PROJECT PART II

Course: Principles of Database Systems

Section: CS-GY 6083 – INET SUMMER 2022

Submission Date: 26th August 2022

Team members:

1. **Niko Ganev (ng2451)**
2. **Bhargav Makwana (bm3125)**
3. **Jash Merchant (jjm9801)**

Table of Contents

Summary

In this project we implemented a new information system for the insurance company **We Do Secure (WDS).** This system is based on a relational database with 12 tables which securely stores the data for users and admins and allow users to apply for insurance policies as well as pay their invoices.

The goal of our application is to increase the number of users reached by WDS through an online portal that can be found online by anyone looking for insurance and convert these potential users into actual paying customers thanks to the easy process and appealing User Interface.

Besides the increase in the number of applications and general awareness of the WDS brand, our solution aims at differentiating WDS from their competition by guaranteeing data security as well as improving worker productivity through simple workflows for both customers and insurance agents.

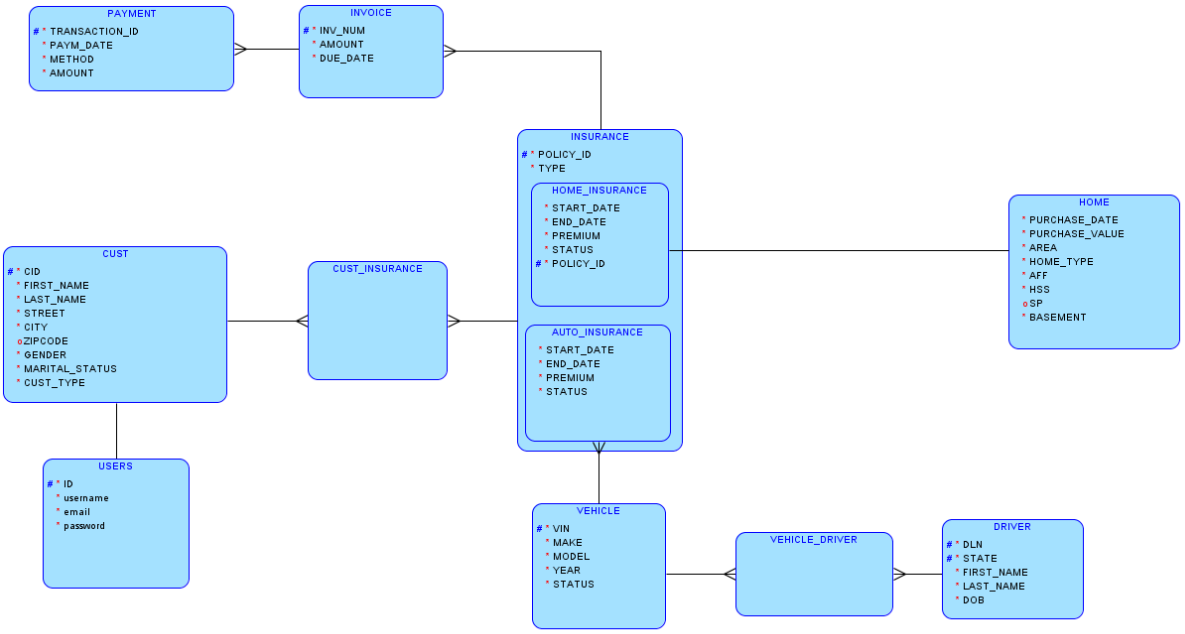
The admin panel allows the insurance agents to monitor their own performance and manage their clients effortlessly empowering an optimal experience for all parties.

Technology Stack

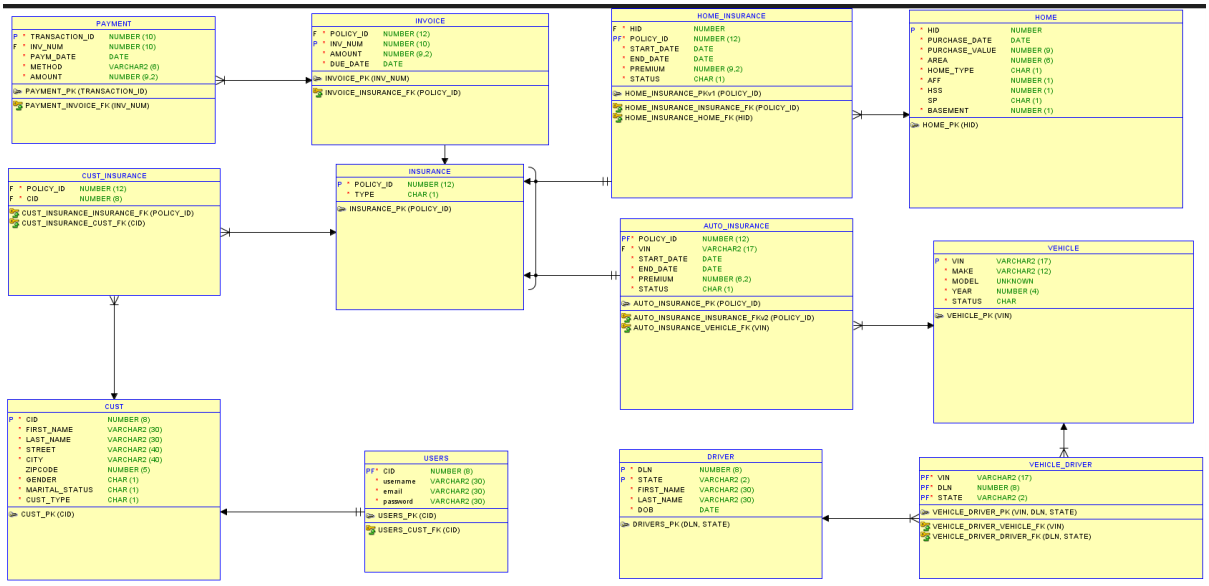
Ours is a mobile-responsive full-stack web application developed for users that want to enrol for Home and Insurance policies, using HTML, CSS, Bootstrap and JavaScript on the frontend while, Python, Flask and sqlite3 on the backend, and is deployed on Heroku (https://pds-project.herokuapp.com/login)

Logical and Relational ERD

Logical ERD



Relational ERD



Assumptions:

1. A customer can enrol in more than one insurance and an insurance contract can have more than one person on contract such as husband and wife and hence customer to insurance becomes a many to many relationship requiring an intersect table which we have named cust_insurance.
2. Similarly, we have considered vehicle and driver to be a many to many relationship needing vehicle_driver as an intersect table.
3. Insurance supertype possess two subtypes - Home insurance and Auto insurance.
4. A single insurance can have multiple invoices generated and each invoice could be paid in several installments.
5. Address is a composite attribute resolved into street, zip code and city.
6. Name is a composite attribute resolved into first and last name

Data Tables

Tables:

```
1  SELECT table_name FROM user_tables;
```

TABLE_NAME
AUTO_INSURANCE
CUST
CUST_INSURANCE
DRIVER
HOME
HOME_INSURANCE
INSURANCE
INVOICE
PAYMENT
USERS
VEHICLE
VEHICLE_DRIVER

Download CSV

12 rows selected.

Total Number of records

```
SELECT COUNT(*) CUST_COUNT
FROM CUST;
```

CUST_COUNT
25

Download CSV

```
SELECT COUNT(*) CUST_INSURANCE_COUNT
FROM CUST_INSURANCE;
```

CUST_INSURANCE_COUNT
30

Download CSV

```
SELECT COUNT(*) AUTO_INSURANCE_COUNT
FROM AUTO_INSURANCE;
```

AUTO_INSURANCE_COUNT
29

Download CSV

```
SELECT COUNT(*) VEHICLE_COUNT
FROM VEHICLE;
```

VEHICLE_COUNT
29

Download CSV

```
1  SELECT COUNT(*) VEHICLE_DRIVER_COUNT
2  FROM VEHICLE_DRIVER;
3
```

VEHICLE_DRIVER_COUNT
40

Download CSV

```
1  SELECT COUNT(*) VEHICLE_COUNT
2  FROM VEHICLE;
3
```

VEHICLE_COUNT
29

Download CSV

```
1  SELECT COUNT(*) HOME_INSURANCE_COUNT
2  FROM HOME_INSURANCE;
3
```

HOME_INSURANCE_COUNT
30

Download CSV

```
1  SELECT COUNT(*) HOME_COUNT
2  FROM HOME;
3
```

HOME_COUNT
30

Download CSV

```
1  SELECT COUNT(*) INVOICE_COUNT
2  FROM INVOICE;
3
```

INVOICE_COUNT
37

Download CSV

```
1  SELECT COUNT(*) INSURANCE_COUNT
2  FROM INSURANCE;
3
```

INSURANCE_COUNT
30

Download CSV

```
1  SELECT COUNT(*) PAYMENT_COUNT
2  FROM PAYMENT;
3
```

PAYMENT_COUNT
50

Download CSV

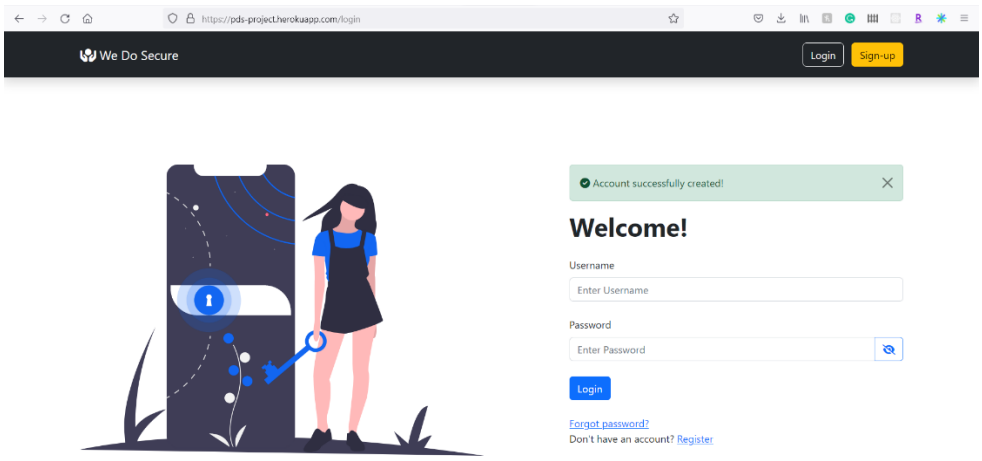
```
1  SELECT COUNT(*) USER_COUNT
2  FROM users;
```

USER_COUNT
14

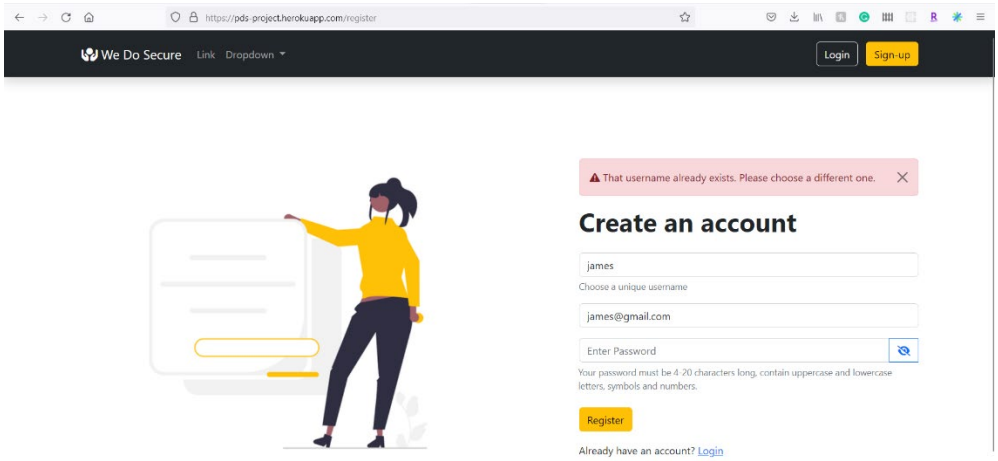
Download CSV

Demo

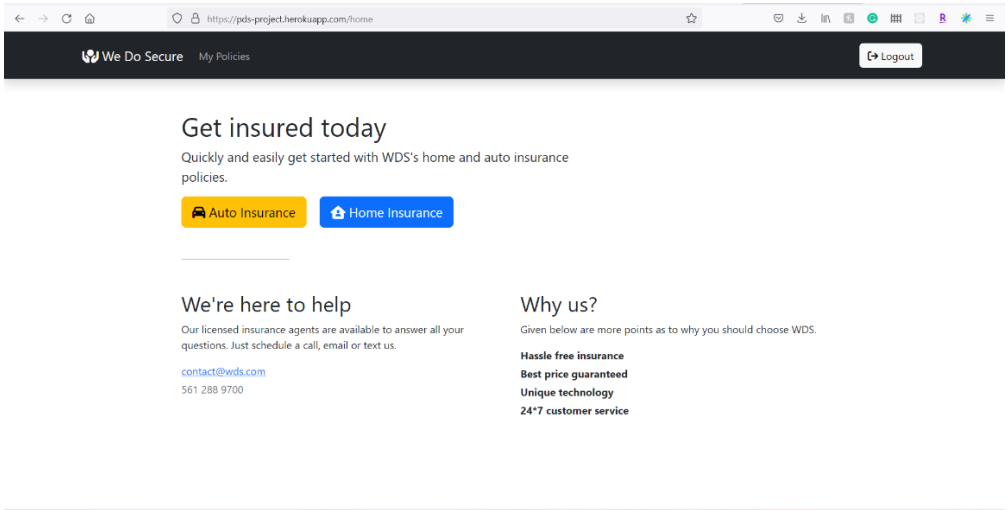
1. Login



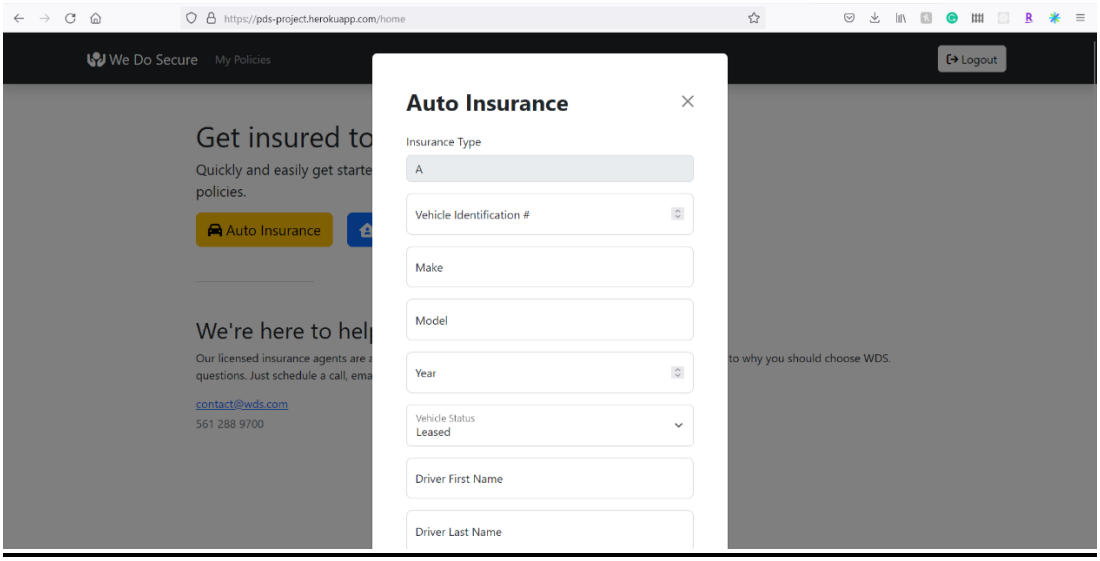
2. Registration



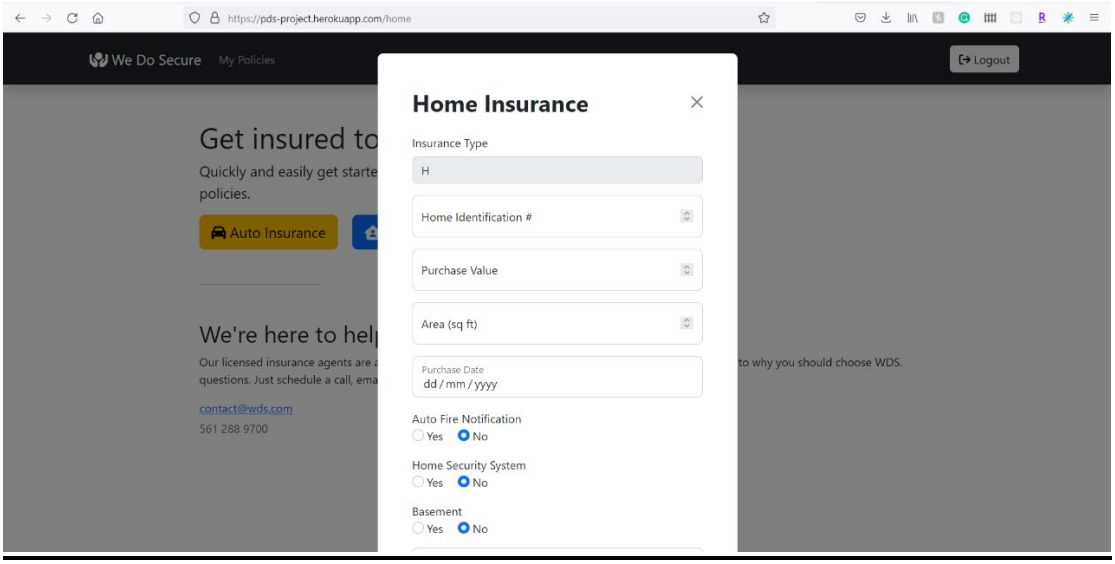
3. Home



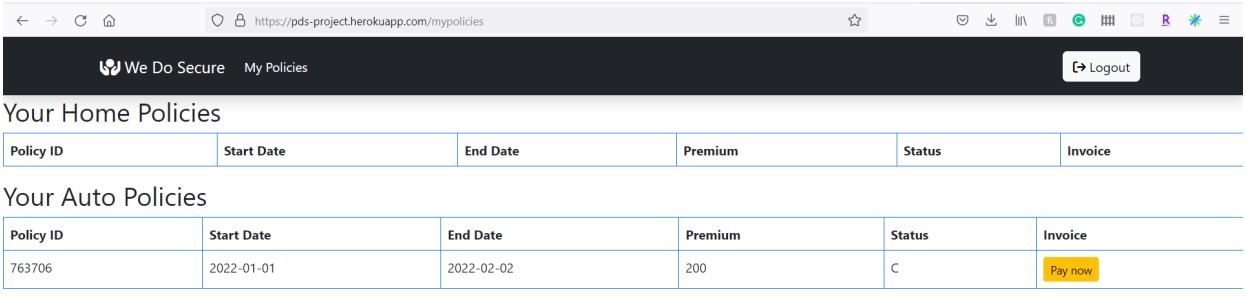
4. Auto Insurance Form



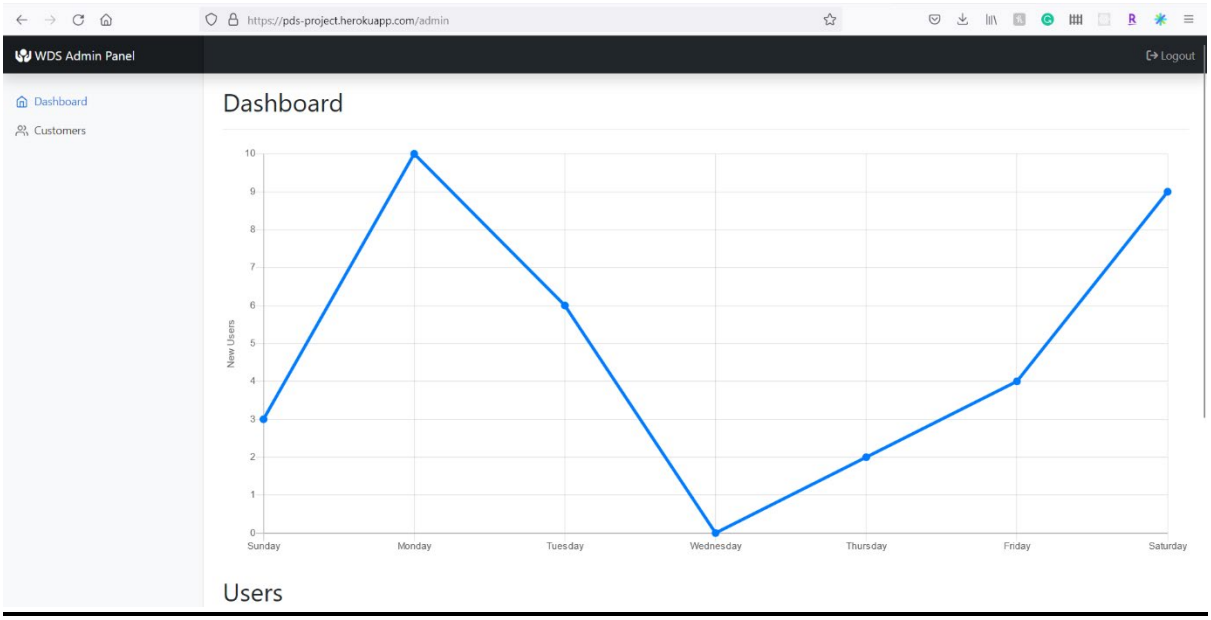
5. Home Insurance Form

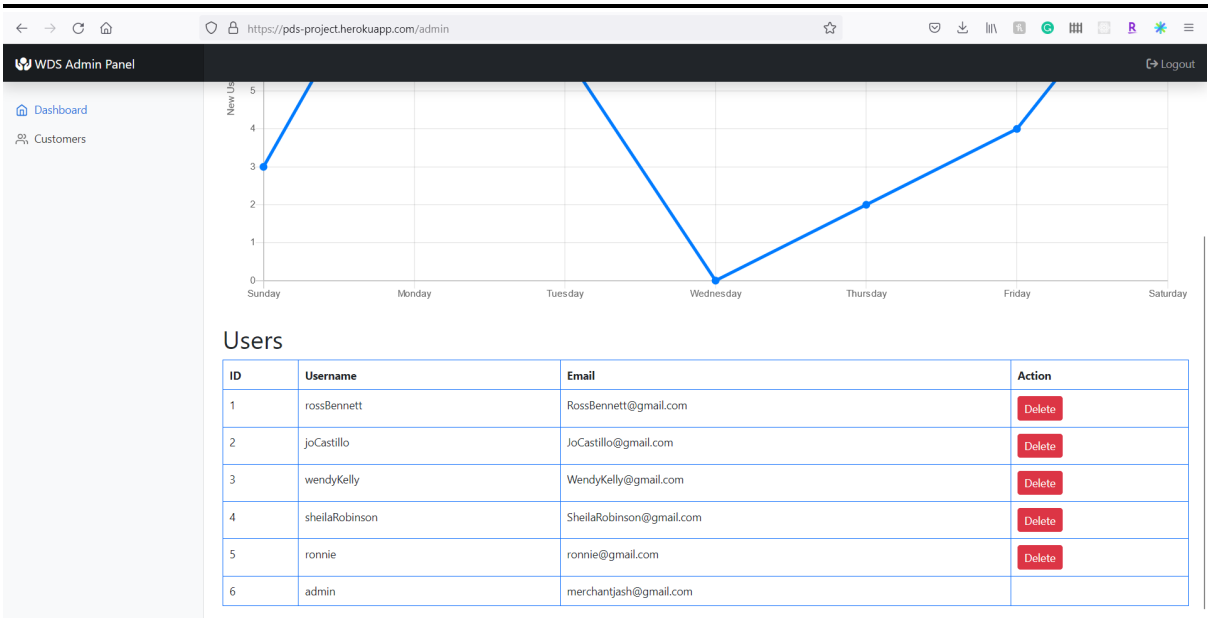


6. My Policies page

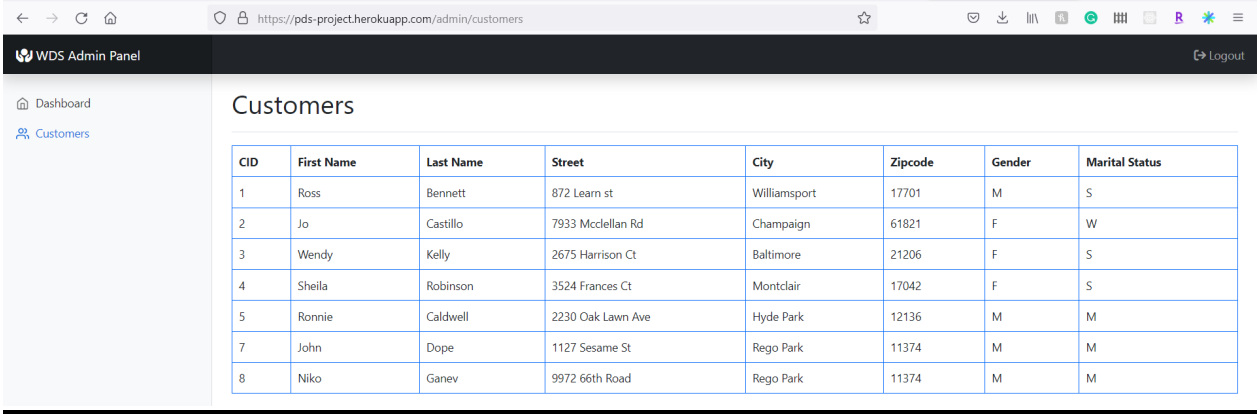


7. Admin Panel

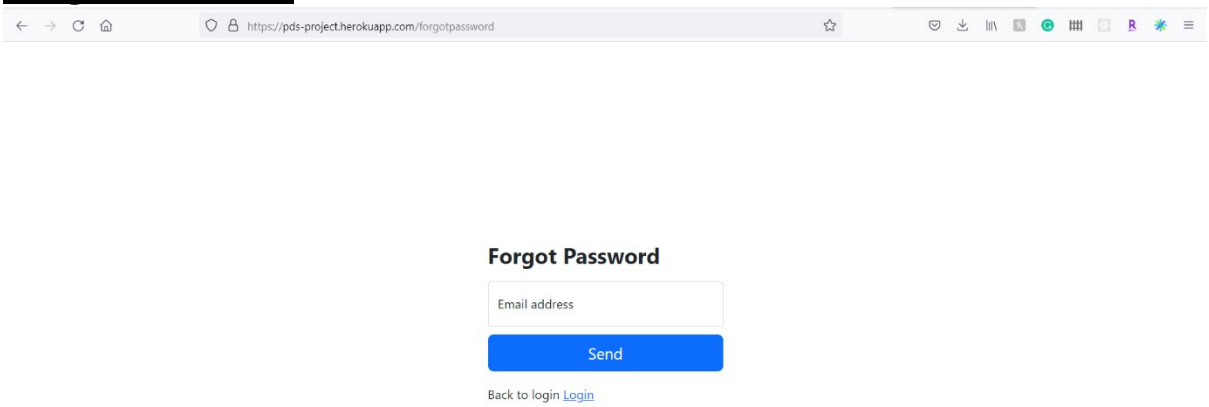




8. Admin Panel (Customers List)



9. Forgot Password



Features

- Users can perform **CRUD (C-Create, R-Read, U-Update, D-Delete)**
- SQL injection prevention using SQLAlchemy library
- Cross site scripting prevention
- Concurrency
- Mobile-responsive UI
- User registration **(C)**
- Password encryption using bcrypt library **(C)**
- Home Insurance application **(C)**
- Auto Insurance application **(C)**
- Frontend and backend error validations **(R)**
- Allow only unique usernames **(R)**
- User login **(R)**
- Graph to display number of users registered each day **(R)**
- My policy page for users to view their respective policies and pay invoice **(R)**
- Admin panel to view users list **(R)**
- Customers section under admin panel to view users who successfully converted to customers by enrolling in a policy **(R)**
- Reset password feature by emailing a unique link **(U)**
- Delete users and corresponding customer entries **(D)**
- Deployed to Heroku (https://pds-project.herokuapp.com/login)

Lessons learned

In the span of the entire project, that is from understanding the Business case to the part of developing a prototype application we learned a lot of useful things jotted down below:

- Understanding the Business requirements and developing an ER model based on it has its own challenges, like what kind of relationship best serves the purpose (1:M, M:N, etc.).

- We enjoyed and the team building and coordination process, realized the importance of a team and its members.

- We learned that implementing an actual application from designing schema is actually fascinating. While it has its own challenges pertaining to the tech stack used, it also helps one to learn the tech skills as well has improve debugging skills.

- The project timeline and deadline play an important role in restricting/scaling the scope of the application.

- I learned how to use Github effectively and realized how important it is while working on a team project.

- I believe we could have done even better if we started working on the project a little early with same seriousness as there were a lot of other things that we wanted to implement and improve but due to time constraints we couldn’t.

- Got to learn a lot too from the teammates, each one of us had different speciality so we focussed on that and worked for a common goal and fortunately things went well.

SQL Queries and results

Q1)

```
1  SELECT DISTINCT c.policy_id, c.cid
2  FROM AUTO_INSURANCE a
3  JOIN INSURANCE b ON a.policy_id=b.policy_id
4  JOIN CUST_INSURANCE c ON c.policy_id=b.policy_id
5  WHERE cid=6;
```

POLICY_ID	CID
106	6
127	6

Download CSV

2 rows selected.

This query returns all insurance policies for customer with id 6.

Q2)

```
1  SELECT b.ipolicy_id, b.amount
2  FROM insurance a JOIN invoice b ON a.policy_id = b.ipolicy_id
3  WHERE a.policy_type = 'H' AND b.amount >
4
5  ALL (SELECT b.amount
6  FROM insurance a JOIN invoice b ON a.policy_id = b.ipolicy_id
7  WHERE a.policy_type = 'A');
8
```

IPOLICY_ID	AMOUNT
103	975.34
123	896.17

Download CSV

2 rows selected.

This query returns all home insurance policies that are more expensive than all of the auto insurance policies.

Q3

```
1
2  SELECT inv_num, amount
3  FROM INVOICE
4  WHERE    amount > (SELECT AVG(amount) FROM INVOICE);
5
```

INV_NUM	AMOUNT
8007482823	769.67
9948137411	633.13
5413330330	975.34
1113990911	818.01
3249609658	526.11
4088251801	632.81
2654430917	678.38
9844216976	728.34
3931781541	541.27
2121573015	520.15

This query returns all invoices that are above the average invoice amount.

Q4)

```
1  SELECT HID FROM HOME
2  WHERE SP = 'O'
3
4  INTERSECT
5
6  SELECT HID FROM HOME_INSURANCE
7  WHERE PREMIUM > 200;
8
```

HID
2078102300
3120205778
4518817008
5990598804
6246892852
9491123506

Download CSV

6 rows selected.

This query returns the ids of all homes with an outdoor pool paying more than 200 dollars in home insurance premium.

Q5)

```
1  WITH x AS
2  (SELECT *
3  FROM INVOICE a JOIN PAYMENT b ON a.inv_num=b.inv_num)
4  SELECT IPOLICY_ID, PAYM_DATE FROM x
5  ORDER BY 2 DESC;
6
```

IPOLICY_ID	PAYM_DATE
108	25-AUG-22
108	25-AUG-22
111	25-AUG-22
124	21-AUG-22
123	21-AUG-22
124	19-AUG-22
115	17-AUG-22
128	17-AUG-22
123	16-AUG-22
111	16-AUG-22

This query returns the IDs and payment date of policies which received a payment the most recently.

Q6)

```
1  SELECT amount,
2   method,
3   rank() over (order by amount desc) as ranked_amount
4  FROM PAYMENT;
5
```

AMOUNT	METHOD	RANKED_AMOUNT
121.92	Credit	1
121.92	Credit	1
121.92	Debit	1
121.92	Debit	1
121.92	Credit	1
112.02	Debit	6
112.02	PayPal	6
108.05	Debit	8
108.05	Debit	8
104.58	PayPal	10
104.58	PayPal	10

This insurance returns the top payment amounts along with the method of payment allowing to look for corelation between payment amount and method.