DATABASE MANAGEMENT SYSTEM
UE19CS301
INTELLIGENT METRO MANAGEMENT SYSTEM

Team-members:

Venuthurla Venkata Pradeep Reddy PES1UG19CS564

Y Nikhil Bharadwaj PES1UG19CS586

Yamajala Siddhardha PES1UG19CS588

Languages used for frontend :

1. HTML
2. CSS

Languages used for Database connection :

1. Python.

Dependencies installed :

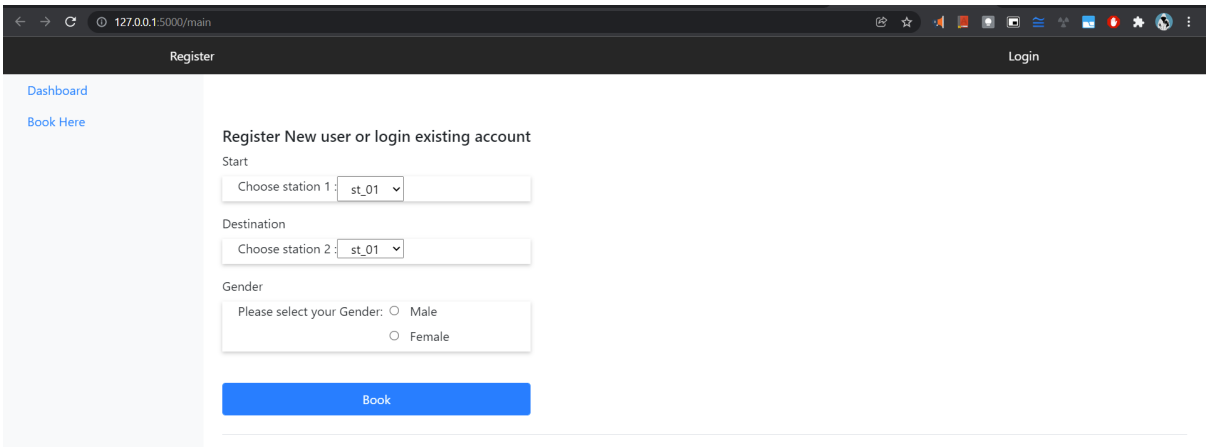
1. Flask: Flask is a web framework, it's a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like url routing, template engine. It is a WSGI web app framework.

2. Psycopg2: Psycopg is the most popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several

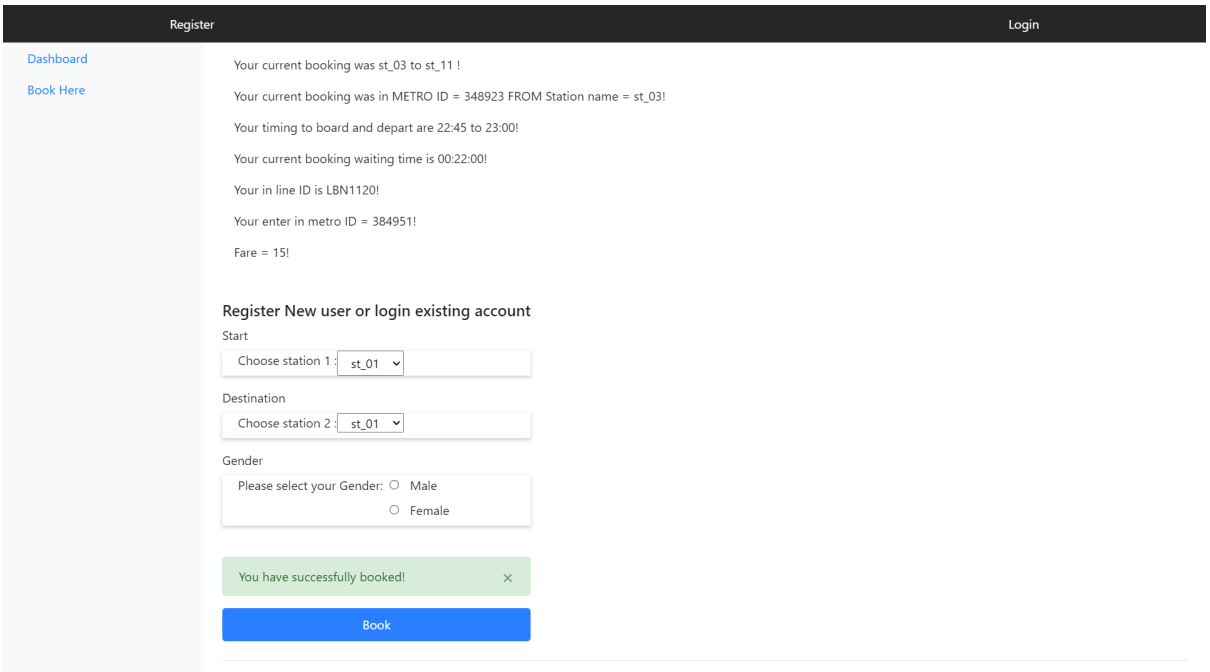
threads can share the same connection). It was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent INSERTs or UPDATEs.

Queries implemented in frontend:

Main page:



This page is used by the unregistered users to book tickets for their metro travel.



Platform table:

Query Editor | Query History

```
SELECT * FROM public.platform
```

Data Output | Explain | Messages | Notifications

	platform_no integer	arrival_time character varying (30)	departure_time character varying (30)	waiting_time character varying (10)	station_id integer	user_id integer
94	1	22:55	23:20	00:27:00	348921	10097
95	1	22:45	23:00	00:22:00	348923	10098
96	1	22:50	23:05	00:23:00	348922	10099

Ticket Table:

```
SELECT * FROM public.ticket
ORDER BY ticket_id ASC
```

Data Output | Explain | Messages | Notifications

	ticket_id [PK] integer	date_cur character varying (15)	start_time character varying (30)	end_time character varying (30)	fare integer	start character varying (30)	destination character varying (30)	user_id integer
93	349483	07:12:2021	22:25:20	23:20	25	st_01	st_11	10096
94	349484	07:12:2021	22:28:40	23:20	25	st_01	st_11	10097
95	349485	08:12:2021	22:23:26	23:00	15	st_03	st_11	10098

Boards Table:

Query Editor | Query History

```
SELECT * FROM public.boards
```

Data Output | Explain | Messages | Notificatic

	metro_id integer	user_id integer	station_id integer
93	384951	10096	348921
94	384951	10097	348921
95	384951	10098	348923

Users Table:

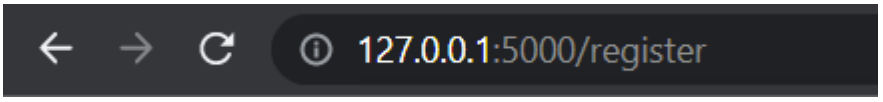
Query Editor Query History

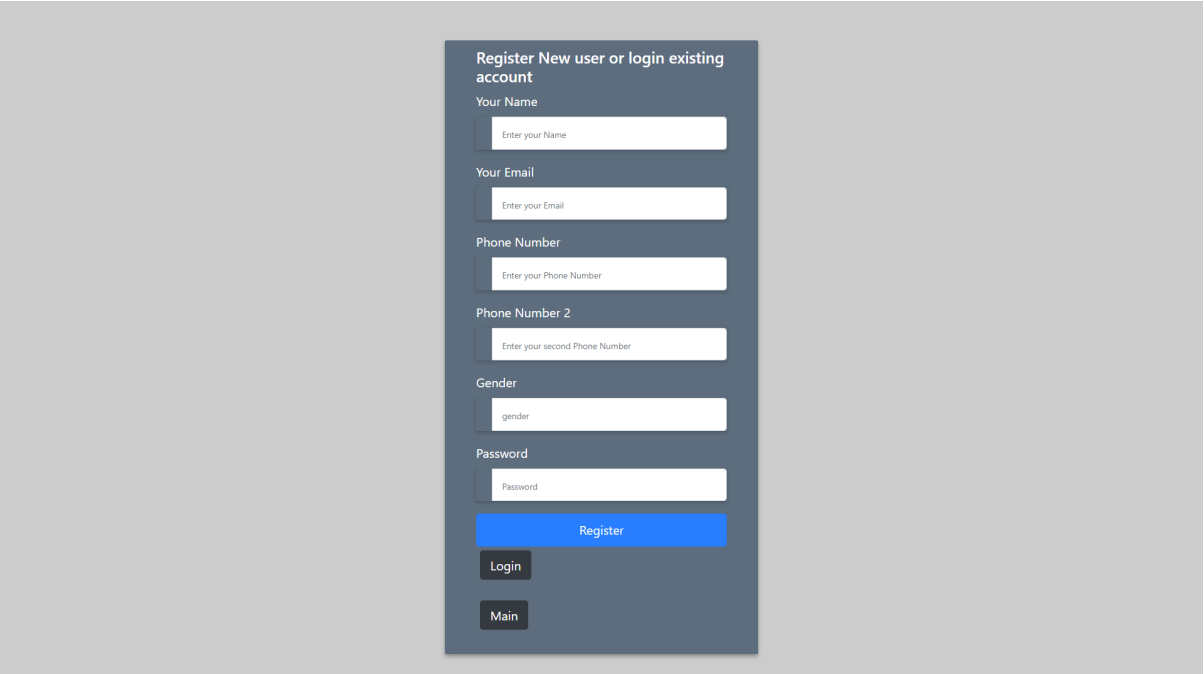
```
1 SELECT * FROM public.users
2 ORDER BY user_id ASC
```

Data Output Explain Messages Notifications

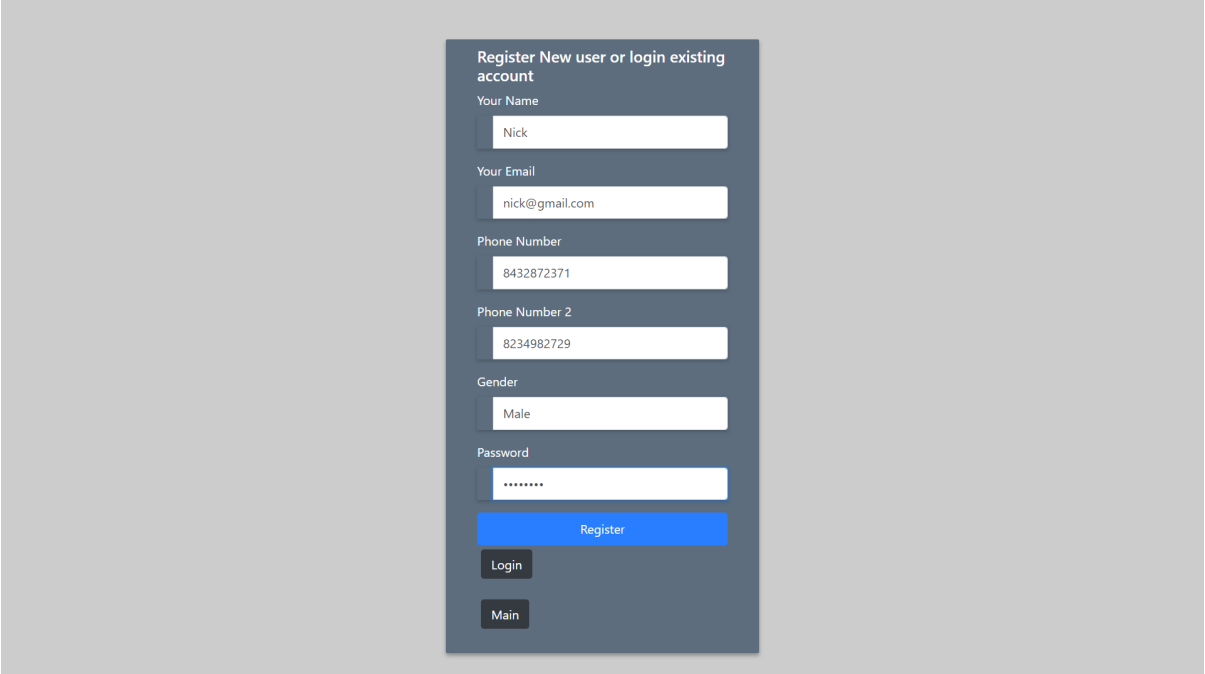
	user_id [PK] integer	gender character varying (10)	station_id integer
96	10096	M	348921
97	10097	M	348921
98	10098	M	348923

"User Registration" is for those who are looking for a metrocard for frequent usage of metro services.

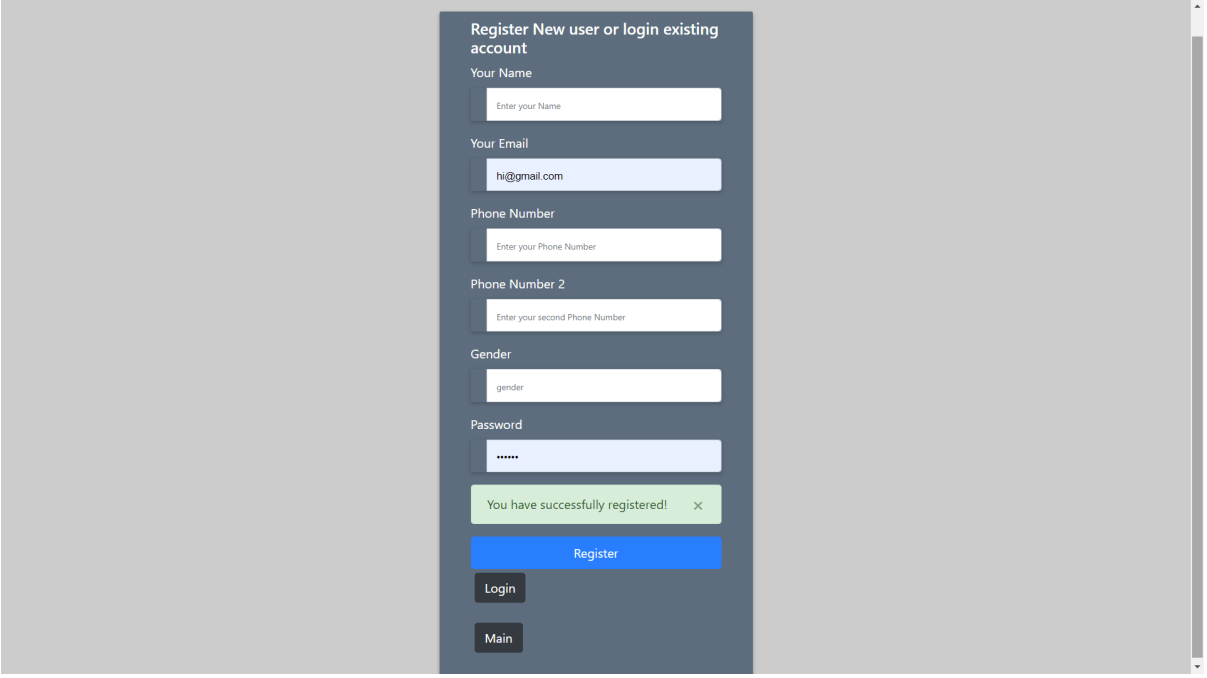




After filling out the fields:



Successfully registered:



Members Table:

```
SELECT * FROM public.members
ORDER BY email_id ASC
```

Data Output Explain Messages Notifications

	email_id [PK] character varying (30)	name character varying (20)	gender character varying (10)	password character varying (30)
10	hi@gmail.com	hi	M	qweqwe
11	kamarthikaran@gmail.com	karan	M	123456
12	nick@gmail.com	Nick	Male	nick@123

Metrocard Table:

```
SELECT * FROM public.metro_card
ORDER BY card_id ASC
```

Data Output Explain Messages Notifications

	card_id [PK] integer	balance integer	start character varying (30)	destination character varying (30)	s_time character varying (30)	d_time character varying (30)	email_id character varying (30)
15	98738405	150	st_01	st_01	17:04:53	17:04:53	vishnu@gmail.com
16	98738406	200125	st_01	st_11	22:55	23:20	sasankcharishma143@gmail.com
17	98738407	635	st_02	st_10	22:50	23:05	nick@gmail.com

Payment Table:

metro11/postgres@PostgreSQL 13

Query Editor Query History

```
SELECT * FROM public.payment
ORDER BY payment_id ASC
```

Data Output Explain Messages Notifications

	payment_id [PK] integer	status character varying (20)	card_id integer
16	1817207	paid	98738405
17	1817208	paid	98738406
18	1817209	paid	98738407

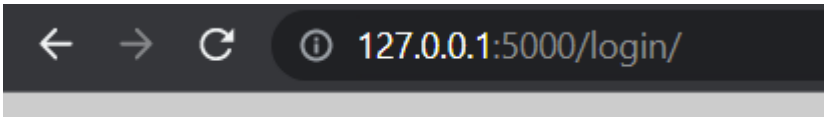
Phone No of Members Table:

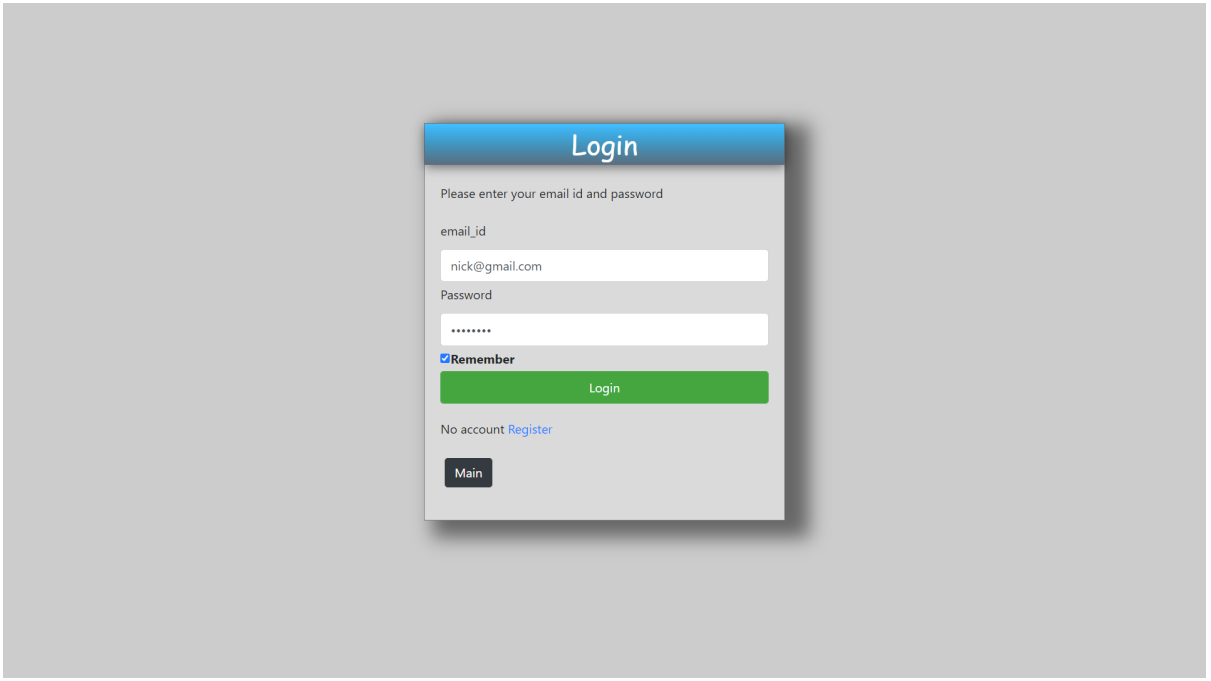
```
SELECT * FROM public.phoneno
```

Data Output | Explain | Messages | Notifications

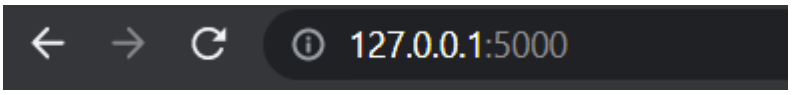
	p_no character varying (15)	p_no2 character varying (15)	email_id character varying (30)
15	9390137310	9000915863	vishnu@gmail.com
16	9411431432	1431431431	sasankcharishma143@gmail.com
17	8432872371	8234982729	nick@gmail.com

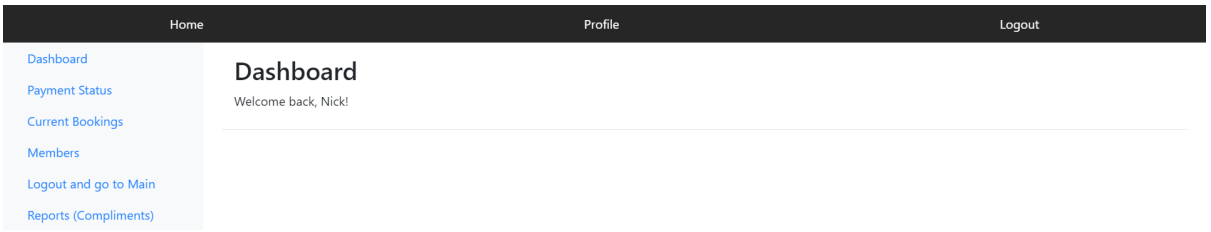
Login Page:



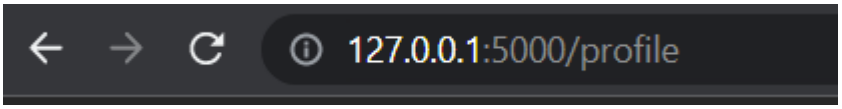


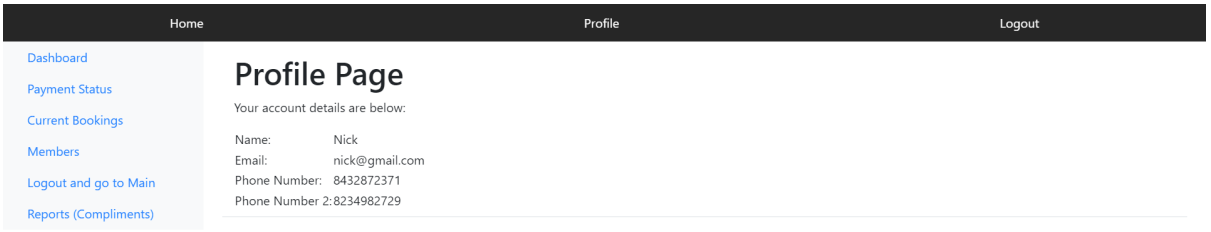
After Login getting inside dashboard:





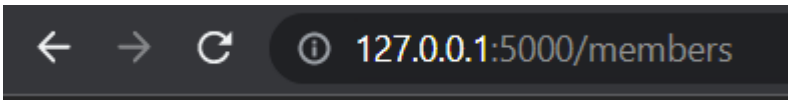
“Profile Page” of Members:

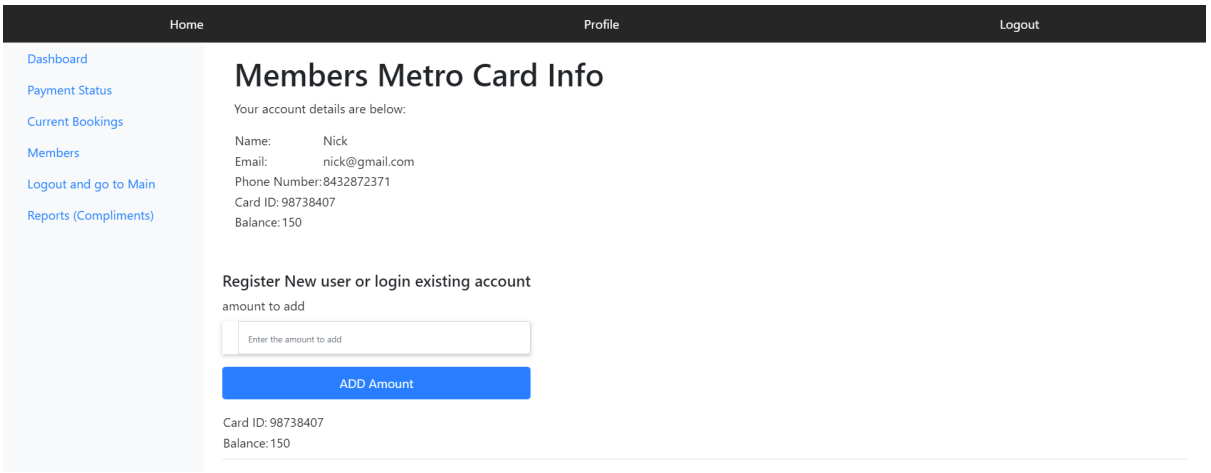




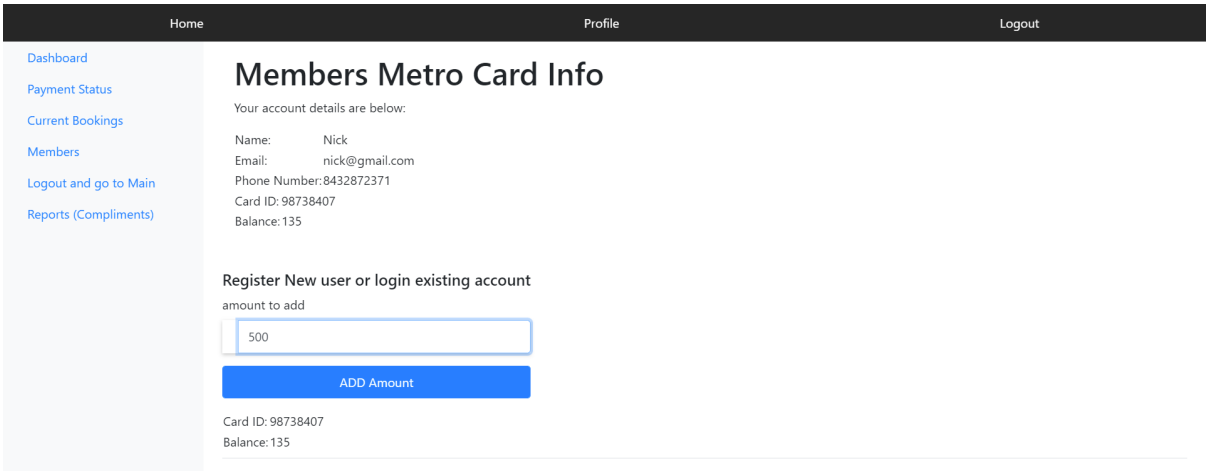
“Members” where member can check balance add balance:

Checking Balance:

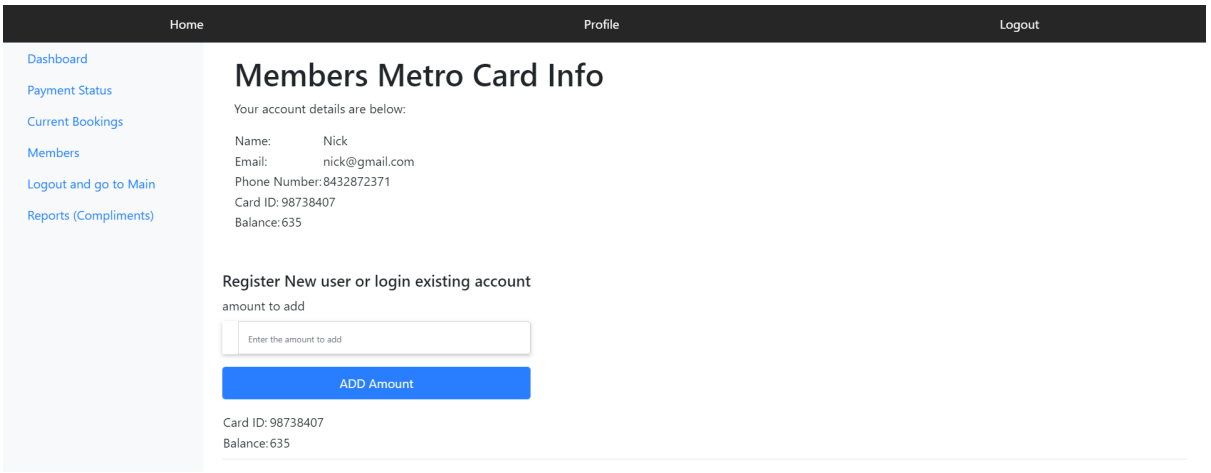




Adding Balance: “ Updating Balance” for the member who logged in.



The balance is updated accordingly in the below image and also in database:

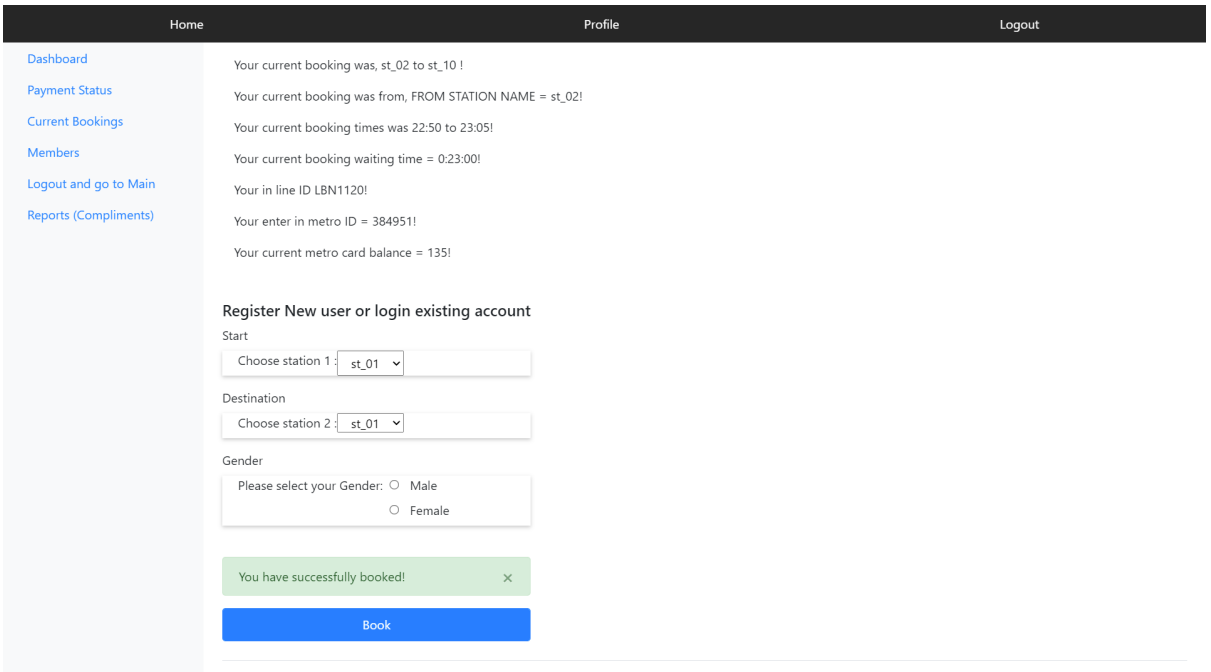


“Bookings” of Members:

Here one can see there previous bookings or current booking if they did now and will be updated with the new booking status:

Here Member can see which metro one has to enter, waiting time for the metro to arrive at the station the member is boarding, start time, end time of the metro journey.

The fare for the current booking will be automatically deducted from the members metrocard balance.



Updated balance Metrocard Table:

Query Editor | Query History | Scratch Pad

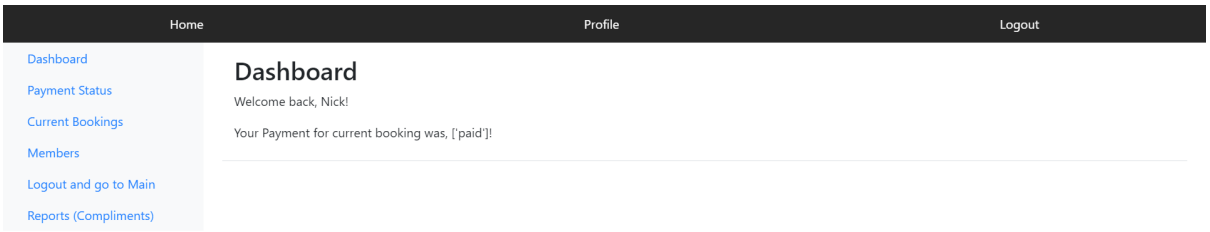
```
SELECT * FROM public.metro_card
ORDER BY card_id ASC
```

Data Output | Explain | Messages | Notifications

	card_id [PK] integer	balance integer	start character varying (30)	destination character varying (30)	s_time character varying (30)	d_time character varying (30)	email_id character varying (30)
17	98738407	635	st_02	st_10	22:50	23:05	nick@gmail.com

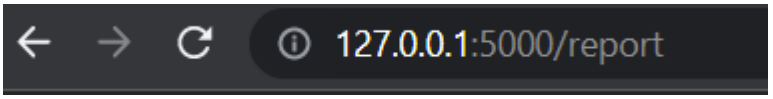
“Current payment status”:

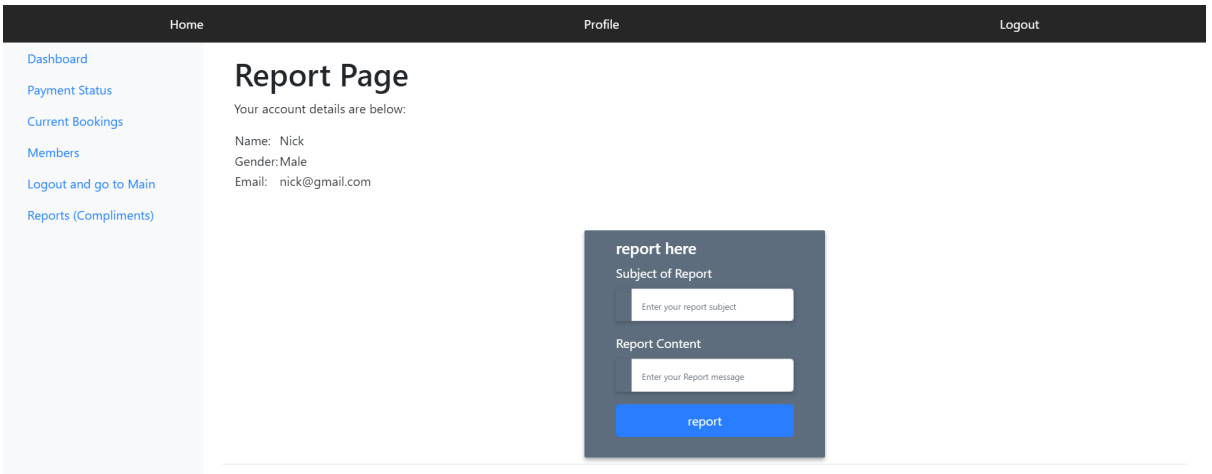
Here the Member can see their payment status:

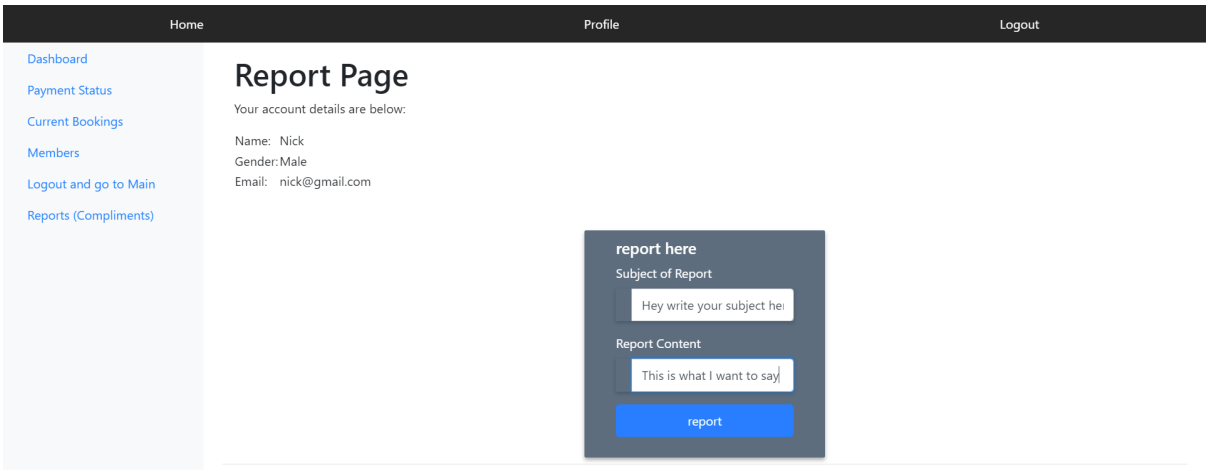


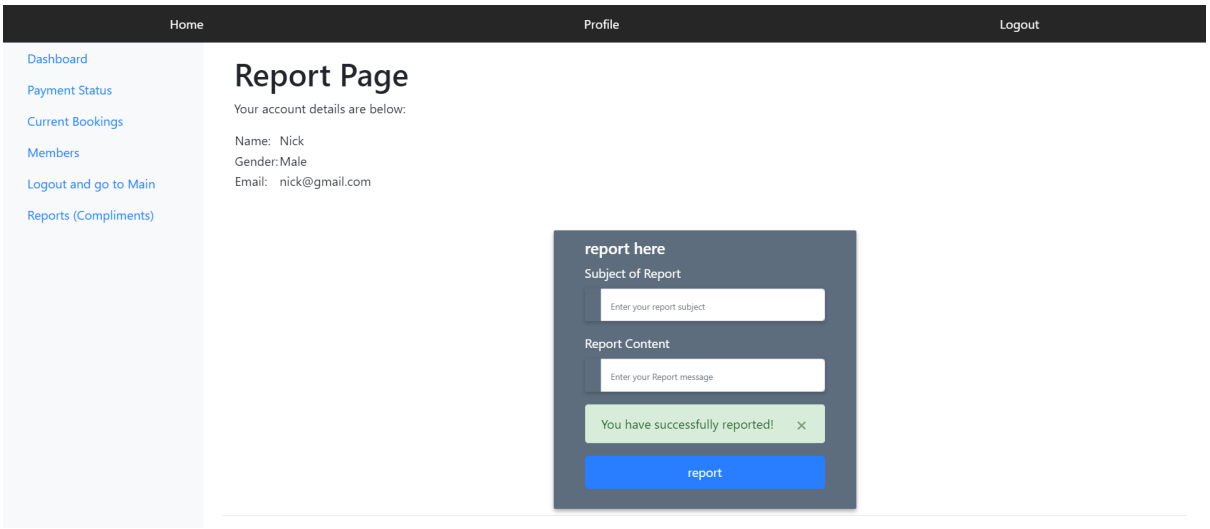
“Members can Report a problem” using Report Complaints page:

By writing Subject and Content of the report:









Reports Table:

```
1  SELECT * FROM public.report
2
```

Data Output Explain Messages Notifications

	report_subject character varying (50)	report character varying (300)	email_id character varying (30)
10	Hey write your subject here	This is what I want to say	nick@gmail.com

> Business expansion can lead to lots of schema and constraint changes in the Database.

For example if we want to give offers for certain users then we have to create a separate table and record the validity of the offer for the specific use. Assuming we need to restrict the number of customers who are taking the rooms to the customers who are having age more prominent than 18 then we want to add the requirement in the data set so customers with age under 18 can't save the rooms.

Database migration :

If the current database should be migrated to an alternative one, we choose mongoDB. A modern, document-based database such as MongoDB can be a great choice over an RDBMS like PostgreSQL.

Because,

It considers the schema changes that would be best for your data, while keeping in mind MongoDB schema best practices and avoiding anti-patterns.

Querying data and their associated data objects is often faster in MongoDB than in SQL, which uses expensive JOIN statements.

MongoDB distributed, scale-out architecture allows your database to grow as your application grows. This is in contrast to PostgreSQL, which is not natively distributed.

MongoDB's flexible schema enables changes to an application's data model or schema to be deployed quickly and flexibly, without the need for migrations or to update query statements in legacy code.

******

Contributions:

All members have contributed equally in all aspects , but we individually worked more on some aspects parallely

Yamajala Siddhardha - Creation of database , insertion of values , Queries , linking the database with front - end ,helped in creation of front - end

Y Nikhil Bharadwaj - E-R diagram , Granting permissions to different users in our database , Queries , Creation of some HTML pages,helped in creation of front-end

V V Pradeep Reddy - Creation of database , insertion of values , linking the database with the front end , Creation of HTML pages + CSS , creation of front - end

Time spent on each aspect :

1) E-R diagram + conversion to relational schema (5-6 hours)
2) Creation of database + insertion of values (4-5 hours)
3) Granting permission and Queries (4-5 hours)
4) front - end (took about a week)