A close-up of a logo

AI-generated content may be incorrect.

**School of Computer Science and Engineering**

**COMMERCE BANK PRO (Static Main)**

**BY**

**I. Introduction** CommerceBankpro is a comprehensive online banking platform designed to provide users with a seamless and secure banking experience. This architecture document aims to delineate the high-level hierarchy, components, and interactions within the system.

**II. System Overview** The architecture of CommerceBankpro comprises several interconnected modules, each serving distinct functionalities essential for a comprehensive online banking experience.

**III. High-Level Hierarchy**

1. **User Interface (UI) Layer:**
   * The UI layer encompasses the front-end components visible to users, including distinct pages such as:
     + **Register for the Account Page:** Allows users to create new banking accounts.
     + **Sign In Page:** Grants access to registered accounts and services.
     + **OpenBankAccount Page:** Facilitates the process of opening new accounts.
     + **UPI Payment Page:** Enables fund transfers between accounts using UPIIDs.
     + **Split Payment Feature:** Divides transaction amounts among multiple sources using UPIID.
     + **Dashboard:** Offers an overview of transactions and user profile details.
2. **Application Layer:**
   * Responsible for handling user requests and processing data, comprising:
     + **Account Management Module:** Manages user accounts, registration, and authentication.
     + **Transaction Module:** Handles fund transfers, payments, and transaction splitting.
     + **Data Processing Module:** Validates user inputs and interacts with the database layer.
3. **Business Logic Layer:**
   * Contains the core logic and rules governing the platform's functionalities, including:
     + **Security and Encryption:** Implements encryption protocols to secure user data.
     + **Validation Rules:** Ensures accuracy and integrity of user-provided information.
     + **Transaction Processing Logic:** Executes fund transfers and payment operations securely.
4. **Data Access Layer:**
   * Manages interaction with the database, including:
     + **MySQL Workbench Database:** Stores user information, transaction data, and account details securely.
     + **Data Access Objects (DAOs):** Handles database queries and operations.
5. **Testing and Quality Assurance Layer:**
   * Utilizes Selenium and Selenium IDE for automated testing and quality assurance:
     + **Automated Test Scripts:** Conducts rigorous testing of the website's functionalities.
     + **Quality Assurance Processes:** Ensures consistent performance and identifies potential issues.

**IV. Interactions Between Layers**

* The UI layer interacts with the Application Layer, sending user inputs and requests for account creation, sign-in, transactions, and dashboard viewing.
* The Application Layer processes these requests, validates data, and interacts with the Business Logic Layer for executing transactions securely.
* The Business Logic Layer ensures adherence to security protocols, validation rules, and transaction processing logic.
* The Data Access Layer handles interactions with the MySQL database, retrieving and storing user data securely.
* The Testing and Quality Assurance Layer continuously tests and verifies the functionalities to maintain system reliability.

**User Interaction Flow:**

1. **Account Registration Process:**
   * The process initiates when a user navigates to the "Register for the Account" page.
   * The user inputs necessary details like personal information, contact information, and creates login credentials.
   * The entered data undergoes validation checks to ensure accuracy and completeness.
   * Upon successful validation, the information is securely stored in the MySQL database, creating a new user account.
2. **User Authentication and Access:**
   * Users access the application through the "Sign In" page by providing their login credentials.
   * The system validates the user's credentials against the stored data in the database.
   * If the authentication is successful, the user gains access to the dashboard and other functionalities.
3. **Account Opening Process:**
   * Users navigate to the "OpenBankAccount" page to apply for a new banking account.
   * They input necessary details and information required for account creation.
   * The system validates the provided data and securely stores it in the MySQL database as a new account entry.
4. **Financial Transactions:**
   * Users utilize the "UPI Payment" page to transfer funds between accounts by inputting UPIIDs and transaction amounts.
   * The system verifies the UPIIDs and processes the transaction securely, updating account balances accordingly.
   * In the case of a "Split Payment," users provide UPIIDs for multiple sources, and the system distributes the transaction amount accordingly.
5. **Dashboard Interaction:**
   * The "Dashboard" collates and displays an overview of transactions and user profile details fetched from the database.
   * Users interact with the dashboard to view transaction history, manage accounts, and perform additional actions.

**Interactions and Dependencies:**

1. **Web Server-Client Interaction:**
   * The web server hosts the application and responds to user requests received through web browsers.
   * It interacts with the client-side scripts and interfaces to facilitate page navigation and data submission.
2. **Database Interaction:**
   * The MySQL Workbench database stores user accounts, transaction details, and other critical information.
   * The application interacts with the database to retrieve, update, and store user data securely.
3. **Page Interaction and Functionality:**
   * Each page (Register, Sign In, OpenBankAccount, UPI Payment, Split Payment, Dashboard) interacts with the server-side scripts and database for specific functionalities.
   * They communicate user inputs, validate data, and perform necessary actions while ensuring data integrity and security.
4. **Selenium Automated Testing:**
   * Selenium and Selenium IDE perform automated tests to ensure the functionality of different pages.
   * The testing process interacts with the web application, simulating user actions and validating expected behaviors.

**Concurrency and Performance:**

* The system ensures concurrency handling to accommodate multiple users accessing various pages simultaneously.
* It maintains high performance by handling multiple requests efficiently, ensuring optimal response times without compromising system stability.

**Scalability and Reliability:**

* CommerceBankpro's architecture is designed to be scalable, accommodating a growing user base without compromising performance.
* The system architecture ensures reliability by implementing robust security measures, backup protocols, and error handling mechanisms to mitigate system failures.