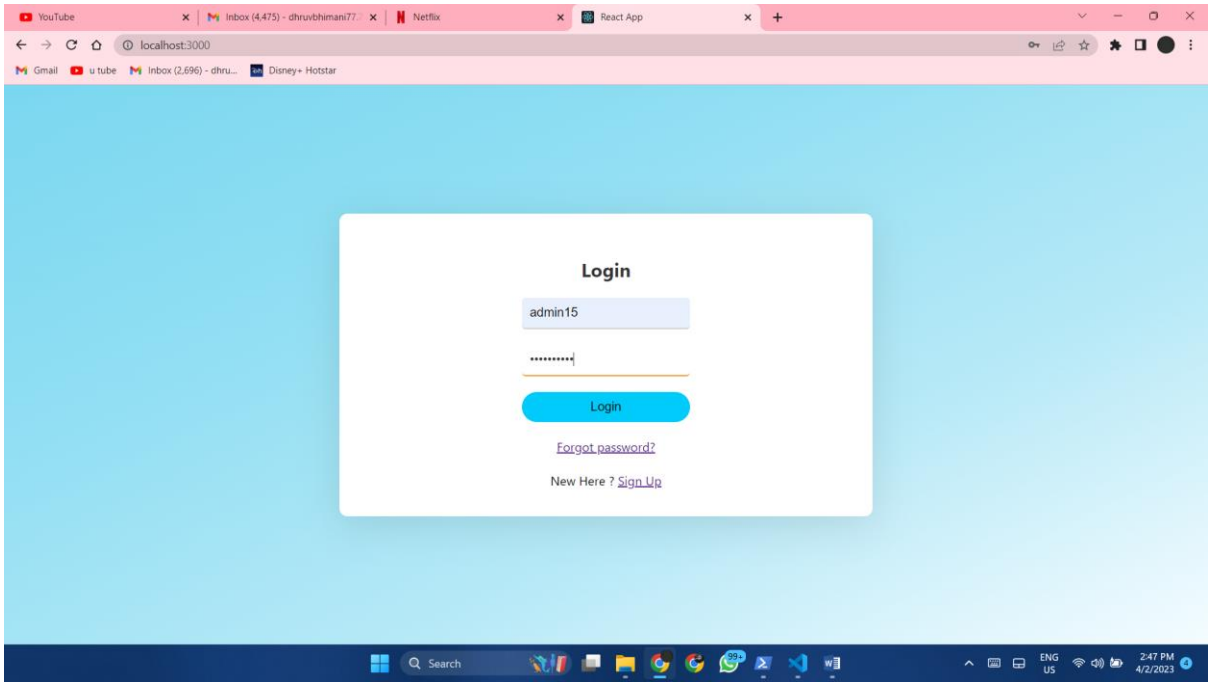
> Manual testing was performed in order to find and fix the bugs in development process.

Sr.No	Test Scenario	Expected Results	Actual Results	Status
1.	Sign up	If details added correctly	User is registered	Success
2.	Sign up	If details are not added correctly	User is getting errors	Success
3.	View details of providers	Should display data of all providers like price	Displayed data of providers with detail	Success
4.	Request provider	If accepted user will assign to particular provider	provider is added to profile of user	Success
5.	Update delivery dates	Delivery date can be updated to deliver or don't deliver	Assigned provider can see updated dates	Success
6.	Send request to launderer ,	If accepted then show	User can see provider has	Success

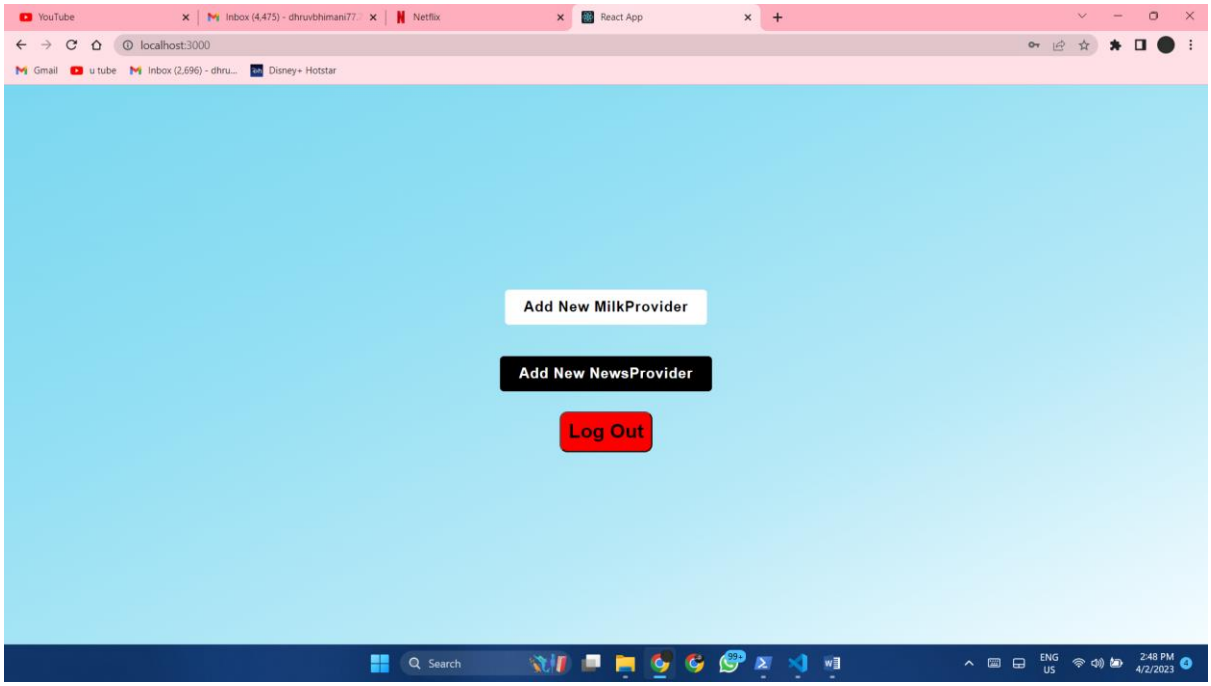
	housekeeper , electrician etc. for visit	date and time of accepted day	accepted the request	

Screen-shots of the System:

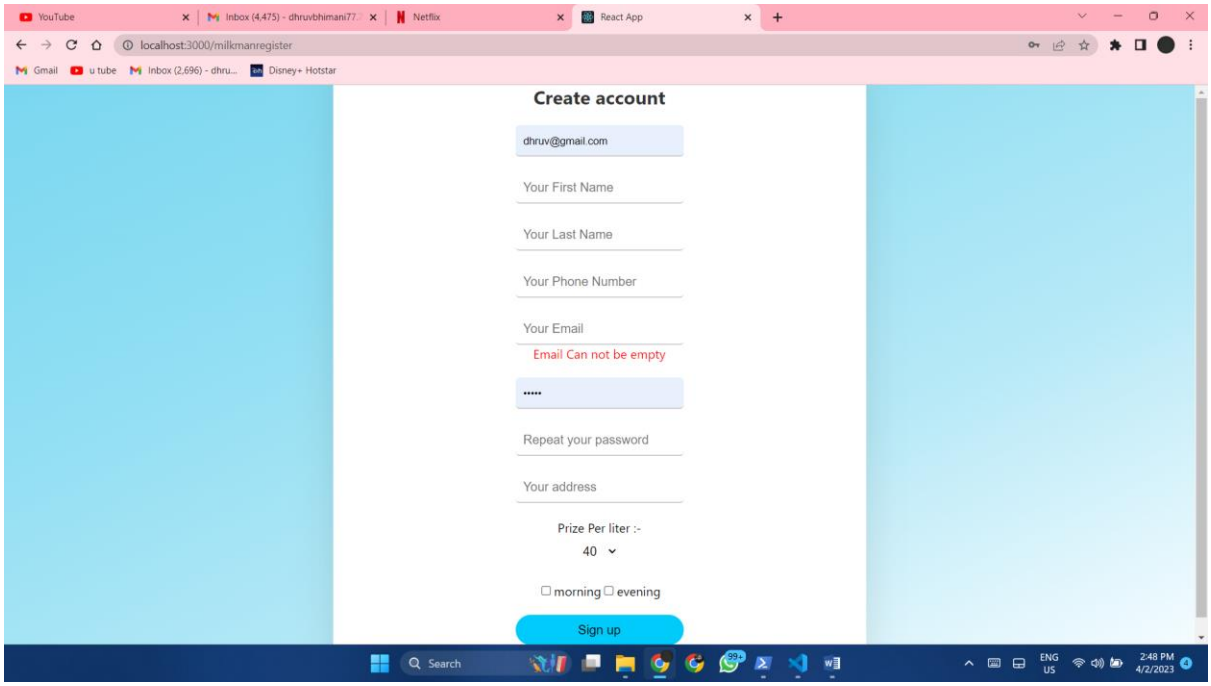
1. Login page:



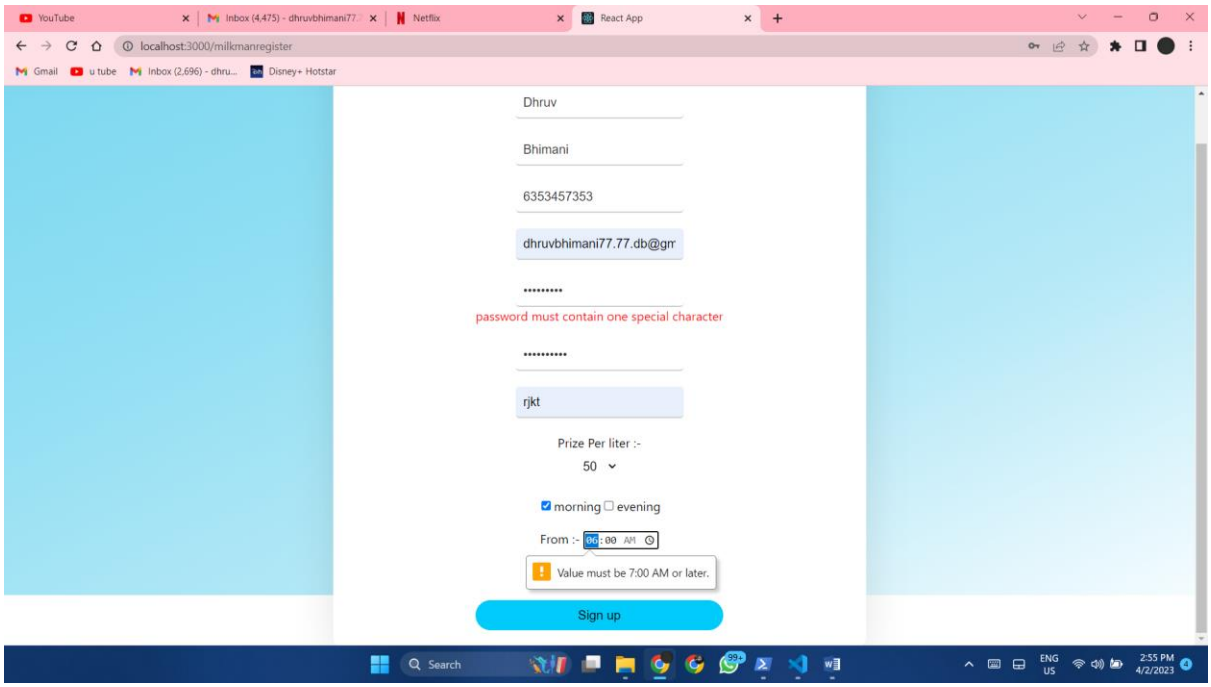
2. Admin dashboard:



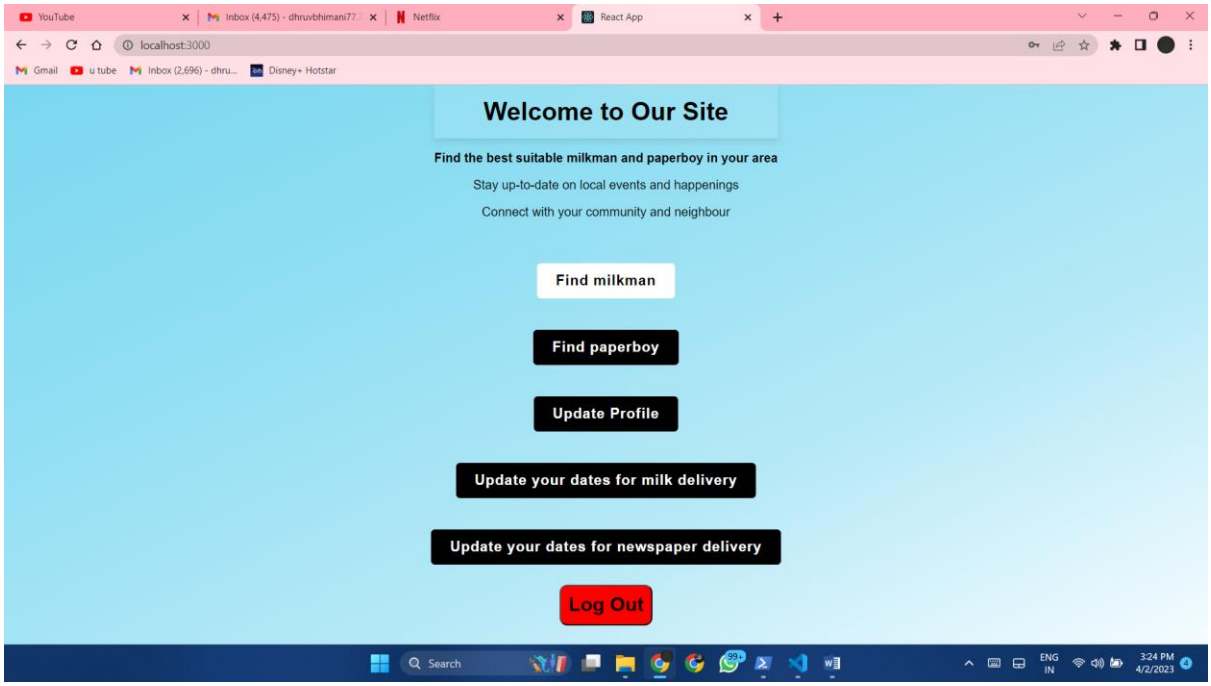
3. Add providers (create account for provider)



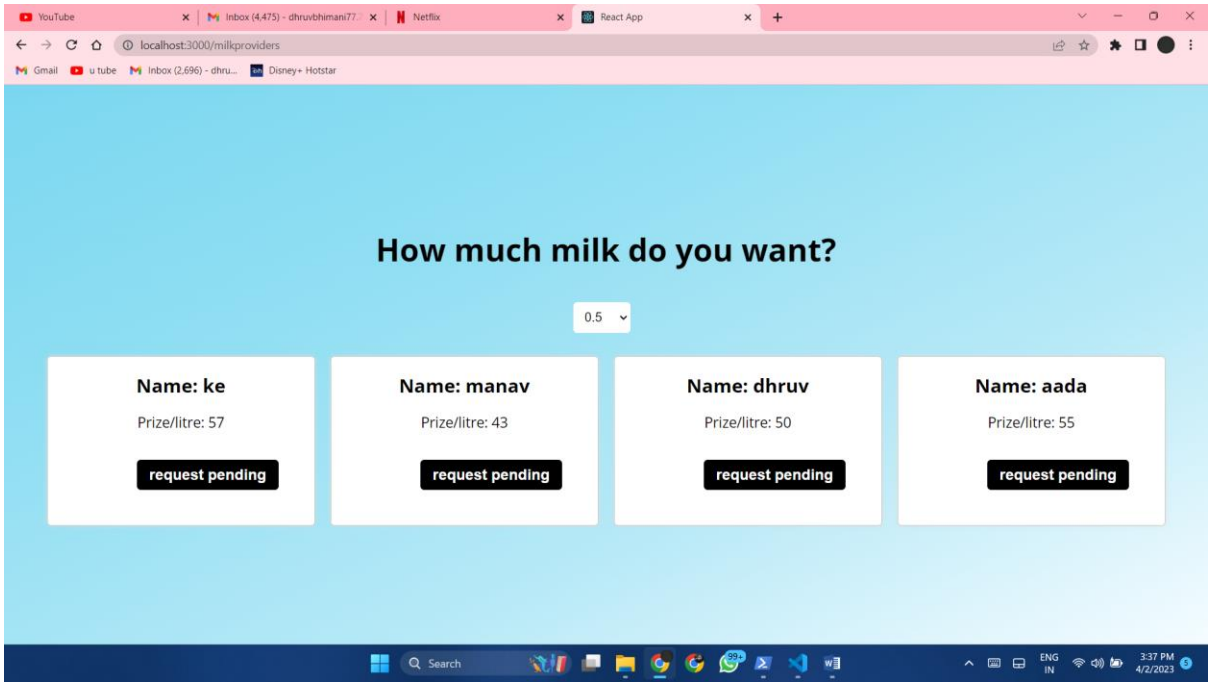
3.1 Some validations



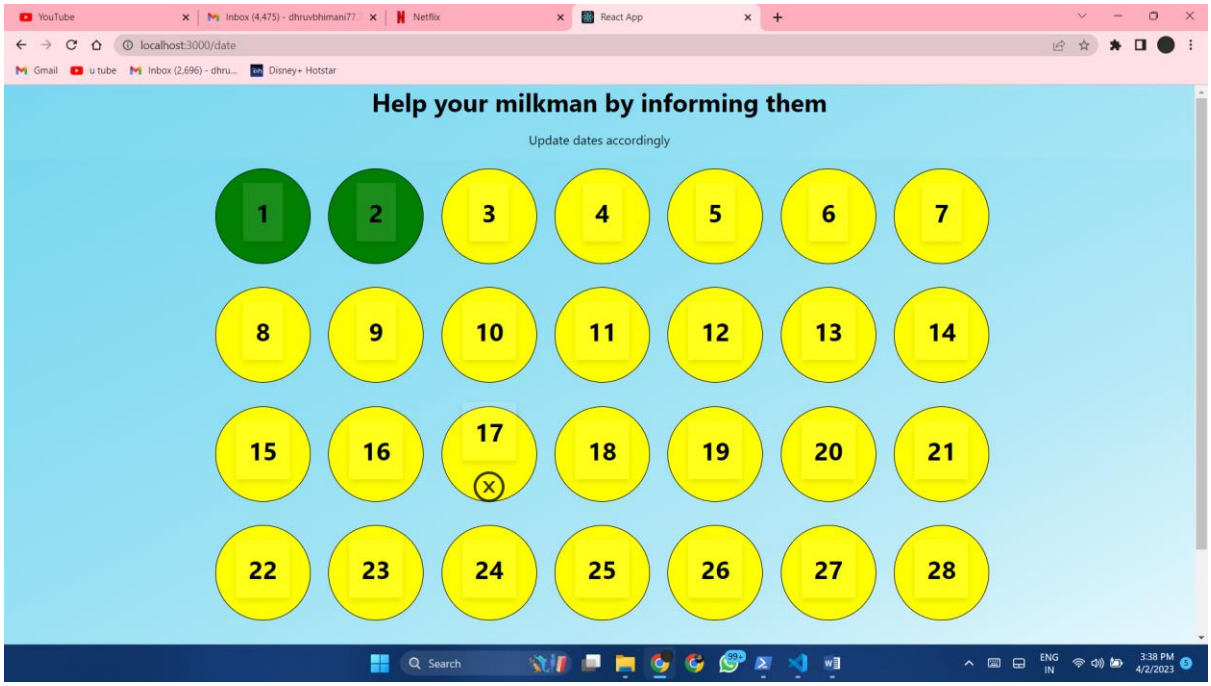
4. User Dashboard



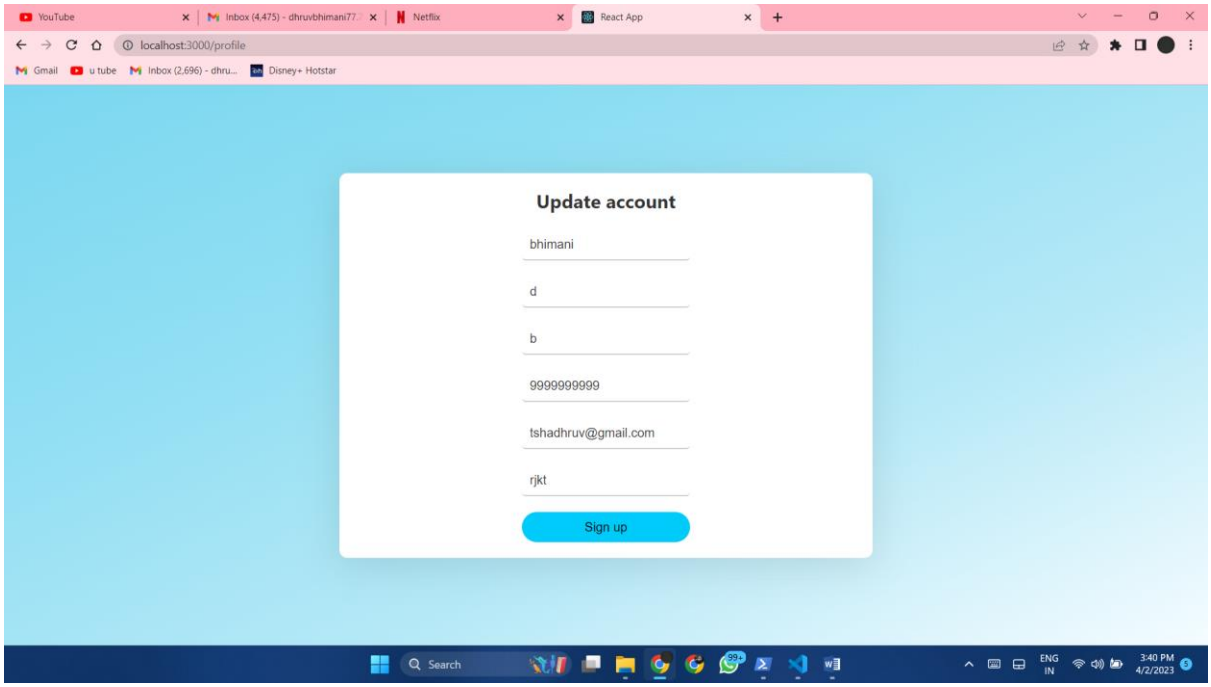
5. Send request to milkman or paperboy



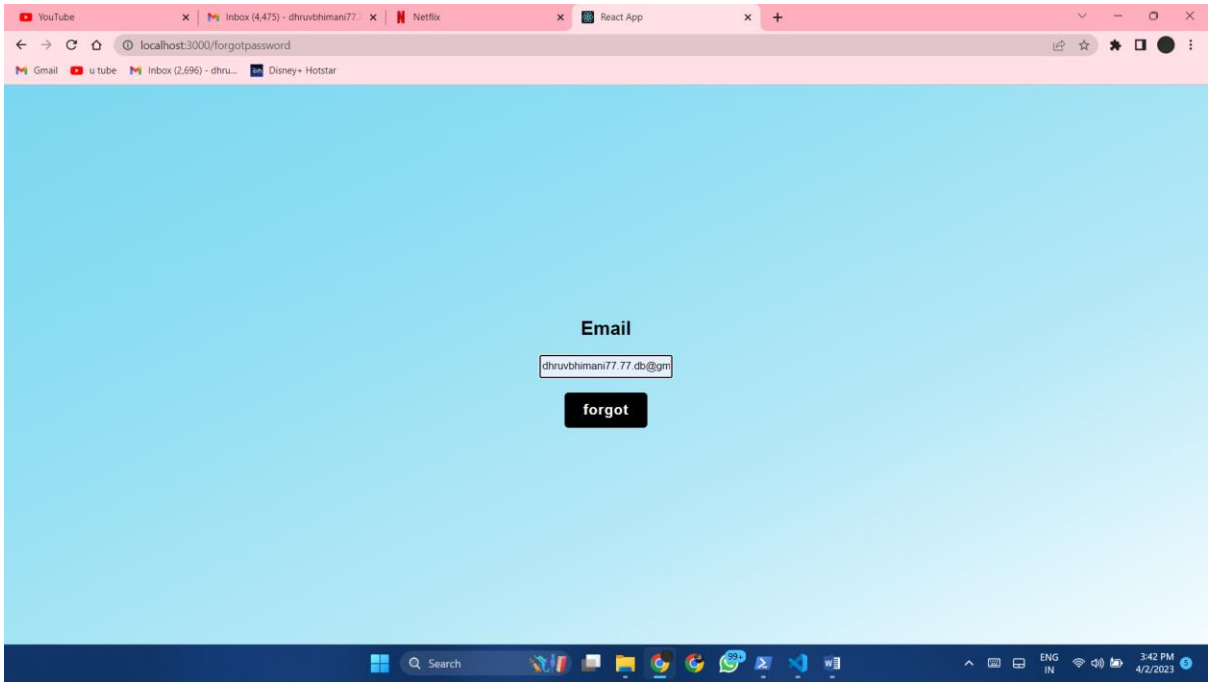
6. Update dates



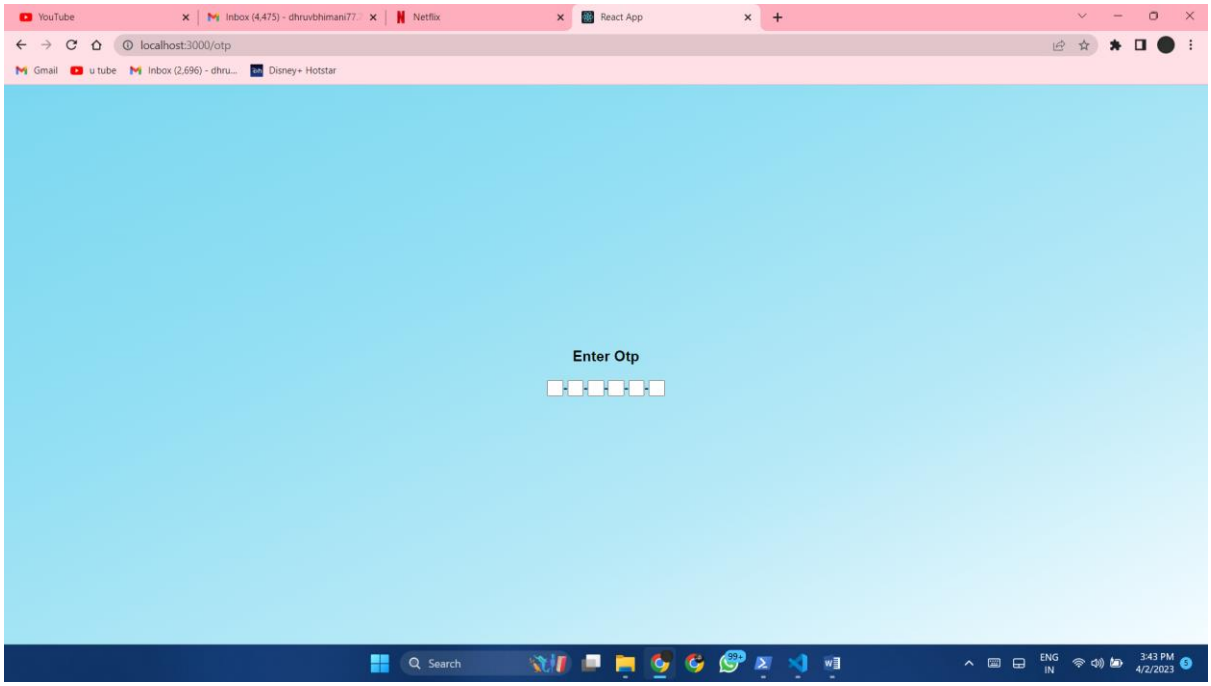
7. Update profile

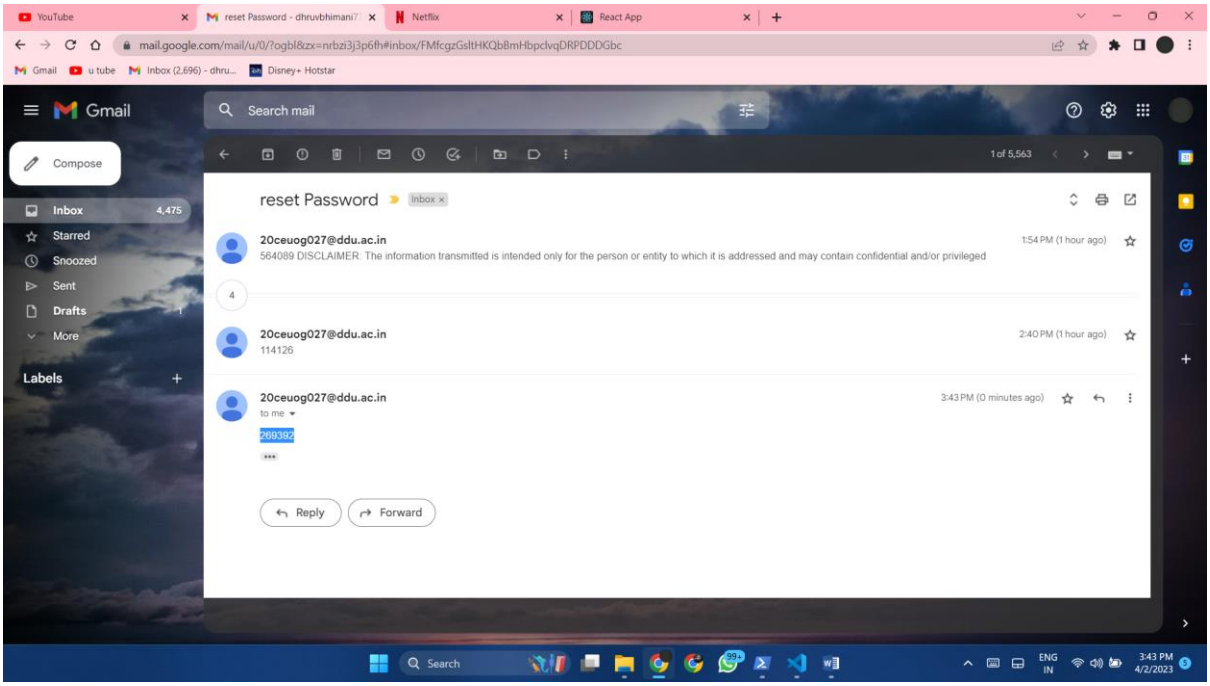


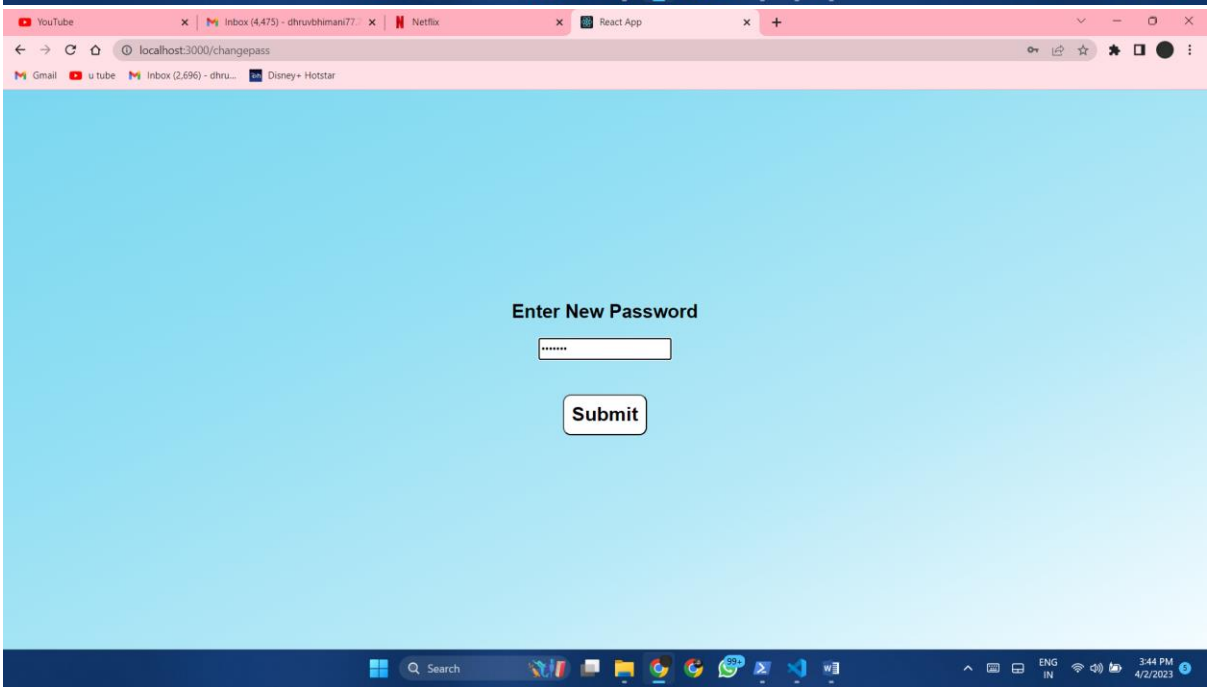
8. Forgot password



9. Enter otp




YouTube
reset Password - dhruvbhimani7
Netflix
React App
mail.google.com/mail/u/0/?ogbl&zx=nrbzi3j3p6fh#inbox/FMfcgzGsltHKQbBmHbpclvqDRPDDDGbc
Gmail
Search mail
Compose
Inbox 4,475
Starred
Snoozed
Sent
Drafts
More
Labels
1 of 5,563
reset Password
Inbox
20ceuog027@ddu.ac.in
1:54 PM (1 hour ago)
564089 DISCLAIMER: The information transmitted is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged
4
20ceuog027@ddu.ac.in
2:40 PM (1 hour ago)
114126
20ceuog027@ddu.ac.in
3:43 PM (0 minutes ago)
to me
269382
Reply
Forward
Search
ENG
IN
3:43 PM
4/2/2023


YouTube
Inbox (4,475) - dhruvbhimani77
Netflix
React App
localhost:3000/changepass
Enter New Password
Submit
Search
ENG
IN
3:44 PM
4/2/2023

Conclusion:

The functionalities are implemented in system after understanding all the system modules according to the requirements.

Functionalities that are successfully implemented in the system are:

- See available providers
- Assign particular provider to user
- Update dates of delivery
- Appoint some other providers when needed
- Final billing at the end of month

After the implementation and coding of system, comprehensive testing was performed on the system to determine the errors and possible flaws in the system.

Limitations and Future Extensions of System:

- In our website, we can add many different functionalities like add distance from user to provider so nearest provider can be shown first.
- Send massage as soon as provider deliver some product so that both can verify easily
- Send bill invoice through mail or whatsapp and generate pdf of the same
- Auto validation for new providers

All these thing can be further extended and we can add more functionalities as per the user requirements in future for their convenience.

Bibliography:

Websites:

- www.youtube.com
- https://code.visualstudio.com/