Insurance Agency Management System

Sanjay Krishnamurthi, Gursimran Kaur and Mohammad Zaza

Introduction to Database Management Systems

Aravind Rokkam

San Jose State University

Insurance Agency Management System

This final project involves creating an application which is designed to serve as a comprehensive management system for an insurance company or agency. It centers around developing a web-based application that supports the core operations of an insurance company. It provides tools for managing customer databases, accessing policy details, and interacting with various insurance carriers. This system is designed to facilitate efficient data retrieval and management, ensuring that insurance agents and administrators can quickly find and process the information they need.

Objective

Our aim is to build a solution that seamlessly integrates the essential functions of an insurance business, from client information management to policy tracking and carrier connections. The application must quickly and concisely handle the company's data and manage it, giving employees an intuitive management portal from which they can add and remove customers and policies.

High Level Design

The high-level design revolves around the combination of solid backend infrastructure and a seamless user interface. The core of the system is a MySQL database. This database stores all the essential information for the insurance company, including customer details, policy information and carrier connections. The database schema is designed to follow the principles of the Third Normal Form (3NF), ensuring that data is logically organized, free from redundancy and easily maintainable.

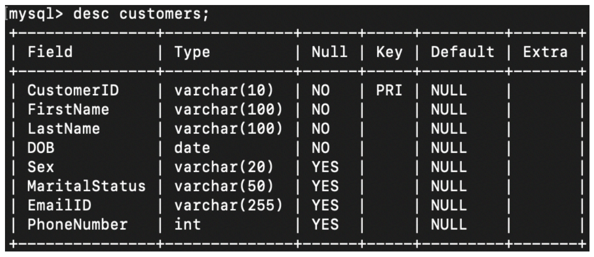
On the application side, Python is used as the primary programming language to interact with the database. Python's versatility makes it easy to create efficient queries to retrieve and manipulate data. The backend handles user requests, performs database operations, and sends the appropriate data back to the user interface.

The user interface is designed to be intuitive and user-friendly, providing a seamless experience for insurance agents and administrators. It allows users to retrieve and update customer and policy information quickly, and manage data with ease. To ensure security and data integrity, the system incorporates user authentication and authorization, ensuring that only authorized personnel can access sensitive information through an authentication portal. Overall, this high-level design creates a reliable and efficient platform for managing an insurance company's operations.

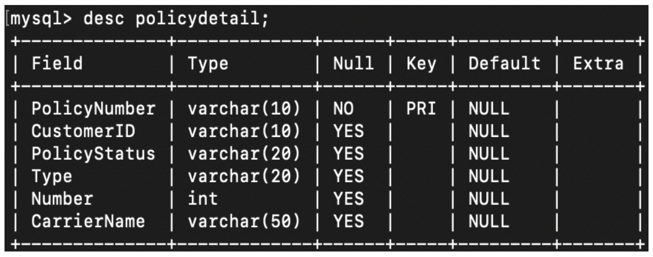
Database Design

The system features a database structure with five primary tables, each serving a specific function within the system. The cornerstone is the Customers table, which holds essential customer details, such as customer ID, name, address, phone number, and email address. Next is the PolicyDetail table, designed to store key information about insurance policies. This includes policy numbers, policy types (such as auto, home, or umbrella insurance), and references to related customers, carriers, and policy statuses. The Carriers table complements this structure by storing information about insurance carriers, including carrier ID, name, and mailing location. The remaining two tables, PolicyStatus and Premiums, add further functionality. The PolicyStatus table tracks the current status of each policy, such as "Active," "Pending," or

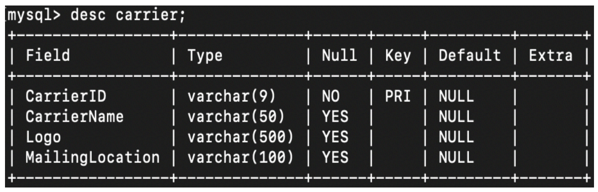
"Canceled." This allows the system to monitor and report on policy states effectively. The Premiums table stores premium values and payment dates, linking them to specific customers for accurate billing and financial tracking. This database design, with its interconnected structure, provides a robust and flexible foundation for our insurance management system, enabling efficient operations and scalable growth. The tables as well as the ER Diagram are shown below.



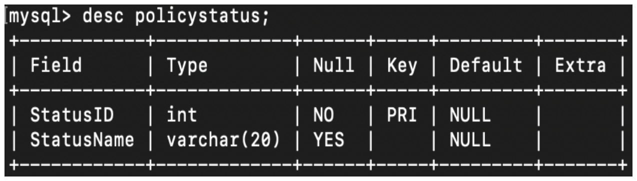
```
[mysql> desc customers;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| CustomerID    | varchar(10)  | NO   | PRI | NULL    |       |
| FirstName     | varchar(100) | NO   |     | NULL    |       |
| LastName      | varchar(100) | NO   |     | NULL    |       |
| DOB           | date         | NO   |     | NULL    |       |
| Sex           | varchar(20)  | YES  |     | NULL    |       |
| MaritalStatus | varchar(50)  | YES  |     | NULL    |       |
| EmailID       | varchar(255) | YES  |     | NULL    |       |
| PhoneNumber   | int          | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```



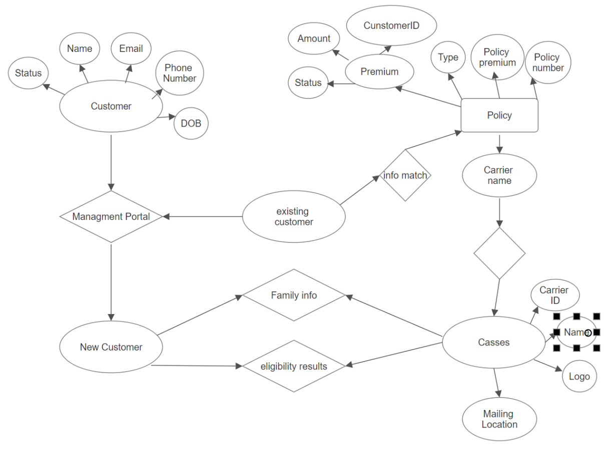
```
[mysql> desc policydetail;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| PolicyNumber | varchar(10) | NO   | PRI | NULL    |       |
| CustomerID   | varchar(10) | YES  |     | NULL    |       |
| PolicyStatus | varchar(20) | YES  |     | NULL    |       |
| Type         | varchar(20) | YES  |     | NULL    |       |
| Number       | int         | YES  |     | NULL    |       |
| CarrierName  | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

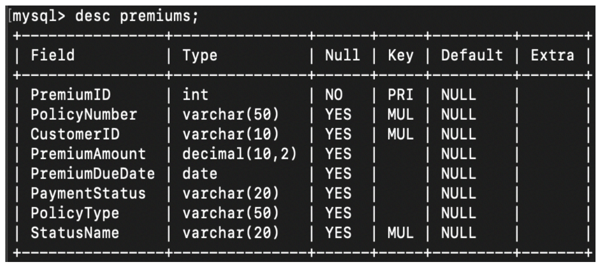


```
mysql> desc carrier;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| CarrierID       | varchar(9)   | NO   | PRI | NULL    |       |
| CarrierName     | varchar(50)  | YES  |     | NULL    |       |
| Logo            | varchar(500) | YES  |     | NULL    |       |
| MailingLocation | varchar(100) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
```



```
[mysql> desc policystatus;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| StatusID   | int         | NO   | PRI | NULL    |       |
| StatusName | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```





```
[mysql> desc premiums;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| PremiumID      | int           | NO   | PRI | NULL    |       |
| PolicyNumber   | varchar(50)   | YES  | MUL | NULL    |       |
| CustomerID     | varchar(10)   | YES  | MUL | NULL    |       |
| PremiumAmount  | decimal(10,2) | YES  |     | NULL    |       |
| PremiumDueDate | date          | YES  |     | NULL    |       |
| PaymentStatus  | varchar(20)   | YES  |     | NULL    |       |
| PolicyType     | varchar(50)   | YES  |     | NULL    |       |
| StatusName     | varchar(20)   | YES  | MUL | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
```

Normalization

Normalization is a critical process in database design that aims to minimize data redundancy and maintain data integrity by organizing the database into related tables. In the Insurance Company

Management System, the normalization principles have been applied to ensure that each table has a distinct purpose and unique set of responsibilities, following the rules of the Third Normal Form (3NF). Normalization also facilitates easier querying and maintenance. With a normalized structure, queries can be more precise and efficient, reducing the computational load on the system.

To meet the requirements of 1NF, the database tables are designed to ensure atomicity—each cell holds a single value. This prevents ambiguity and data duplication. For example, in the Customers table, there's a unique identifier (customer ID) that serves as the primary key, and each column contains only one piece of information, such as the customer's name, address, or phone number. Similarly, the PolicyDetail table has unique policy numbers, ensuring there's no duplicated data. By adhering to these rules, we maintain a clean, well-organized table structure with no repeated rows or columns.

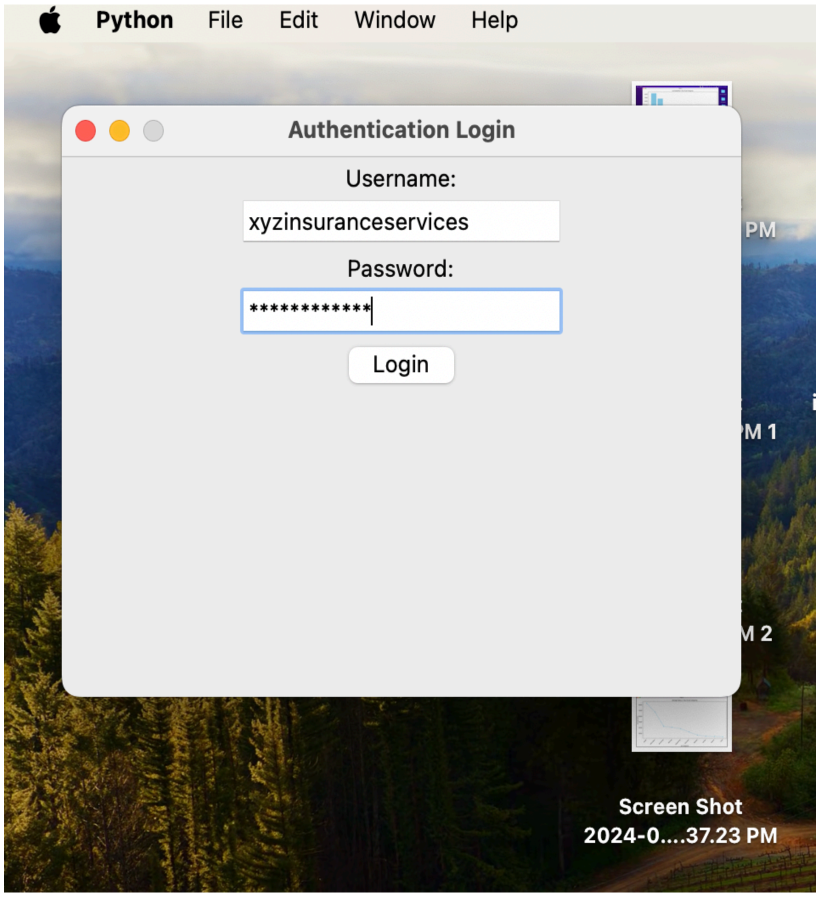
A database in 2NF must first comply with 1NF and then ensure that all non-key attributes are fully dependent on the primary key, eliminating partial dependencies. In the database, each table has a primary key, and all non-key attributes depend entirely on it. For example, in the PolicyDetail table, the policy number is the primary key, and all other information—such as the policy type, effective date, and customer ID—relates directly to this key. This structure avoids partial dependencies, where a non-key attribute might rely only on part of a composite primary key.

To achieve 3NF, a database must be in 2NF and have no transitive dependencies. A transitive dependency occurs when a non-key attribute depends on another non-key attribute instead of the primary key. Our design eliminates these dependencies by ensuring that all attributes in each table are either primary keys or directly dependent on them. For example, the Premiums table has a primary key (a unique premium ID) with attributes like the premium amount and payment date. These attributes depend solely on the primary key, with no transitive relationships.

Results

In the end, we were able to successfully create the authentication portal as well as the application which can be used by the employees of an insurance agency. The authentication portal inputs the username and password and allows the user into the system if it’s correct. Once they enter the system, they are able to access the features, including a search bar where a customer’s information can be looked up quickly. It also contains many buttons to carry out the functions which an employee may need to use. These include buttons to add and remove customers, buttons to show their policy and premium statuses, buttons to show and issue policies, as well as a button to display the available carriers. There are also buttons which filter out and display both the paid and unpaid customers and their details. This allows the agency to quickly retrieve a list of all unpaid customers so that they can send mass reminder emails or text messages without having to go through each customer individually. Overall, the system allows the user to carry out their jobs easily and without difficulty.

Screenshots

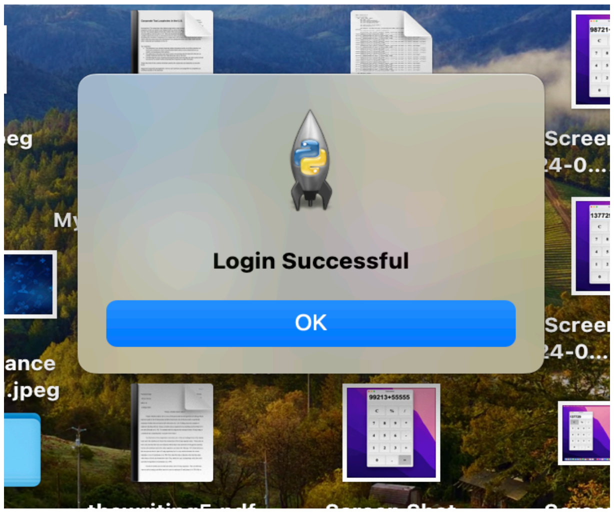


Username:

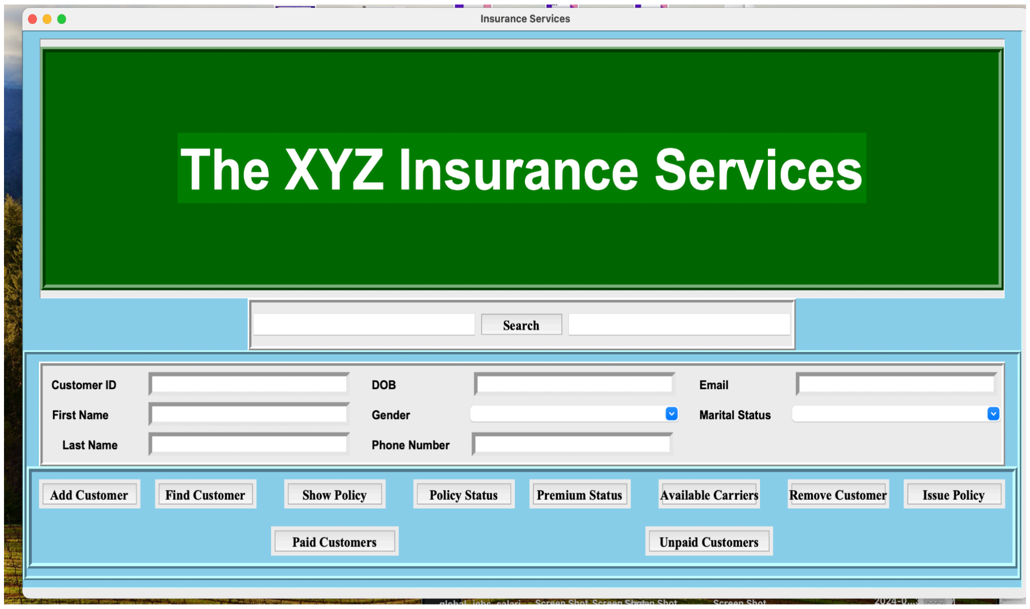
xyzinsuranceservices

Password:

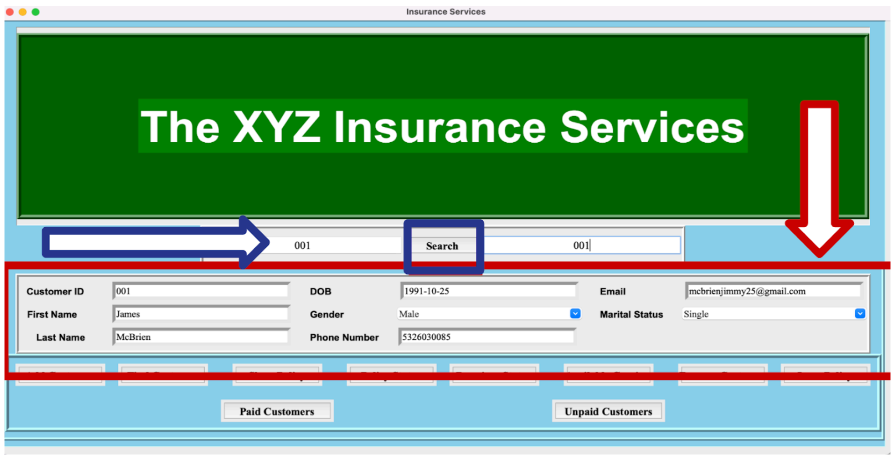
****



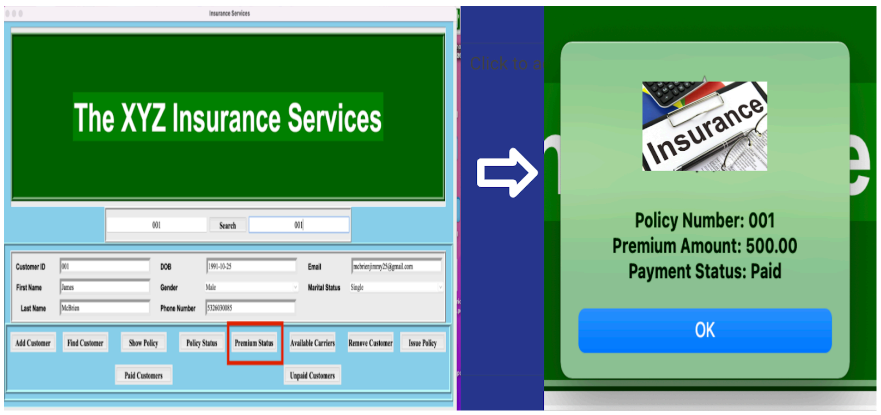
Login Successful



Customer ID		DOB		Email	
First Name		Gender		Marital Status	
Last Name		Phone Number			



Customer ID	001	DOB	1991-10-25	Email	mcbrienjimmy25@gmail.com
First Name	James	Gender	Male	Marital Status	Single
Last Name	McBrien	Phone Number	5326030085		



Customer ID	001	DOB	1991-10-25	Email	mcbrienjimmy25@gmail.com
First Name	James	Gender	Male	Marital Status	Single
Last Name	McBrien	Phone Number	5326030085		

Policy Number: 001
Premium Amount: 500.00
Payment Status: Paid

Contributions

All of the team members worked together to come up with the idea as well as creating the mySQL database by making the individual tables and filling them with values and figuring out how to connect them using the ER diagram. The remaining individual contributions are listed out below.

Sanjay: In addition to working on the database and the backend, also worked on preparing the presentation and creating the project report.

Gursimran: Worked on the frontend and backend, presentation.

Mohammad: working on the database and the backend, helped with the presentation slides.

References/Sources

https://youtu.be/DUNn0YAIh1U?feature=shared

https://youtu.be/X9reTI_Mckk?feature=shared

https://youtu.be/-fWXAN9xcVg?feature=shared

https://www.geeksforgeeks.org/

https://stackoverflow.com/