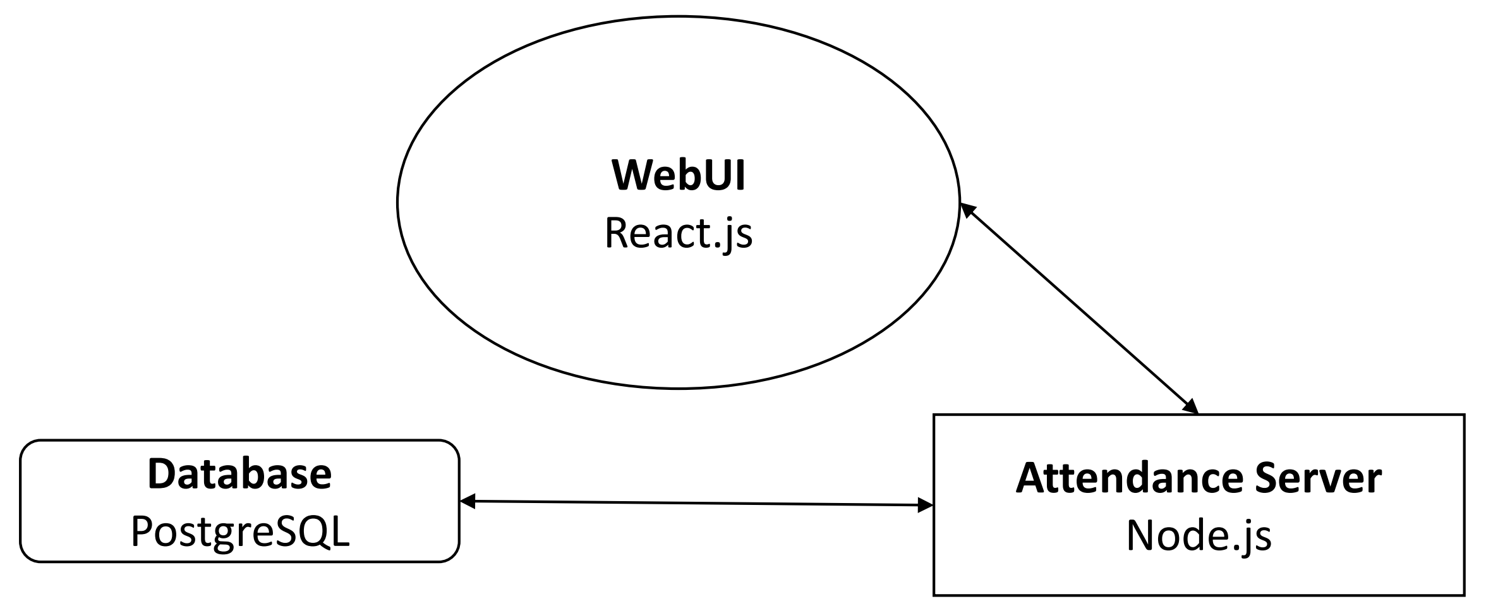
A10Dance Tracker

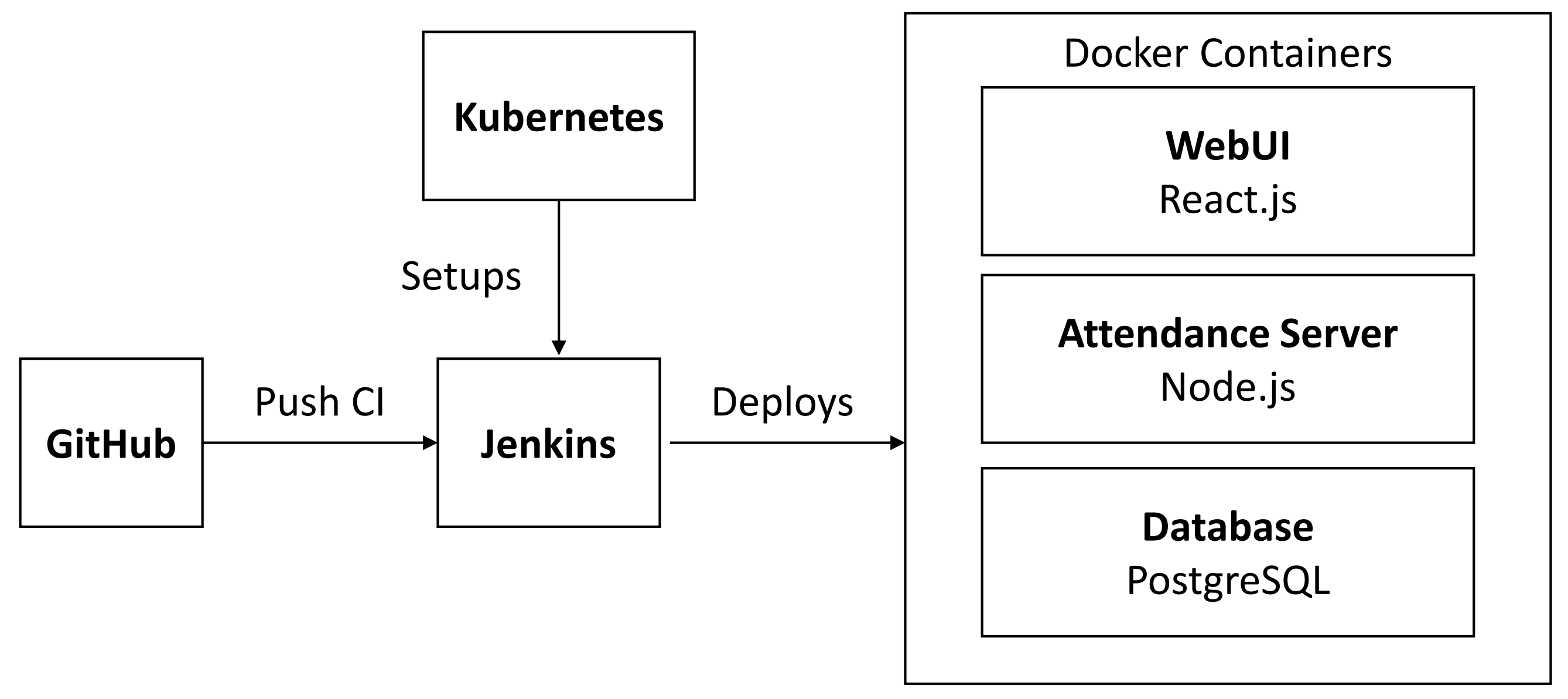
By: Josue Camilo, Danielle Rivas,
Kyara Scott, Long Vu, Jackson Wertz
https://github.com/josuecamilo/Attendance

Project Architecture



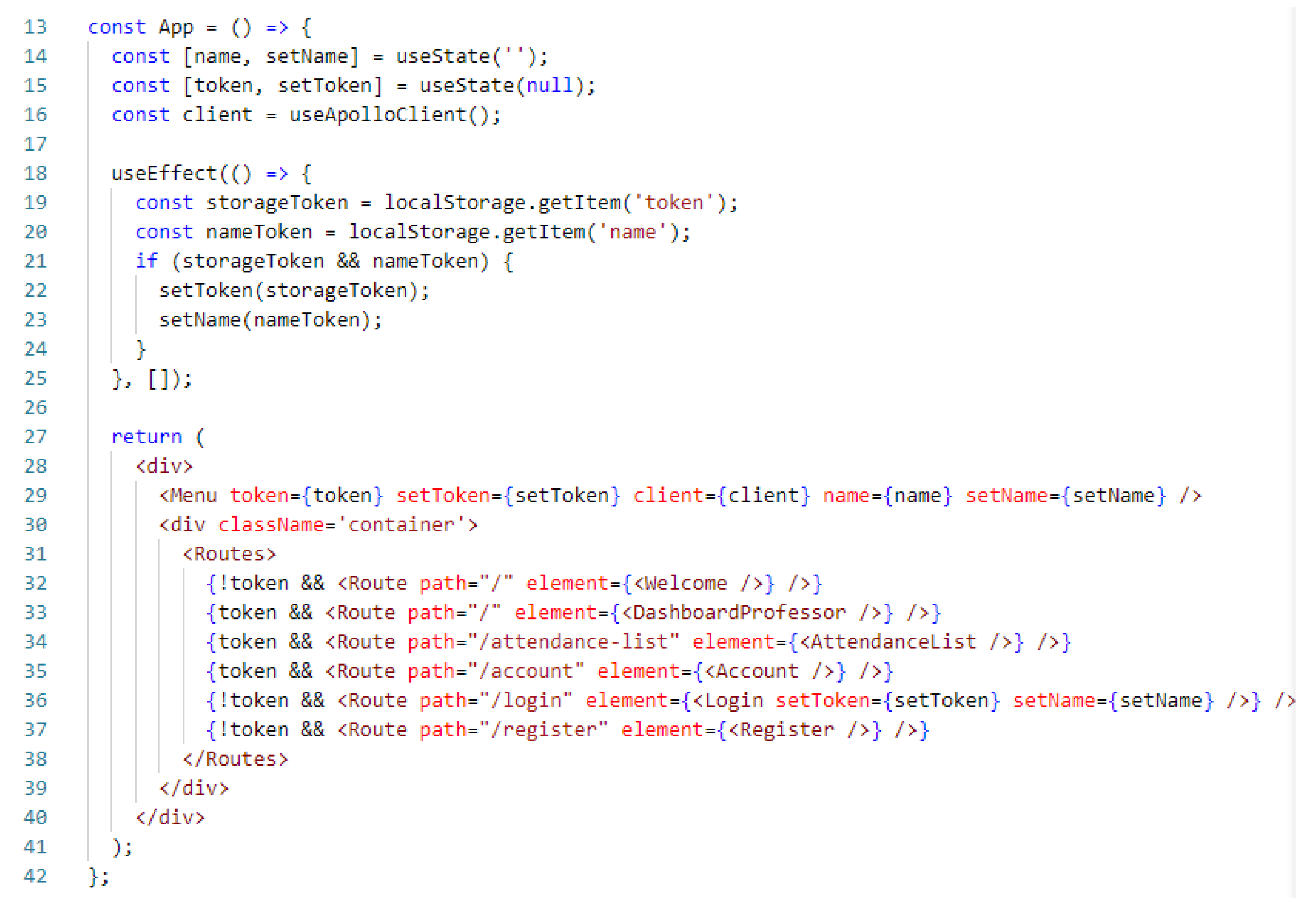
- The graph demonstrates how the individual components of the program works together.
- The database contains user information, classes, and attendance statistics for each student. The attendance server handles accessing and modifying information from the database. The WebUI can query the attendance server for information and tell it to make a modification to the database.

Cloud Deployment



- Runs using CloudLab servers
- Using a combination of Docker and Kubernetes
- Jenkins:
  - Deployment on Kubernetes
  - Integration with GitHub
  - Used for continuous integration in order to automatically push out changed content to users

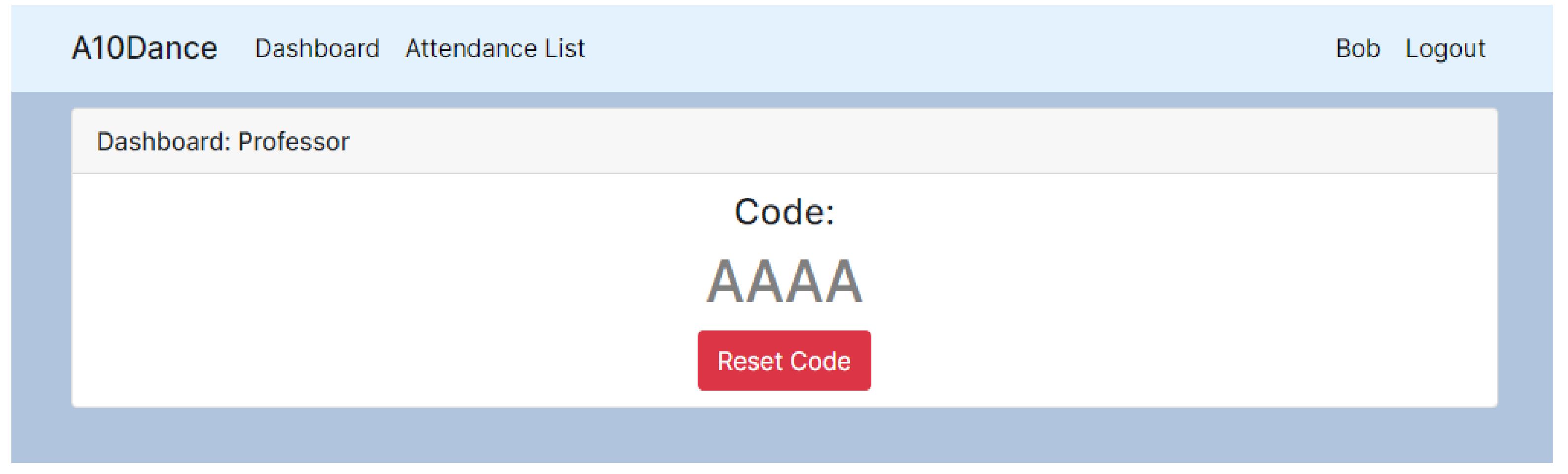
Attendance Server



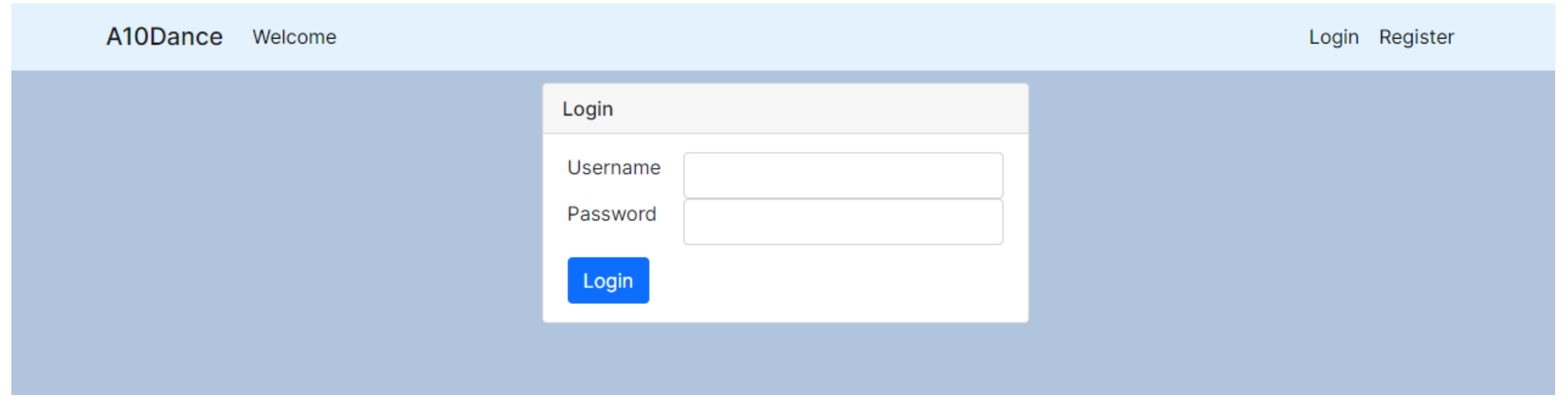
```
const App = () => {
  const [name, setName] = useState('');
  const [token, setToken] = useState(null);
  const client = useApolloClient();

  useEffect(() => {
    const storageToken = localStorage.getItem('token');
    const nameToken = localStorage.getItem('name');
    if (storageToken && nameToken) {
      setToken(storageToken);
      setName(nameToken);
    }
  }, []);

  return (
    <div>
      <Menu token={token} setToken={setToken} client={client} name={name} setName={setName} />
      <div className='container'>
        <Routes>
          {!token && <Route path="/" element={<Welcome />} />}
          {token && <Route path="/" element={<DashboardProfessor />} />}
          {token && <Route path="/attendance-list" element={<AttendanceList />} />}
          {token && <Route path="/account" element={<Account />} />}
          {!token && <Route path="/login" element={<Login setToken={setToken} setName={setName} />} />
          {!token && <Route path="/register" element={<Register />} />}
        </Routes>
      </div>
    </div>
  );
};
```

- The backend receives login information from the WebUI and return queries asked for after identification.
- GraphQL is used to handle queries between the WebUI and attendance server.
- If a user has logged in, the attendance server will allow them to access specific queries depending on their role like generating a code if they are a professor or entering a code for students.

WebUI



- The screenshot shows a demo of the professor dashboard displaying an attendance code which students can use to check-in to the class.



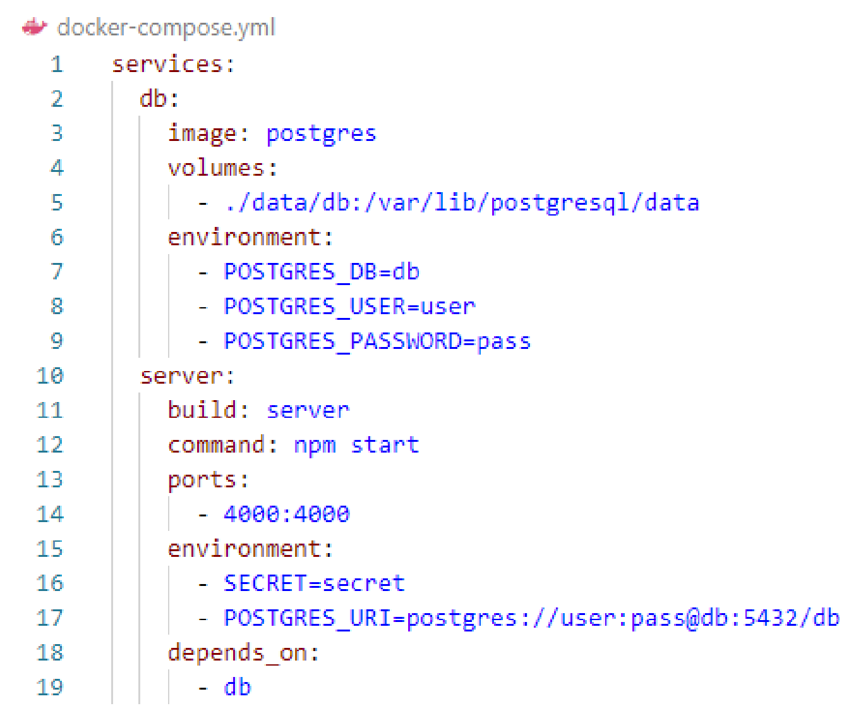
- Students and professors can create new accounts and login to their created one.
- Students will be able to input a code generated by the professor to check-in.
- The username and password inputted will be sent to the attendance server which will query the database for a valid user and return if the user is valid with correct credentials or warn if they entered an invalid password or username.
- The front end is written using the following languages and libraries:
  - React.js/JSX
  - CSS
  - JavaScript
  - Bootstrap

Docker

```
FROM node:16
WORKDIR /server
COPY package*.json ./
RUN npm install
# RUN npm ci --only=production
COPY . .
```

Web Attendance Dockerfile

- Docker builds and runs components required by the project
- Dockerfile created for running Node/NGINX



docker-compose.yml

```
services:
  db:
    image: postgres
    volumes:
      - ./data/db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=db
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
  server:
    build: server
    command: npm start
    ports:
      - 4000:4000
    environment:
      - SECRET=secret
      - POSTGRES_URI=postgres://user:pass@db:5432/db
    depends_on:
      - db
```

Docker Compose File

- Docker Compose ties different parts together
  - NGINX for hosting the WebUI
  - Node.js for Attendance Server
  - PostgreSQL database