Seyed Ahmad Farsad

Task Management Project Overview

This Task Management project is a simple web-based application developed using Django for the backend and React for the frontend. The application allows users to create and view tasks, which are stored in an SQLite database. The application follows a 3-tier architecture, where the React frontend communicates with the Django backend, and the backend interacts with the SQLite database to store and retrieve tasks.

How to Run the Project

1. **Set Up the Backend (Django)**

   - Navigate to the backend project directory:

     ```
     cd simple_backend
     ```

   - Install the required Python packages:

     ```
     pip install -r requirements.txt
     ```

   - Apply the database migrations to set up the SQLite database:

     ```
     python manage.py migrate
     ```

   - Run the Django development server:

     ```
     python manage.py runserver
     ```

   - The backend server will start running at http://127.0.0.1:8000/.

2. **Set Up the Frontend (React)**

   - Navigate to the frontend project directory:

     ```
     cd simple_frontend
     ```

   - Install the required Node.js packages:

     ```
     npm install
     ```

   - Start the React development server:

     ```
     npm start
     ```

- The frontend server will start running at http://localhost:3000/.

3. Access the Application

- Open a web browser and navigate to http://localhost:3000/ to interact with the Task Management application.

How to Check the Database

1. Access the SQLite Database

- The SQLite database file is located in the simple_backend directory as db.sqlite3.
- You can open this file using a SQLite viewer such as DB Browser for SQLite.
- In the viewer, navigate to the api_task table to view the tasks that have been created.

2. Checking Data

- You can manually inspect the tasks in the api_task table or run SQL queries to retrieve and manipulate the data.

How to Test the Application

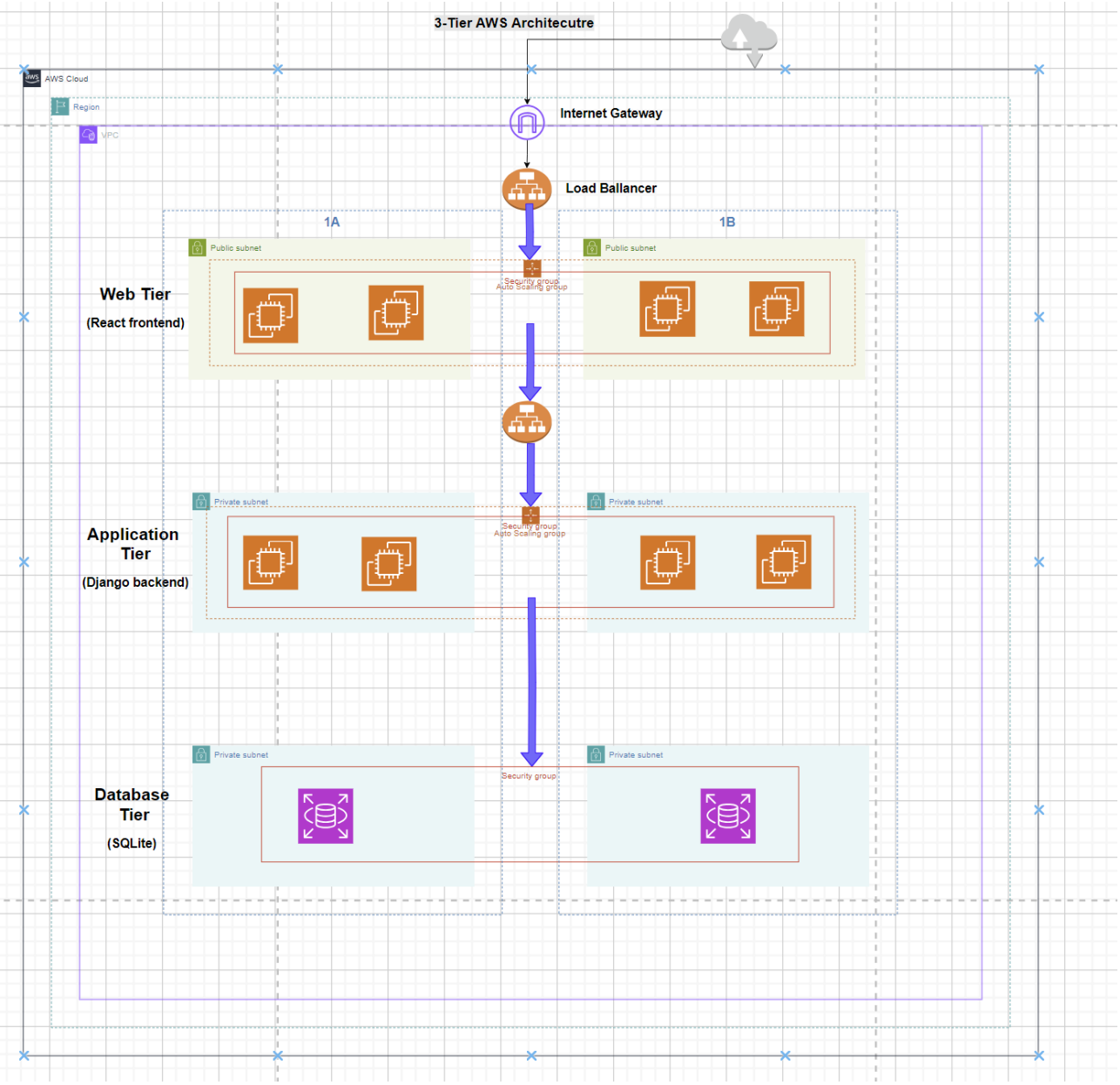
1. Testing Task Creation (from React)

- **Step 1:** Create a new task using the React frontend.
- **Step 2:** Once the task is created in React, go to the Django REST framework page (typically at http://127.0.0.1:8000/api/tasks/) and click the GET button.
- **Step 3:** Verify that the new task appears in the list on the Django REST framework page.
- **Step 4:** Optionally, check the api_task table in the SQLite database to ensure the task was added successfully.

2. Testing Task Creation (from Django REST Framework Interface)

- **Step 1:** On the Django REST framework page, create a new task using the POST section.
- **Step 2:** After submitting the task, switch to your React frontend and refresh the page.
- **Step 3:** Confirm that the newly created task from the Django REST interface is displayed in the React task list.

3-tier AWS Architecutre Diagram



1. **Web Tier (React Frontend)**: This is where the user interacts with the task management application. The React frontend handles the user interface and communicates with the backend (Django).

2. **Application Tier (Django Backend)**: This is where the business logic and data processing happen. The Django backend processes requests from the frontend and interacts with the database.

3. **Database Tier (SQLite)**: This is where all the task data is stored. SQLite is a lightweight database that works well for development and small-scale applications.

4. **Load Balancer**: In a real-world deployment on AWS, a load balancer would distribute incoming requests to multiple instances of the web and application servers. This ensures that the application can handle more traffic and provides fault tolerance.

5. **Auto Scaling Group**: Auto scaling helps to automatically increase or decrease the number of EC2 instances in response to traffic patterns. This helps to maintain performance and minimize costs.

6. **Public and Private Subnets**: Public subnets are used for components that need to be accessible from the internet (like your React frontend), while private subnets are used for components that should not be directly accessible from the internet (like your Django backend and SQLite database).

7. **Security Groups**: These act as virtual firewalls to control inbound and outbound traffic for the instances. They are crucial for securing the application.

Git:

https://github.com/ahmad-farsad-cs/Web-Site-roulettech.git

Thank you