FinFit

Group 8:
Nihal Adhikary
Christopher Berns
Diptesh Mool
Dhruv Mukherjee
Shlok Singh
Evan Stark

ChatGPT conversations:

Document format, input: https://chatgpt.com/share/67afd21f-f18c-8001-9c95-738ac5929942

Document format, output: https://chatgpt.com/canvas/shared/67afd18fcff0819181b6a73fabdb86fc

Generating a subsystem architecture diagram: https://chatgpt.com/share/67b382f1-8cdc-8001-8b3b-bcfe308d506f
https://www.eraser.io/diagramgpt

Project Overview

Objective: Develop a web platform that gamifies financial health and literacy to engage users in improving their financial knowledge and habits.

Technology Stack

- Frontend: React, JavaScript
- Backend: Django, Python
- Database: MySQL
- Hosting: TBD

Key Features

- User authentication (log-in page)
- Profile management
- Main dashboard with a leaderboard
- Games to enhance financial literacy
- Resources for financial education
- Navigation with upper and/or lower tabs for easy access

Stakeholders

- Niner Finances and Financial Institutions:
- Christopher Berns: product owner

System Architecture

Subsystem architecture refinement:

https://app.eraser.io/workspace/5xuklp2csZSUKt2m7xJA?origin=share

Architecture Overview

The system follows a client-server architecture:

- Frontend: React-based UI for user interactions.
- Backend: Django for business logic and API endpoints.
- Database: MySQL for persistent data storage.

- Communication: RESTful API and CRUD operations between the frontend and backend.

Alternative designs considered:

Subsystem Architecture

Authentication Subsystem

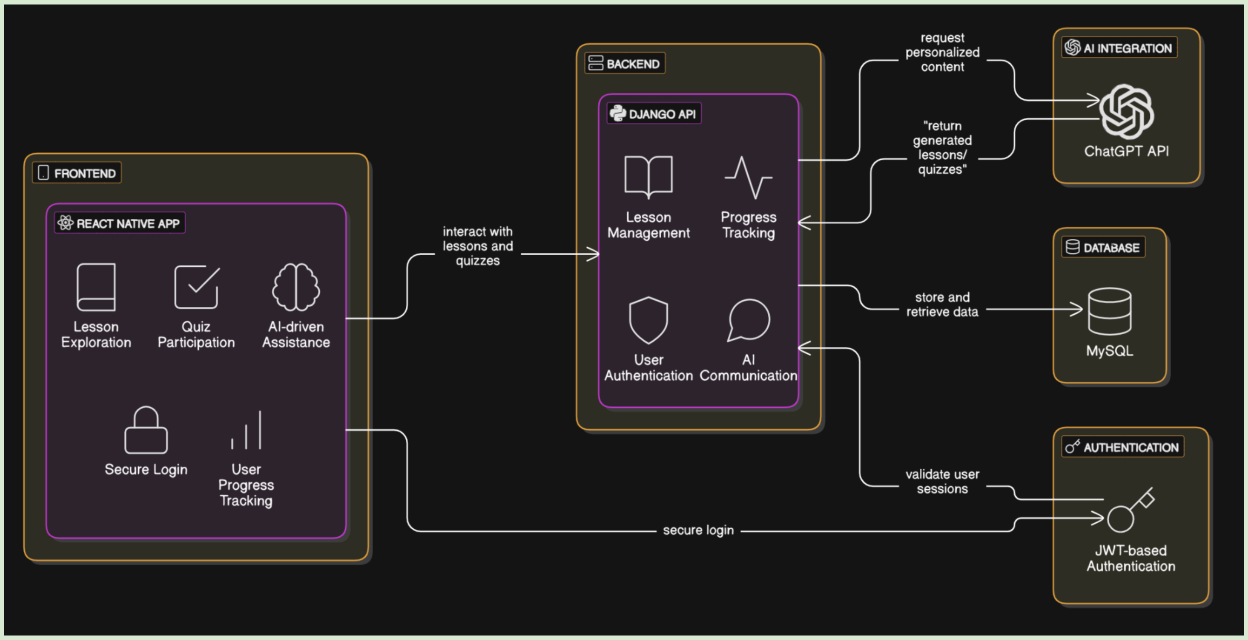
- User registration and login (Django Rest Framework (DRF))
- Profile management
- Secure password handling

Gamification Subsystem

- Leaderboard management (One overall leaderboard, users have points)
- Game mechanics and scoring system
- Financial quizzes and challenges

Educational Resource Subsystem

- Article and video repository
- Interactive tutorials (is this feasible to do?)



Deployment Architecture (create a diagram for it)

- Frontend Hosting: Vercel/Netlify (TBD)
- Backend Hosting: AWS/Render/DigitalOcean (TBD)
- Database Hosting: Managed MySQL instance
- CI/CD: Docker for automated deployments

Persistent Data Storage (need file formats)

- User Data: Profiles, authentication details
- Game Data: User scores, progress tracking
- Resource Data: Educational content repository
- Leaderboard Data: Rankings based on user performance (One overall leaderboard, users have points)

Global Control Flow

- User authentication and session management
- API calls between frontend and backend
- Data validation and error handling
- Procedural or event-driven:
- Time dependency:
- Concurrency:

Application Stack Configuration

The system will be divided into three main layers:

Frontend (React Native)

- Purpose: User interface for mobile (and optionally web).
- Tech Stack: React Native (Expo), UI Kitten, Axios (for API requests).
- Key Features:
  - User authentication (Sign up, Login, Profile).
  - AI-powered chatbot for financial learning.
  - Personalized learning paths.
  - Quiz & progress tracking.

AI Learning Engine (ChatGPT API)

- Purpose: Provides AI-generated financial lessons, quizzes, and explanations.
- Tech Stack: OpenAI GPT-4 API, custom prompt engineering.
- Key Features:
    - AI-generated financial literacy lessons.
    - Context-aware financial advice.
    - Interactive quizzes with AI-generated explanations.

Backend (Django + Django REST Framework)

- Purpose: Handles API requests, user data, and AI interactions.
- Tech Stack: Python, Django, Django REST Framework, PostgreSQL.
- Key Features:
    - API endpoints for user management.
    - AI response handling (integrating ChatGPT).
    - Learning progress tracking.
    - Quiz generation & evaluation.

Tables

A. Users Table

Field	Type	Description

id	INT AUTO_INCREMENT	Primary Key
username	VARCHAR(255) UNIQUE	Unique username
email	VARCHAR(255) UNIQUE	Unique email
password	VARCHAR(255)	Hashed password
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Account creation date

B. Learning Progress Table

Field	Type	Description
id	INT AUTO_INCREMENT	Primary Key
user_id	INT	Foreign Key (Users)
lesson_id	INT	Foreign Key (Lessons)
progress	INT	Percentage of lesson completed
completed_at	TIMESTAMP NULL	Timestamp when lesson is completed

C. Lessons Table

Field	Type	Description
id	INT AUTO_INCREMENT	Primary Key

title	VARCHAR(255)	Lesson title
content	TEXT	AI-generated lesson text
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Timestamp

D. Quizzes Table

Field	Type	Description
id	INT AUTO_INCREMENT	Primary Key
lesson_id	INT	Foreign Key (Lessons)
question	TEXT	Quiz question
options	JSON	Multiple-choice options
correct_ans	VARCHAR(255)	Correct answer

Create a **multi-tier system architecture** for an **AI-driven learning platform** using the **ChatGPT API** for personalized learning. The system consists of:

- **Frontend (React Native App)** for users to interact with lessons, quizzes, and AI-driven learning assistance.
- **Backend (Django with MySQL)** serving APIs to manage users, lessons, and progress tracking.
- **AI Integration (OpenAI ChatGPT API)** to generate dynamic lessons and quizzes based on user input.
- **Database (MySQL)** for storing user progress, lesson content, and quiz data.
- **Authentication (JWT-based)** for secure user login and session management

---

Detailed System Design

User Authentication Flow

- User enters credentials
- The backend verifies and returns JWT token
- The frontend stores token for the session

Leaderboard System

- Users earn points through games
- Backend updates leaderboard rankings
- Frontend displays a real-time leaderboard

Static View

- Frontend Components: Login page, Profile page, Main dashboard, Games page, Resources page

- Backend Modules: User management, Game logic, Content management

Dynamic View

- User Interactions: Logging in, playing games, accessing resources
- Data Flow: API interactions for authentication, game progress, leaderboard updates