

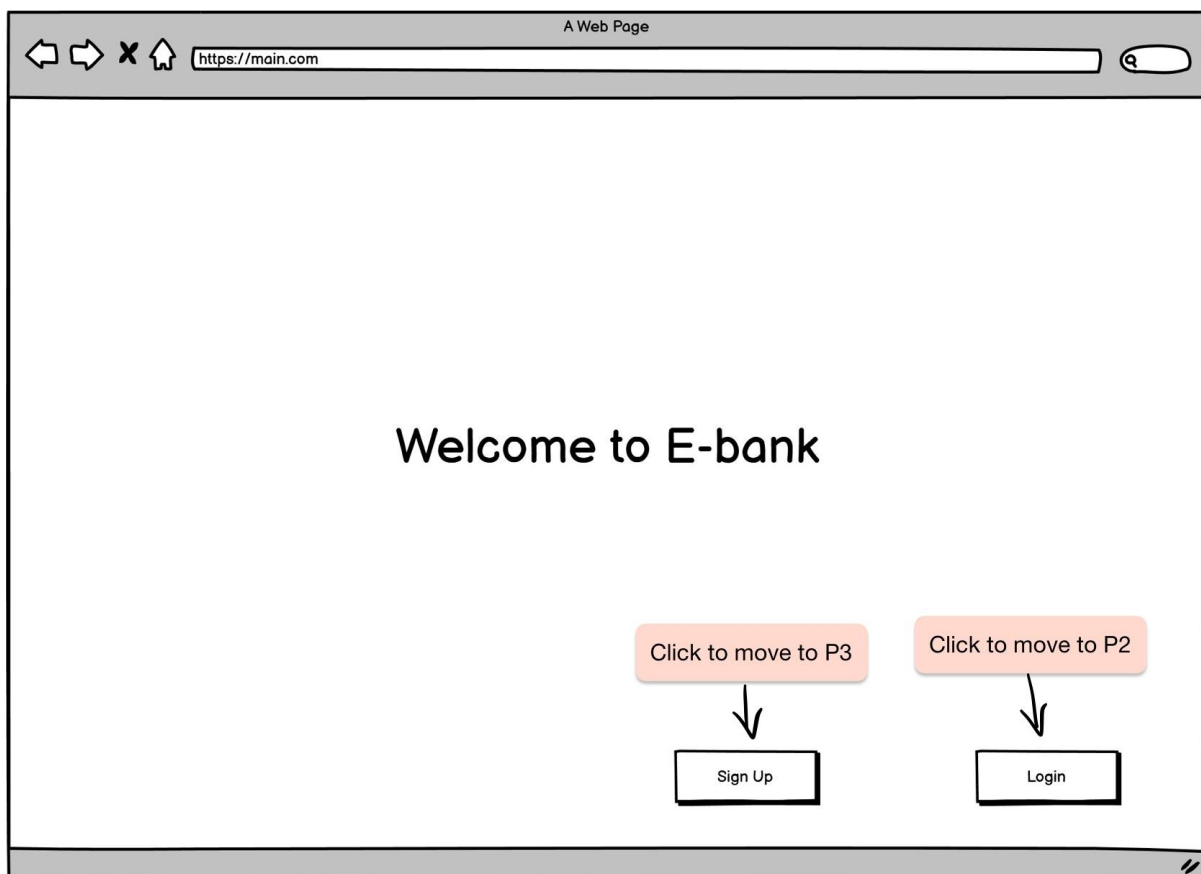
Online Banking Systems on AWS

Overview

The purpose of this project is to familiarize students with building modern applications on Cloud Platform. In this project, students will build an online banking system from scratch, including low level design, database schema design, infrastructure setup, implementation, testing and monitoring.

Low Level Design

UI Mock-up



A Web Page

https://main.com

Welcome to E-bank

Sign Up

Click to move to P1

Click to move to P5

Login

Email address

Password

Login

Close

A Web Page

https://main.com

Welcome

Click to move to P4

Click to move to P1

Sign Up

Name

Email address

Password

Confirm Password

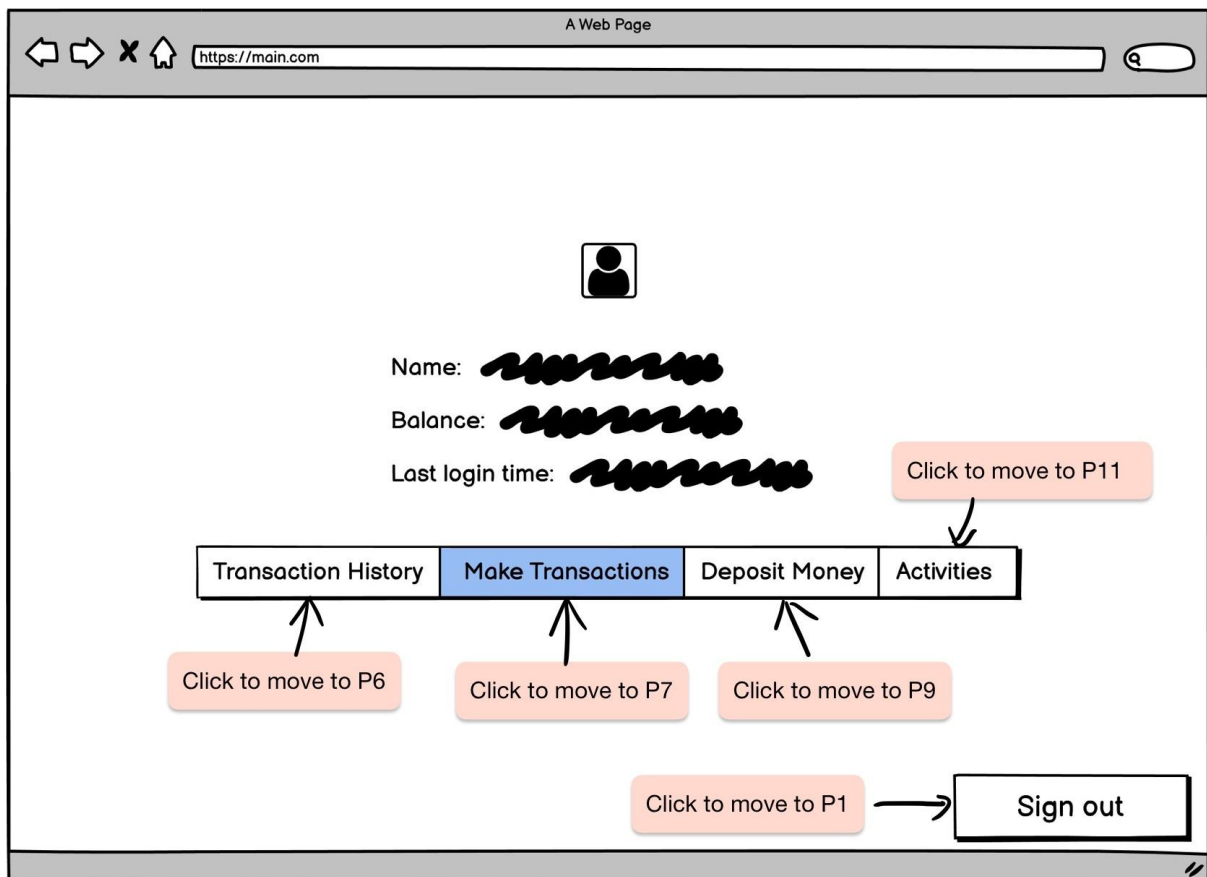
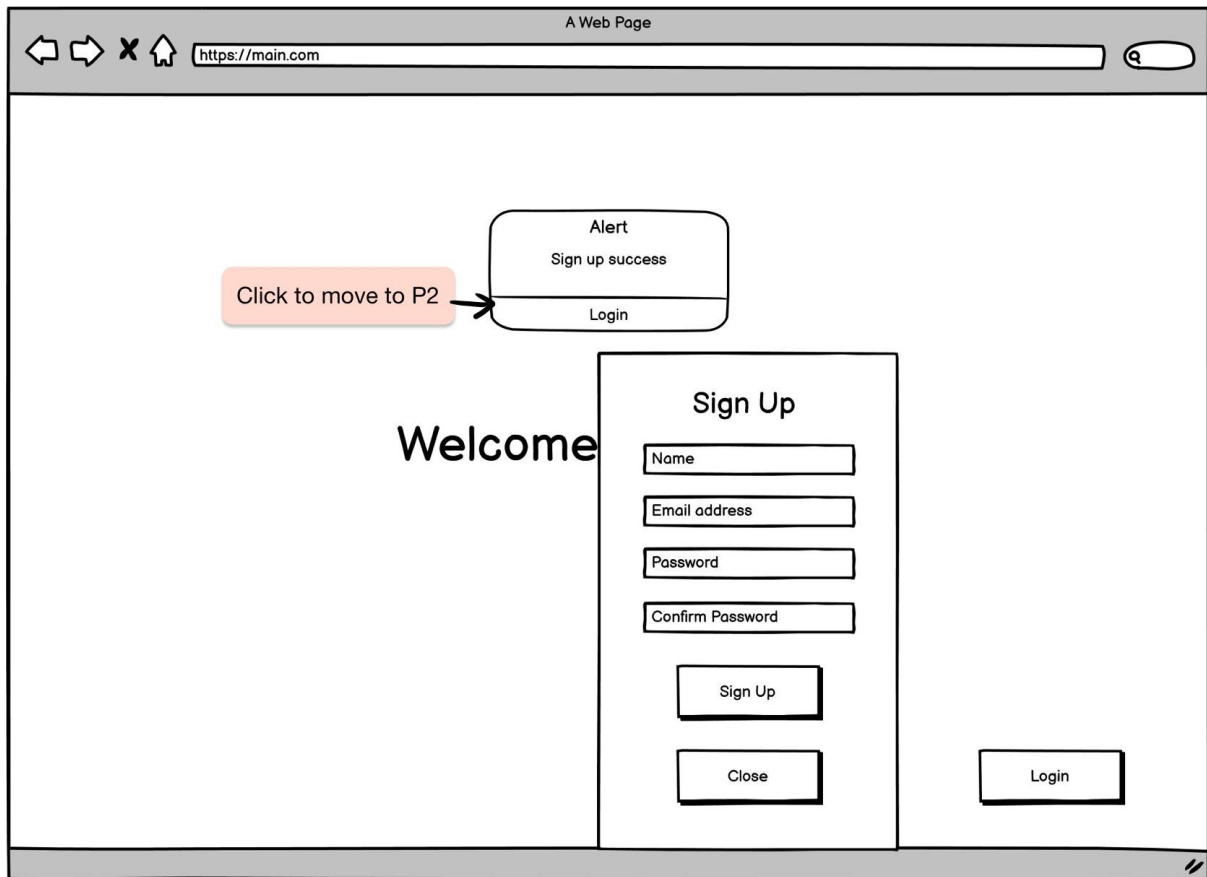
Phone Number

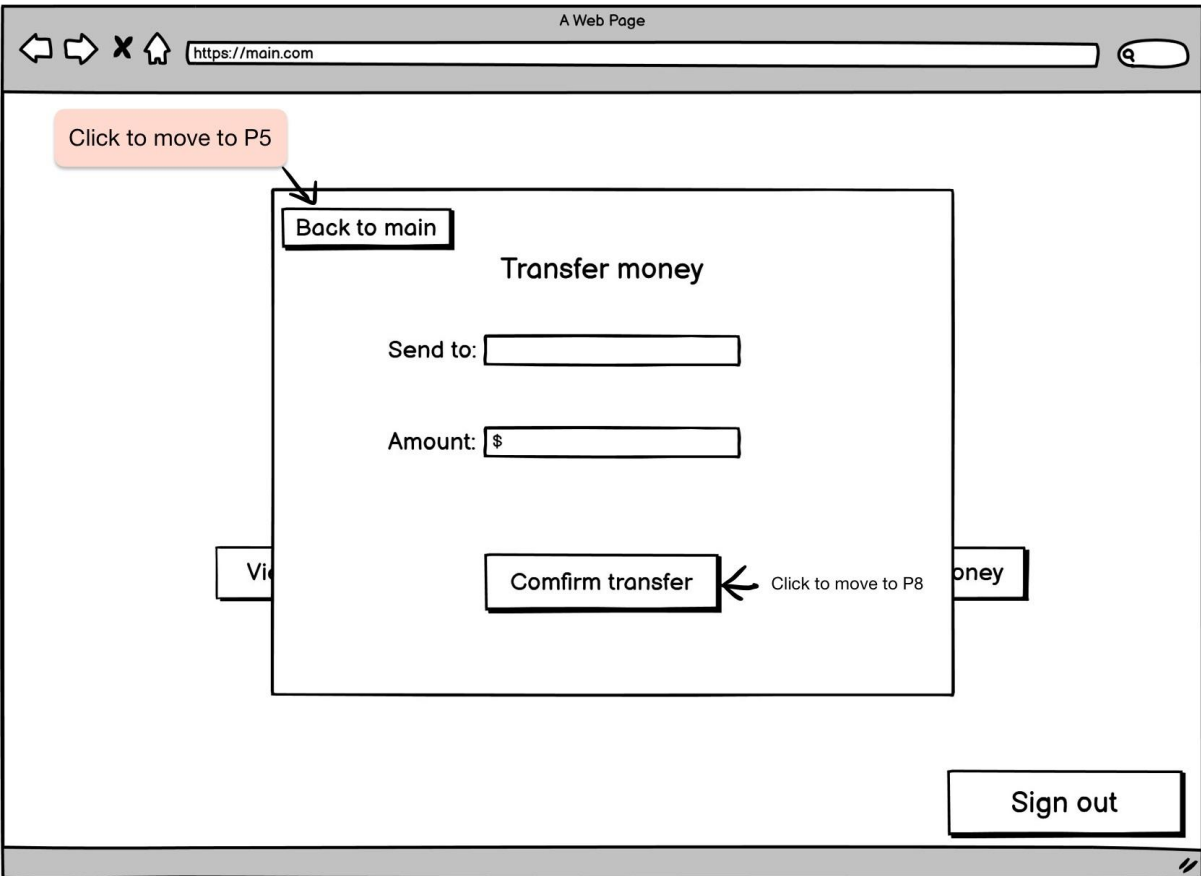
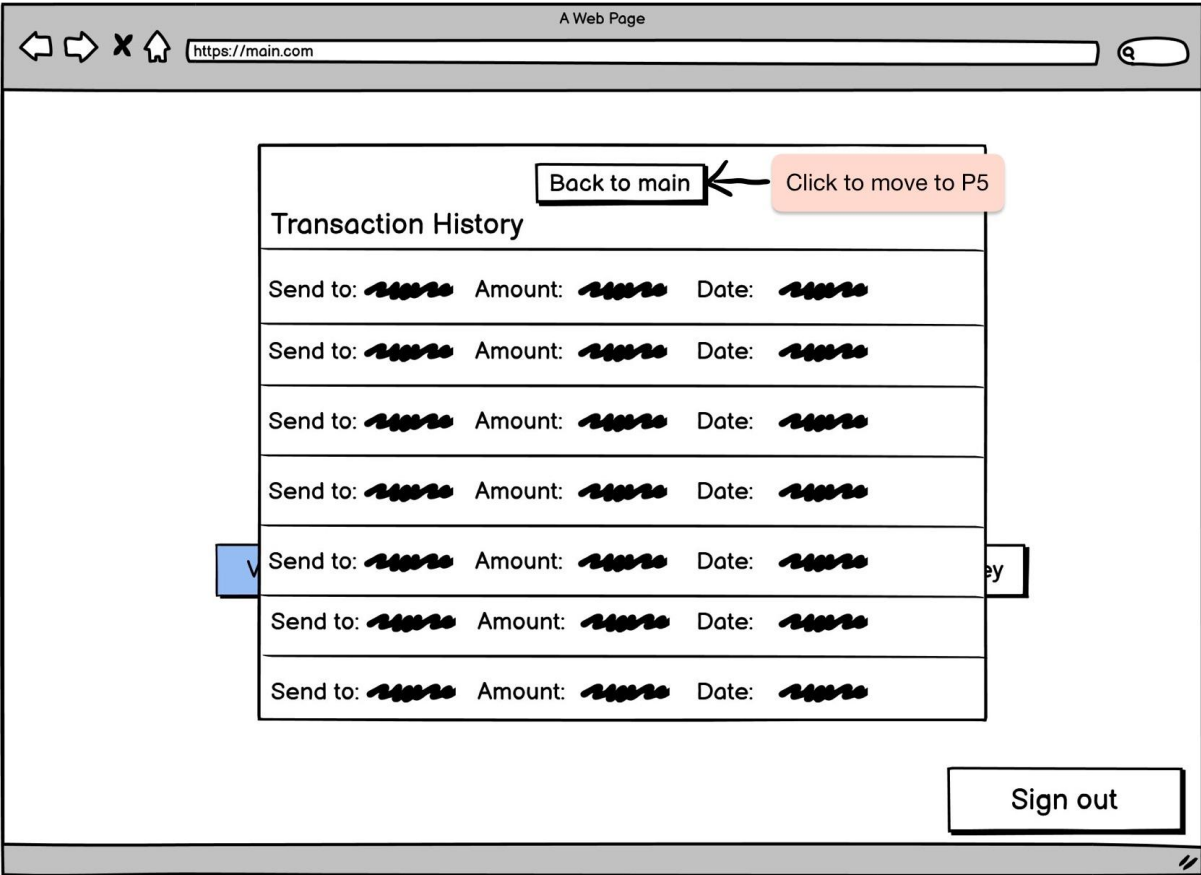
Address

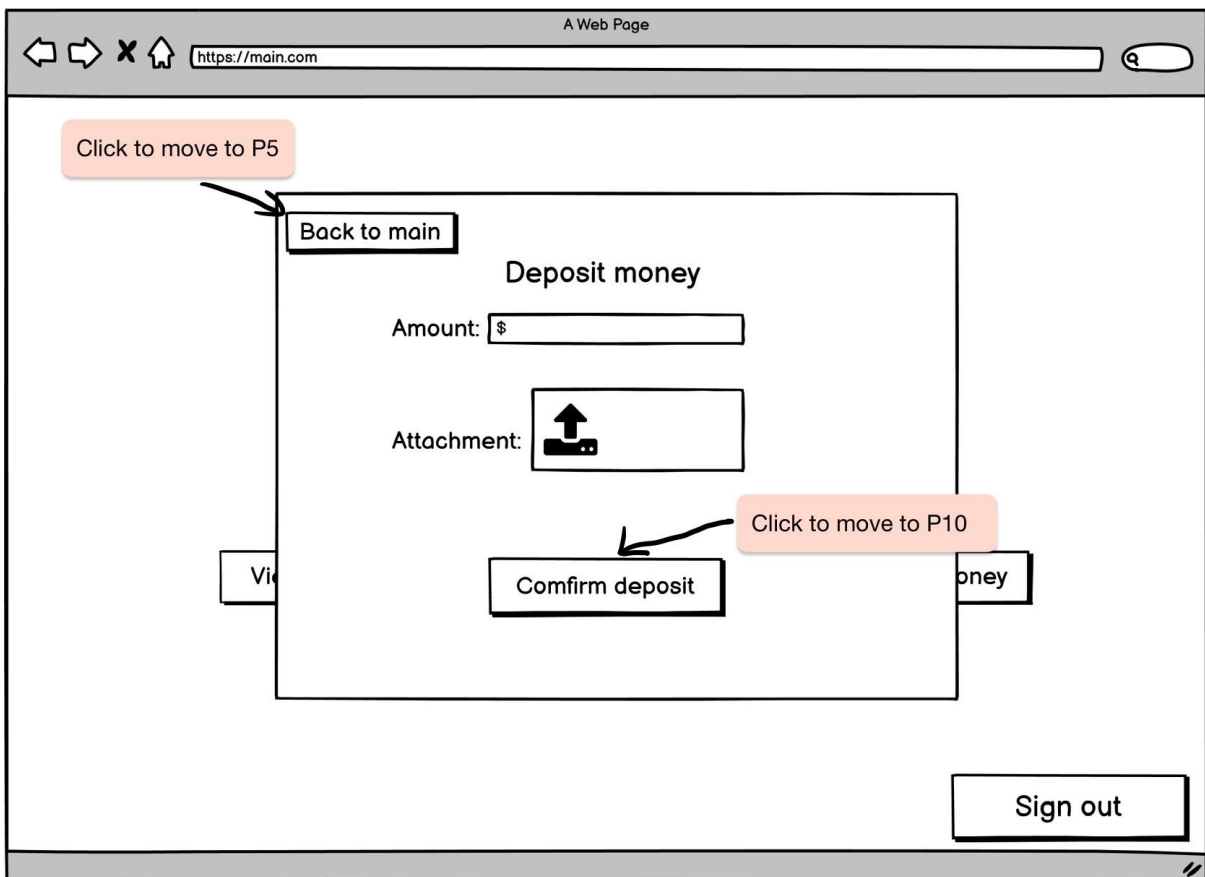
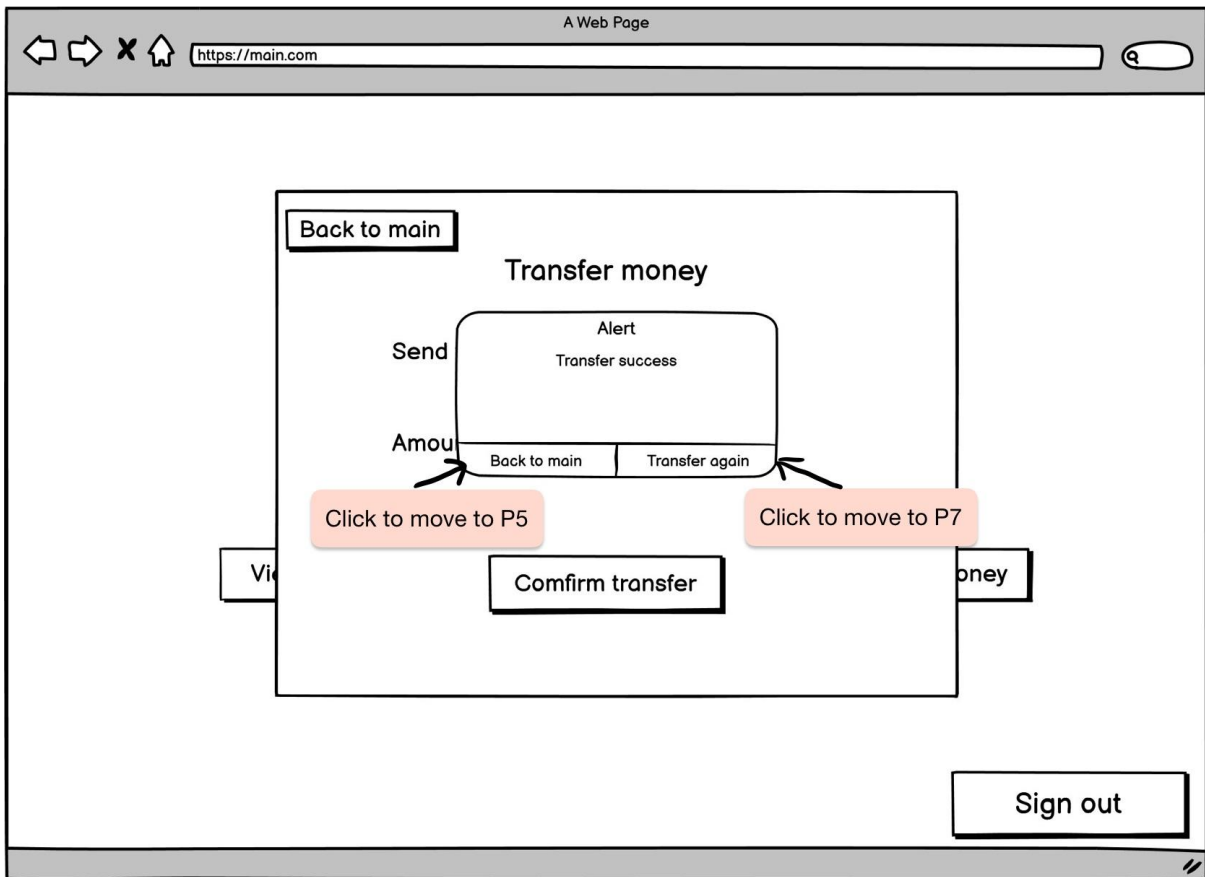
Sign Up

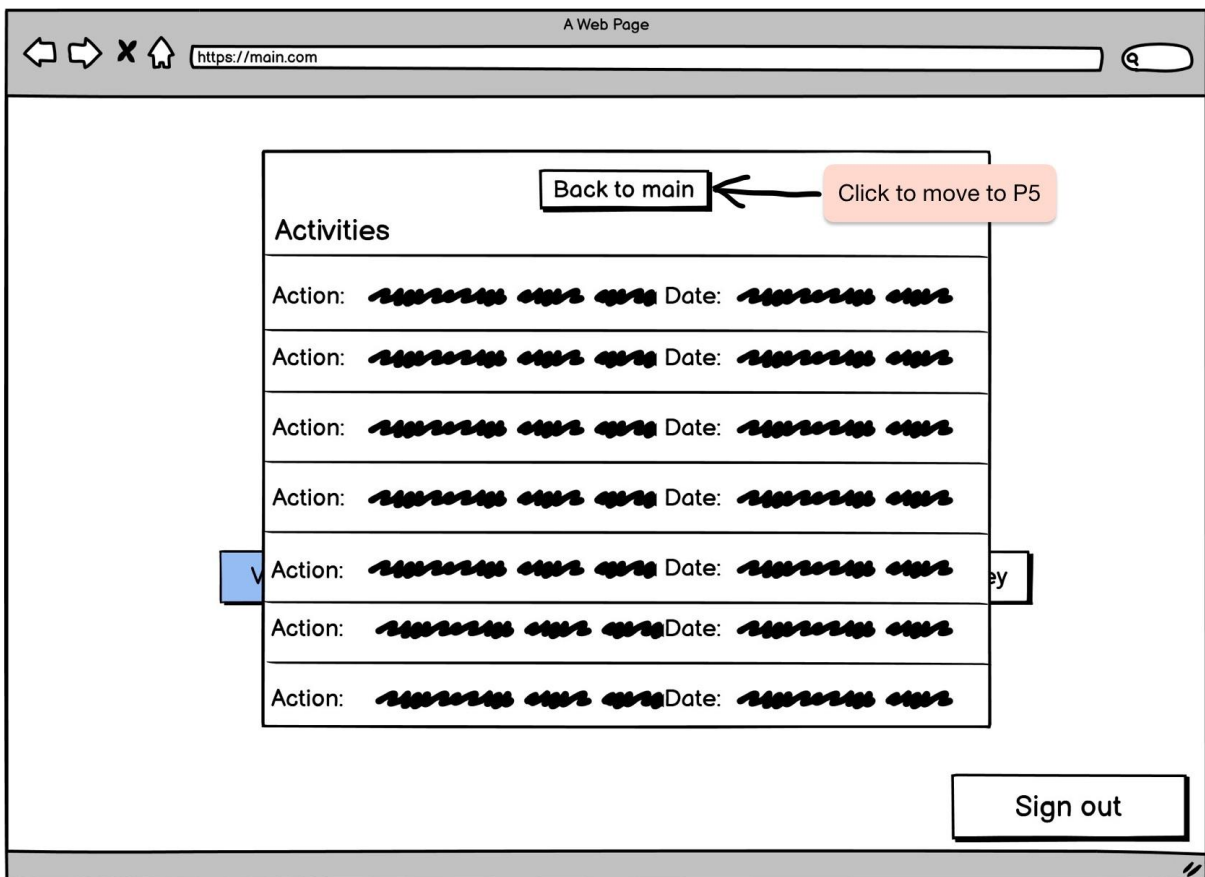
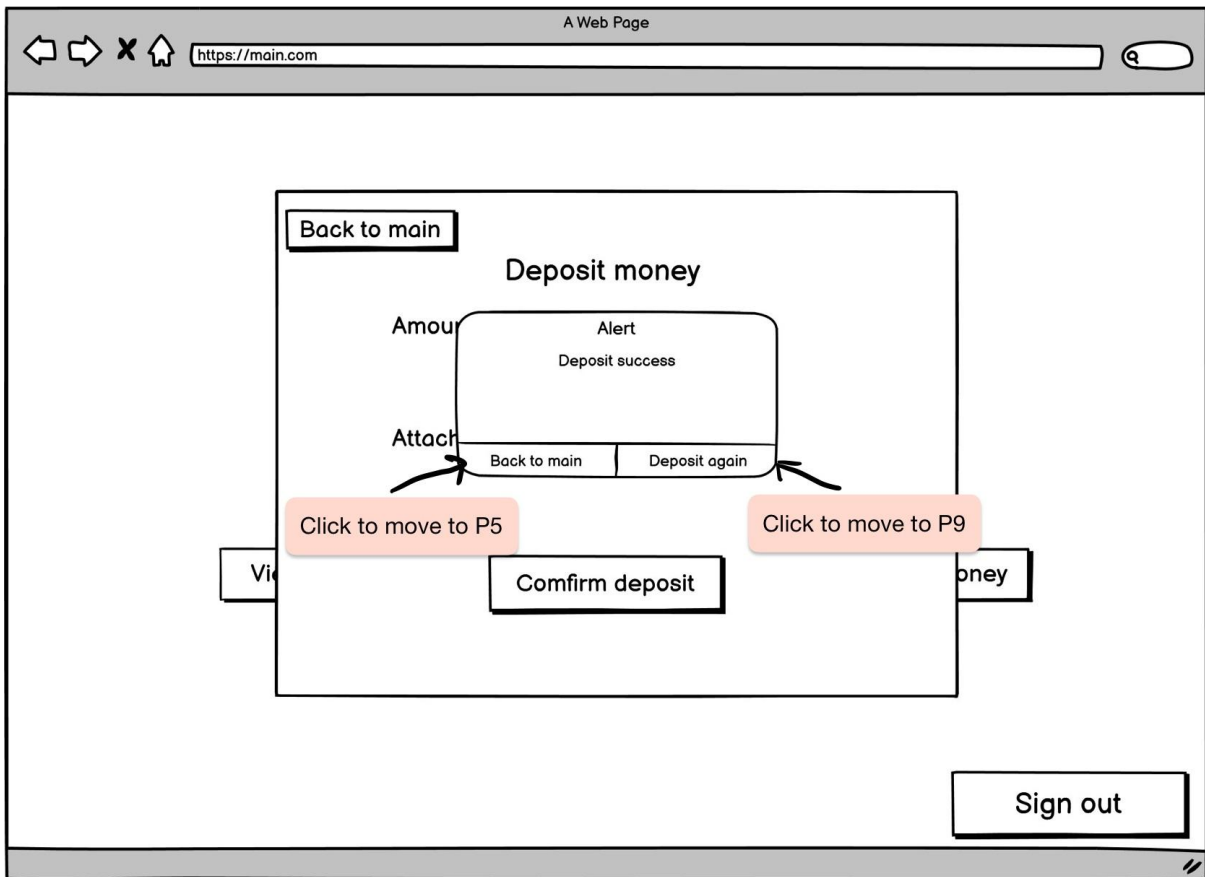
Close

Login

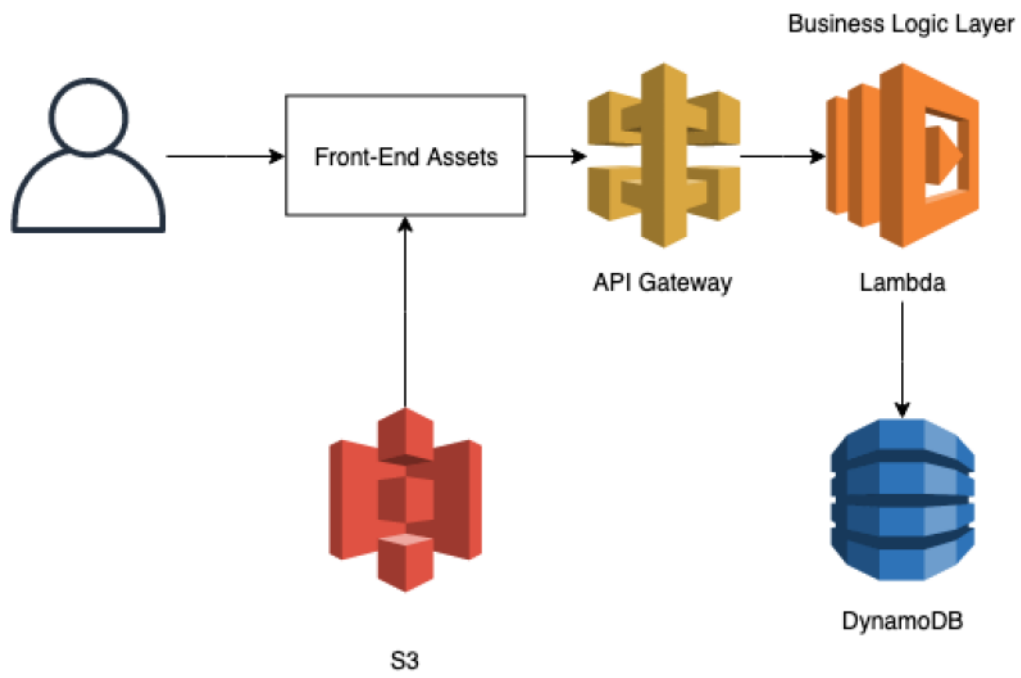








System architecture



Api design

API type: REST API

Function: Sign-Up

HTTP Methods: POST

Api request: **String** Email, **String** Password, **String** Name

Api response: **String** Description.

Method and Description:

signUp()

Returns the description related to the registration information. This description describes the reason for the registration failure.

Function: Login

HTTP Methods: GET

Api request: **String** Email, **String** Password

Api response: **String** Description, **String** auth token

Method and Description:

login()

Returns the description related to the login information. This description describes the reason for the login failure, as well as the account balance, user name, account operation audit trail information for successful login. If the login is successful, the API will return an Auth token for further use, otherwise, it will return null.

Function: getHistory

HTTP Methods: GET

Api request: **String** UserID, **String** auth token

Api response: **List of strings** Account Transaction History

getHistory()

Returns a list of strings related to the user ID. Each element string in the list indicates a transaction, which contains the date, amount, and payee.

Function: getActivity

HTTP Methods: GET

Api request: **String** UserID, **String** auth token

Api response: **List of strings** Account Activity

getHistory()

Returns a list of strings related to the user ID. Each element string in the list indicates an activity, which contains a date and an operation.

Function: Transfer money

HTTP Methods: POST

Api request: **String** Sender User ID, **String** Receiver User ID, **Number** Amount of money, **String** auth token.

Api response: **String** Description.

transferMoney()

Returns the description describing the reason for the login failure.

Function: Deposit money

HTTP Methods: POST

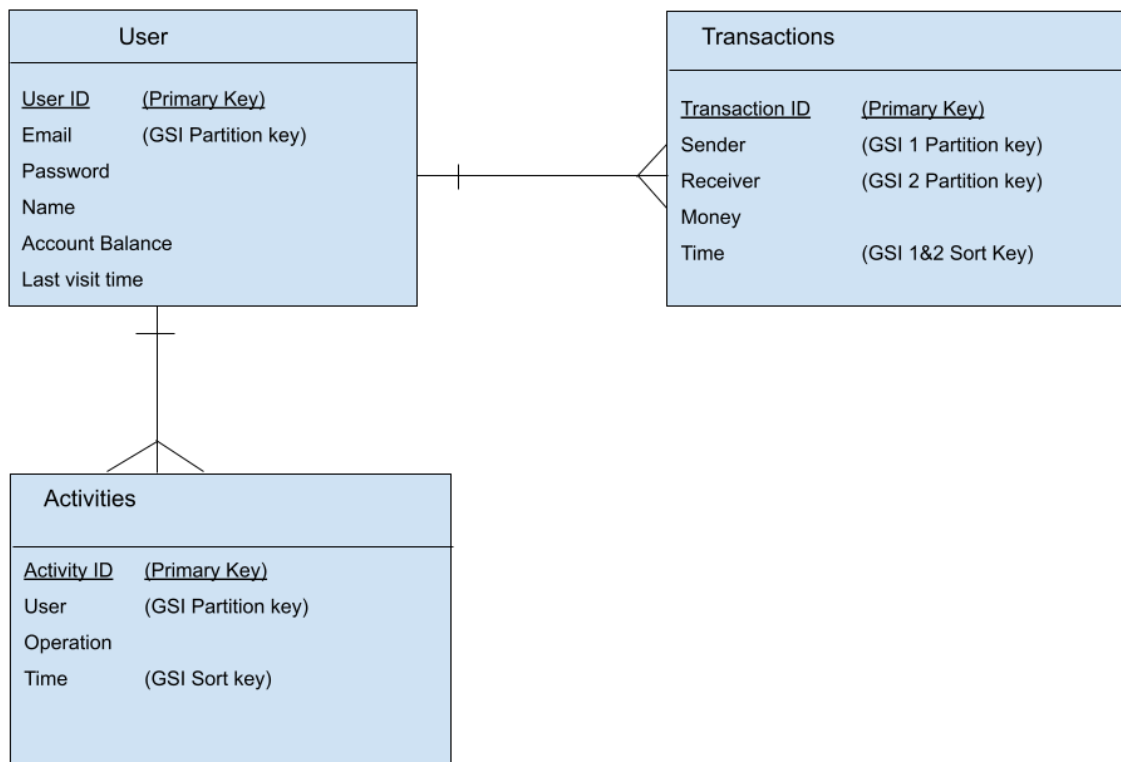
Api request: **String** User ID, **Number** Amount of money, **String** auth token, **File** Attachment

Api response: **String** Description.

depositMoney()

Returns a status code indicating the status of the deposit (0 means failure, 1 means success). The description describes the reason for the login failure.

Database Schema Design



Things to notice

We need to make sure all transactions have ACID properties:

In order to ensure the ACID properties in the database, the project will use all-or-nothing DynamoDB transactions to update the database. The project will also use version numbers for optimistic locking to ensure isolation and concurrency issues.

We need to make sure all data are stored and transmitted securely (at least needs to be encrypted):

The data communication between the server and the client will be securely protected through HTTPS, and sensitive data will be encrypted in the program and stored in the database. In order to prevent malicious users from performing unauthorized operations, the project will use AWS KMS encryption to generate tokens that can authenticate users.