Project 3 Architecture Document

Team Members: Connor Anderson, Mohamed Ashraq, Owen Berkholtz, Matthew Eagleman, Bryson Toubassi

Synopsis: A web application that helps students build and save valid schedules around the courses they want to take.

Architecture Description:

Students shall be able to create/manage their own schedules using a full-stack web application. The schedule maker can be broken down into three primary components: frontend, backend, and database. The following is a description of those three components, beginning with the bottom layer, the database, and moving upward to the frontend.

Database (SQLite): The SQLite database is at the very heart of the application. It stores all of the key data necessary for the system to function. There shall be three data models saved in the database: course data, schedule data, and user data. Course data contains all of the relevant information about a course for the user. This can include course title, course number, meeting times, instructor, etc. Users may also add custom courses that do not already exist in the database. The schedule data model will keep track of the user's schedules. A schedule may be saved, modified, or deleted at the whim of the user. The final data model is the user data. This stores all information related to the user, such as their username, password, and any preferences. The backend utilizes Django, which enables communication with SQLite. Django does this by using its Object-Relational-Mapper (ORM) to convert SQLite data models into Pythonic objects.

Backend (Django): Our application's backend will be built using the Django framework. It will interact with both the React frontend and the SQLite database. As mentioned earlier, it will manage the tables stored in the

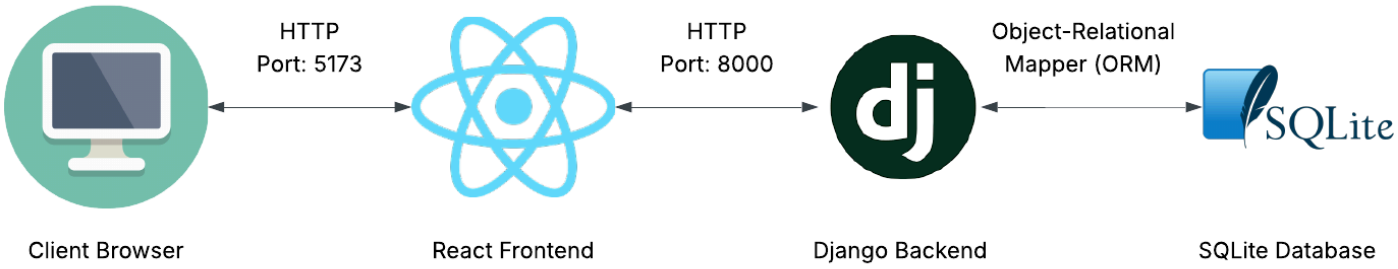
database as models. Each class in the models.py file shall map to a table using the ORM. Whenever the user interacts with the frontend, React will send an HTTP request to some Django API endpoint. Once Django handles the request, it sends a JSON response back to the frontend. The user must be able to search for courses they wish to add to their schedule. Django will query the database and find any matches to the user's search. In addition, the server shall also keep a list of Selected Courses and detect if there are any scheduling conflicts. Users will also be able to add and remove courses from the Selected Courses list. When the user selects a number of courses, all valid schedules will be generated. The user may save any of these schedules and then load them. Django shall also handle user authentication/authorization. New users should create their own username and password. An existing user should be able to log into their account by entering the correct credentials. There will be two kinds of users: Students and Admins. Students will be able to save and load their schedules; they can also add their own custom courses. Admins have all the powers of a student, but in addition, can add/remove things from the database.

Frontend (React): The final layer will be the frontend using React. This is the means by which the user can interact with the web application through their browser. The UI should send a request to the backend (Django) and re-render itself according to the JSON response it receives. There will be three main pages (URLs) on this web app: login, dashboard, and schedule-maker. When the user runs the application, the first page to appear must be the login screen. When the user enters the correct username and password, a request is sent to direct the user to the dashboard. Otherwise, display an error message. The dashboard will display the user's saved schedules. They may create a new schedule or modify an existing one. Creating/making a new schedule will send the user to the schedule maker. The user can search the database for courses or add custom ones. Selected courses will be shown in a list as well as on a weekly calendar. Users may page through all valid schedules in the calendar. There will be buttons allowing schedules to be saved, reset, or

exported. Certain courses can be blacklisted or pinned to the schedule. Schedules can be given custom names. At any point, users shall be able to log out, open the settings panel, or return to the dashboard.

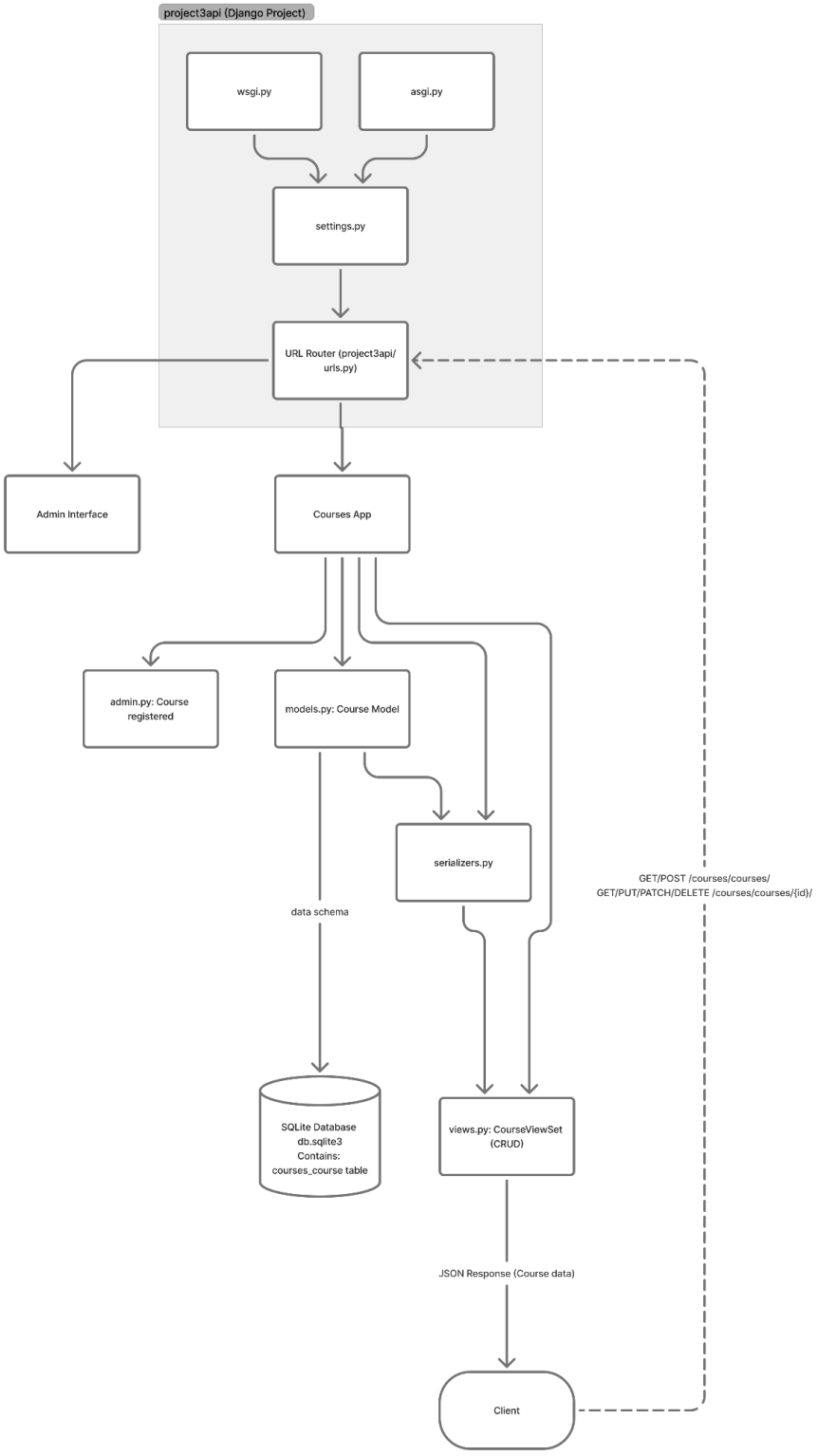
UML Diagrams:

High-Level Project Diagram:



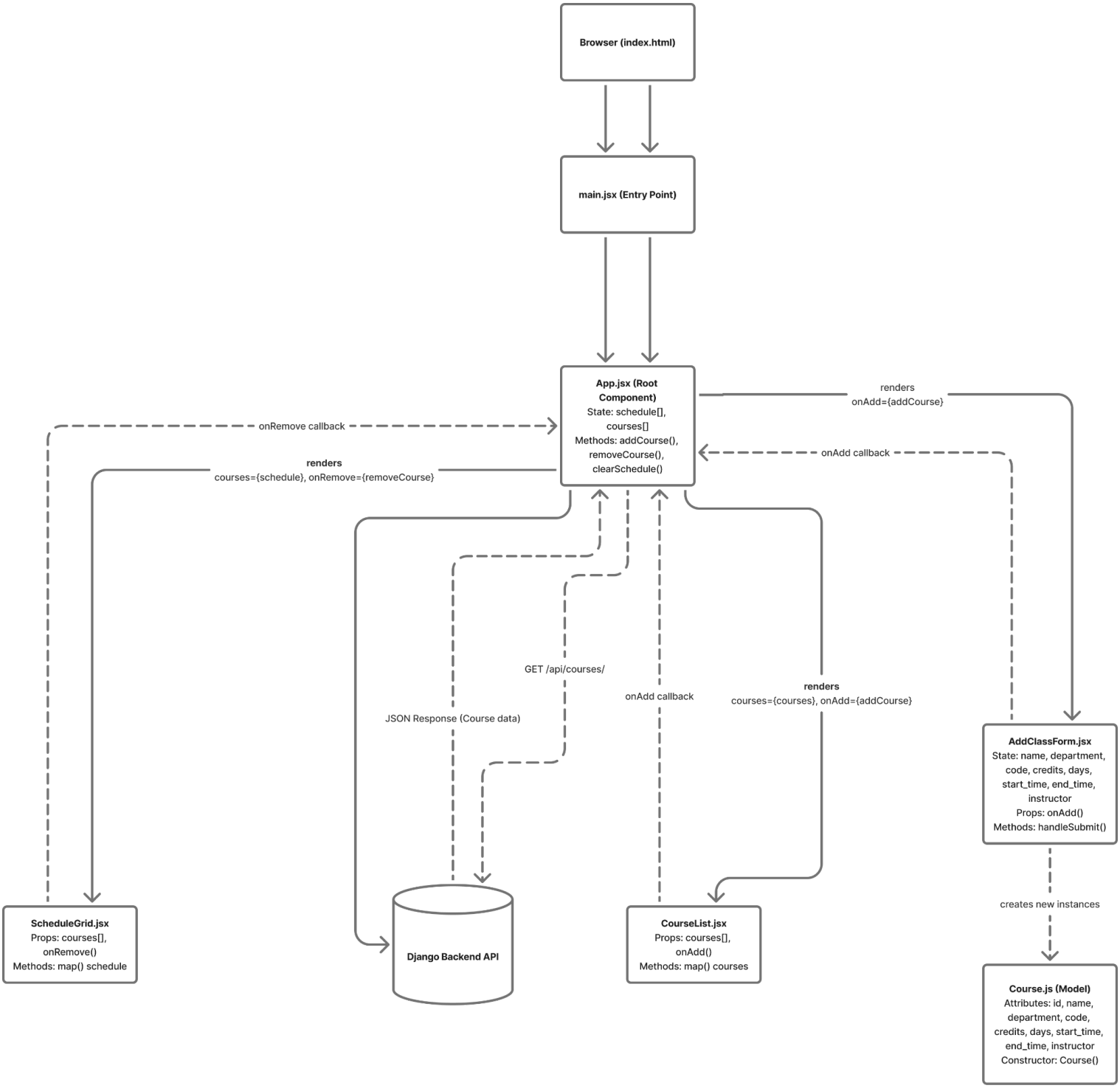
Group 4

Backend Diagram



Group 4

Frontend Diagram



Group 4

Use Case Diagram