Worksheet for Web Banking Application

Goal

Develop a **full-stack web banking platform** that allows users to **register, verify their identity via SMS or Email**, log in securely, manage their account, transfer money to other users, and interact with a **Digital Customer Service** system — all through a modern and responsive web interface.

Keywords

Development & Testing Keywords	**Security & Architecture Keywords**	**Functional / Feature Keywords**	**Technical Stack Keywords**
unit testing	authentication	user registration	React
integration testing	encryption	login system	TypeScript
end-to-end testing	JWT tokens	logout functionality	Node.js
Cypress testing	secure REST API	account verification	Express.js
backend testing	data validation	money transfer	MongoDB
frontend testing	session security	protected routes	Mongoose
CI/CD pipeline	role-based access	unique email validation	JWT authentication
Docker containerization	API rate limiting	session management	REST API
API documentation	CORS	JWT tokens	Swagger
Swagger UI		API integration	Postman
Postman testing		error handling	Docker
		database schema	Material UI
		customer service chat bot	Vercel
			Render
			Railway
			Cypress
			Jest
			Supertest
			Twilio API
			Socket.IO
			Jitsi
			Nodemailer
			Figma

Questions

Technical Stack Keywords

- What is React and how does it help build interactive UIs?

- How does TypeScript improve reliability and maintainability in large projects?

- How does Express.js simplify server-side routing and middleware?

- What is MongoDB and how is data stored differently than in SQL databases?

- What problem does JWT authentication solve, and how does it work?

- What is a REST API and what are its core principles?

- How does Swagger help document and test APIs?

- What is Postman used for in backend development?

- How does Docker help in deploying consistent environments?

- Why use Material UI in a React project?

- What services like Vercel, Render, or Railway offer for app hosting?

- What is Cypress and how is it used for frontend testing?

- What is Jest and what types of tests can it perform?

- What is Supertest and how is it used in backend API testing?

- What is Socket.IO and how does it enable real-time communication?

- How is Jitsi used for implementing video calls in a web app?

- What is Nodemailer and how do you send emails from Node.js?

- Why would you use Figma in a full-stack project workflow?

Functional / Feature Keywords

- How does user registration typically work in web applications?

- What are the steps in a secure login system?

- What should happen when a user logs out?

- Why is account verification important and how can it be implemented?

- How does a money transfer process ensure data accuracy and security?

- How do you ensure that each email is unique in user registration?

- What is session management and how does JWT help with it?

- How are JWT tokens created, validated, and refreshed?

- How should errors be handled gracefully in a web application?

- How can a customer service chatbot improve user experience?

- How Password kept in DB ?

Security & Architecture Keywords

- What is authentication and how is it implemented in Node.js apps?

- What is encryption and where is it used in a banking system?

- What role do JWT tokens play in securing communication?

- How can you secure a REST API against unauthorized access?

- Why is data validation important, and what libraries help achieve it?

- How does session security protect user data?

- What is CORS, and why must it be configured properly in web apps?

<u>Develop a web banking application</u>

Overview

In this task, you need to build a web application for a bank. The user will be able to sign-up & sign in to the bank using a username & password, and validating their phone number/ email using a one-time passcode sent by SMS/Email message.

Technical notes

1. Use React for frontend, NodeJS for server, Mongo for DB.
2. Implement the Figma designs for the application UI.
3. Use Twilio integration for SMS service Or email validation.
4. The web application needs to be hosted on a platform of your choice.
5. **<u>You can use AI tools !</u>**

Requirements

1. Sign up

a) The user can sign up to the bank by choosing an email address, a password, and inserting their phone number.

b) user can’t register with the same email address more than once (otherwise, present an error message).

c) Validate that the entered email address format and the phone number format are valid.

2. Phone/ Email validation

a) During sign up, an SMS / Email message with a passcode (6 random digits) is sent to the user in order to validate their phone number / email.

b) Once the user enters the correct passcode, the registration process is succeeded and the user is redirected to the dashboard.

c) If the passcode is not correct, show an error message.

3. Dashboard

a) The dashboard page is protected, access only after sign up / sign in

b) A random account balance will be set once the user was sign up.

c) The user can see their balance and recent transactions.

d) The user can sign out by pressing the Sign Out button, and redirect to the login page.

4. Login

a) The user can sign in by inserting their correct email & password.

b) If validated, the user is redirected to the dashboard. Otherwise, present an error message.

5. Transaction

a) The user can send money to another registered user, by typing the user’s email address & amount, and press Submit.

b) If the user doesn’t have enough balance, or the email address doesn’t exist, show an error message.

c) Once submitted successfully, present another transaction record

1. for sender: the receiver email & the sent amount with ‘-’ sign.
2. for receiver: the sender email & the received amount with ‘+’ sign.

Phases

Phase	Days	Prerequisite	
1	2	HTTP REST API + CRUD Principle Postman Swagger JWT - theory	● Define Rest API using swagger ● Relevant Sequence Diagram
2	2	JS TypeScript NODE.JS Express JWT	● Implementation of BE (no DB only use memory to store / retrieve data) ● Test using Postman
3	1	MongoDB Mongoose	● Design DB schema using swagger ● Implementation
4	1	Docker	● Run Backend on using docker
5	0.5	Figma	● Design UI / UX using Figma for 5 web pages. - Registration - Login - Logout - Money Transfer Screen - Dashboard
6	2	HTML, CSS, JS TypeScript REACT Material UI	Implement Front End
7	2	Web Socket Socket.IO	Implement transfer money notification between customers
8	2	Jitsi	Implement Video Call between a person who sends the money to the person who receives the money
9	1	Open AI Package	implement customer service chat bot
10	1	jest + super test	Implement Testing for BE
11	1	Cypress	Add front end testing