Straive 2025 – Full Stack Project Work

Duration: 16 Hours

Domain: Banking

Project Title: Customer Onboarding & Account Management

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Scenario

Develop a digital banking module that allows new customers to register, log in securely, and manage their profile and account details. The system must include JWT-based authentication and support KYC document upload.

# Expected Deliverables

1. Implement frontend screens for registration, login, and profile management.
2. Design relational database with tables: customers, accounts, kyc\_documents.
3. Develop RESTful APIs for register, login, and profile update operations.
4. Ensure secure password handling, data encryption, and input validation.
5. Create CI/CD pipeline with unit tests achieving at least 50% coverage.
6. Deploy the application using Docker
7. Document all APIs using Swagger/OpenAPI and maintain Git Hub branching workflow.

# Validations (Client & Server)

* Field-level validations: required fields, min/max length, numeric ranges.
* Format validations: email regex, phone number format, PAN/Aadhaar pattern (if applicable).
* Business-rule validations: sufficient balance, daily transfer limits, unique constraints.
* File validations: allowed file types, max size, virus scan for KYC uploads.
* Cross-field validations: start date < end date, password confirmation match.
* Server-side re-validation: never trust client-side only — validate again on server.
* API input validation: strict JSON schema validation (use tools like Joi, Marshmallow, or Spring Validation).

# Exception Handling & Logging

* Centralized error handling middleware (e.g., Express error handler, @ControllerAdvice in Spring).
* Categorize errors: ValidationError (400), AuthenticationError (401), AuthorizationError (403), NotFound (404), Conflict (409), ServerError (500).
* Use structured logging with correlation IDs for tracing across services (request id).
* Don't log sensitive data (PII, credentials); mask before logging.
* Retry strategies for transient failures and circuit breaker for model API integration.
* Expose safe error messages to clients; keep internal details in logs.
* Capture metrics for exceptions and alert on anomalous rates.

# Unit, Integration, and End-to-End Testing

* Unit tests: test individual functions, validators, and business logic (JUnit, Jest, pytest).
* Integration tests: test API endpoints with in-memory DB or test containers (Testcontainers).
* Contract tests: validate integration with external Model API (use Pact or stub servers).
* End-to-end tests: simulate user flows using Cypress or Selenium for critical paths.
* Mock external dependencies (Model API, Payment Gateway) during unit tests.
* Coverage targets: aim for >=50% coverage and include coverage report in CI pipeline.
* Automate tests in GitLab CI with separate stages for unit, integration, and e2e.

# JWT Authentication & Authorization

* Auth flow: Login -> issue access token (short-lived) and refresh token (longer-lived).
* Store refresh tokens securely server-side or use rotating refresh tokens.
* Access token content: minimal claims (user id, roles, expiry), avoid storing sensitive data.
* Use strong signing algorithm (RS256 preferred) and rotate keys periodically.
* Token revocation: maintain blacklist or use token versioning in DB to invalidate tokens.
* Protect tokens in transit (HTTPS) and in client storage (HttpOnly secure cookies recommended).
* Implement role-based access control (RBAC) and scope checks on APIs.

# Visual Design & UX Tasks (Common to All Use Cases)

* Follow the given UX prototype or create a simple wireframe in Figma.
* Apply consistent color palette, typography, and layout spacing.
* Ensure responsive design for desktop, tablet, and mobile.
* Use micro-interactions (hover, button click, form validation) to enhance UX.
* Optimize images and assets for faster rendering.
* Demonstrate at least one accessibility test using Chrome DevTools or Lighthouse.

# Submission Requirements

1. GitHub repository link with all source code, CI pipeline, and Dockerfile.  
2. Screenshots and short video (max 5 mins) demonstrating the app and CI/CD pipeline.  
3. API documentation (Swagger/OpenAPI) and Postman collection.  
4. Test reports (coverage, unit/integration/e2e) and vulnerability scan report.  
5. Technical documentation including architecture, schema diagrams, and setup steps.