FinFit

Group 8 Design Document:

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

ChatGPT Conversations	
Document format, input, rough draft	https://chatgpt.com/share/67afd21f-f18c-8001-9c95-738ac5929942
Document format, output, rough draft	https://chatgpt.com/canvas/shared/67afd18fcff0819181b6a73fabdb86fc
Document format, input, final draft	https://docs.google.com/document/d/1QwaXa51ljaJTefh-_nwAhjvca3BsbUYD89RLMM4uKN4/edit?usp=sharing
Document format, input/output, final draft	https://chatgpt.com/share/680693d5-dc5c-8001-b029-a266048b9513
Generating a subsystem architecture diagram, first draft	https://chatgpt.com/share/67b382f1-8cdc-8001-8b3b-bcfe308d506f https://www.eraser.io/diagramgpt

Project Overview

Objective:

FinFit is a platform designed to make financial literacy education both engaging and interactive. Through a combination of quizzes, games, and personalized AI-generated content, FinFit empowers users to acquire important financial skills. The system tracks user progress through a real-time leaderboard, awarding points for engaging with the educational content. The platform ensures personalized learning paths that adapt to users at various knowledge levels, allowing them to progress through their financial education journey at a suitable pace. This approach aims

to meet diverse learning needs and promote the long-term retention of financial concepts in an enjoyable and effective way.

Technology Stack:

FinFit is built with a robust technology stack. The frontend of the platform is developed using **React**, a JavaScript framework that offers dynamic rendering for an interactive and responsive user experience. React ensures smooth transitions between different pages and allows real-time updates of content, making it highly suitable for mobile and desktop experiences. For the backend, **Django** (a Python web framework) is used alongside **Django REST Framework (DRF)** to handle the business logic and API management, ensuring secure and efficient communication between the frontend and backend systems. The **MySQL database** is employed to store persistent data such as user profiles, quiz scores, and progress tracking information, allowing the system to maintain consistent data across sessions.

Key Features:

FinFit incorporates several features aimed at enhancing the learning experience. One of the key features is **user authentication**, where users can securely log in and manage their profiles, tracking their progress, including completed quizzes and lessons. The **leaderboard** allows users to compete with others based on the points they have earned through quizzes and games, providing an additional layer of motivation and engagement. The **financial literacy games** feature includes various interactive quizzes and challenges designed to test and expand users' understanding of essential financial concepts. Additionally, The **navigation** of the platform is designed to be intuitive, offering easy access to all sections such as games, quizzes, the

leaderboard, and educational resources.

Stakeholders:

The platform benefits a wide array of stakeholders. **Niner Finances**, a financial literacy initiative, plays a significant role by supporting and promoting financial education within the university community. Financial institutions are also key partners, contributing valuable educational content and tips to help users build better financial habits. The **Product Owner**, Christopher Berns, oversees the development of the project, ensuring it aligns with educational goals and meets the needs of the end-users.

Part 2 - System Architecture

Subsystem architecture refinement	https://app.eraser.io/workspace/5xuklp2csZSUKt2m7xJA?origin=share Subsystem architecture diagram, final draft:
Subsystem architecture diagram, final draft.	https://app.eraser.io/workspace/2vpMG8aHDI7JMQpbgzVH?origin=share&elements=zgA4mV81I-J5kR2LnIfpSA

Architecture Overview

The architecture of FinFit follows a **client-server** model, which ensures clear separation between the frontend, backend, and the database. Each component of the system operates independently but interacts seamlessly through well-defined APIs.

The system follows a client-server architecture:

- **Frontend (React):**

  The frontend serves as the user interface, handling user interactions and dynamic

rendering of content. It is built using React, which is optimal for creating highly interactive, real-time web applications. React components like useState and useEffect manage dynamic content such as quiz results, leaderboard scores, and user data. React Router facilitates navigation between various pages like sign-up, login, and the main dashboard, while Tailwind CSS ensures that the pages are visually appealing and responsive across all devices.

- **Backend (Django + DRF):**

  The backend, powered by Django and Django REST Framework (DRF), handles the business logic, user authentication, and API operations. It serves as the core of the application, processing user data, validating requests, and ensuring secure access to the system. Django's model-view-controller (MVC) architecture makes it easy to manage user authentication, API routes, and database interactions, while DRF allows the backend to serve RESTful APIs to the frontend, enabling smooth data flow between the client and server.

- **Database (MySQL):**

  The MySQL database is responsible for persistent data storage. It stores essential user data such as authentication details, quiz scores, and financial progress. The database schema includes tables for user information, leaderboard rankings, and financial data (e.g., user transactions, points earned). The backend interacts with MySQL through Django ORM, enabling CRUD (Create, Read, Update, Delete) operations on the data.

Part 2.1 - Subsystem Architecture

Authentication Subsystem

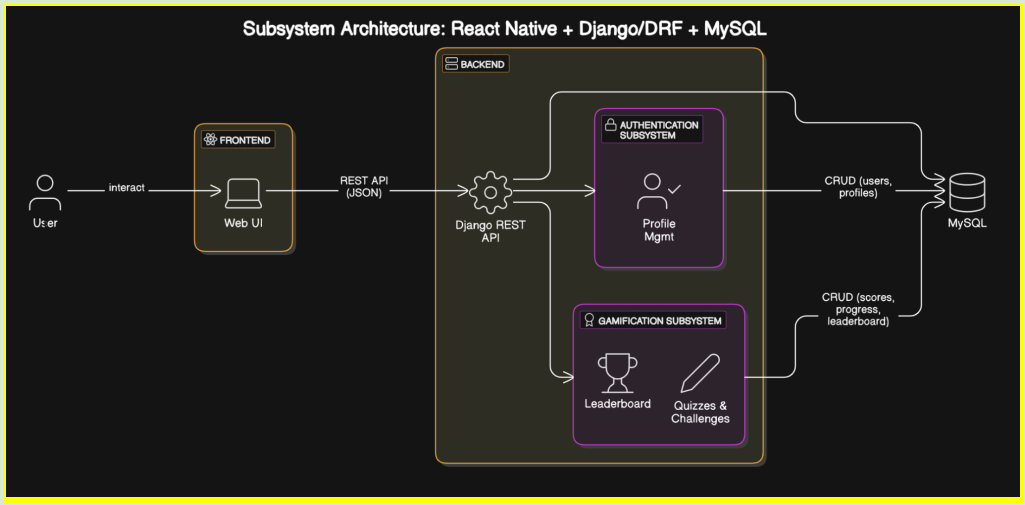
Authentication Subsystem:

- This subsystem is responsible for managing user authentication. Built using **Django REST Framework (DRF)** it ensures secure user login and profile management. This subsystem handles registration, login, and password hashing.
- **Features**: User registration, login, password hashing, and profile management (including avatars).

Gamification Subsystem

- The **gamification subsystem** focuses on enhancing user engagement through a global leaderboard. Users earn points by completing financial literacy games, quizzes, and challenges. The leaderboard ranks users based on the points accumulated, providing real-time feedback on their performance. This subsystem is designed to encourage friendly competition and continuous learning. Additionally, the system incorporates game mechanics where points are awarded for correct answers and participation, motivating users to stay engaged and progress in their learning journey.

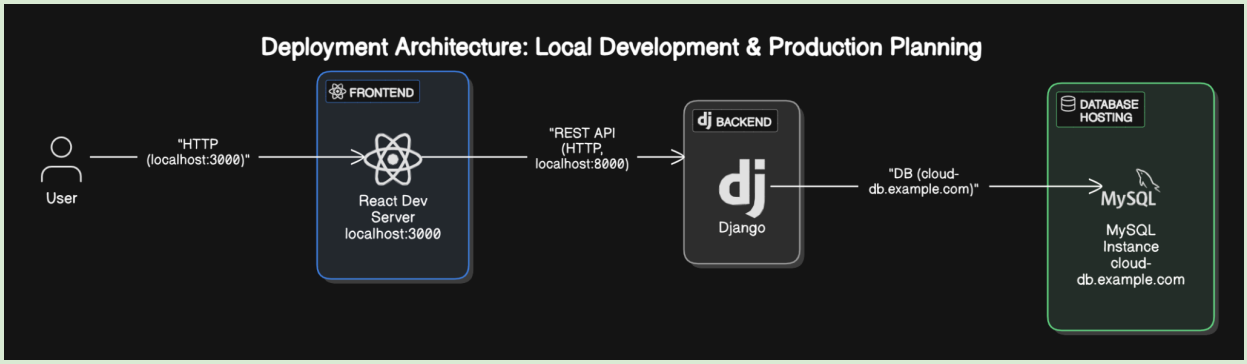
NEW:



Part 2.2 - Deployment Architecture

Deployment Architecture Diagram Generation: https://app.eraser.io/workspace/UJQuj9yJH6765X5CFrQX?origin=share

- The deployment architecture for FinFit involves hosting both the frontend and backend components in a local development environment initially, with plans for cloud-based hosting in production. The **frontend** is hosted on a React development server, typically accessible through **localhost:3000**, which allows for real-time updates and hot-reloading during development. The **backend** is hosted on **localhost:8000**, running Django's development server, and exposed through RESTful APIs for communication with the frontend. **Django REST Framework** handles all API requests, allowing the frontend to retrieve user data, leaderboard scores, and other dynamic content. This setup is scalable and can easily be deployed to cloud platforms such as **AWS** or **DigitalOcean** once the application is production-ready.



Part 2.3 - Persistent Data Storage

The data required for the functioning of FinFit is stored in **MySQL**. The platform relies on **three main data types**:

- **User Data**: This includes user account details, authentication credentials, preferences, and learning progress.
- **Game Data**: This stores quiz scores, completed lessons, and the points accumulated by the user.
- **Leaderboard Data**: This includes rankings and points information, which is updated in real-time based on user participation.

All of this data is managed through **Django ORM**, allowing seamless integration between the application's backend and the database.

Part 2.4 - Global Control Flow

The FinFit system follows an **event-driven model**, where user interactions trigger various backend processes. For instance, when a user logs in or submits a quiz, an API request is sent to the backend, which processes the request and updates the database accordingly. The frontend then displays the updated information to the user in real-time. The backend is capable of handling multiple simultaneous requests and is designed to scale horizontally to support increasing traffic. Error handling is integrated at both the frontend and backend to ensure that users receive appropriate feedback if any issue occurs, and retries are available for failed requests.

Application Stack Configuration

Three-Tier System Architecture:

The **Frontend (React Native)** serves as the user interface for the FinFit platform, designed to be accessible on both mobile devices and optionally as a web application. The primary purpose of the frontend is to provide a seamless user experience for interacting with the platform's features. It allows users to register and log in to their accounts, access various quizzes, and track their progress over time. Additionally, the frontend supports personalized learning paths that adapt to the user's progress, ensuring an engaging and tailored educational experience.

Backend (Django + DRF)

The **Backend (Django + DRF)** is responsible for managing the application's APIs, handling user states. Built using **Python, Django, DRF (Django Rest Framework)**, and **MySQL**, the backend is the backbone of the system, ensuring the smooth flow of data between the user interface and the database. The backend manages critical functionalities such as user authentication, quiz evaluation, and progress tracking. It also handles the updating of leaderboard information based on users' quiz results and activities. This centralized structure ensures that all user interactions are processed efficiently, with real-time updates to the system.

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
points	INT DEFAULT=0	Number of points the user has.

Part 3 - Detailed System Design

User Authentication Flow:

1. The user enters their login or signup credentials.
2. The backend stores users' credentials.

Leaderboard System:

1. The user earns points through quizzes and educational games.
2. The backend updates the leaderboard with the user's new score and ranking.
3. The frontend dynamically retrieves the latest leaderboard data from the backend and displays it to the user.

Part 3.1 - Static View

- The static view consists of pages like **Login**, **Profile**, **Dashboard**, and **Games**. These pages are served to the user in a structured and visually appealing way, with static content that doesn't change unless the user interacts with the system (e.g., submitting a quiz or completing a challenge).

Part 3.2 - Dynamic View/UML Use Case

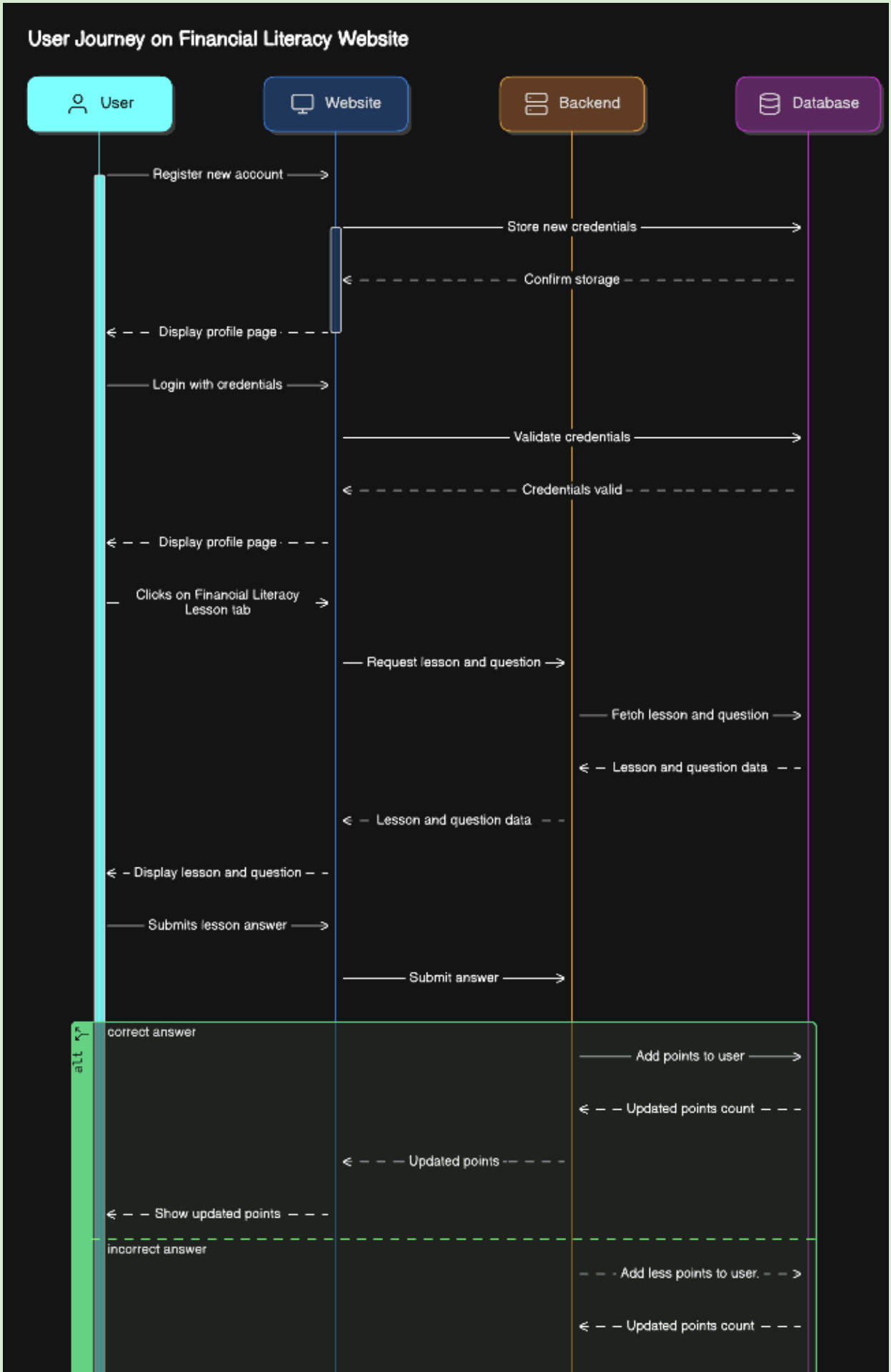
UML Use Case:

https://app.eraser.io/workspace/L953aCOczOo7PJ2rb5Ji?origin=share

The dynamic view includes pages that update in real-time based on user actions, such as **Leaderboard** and **Progress Tracking**. These pages pull the latest data from the backend and display it instantly, providing users with immediate feedback on their learning progress.

Data Flow:

- API request → Backend validation → DB interaction → Response
- Responses returned to frontend and displayed dynamically


User Journey on Financial Literacy Website
User
Website
Backend
Database
Register new account
Store new credentials
Confirm storage
Display profile page
Login with credentials
Validate credentials
Credentials valid
Display profile page
Clicks on Financial Literacy Lesson tab
Request lesson and question
Fetch lesson and question
Lesson and question data
Lesson and question data
Display lesson and question
Submits lesson answer
Submit answer
alt
correct answer
Add points to user
Updated points count
Updated points
Show updated points
incorrect answer
Add less points to user.
Updated points count