Vivid

... is a full-stack web application for any general amateur or professional club that allows administrators to manage the finances, membership, and coaches of their club. It includes user authentication, a messaging platform for all members and administrators, an expense tracking panel, and a membership management system for administration.

Developers				
Andre Arcaina	Tristan Cheng	Jack Heintz	Joseph Leung	Felipe Quiroga

Tech Stack + How it Works

The frontend is developed using `JavaScript` and the Next.js framework, with Tailwind CSS as the chosen `CSS` framework.

The backend framework chosen for this project is Django, a high level `Python` -based framework. In Django, we create database schemas and API endpoints to communicate with the frontend (the client). These endpoints are designed for retrieving data from a PostgreSQL database structure, which is hosted and created using Supabase. Built on a RESTful API architecture, Vivid leverages HTTP methods such as POST, GET, PUT, and DELETE, adhering to the CRUD (Create, Read, Update, Delete) operations paradigm.

Our application also utilizes Django Channels to implement WebSockets, enabling real-time chat functionality seamlessly integrated into the user experience. With Django Channels, we establish persistent connections between the client and the server, allowing bidirectional communication without the need of traditional HTTP requests.

For more information, check the documentation.

Functionality

1. **`User Authentication`** :
    - [x] Members can register accounts and log in securely to access the application.
    - [x] Passwords are securely hashed and stored in the database.

- [x] Administrators can manage member accounts, including resetting passwords and deactivating accounts if necessary.

2. **`Messaging Platform`** :

   - [x] All members and administrators have access to a messaging platform within the application.
   - [x] Administrators, notably the coaches, can broadcast annoucements to all members.
     - [x] Administrators can send and receive messages.
   - [x] Members can send and receive messages, and access announcements from coaches.

3. **`Expense Tracking Panel`** :

   - [x] Administrators, particularly the treasurer, have access to an expense tracking panel.
   - [x] Expenses incurred by the club, such as rent for the meeting hall, coach salaries, and other operational costs, can be recorded and tracked.
   - [x] The panel includes a log of any unpaid debts from previous months, including instances where the rent for the hall was not paid in full or not paid at all, as well as any unpaid coach expenses.
     - [x] The treasurer can review and prioritize outstanding debts, follow up with relevant parties, and take necessary actions to settle outstanding balances.
   - [ ] Historical data on unpaid debts allows the treasurer to identify trends, assess financial risks, and implement strategies to improve financial management and accountability within the club.
   - [ ] The panel also provides information on the current month's account payables, including members who have paid in advance. This allows the treasurer to maintain accurate records of prepaid fees, track membership dues, and ensure timely invoicing and collection processes.
   - [x] The treasurer has access to managing the coaches salary in this panel.

4. **`Membership Management System`** :

   - [x] Administrators, including the treasurer and coaches, have tools to manage club memberships.
   - [x] New members can register and apply for membership through the application.
   - [x] Coaches can approve (add) or reject (remove) membership applications, and manage membership statuses.
   - [ ] Membership dues, costs and fees can be collected securely through the application.
   - [x] Coaches can communicate with members regarding membership-related matters, such as costs, sessions, and events.
   - [x] Both the treasurer and coaches can sort the membership list by attendance, and number of unpaid/paid times.

        - [ ] Give a discount for members higher on the database, and a charge for those on the bottom.

5. **`Member Scheduling System`** :
    - [x] Members can schedule an existing practice session and quickly pay on the application.
    - [ ] Members can update their profiles as well.
6. **`Coach Management System`** :
    - [x] Administrators, particularly the treasurer, have tools to manage coach lists.
    - [x] This ties to #3: **`Expense Tracking Panel`** .
7. **`Coach Scheduling System`** :
    - [x] Coaches have a section to create new classes and manage their sessions.
    - [ ] Coaches can mark their availability for practice sessions.
    - [ ] Administrators have oversight of coach activities, including scheduling, attendance, and performance evaluations.

Quality Assurance and Test Reports

Our plan for testing if our system and our implementation works will consist of 5 different types of tests:

1. User Authentication
2. Finance Tracking
3. Membership Logging
4. Message Platform
5. Database Tracking

These tests will ensure that our product in mind is to the client's expectation.

Installation Process

To clone this application, you'll need:

- Git
- Node.js (which comes with npm)
- Python

Then, from your command line:

```
# Clone this repository
$ git clone https://github.com/andrearcaina/vivid.git

# Go into the backend folder
$ cd vivid/server

# install virtual environment
$ python3 -m venv .venv # for linux
> py -3 -m venv .venv   # for windows

# make sure you activate virtual environment
$ source .venv/bin/activate # for linux
> .venv\Scripts\activate # for windows

# install necessary dependencies
$ pip install -r requirements.txt

# go to the frontend folder
$ cd ../client

# install necessary packages
$ npm install # using npm
$ bun install # using bun
$ pnpm install # using pnpm
$ yarn install # using yarn

# run the development server and go to localhost:3000
$ npm run dev
$ bun run dev
$ pnpm run dev
$ yarn run dev

# run the server
$ cd ../server
$ python manage.py runserver
```