Grab Customer Report System

Jezreel James Hallasgo
Ruth Margel Mendoza
Kyle Angelo Racuya

National University Philippines

June 24, 2024

I. Introduction

Project Overview

Grab(formerly known as Grab Taxi) is a technology company that offers a wide range of ride-hailing and logistics services through its app in Southeast Asia, specifically in Malaysia, Philippines, Singapore, Thailand, Vietnam and Indonesia. Grab is Southeast Asia's first decacorn and the biggest technology startup in the region. It became publicly traded on the NASDAQ in 2021, following the largest SPAC merger at the time. In 2023, Fast Company listed Grab amongst the most innovative companies in the Asia-Pacific region.

Objectives

By implementing a database-driven approach to identifying inconsistencies in records, the company can maintain trust with both riders and passengers by ensuring fair and transparent operations.

Scope

To ensure the database infrastructure will accommodate growth and adapt to evolving business needs, ensuring long-term viability and effectiveness. The system database-driven approaches to maintain high standards of record consistency, fostering trust, efficiency, and reliability in their operations.

Technologies Used

Technology	**Purpose**
Java	Main Language
MySQL	Database
JavaFX	Front End
SceneBuilder	Design

II. System Design

Database Design

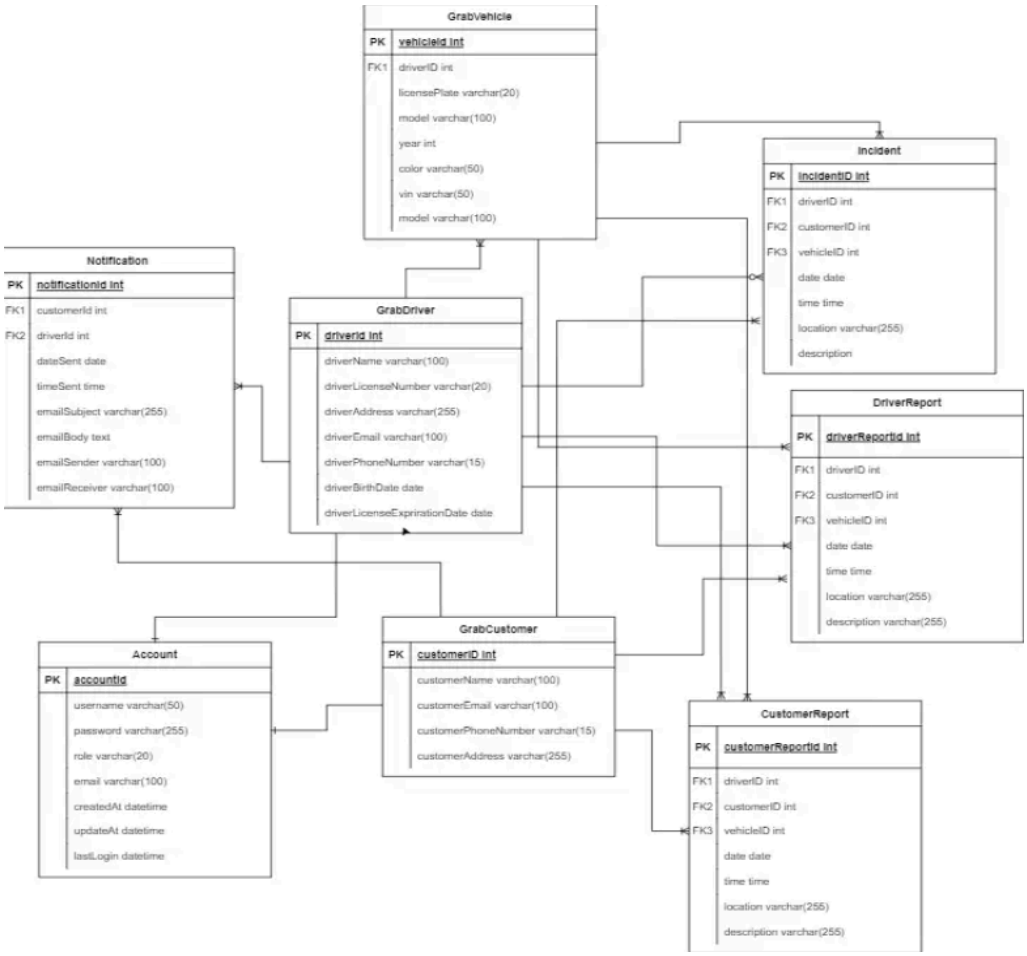
The data that has been stored is very relational, and that is why a relational database was used for the database, this project was created using MySQL. There are 4 Main entities.

1. **Account:** This entity holds the accounts in the database.
2. **Customer:** This entity holds the customer's role in the database.
3. **Driver:** This entity holds the driver's role in the database.
4. **Vehicle:** This entity holds the registered vehicle in the database.

The other entities are created for normalization. These are the entities.

1. **Customer Report:** Holds all the feedback from the customers.
2. **Driver Report:** Holds all the feedback from the drivers.
3. **Notification:** Holds all notifications associated with an account. Each notification has its own corresponding action.
4. **Reference Number:** Holds all reference numbers from the customer's ride, and driver's vehicle, and customer.

Schema Design and ER Diagram



Normalization

1. **First Normal Form (1NF)**

   Every table in the database needs to have a main key, and all the values in the fields need to be simple and atomic in order to reach the First Normal Form. The above Entity-Relationship Diagram (ERD) makes it clear that each table has a main key and that the values in each column are simple and can't be split. In this way, the tables meet the requirements of the First Normal Form.

2. **Second Normal Form (2NF)**

A table must first meet the requirements of the First Normal Form before it can also meet the requirements of the Second Normal Form [1]. This is also necessary for any traits that aren't main keys to depend on the primary key completely. Because of this, each item in a table should only be related to the main key. The emailSubject, emailBody, emailSender, and emailReceiver are the only fields in the Notification table that should be dependent on the notificationID. It is evident from the Entity-Relationship Diagram (ERD) that all tables adhere to this principle. This means that they meet the requirements of the Second Normal Form.

3. **Third Normal Form (3NF)**

A table must stick to the Second Normal Form requirements before the Third Normal Form may be applied. It's crucial that non-primary key characteristics rely only on the main key and not on any other non-primary key qualities. This prevents the needless repetition of data. By way of illustration, the GrabDriver database shows that the driverID is linked to a plethora of information, such as driverName, driverLicenseNumber, address, email, phone number, age, etc. The Entity-Relationship Diagram (ERD) ensures that all tables follow this rule. This means that they all meet the Third Normal Form standards.

4. **Fourth Normal Form (4NF)**

It is against the Fourth Normal Form for a table to have more than one separate multi-valued fact about an object. To put it simply, columns that hold groups of values shouldn't be in a table. The given Entity-Relationship Diagram (ERD) makes sure that each property in each table has only one value. In particular, the color field of the

GrabVehicle table only has one color for each car. As a result, the tables follow the Fourth Normal Form.

5. **Fourth Normal Form (4NF)**

To get to the Fifth Normal Form, a table must be set up in a way that stops it from being broken down into smaller tables without losing any information. This makes sure that all the data can be put back together from smaller tables without any problems. Each record in the Account table, for instance, has its own unique accountID and contains information like login, password, job, email, createdDate, updatedAt, and lastLogin. The Entity-Relationship Diagram (ERD) makes it clear that each table is set up in a way that keeps all the important data and can't be split up any further without affecting the security of the data. This means that the ERD meets the standards of the Fifth Normal Form.

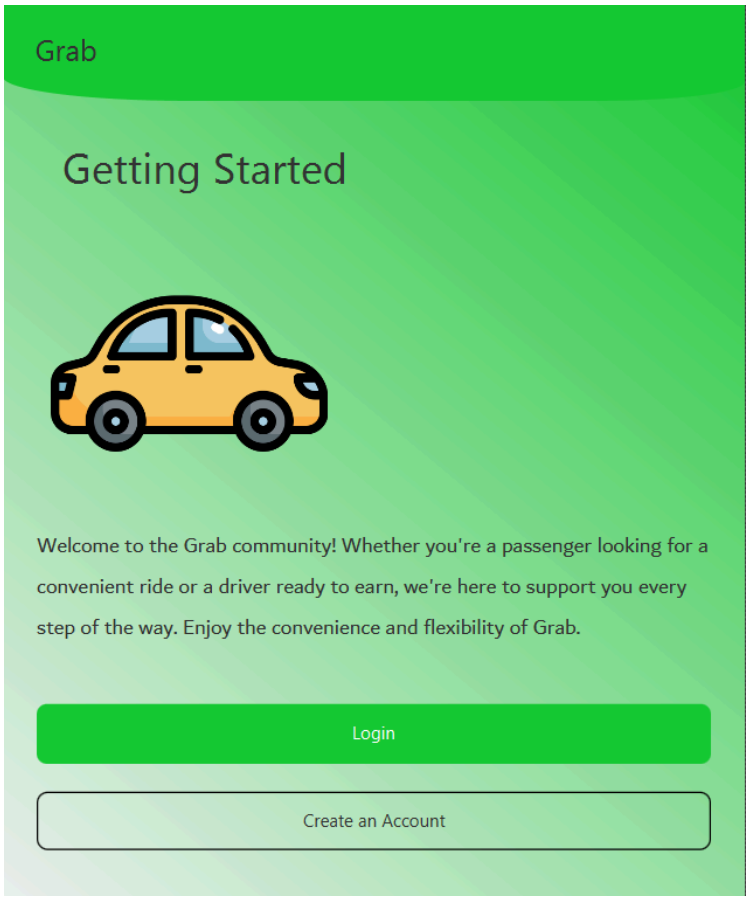
6. **Final Adjustments**

We can confirm that the field Entity-Relationship Diagram (ERD) meets all the requirements for the First, Second, Third, Fourth, and Fifth Normal Forms after a careful review. There is a major key in every table that holds all the basic numbers in all the fields and can't be split. Also, all traits that aren't the main key depend on the primary key directly and completely. There are no characteristics that can have more than one number, and the tables are broken down correctly so that no data is lost. This makes the information better organized and works better.

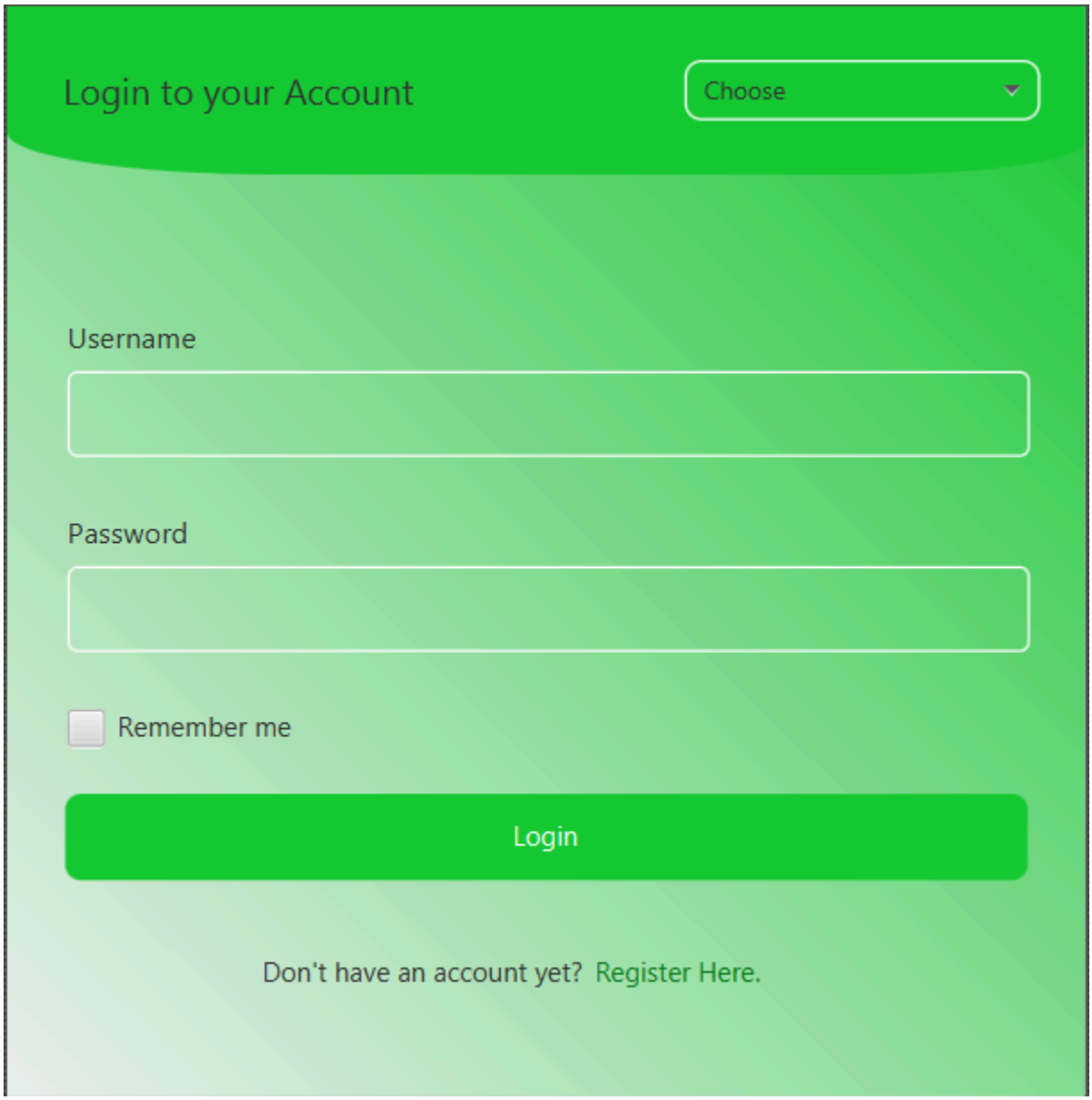
III. User Interface Design

Mockup

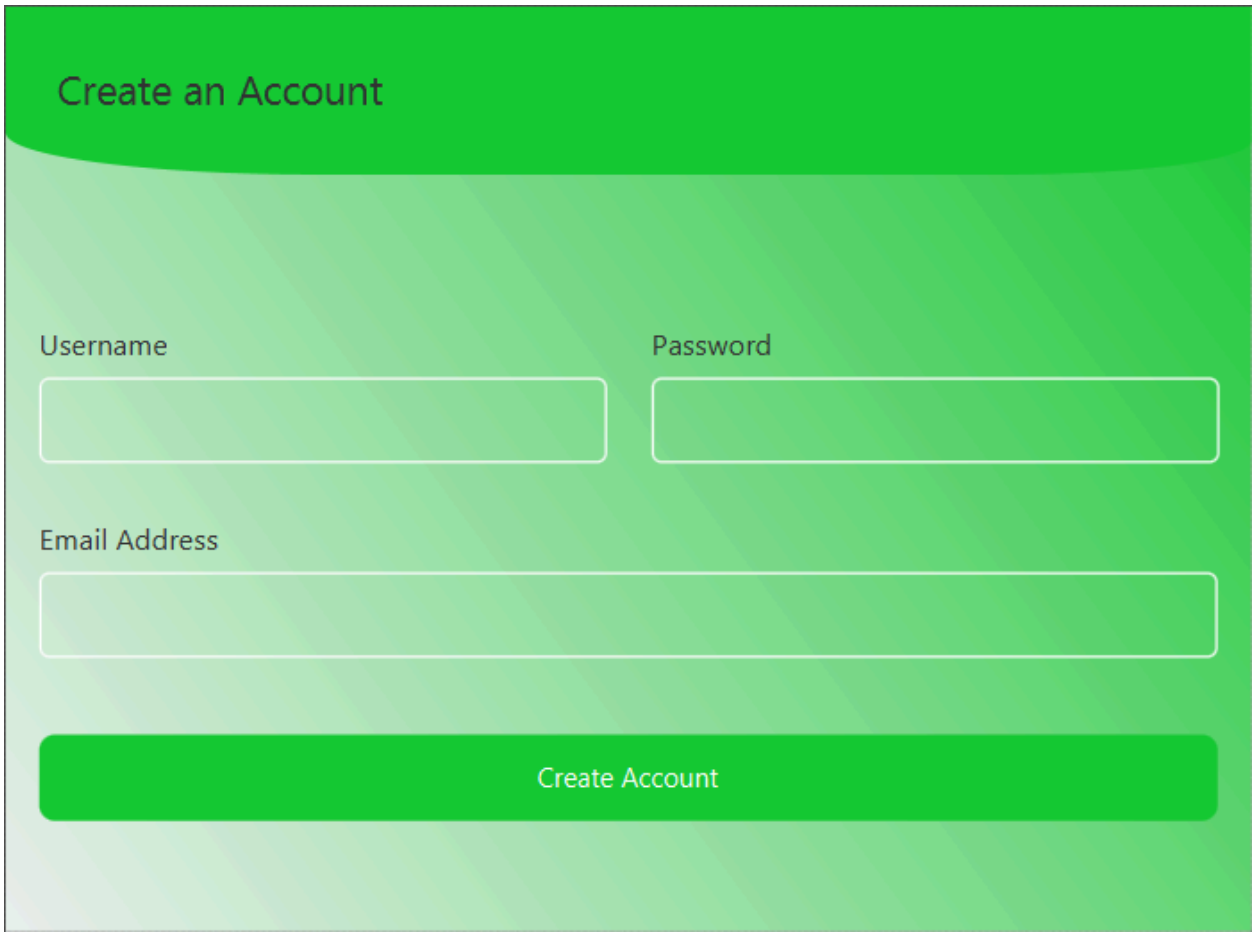
1. Start



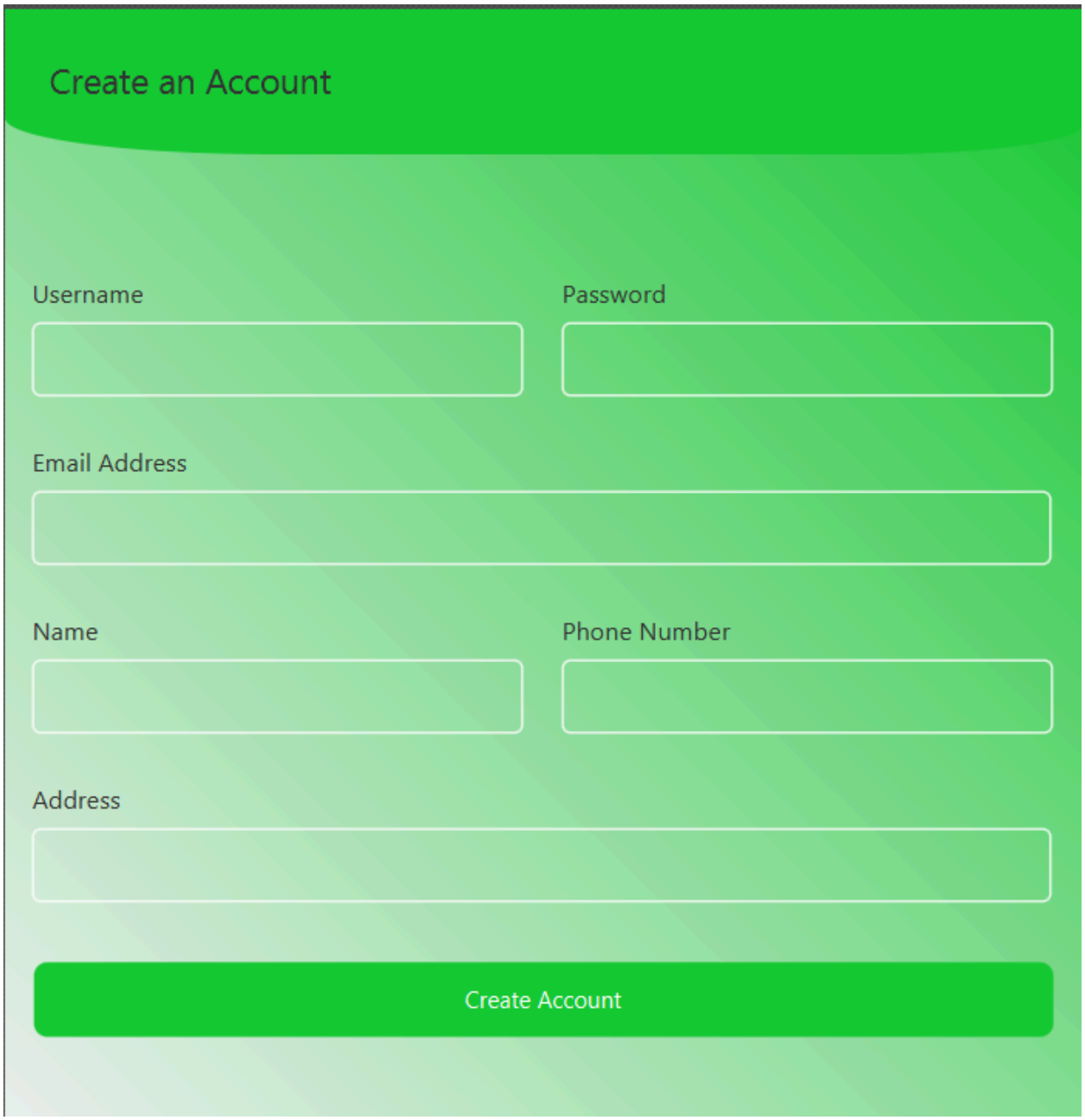
2. Login



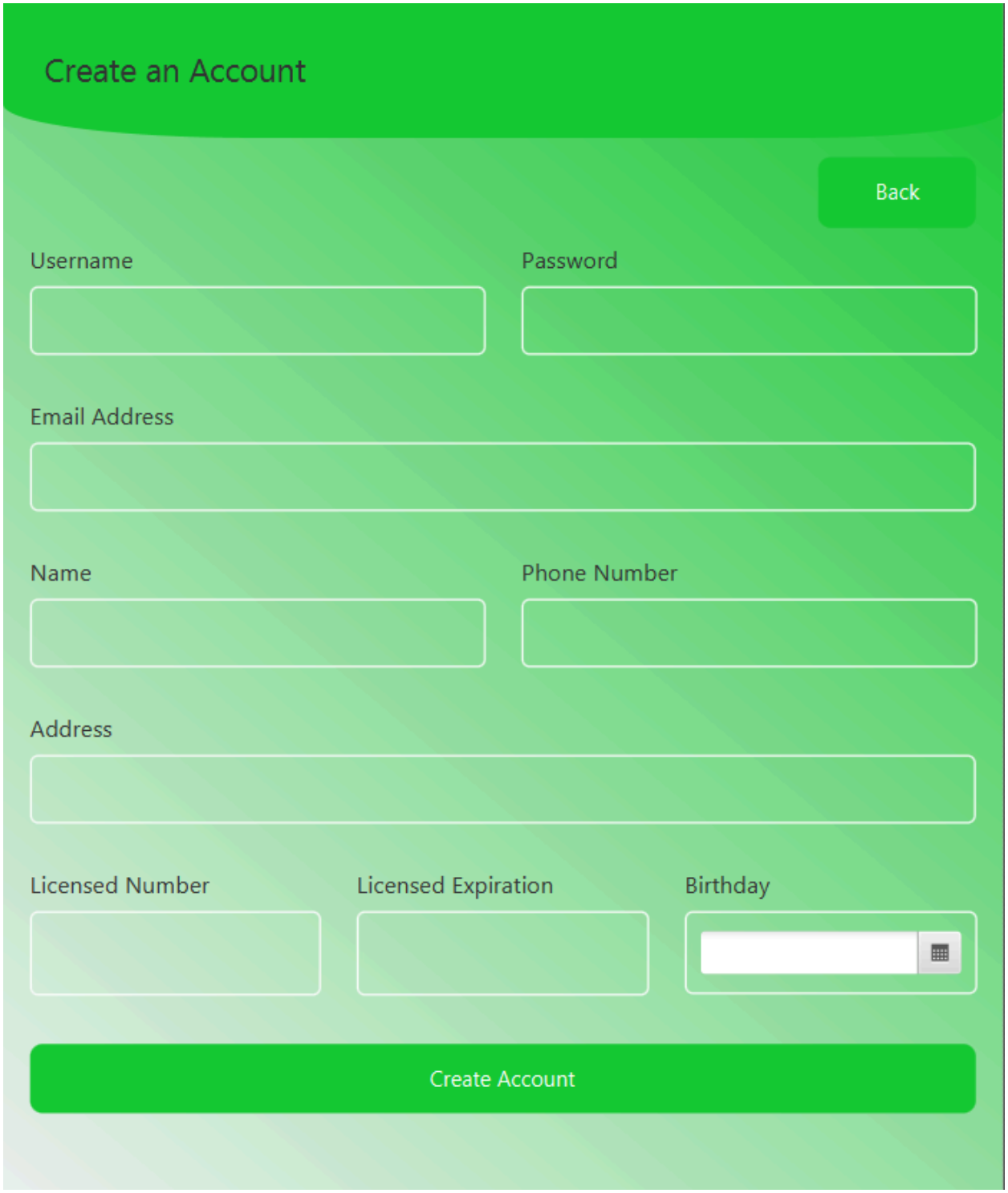
3. Admin Registration



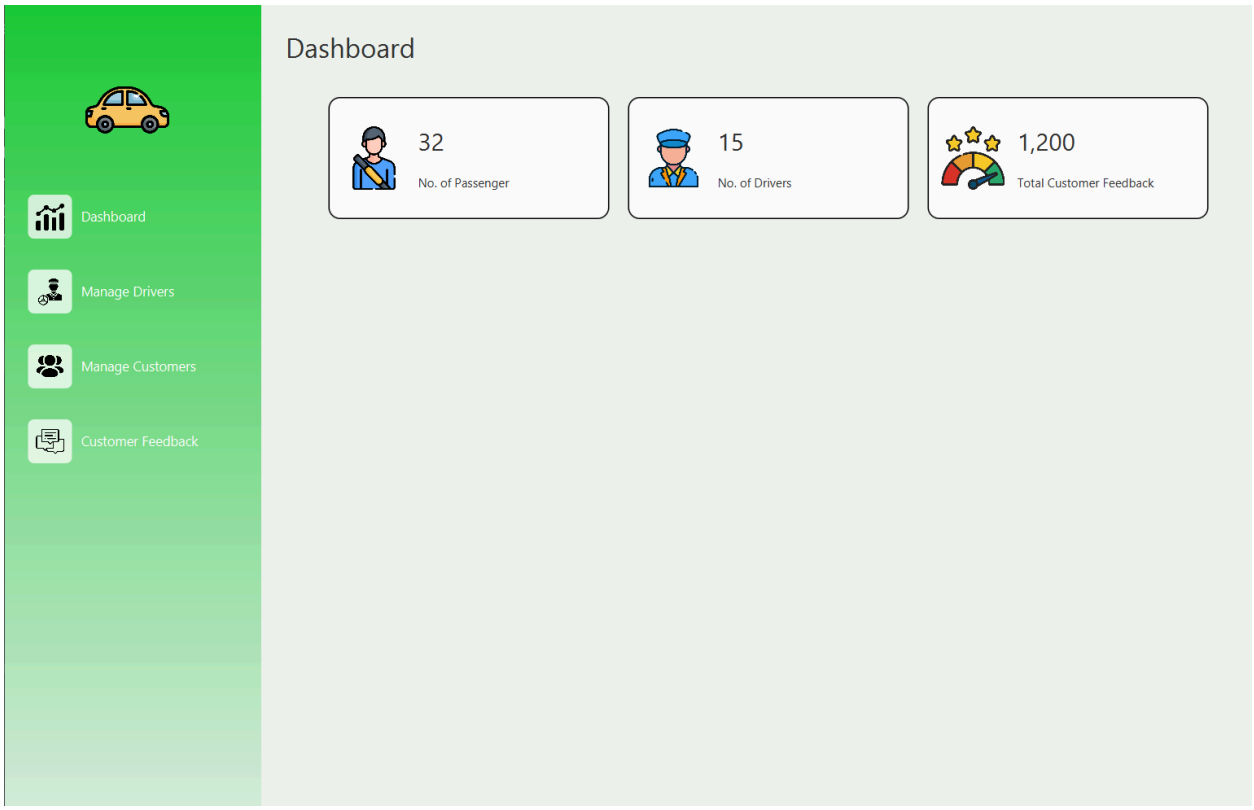
4. Customer Registration



5. Driver Registration


Create an Account
Back
Username
Password
Email Address
Name
Phone Number
Address
Licensed Number
Licensed Expiration
Birthday
Create Account

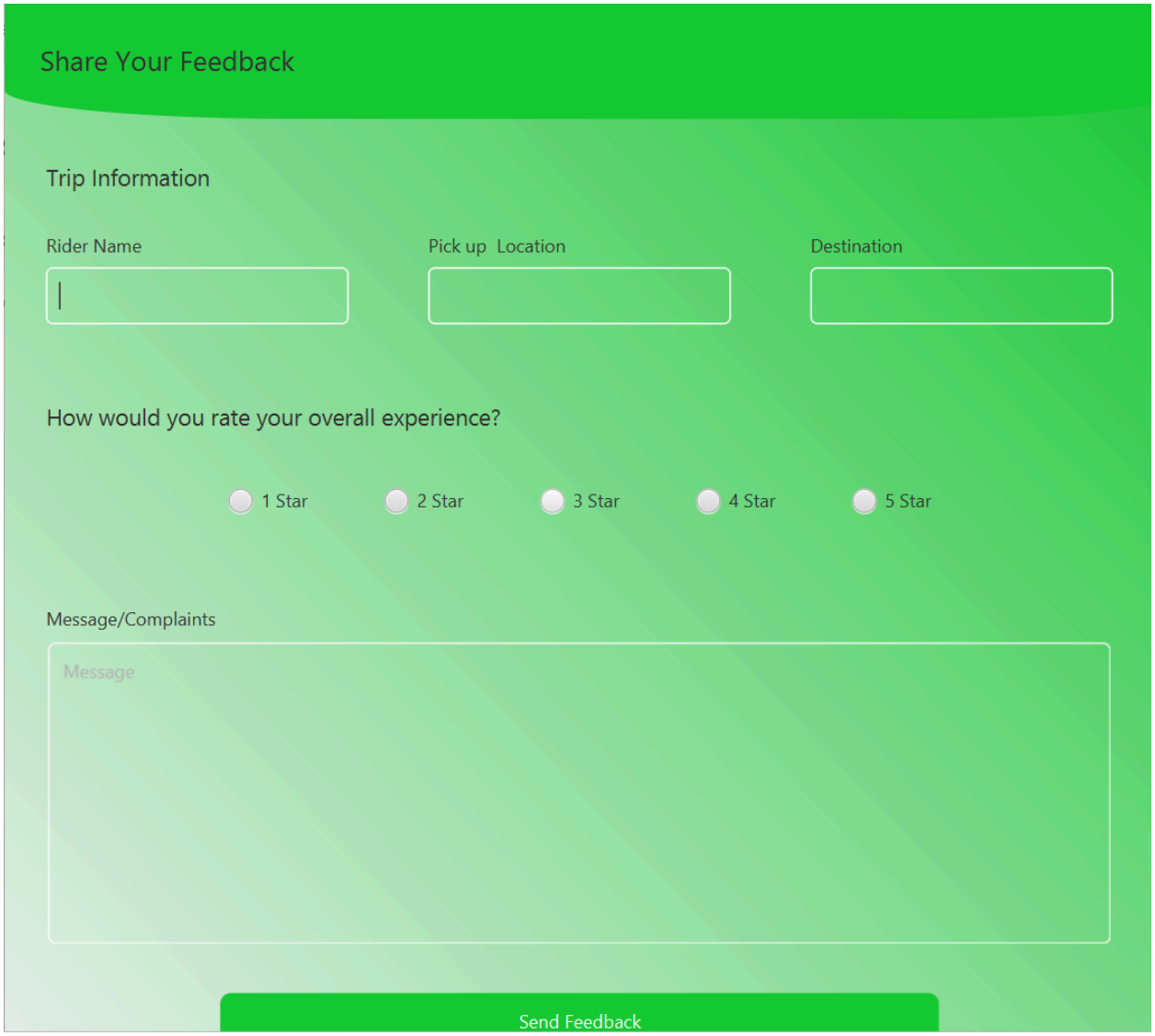
6. Main



7. Customer Log in


Grab (Passenger)
Grab
Search...

8. Customer Feedback



9. Driver Log in


Grab (Driver)
Grab
Search...

IV. Implementation

Database Creation

1. Account

```
CREATE TABLE `account` (
  `acc_id` int NOT NULL,
  `acc_username` varchar(50) NOT NULL,
  `acc_pass` varchar(255) NOT NULL,
  `acc_role` varchar(20) NOT NULL,
  `acc_email` varchar(100) NOT NULL,
  PRIMARY KEY (`acc_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

2. Customer

```
CREATE TABLE `customer` (
  `cus_id` int NOT NULL,
  `cus_name` varchar(100) NOT NULL,
  `cus_email` varchar(100) NOT NULL,
  `cus_phonenum` varchar(15) NOT NULL,
  `cus_address` varchar(255) NOT NULL,
  PRIMARY KEY (`cus_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

3. Driver

```
CREATE TABLE `driver` (
  `driver_id` int NOT NULL,
  `driver_name` varchar(10) NOT NULL,
  `driver_lice_num` varchar(20) NOT NULL,
  `driver_address` varchar(255) NOT NULL,
  `driver_email` varchar(100) NOT NULL,
  `driver_phone_num` varchar(15) NOT NULL,
  `driver_birth_date` date NOT NULL,
  `driver_lice_exdate` date NOT NULL,
  PRIMARY KEY (`driver_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

4. Vehicle

```
CREATE TABLE `vehicle` (
  `vehicle_id` int NOT NULL,
  `driver_id` int NOT NULL,
  `lice_plate` varchar(20) NOT NULL,
  `model` varchar(100) NOT NULL,
  `year` date NOT NULL,
  `color` varchar(50) NOT NULL,
  `vin` varchar(50) NOT NULL,
  PRIMARY KEY (`vehicle_id`),
  KEY `driver_id_idx` (`driver_id`),
  CONSTRAINT `driver_id` FOREIGN KEY (`driver_id`) REFERENCES `driver` (`driver_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

5. Customer Report

```
CREATE TABLE `customer_report` (
  `customer_reportid` int NOT NULL,
  `cust_rep_driver` int NOT NULL,
  `cust_rep_customer` int NOT NULL,
  `cust_rep_vehicle` int NOT NULL,
  `date` date NOT NULL,
  `time` varchar(45) NOT NULL,
  `location` varchar(255) NOT NULL,
  `decsription` varchar(255) NOT NULL,
  PRIMARY KEY (`customer_reportid`),
  KEY `cust_rep_vehicle_idx` (`cust_rep_vehicle`),
  KEY `cust_rep_customer_idx` (`cust_rep_customer`),
  KEY `cust_rep_driver_idx` (`cust_rep_driver`),
  CONSTRAINT `cust_rep_customer` FOREIGN KEY (`cust_rep_customer`) REFERENCES `customer` (`cus_id`),
  CONSTRAINT `cust_rep_driver` FOREIGN KEY (`cust_rep_driver`) REFERENCES `driver` (`driver_id`),
  CONSTRAINT `cust_rep_vehicle` FOREIGN KEY (`cust_rep_vehicle`) REFERENCES `vehicle` (`vehicle_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

6. Driver Report

```
CREATE TABLE `driver_report` (
  `driver_reportID` int NOT NULL,
  `driver_rep_driver` int NOT NULL,
  `driver_rep_customer` int NOT NULL,
  `driver_report_vehicle` int NOT NULL,
  `date` date NOT NULL,
  `time` varchar(45) NOT NULL,
  `location` varchar(255) NOT NULL,
  `decsription` varchar(255) NOT NULL,
  PRIMARY KEY (`driver_reportID`),
  KEY `driver_rep_vehicle_idx` (`driver_report_vehicle`),
  KEY `driver_rep_customer_idx` (`driver_rep_customer`),
  KEY `driver_rep_driver_idx` (`driver_rep_driver`),
  CONSTRAINT `driver_rep_customer` FOREIGN KEY (`driver_rep_customer`) REFERENCES `customer` (`cus_id`),
  CONSTRAINT `driver_rep_driver` FOREIGN KEY (`driver_rep_driver`) REFERENCES `driver` (`driver_id`),
  CONSTRAINT `driver_rep_vehicle` FOREIGN KEY (`driver_report_vehicle`) REFERENCES `vehicle` (`vehicle_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

7. Reference Number

```
CREATE TABLE `reference_number` (
  `reference_numberID` int NOT NULL,
  `ref_num_driver` int NOT NULL,
  `ref_num_customer` int NOT NULL,
  `ref_num_vehicle` int NOT NULL,
  `date` date NOT NULL,
  `time` varchar(45) NOT NULL,
  `location` varchar(255) NOT NULL,
  `decsription` varchar(255) NOT NULL,
  PRIMARY KEY (`reference_numberID`),
  KEY `ref_num_vehicle_idx` (`ref_num_vehicle`),
  KEY `ref_num_customer_idx` (`ref_num_customer`),
  KEY `ref_num_driver_idx` (`ref_num_driver`),
  CONSTRAINT `ref_num_customer` FOREIGN KEY (`ref_num_customer`) REFERENCES `customer` (`cus_id`),
  CONSTRAINT `ref_num_driver` FOREIGN KEY (`ref_num_driver`) REFERENCES `driver` (`driver_id`),
  CONSTRAINT `ref_num_vehicle` FOREIGN KEY (`ref_num_vehicle`) REFERENCES `vehicle` (`vehicle_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

8. Notification

```
CREATE TABLE `notification` (
  `notif_id` int NOT NULL,
  `notif_cust_id` int NOT NULL,
  `notif_driver_id` int NOT NULL,
  `date_sent` date NOT NULL,
  `time_sent` varchar(45) NOT NULL,
  `email_sub` varchar(255) NOT NULL,
  `email_body` varchar(45) NOT NULL,
  `email_sender` varchar(100) NOT NULL,
  `email_receiver` varchar(45) NOT NULL,
  PRIMARY KEY (`notif_id`),
  KEY `cust_id_idx` (`notif_cust_id`),
  KEY `driv_id_idx` (`notif_driver_id`),
  CONSTRAINT `notif_cust_id` FOREIGN KEY (`notif_cust_id`) REFERENCES `customer` (`cus_id`),
  CONSTRAINT `notif_driver_id` FOREIGN KEY (`notif_driver_id`) REFERENCES `driver` (`driver_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Data Insertion

Function to add Customer

```
1 usage
public void addCustomer(String acc_id, String cus_id, String cus_name, String cus_email, String cus_phonenum, String cus_address) throws SQLException {
    String sql = "insert into customer (acc_id, cus_id, cus_name, cus_email, cus_phonenum, cus_address) values (?, ?, ?, ?, ?, ?)";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setString( parameterIndex: 1, acc_id);
    preparedStatement.setString( parameterIndex: 2, cus_id);
    preparedStatement.setString( parameterIndex: 3, cus_name);
    preparedStatement.setString( parameterIndex: 4, cus_email);
    preparedStatement.setString( parameterIndex: 5, cus_phonenum);
    preparedStatement.setString( parameterIndex: 6, cus_address);
```

Function to add CustomerReport

```
1 usage
public void addCustomerReport(String customer_reportid, String cust_rep_driver, String cust_rep_customer, String cust_rep_vehicle, Date date, Time time, String description
    String sql = "insert into customer_report (customer_reportid, cust_rep_driver, cust_rep_customer, cust_rep_vehicle, date, time, decsription, reportType, trip_ref_num)
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setString( parameterIndex: 1, customer_reportid);
    preparedStatement.setString( parameterIndex: 2, cust_rep_driver);
    preparedStatement.setString( parameterIndex: 3, cust_rep_customer);
    preparedStatement.setString( parameterIndex: 4, cust_rep_vehicle);
    preparedStatement.setDate( parameterIndex: 5, date);
    preparedStatement.setTime( parameterIndex: 6, time);
    preparedStatement.setString( parameterIndex: 7, description);
    preparedStatement.setString( parameterIndex: 8, reportType);
    preparedStatement.setString( parameterIndex: 9, trip_ref_num);

    preparedStatement.executeUpdate();
}
```

Application Development

1. Planning and Requirements Analysis

- **Objectives: Define the purpose, scope, and goals of the database system.**
- **Tasks:**
    - **Identify stakeholders and gather requirements.**
    - **Create a requirements specification document.**
    - **Determine the system's functional and non-functional requirements.**

2. Conceptual Design

- **Objectives: Develop a high-level model of the database system.**
- **Tasks:**
    - **Create an Entity-Relationship Diagram (ERD) to visualize data relationships.**
    - **Define entities, attributes, and relationships.**
    - **Identify primary keys and foreign keys.**

3. Logical Design

- **Objectives: Translate the conceptual design into a logical data model.**
- **Tasks:**
    - **Normalize the data to reduce redundancy and improve integrity.**
    - **Define the schema and database tables.**
    - **Develop data integrity constraints (e.g., unique, not null, referential integrity).**

4. Physical Design

- **Objectives: Convert the logical design into a physical database.**
- **Tasks:**
    - **Choose the database management system**
    - **Design the physical storage structures (tablespaces, files).**

5. Development

- **Objectives: Build the database and the application interfaces.**
- **Tasks:**
    - **Write SQL scripts for database creation and data manipulation.**
    - **Develop stored procedures, functions, triggers, and views.**
    - **Create application logic for interacting with the database**
    - **Design and develop the user interface (UI) if applicable.**
    - **Implement transaction management and error handling.**

6. Testing

- **Objectives: Ensure the database system functions correctly and meets requirements.**
- **Tasks:**
    - **Perform unit testing on individual components.**
    - **Conduct integration testing to verify that components work together.**

V. Conclusion

Summary of Work Done

Four weeks of work were put into the project and paper's research and improvement. Included in the research are the following:

- Research of tools and technologies
- Creation of database
- Design using SceneBuilder
- Application Development using IntelliJ, and JavaFX
- Testing

Challenges Faced

The challenges that were faced while creating this project have numerous factors. The researchers had a hard time choosing what tools were going to be used for this project, limited time for testing everything that would be able to be put in the project, and advantages and disadvantages of different researchers. The project was also created during the end of the term and unfortunately got bumped into the other projects from different courses which led to splitting the attention, and time of the researchers. Through the challenges that the researchers faced, the project was complete.

Lesson Learned

The researchers encountered a lot of problems in the making of the project and because of those problems, the researchers learned a lot. The researchers learned techniques on what a more understandable database system could be. Above all, researchers also learned about the essence of time management and planning. A lot can happen with just a manner of 1 day of progress with a single project, and planning on what will be the next step in the project.

Future Enhancements

The researchers suggest adjustment for the back end. The back end in the project had a lot of problems with the data samples, causing the front end to be empty, also the dashboard of the project is another adjustment. The dashboard has a lot of space and no way of seeing if the feedback of the customers are good or bad feedback. Overall, there are no further enhancements needed except the data samples and back end of the project.