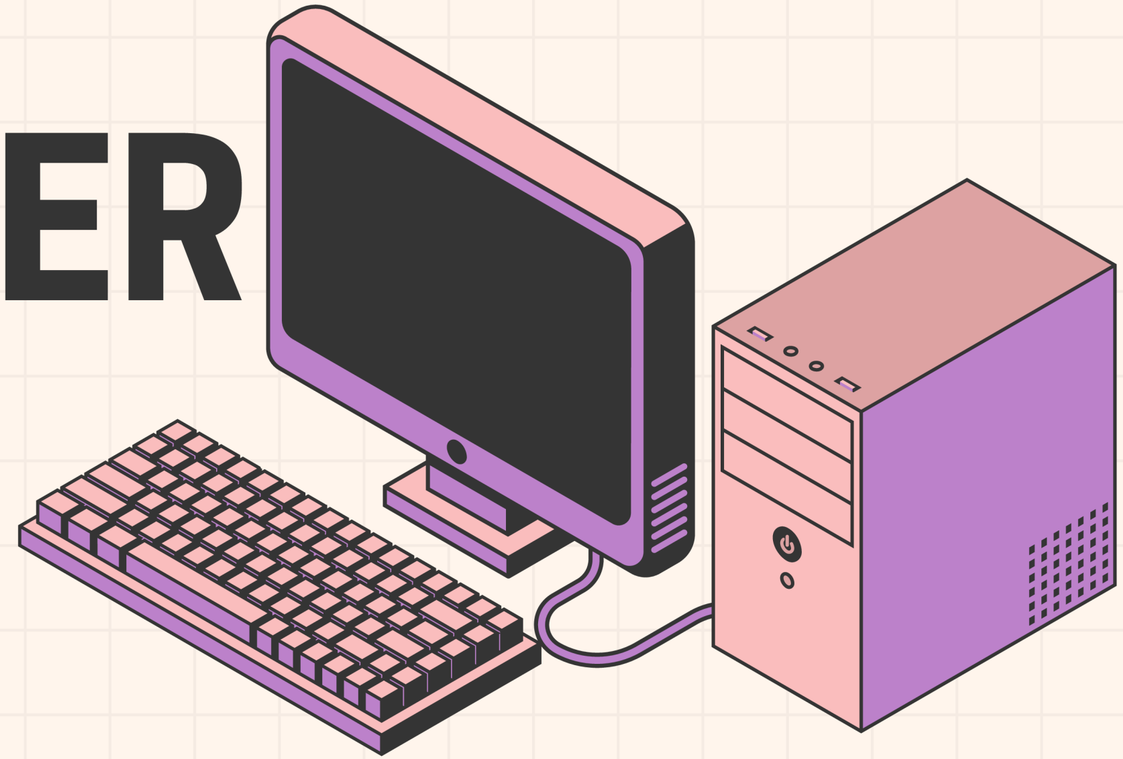
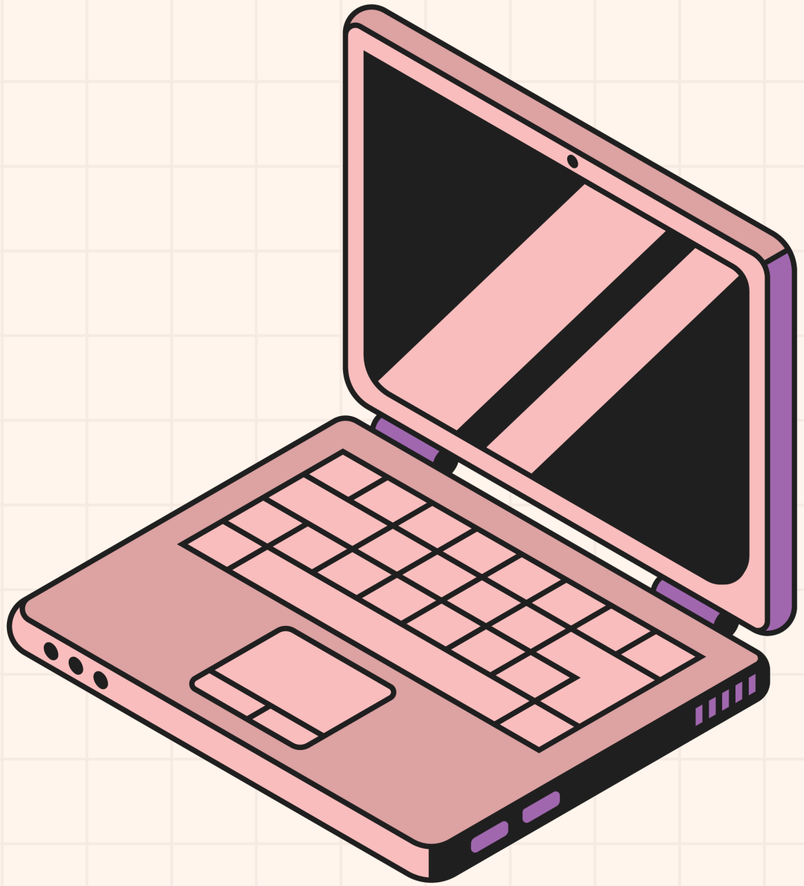
CLIENT SERVER
BANKING APPLICATION



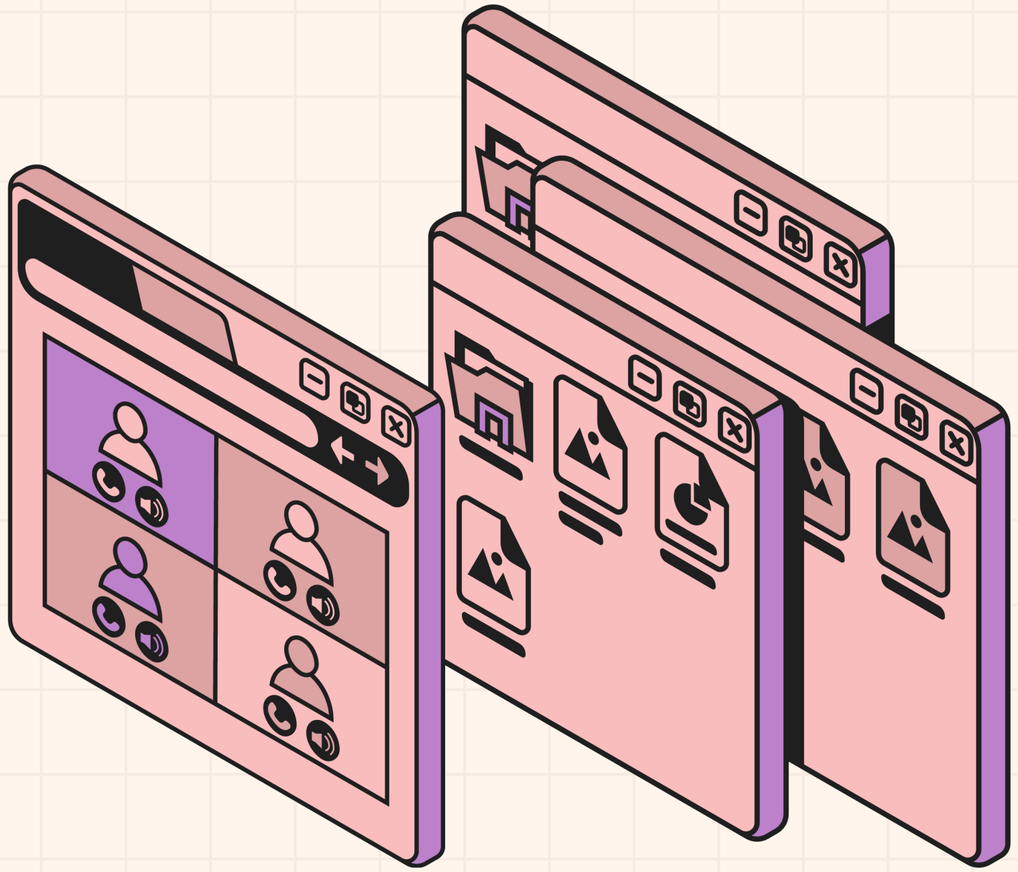
OVERVIEW

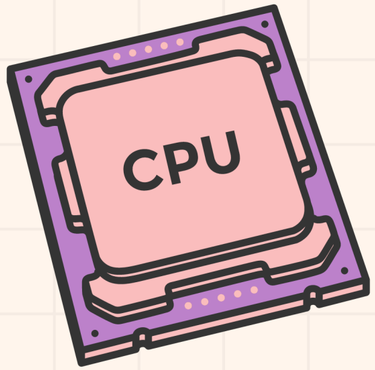
This project simulates a simplified online banking system using client-server architecture, which allows users to manage their bank accounts, perform transactions, and receive real-time notifications. This system leverages TCP for secure data transfer and UDP for fast, real-time notifications.

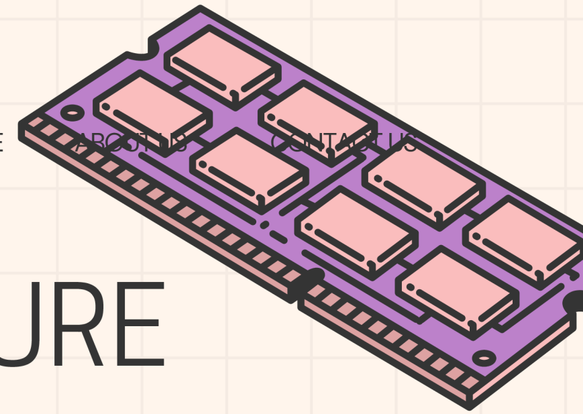


PROJECT REQUIREMENTS:

- Operating System: Ubuntu (for both server and client virtual machines).
- Networking Protocols: TCP for secure, reliable communication; UDP for faster, non-critical messaging.
- Programming Language: Python for socket programming.
- Environment: VirtualBox to simulate two machines (one client, one server).
- Security: Basic encryption for secure transactions.







CLIENT SERVER ARCHITECTURE

CLIENT

- The client side of your application connects to the server to perform banking operations.
- The client is responsible for:
  - Displaying menus for user interaction (e.g., sign up, login, view account details).
  - Sending requests to the server (e.g., creating an account, logging in, transferring funds).
  - Receiving responses from the server and updating the user interface accordingly.
  - Listening for real-time notifications via UDP (for fund transfers).

SERVER

- The server is a centralized component that handles all banking operations and data management. It processes client requests and responds with the appropriate information or actions.
- The server performs the following tasks:
  - Accepts incoming client connections and manages them in separate threads, allowing multiple users to interact with the server simultaneously.
  - Processes various banking operations such as user registration, login, fund transfers, balance checks, and account deletions.
  - Sends notifications to users via UDP, informing them of events like received funds.

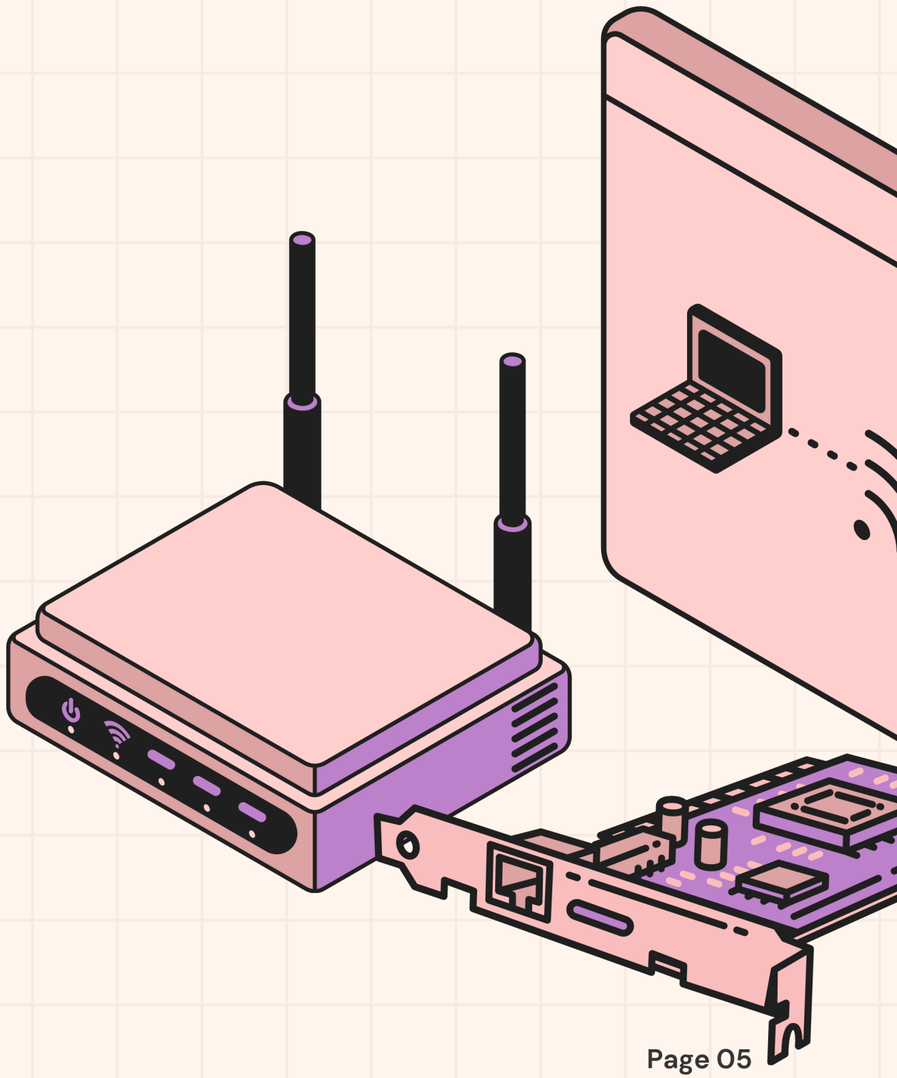
PROJECT MODULES

Client Module (Bank User Interface):

- Functionality: Allows users to log in, view account details, and perform transactions.
- Network Type: TCP connection for reliable, secure communication.
- Connections: 1 TCP connection per session for secure data transfer.

Server Module (Bank Central System):

- Functionality: Handles client requests ,and stores account details.
- Network Type: TCP for client requests, UDP for notifications.
- Connections: Multiple TCP connections to handle concurrent clients; 1 UDP connection for real-time notifications.



PROJECT MODULE

Transaction Processing Module:

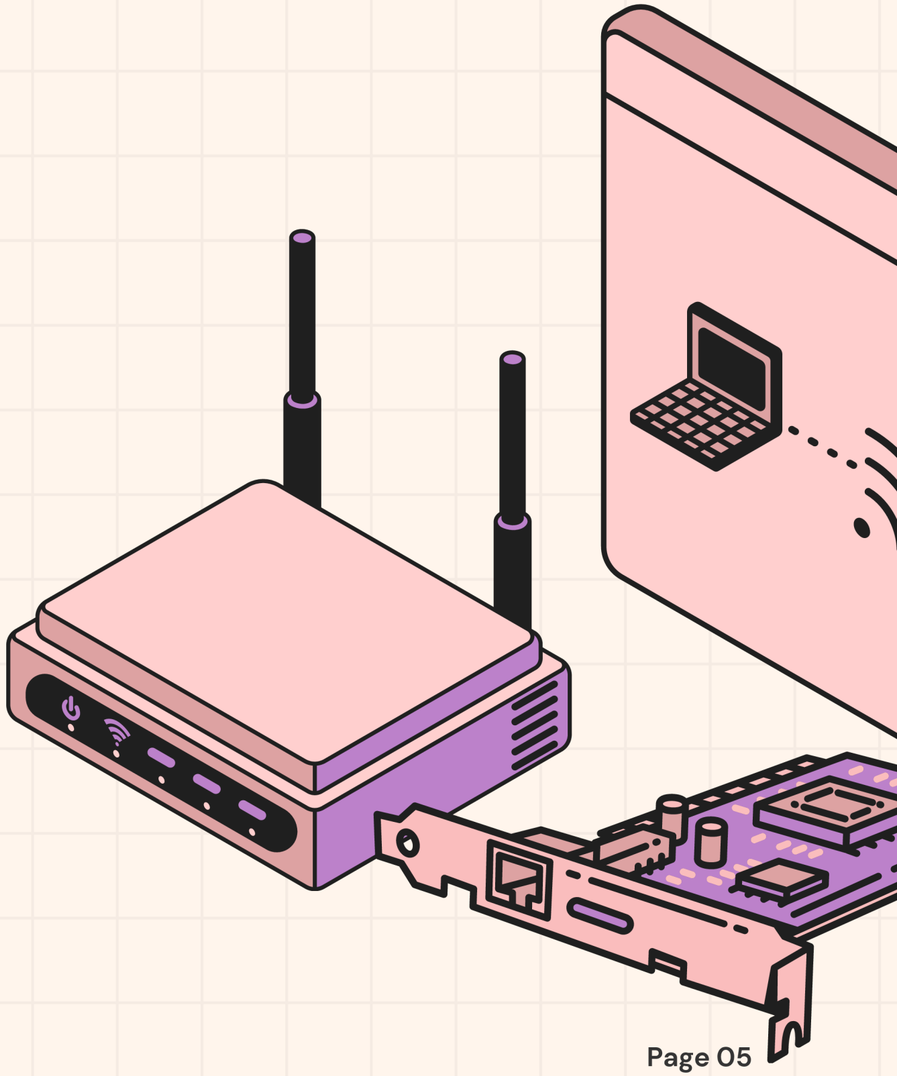
- Functionality: Processes various client transactions such as balance checks and fund transfers.
- Network Type: TCP to ensure transactional data integrity.
- Connections: Shared TCP connection with client-server communication.

Security Module:

- Functionality: Ensures secure communication by validating client credentials before processing requests.
- Network Type: TCP to maintain a secure channel.
- Connections: 1 secure TCP connection per client login session.

Notification/Alert Module:

- Functionality: Sends real-time alerts (e.g., transaction confirmations, suspicious activity) to clients.
- Network Type: UDP for faster, less reliable delivery.
- Connections: 1 UDP connection for broadcasting alerts to clients.



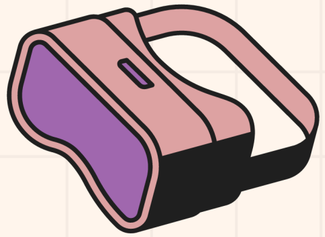
SELECTION OF TCP/UDP CONNECTIONS:

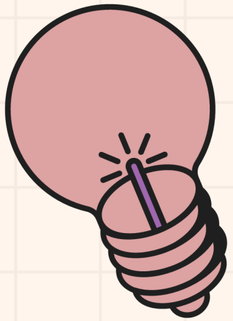
TCP

- TCP is used for all core banking operations such as user login, account management, fund transfers, and balance inquiries.
- When the client sends a request (like a login request), it establishes a TCP connection with the server. The server processes the request and sends back the response through the same connection, ensuring that messages are delivered reliably and in order.

UDP

- UDP is used for sending notifications, such as alerts for successful fund transfers.
- When a user receives money, the server sends a UDP notification to the recipient, allowing for immediate feedback without the overhead of establishing a TCP connection.





THANK YOU