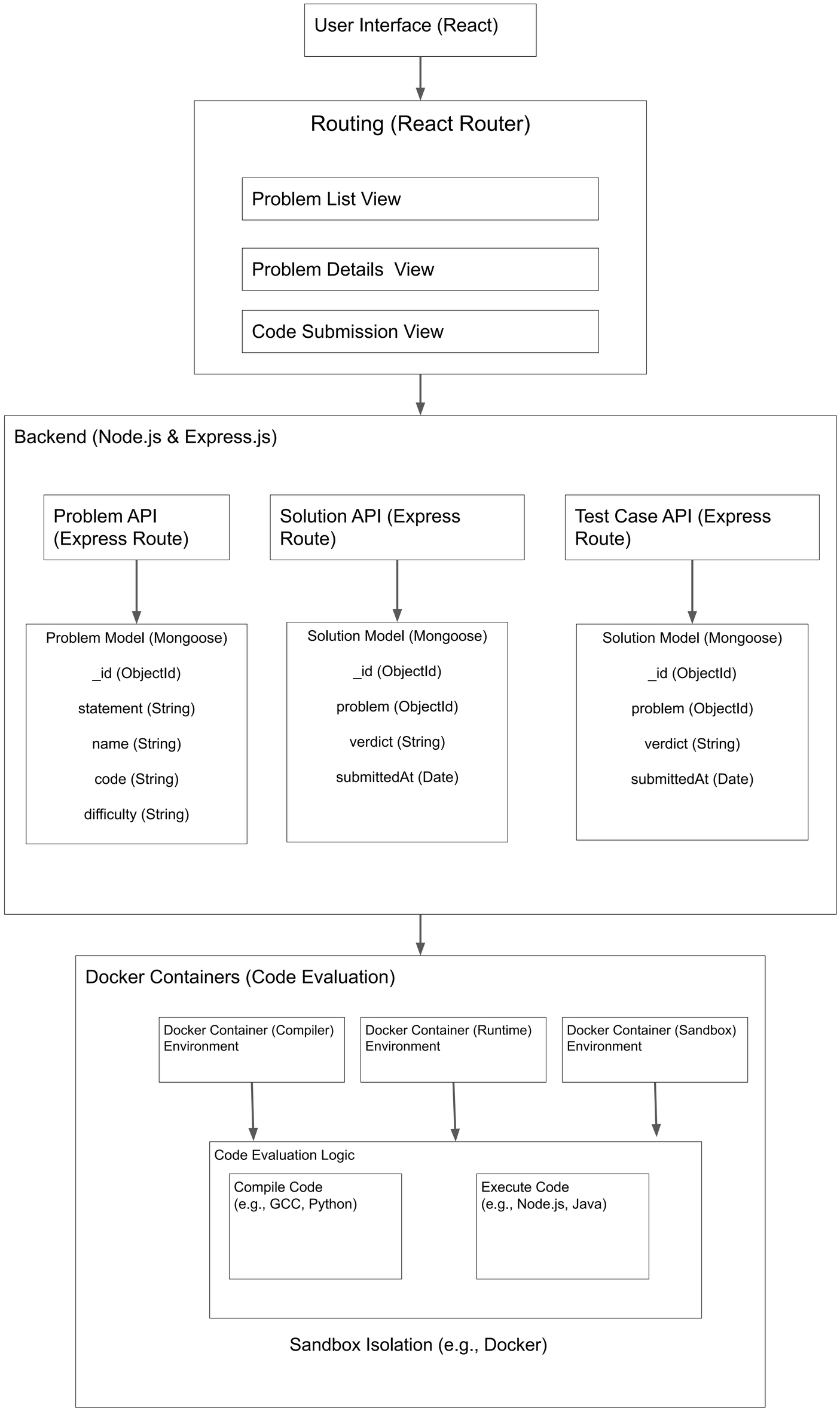
User Interface (React)
Routing (React Router)
Problem List View
Problem Details View
Code Submission View
Backend (Node.js & Express.js)
Problem API (Express Route)
Solution API (Express Route)
Test Case API (Express Route)
Problem Model (Mongoose)
_id (ObjectId)
statement (String)
name (String)
code (String)
difficulty (String)
Solution Model (Mongoose)
_id (ObjectId)
problem (ObjectId)
verdict (String)
submittedAt (Date)
Solution Model (Mongoose)
_id (ObjectId)
problem (ObjectId)
verdict (String)
submittedAt (Date)
Docker Containers (Code Evaluation)
Docker Container (Compiler) Environment
Docker Container (Runtime) Environment
Docker Container (Sandbox) Environment
Code Evaluation Logic
Compile Code (e.g., GCC, Python)
Execute Code (e.g., Node.js, Java)
Sandbox Isolation (e.g., Docker)

High-Level Design Overview:

This high-level design represents an Online Judge application built using the MERN stack (MongoDB, Express.js, React, and Node.js). The design encompasses the user interface, routing, backend API, database, and code evaluation system using Docker containers.

Key Components and Flow:

1. User Interface (React):

- The application's frontend is built using React.js, providing a dynamic and interactive user interface.

- Users interact with the UI to view problem lists, problem details, and submit their code solutions.

2. Routing (React Router):

- React Router handles client-side routing, enabling users to navigate between different views (Problem List, Problem Details, Code Submission) without full page reloads.

3. Express.js Backend:

- The Express.js backend serves as the intermediary between the frontend and the database. It handles API routes and controllers for various functionalities.

4. Database (MongoDB):

- MongoDB stores problem data, user information, and code submissions.

- Three primary collections (tables) in the database are represented: `Problem`, `Solution`, and `Test Cases`.

5. Problem Model (Mongoose):

- Represents the schema for the `Problem` collection in MongoDB.

- Fields include `_id` (automatically generated), `statement` (problem statement), `name` (problem name/title), `code` (problem code), and `difficulty` (problem difficulty level).

6. Solution Model (Mongoose):

- Represents the schema for the `Solution` collection in MongoDB.

- Fields include `_id` (automatically generated), `problem` (reference to the corresponding problem), `verdict` (result/verdict of the solution), and `submittedAt` (timestamp of submission).

7. Test Cases Model (Mongoose):

- Represents the schema for the `Test Cases` collection in MongoDB.

- Fields include `_id` (automatically generated), `input` (input test case data), `output` (expected output for the test case), and `problem` (reference to the corresponding problem).

8. Code Submission (UI):

- Users can submit their code solutions through the UI, selecting a specific problem.

9. Docker Containers (Code Evaluation):

- Docker containers are used to isolate and execute user-submitted code securely.
- Three containers are depicted: Compiler Environment, Runtime Environment, and Sandbox Environment.
- These containers are used for code compilation, execution, and sandboxing, respectively.

10. Code Evaluation Logic:

- Within the Docker containers, code evaluation logic is implemented.
- Compilation of code is performed using compilers such as GCC or interpreters like Python.
- Execution of code is carried out in runtime environments like Node.js or Java.
- Sandbox isolation ensures the security and isolation of code execution.

Flow of Interactions:

1. Users interact with the React-based User Interface, navigating between problem lists, individual problem views, and code submission forms using React Router.
2. The Express.js backend serves as the bridge between the frontend and the database, providing API routes and controllers.
3. The MongoDB database stores problem data, code submissions, and test cases.
4. When users submit code solutions, Docker containers are used for code evaluation in a secure and isolated environment.
5. Code evaluation logic within the Docker containers compiles and executes user-submitted code, capturing the results.
6. The results of code evaluations, including the verdict (pass/fail), are stored in the MongoDB database.

This high-level design and flow illustrate the core components and interactions within an Online Judge application. Users can access and interact with problems, submit code solutions, and receive verdicts on their submissions. The code evaluation system, utilizing Docker containers, ensures the security and reliability of code execution.