**Project Report: ATM Application**

**1. PRE-BUILT DATABASE: banking\_atm**

**Overview**

The banking\_atm database is designed to support the ATM application's operations by storing account-related information. It provides the necessary backend structure to manage transactions and account details.

**Database Structure**

The database consists of a table named details, which holds essential information about each account, including account number, account holder's name, date of birth, balance, city, and mobile number.

**2. TABLE: details**

**Schema Description**

* **Account\_Number**: A unique number assigned to each account, serving as the primary key for identifying accounts.
* **Name**: The account holder's full name.
* **Date\_of\_Birth**: The account holder's date of birth.
* **Balance**: The amount of money currently in the account.
* **City**: The city where the account holder resides.
* **Mobile\_Number**: The contact phone number of the account holder.

**3. PROCEDURE TO SET UP DATABASE CONNECTION**

To connect the ATM application with the banking\_atm database:

1. **Install Required Library**: Ensure that the Python library pymysql is installed. This library is used to facilitate communication between the Python application and the MySQL database.
2. **Establish Connection**: The application establishes a connection to the database using connection parameters like the host (server address), user credentials, and the database name. This connection allows the application to execute SQL queries for operations like retrieving and updating account information.

**4. EXPLANATION OF EACH WINDOW PAGE**

**Main Menu**

**Purpose**: Acts as the central control panel for the ATM application. It provides users with access to various functions, including withdrawing money, depositing funds, changing pin, checking their balance, and viewing account details.

**Features**:

* **Withdraw**: Initiates the withdrawal process.
* **Deposit**: Allows users to deposit money into their account.
* **Balance Enquiry**: Enables users to check their current account balance.
* **PIN Change**: Intended for changing the user’s PIN.
* **Account Details**: Provides access to detailed information about the user's account.

**Withdraw**

**Purpose**: Facilitates the withdrawal of funds from the user's account.

**Features**:

* **Input Field**: Users specify the amount they wish to withdraw.
* **Proceed Button**: Processes the withdrawal request and updates the account balance accordingly.

**Deposit**

**Purpose**: Enables users to deposit money into their account.

**Features**:

* **Input Field**: Users enter the amount they want to deposit.
* **Proceed Button**: Processes the deposit and updates the account balance.

**Balance Enquiry**

**Purpose**: Allows users to check their current account balance.

**Features**:

* **Check Balance Button**: Retrieves and displays the current balance of the user’s account.

**Account Details**

**Purpose**: To display detailed information about the account holder’s account based on their account number.

**Features**:

* **Input Field**: Users enter their account number to retrieve specific account details.
* **View Details Button**: Fetches and displays the account details such as the account holder’s name, balance, and contact information.

**5. OBSERVATION**

The ATM application offers a user-friendly interface for performing basic banking operations. It successfully integrates graphical user interface elements with backend database operations, providing an intuitive experience for users. The application handles real-time transactions and updates efficiently, although some advanced features (like PIN change) implemented.

**6. ABSTRACTION**

**User Perspective**

For users, the ATM application abstracts the complexities of database operations and connectivity. They interact with a straightforward graphical interface to perform transactions without needing to understand the underlying technical details.

**Technical Perspective**

From a developer’s viewpoint, the application abstracts database interactions into simple functions. The frontend (user interface) communicates with the backend (database) through these functions, simplifying the process of managing user inputs and updating account information.

**7. DESCRIPTION**

The ATM application simulates the functionalities of an automated teller machine. It allows users to perform transactions such as withdrawing and depositing money, checking their account balance, and viewing detailed account information, changing pin. The application uses a graphical interface created with Tkinter and a MySQL database for data management, offering a realistic simulation of ATM operations.

**8. SUMMARY**

The ATM application is a functional simulation of basic banking operations, providing a user-friendly interface for handling transactions and managing account details. It integrates Tkinter for the GUI and MySQL for data storage, offering an effective tool for simulating ATM functionality. Future improvements could include additional features and enhanced security measures.

**9. ARCHITECTURE**

**Components**

* **Frontend**: Built using Tkinter, which provides the graphical user interface for user interactions.
* **Backend**: Python code that manages database interactions and business logic.
* **Database**: MySQL database banking\_atm, which stores account information and transaction records.

**Workflow**

1. **User Interaction**: Users interact with the Tkinter-based interface to initiate transactions.
2. **Data Handling**: Python functions process user inputs, execute SQL queries, and manage data updates.
3. **Database Operations**: The MySQL database handles storage and retrieval of account information.

**10. FEATURES**

* **Withdrawal**: Enables users to withdraw money from their account.
* **Deposit**: Allows users to deposit funds into their account.
* **Balance Enquiry**: Provides current balance information.
* **Account Details**: Displays detailed information about the account holder.
* **Change PIN**: Allows users to change their current pin.

**11. TOOLS AND SOFTWARE USED**

* **Programming Language**: Python
* **Libraries**:
  + **Tkinter**: For creating the graphical user interface.
  + **pymysql**: For connecting to and interacting with the MySQL database.
* **Database**: MySQL
* **IDE**: Python-compatible Integrated Development Environment (e.g., PyCharm, VSCode).

**12. CONCLUSION**

The ATM application demonstrates the essential functions of an automated teller machine through a well-designed graphical interface. By integrating Tkinter for the frontend and pymysql for database interactions, the application effectively manages account transactions and data. While the core functionalities are robust, further enhancements could improve security and add more advanced features.