AMERICAN UNIVERSITY OF BEIRUT
MAROUN SEMAAN FACULTY OF ENGINEERING & ARCHITECTURE

American University of Beirut School of Engineering and Architecture

Department of Electrical and Computer Engineering

EECE503 M – Software Security Project

The Online Banking System is a web-based application built using Flask, allowing users to manage their bank accounts, perform transactions, pay bills, and track financial activity. The system incorporates Role-Based Access Control (RBAC) to ensure that different categories of users have appropriate capabilities and restrictions.

The system contains four user roles, each with specific permissions:

- **Customer:** A bank customer who can create and manage their accounts.
- **Support Agent:** Customer support staff who can access user accounts for troubleshooting, but cannot perform financial operations.
- **Auditor:** Read-only access for internal auditing, compliance, and fraud review.
- **Admin:** Super administrator with full access to manage users, accounts, system settings, and security controls.

User Registration:

- Users create accounts by providing:
  - Full name
  - Email
  - Phone number
  - Password

each service has its own db

Admin:

- The system comes with a default admin configured in the database. The admin logs in for the first time and changes his username and password.
- Admin can edit any user's profile and role.
- Admin can create and edit support agents and editors.

🏗 Architecture
The admin service follows a proxy patter

Stateless service (no database)
Forwards authenticated requests to Auth
Service
All requests require admin permission
Uses JWT tokens for authentication
Follows the same structure as other services

Account Creation

- Customers can open new accounts.
- Admin can open accounts for any user.
- Each account has:
  - Auto-generated account number
  - Type (checking/savings)

    - Opening balance
    - Status: Active, Frozen, Closed

Account Dashboard

- Customers see all their accounts with:
    - Current balance
    - Recent 5 transactions
    - Quick links for transfers or bill payment ?

Account Status Management (Admin Only)

- Admin can freeze/unfreeze any account.
- Frozen accounts cannot send or receive transfers.

Internal Transfers

- Customer transfers money between their own accounts.
- Validation:
    - Sufficient balance
    - Account status must be Active

External Transfers

- Customer transfers money to another user's account.
- Requirements:
    - Validate account number
    - Validate available balance
    - Store description + timestamp

Transaction History

- Filter by:
    - Date range
    - Transaction type
    - Amount range
- Export to PDF is optional.

Transaction Model Stores:

- Transaction ID
- Sender account
- Receiver account
- Amount
- Type (credit/debit)
- Timestamp

Support Management (Support Agent Role) Model

Support Agent Capabilities:
☑ Register/login
☑ Manage own profile
☑ View own bank account
☑ View ALL customer accounts (read-only)
☑ View own transactions
☑ View ALL transactions (read-only)
☑ View all support tickets
☑ Update ticket status (Open → In Progress → Resolved → Closed)
☑ Assign tickets to themselves or other agent
☑ Add notes (public or internal)
☑ Filter tickets by status, priority, assignment

- Customers open tickets when they need assistance.
- Support agents can:

testing ->

  - View all open tickets
  - Update ticket status to:
    - Open
    - In Progress
    - Resolved
  - Add notes for customer communication

Audit & Security Module (Auditor Role)

System must log: Model

- Login attempts
- Failed login attempts
- Account freezes/unfreezes
- Suspicious transactions
- Admin operations

Admin Panel

Admin capabilities include:

- Manage users (add, delete, update profile)
- Assign or change user roles
- Freeze/unfreeze accounts
- View audit logs
-

RBAC Permission Matrix

Feature / Action	**Customer**	**Support Agent**	**Auditor**	**Admin** ✓
Register/Login	OK	OK	OK	OK
Manage own profile	OK ←	OK	X	OK
View own accounts	OK	OK	OK	OK
View all user accounts	X	OK	OK	OK
Create accounts	OK	X	X	OK
Internal transfers	OK	X	X	OK
External transfers	OK	X	X	OK
View own transactions	OK	OK	OK	OK

Feature / Action	**Customer**	**Support Agent**	**Auditor**	**Admin**
View all transactions	X	OK	OK	OK
Freeze/unfreeze accounts	X	X	X	OK
Assign/change user roles	X	X	X	OK
View audit/security logs	X	X	OK	OK
Manage support tickets	X	OK	X	OK

Technology stack and architecture

- You should be using Python Flask as backend.
- Choose the frontend technology of your choice
- The application should be multiservice. The backed should not be a one service running all the application. For example, you can have the DB, customers' module, Admin and RBAC on separate services.

Delivery plan and evaluation

- You should take into consideration all the security measures covered in class (e.g., Injection, SSRF, Authentication, Authorization, RBAC, Cryptography failures, etc.).
- Your code should be both secure and functional.
- The evaluation of the deliverables will involve analyzing your code from a security perspective, as well as an interview/presentation.
- You should clearly state each team member's contribution to the project.
- Each team should work independently. No code exchange between groups is allowed, and this will be checked during the evaluation.