Starting the Application

Start the application using the `run.sh` script.

Make the Script Executable:

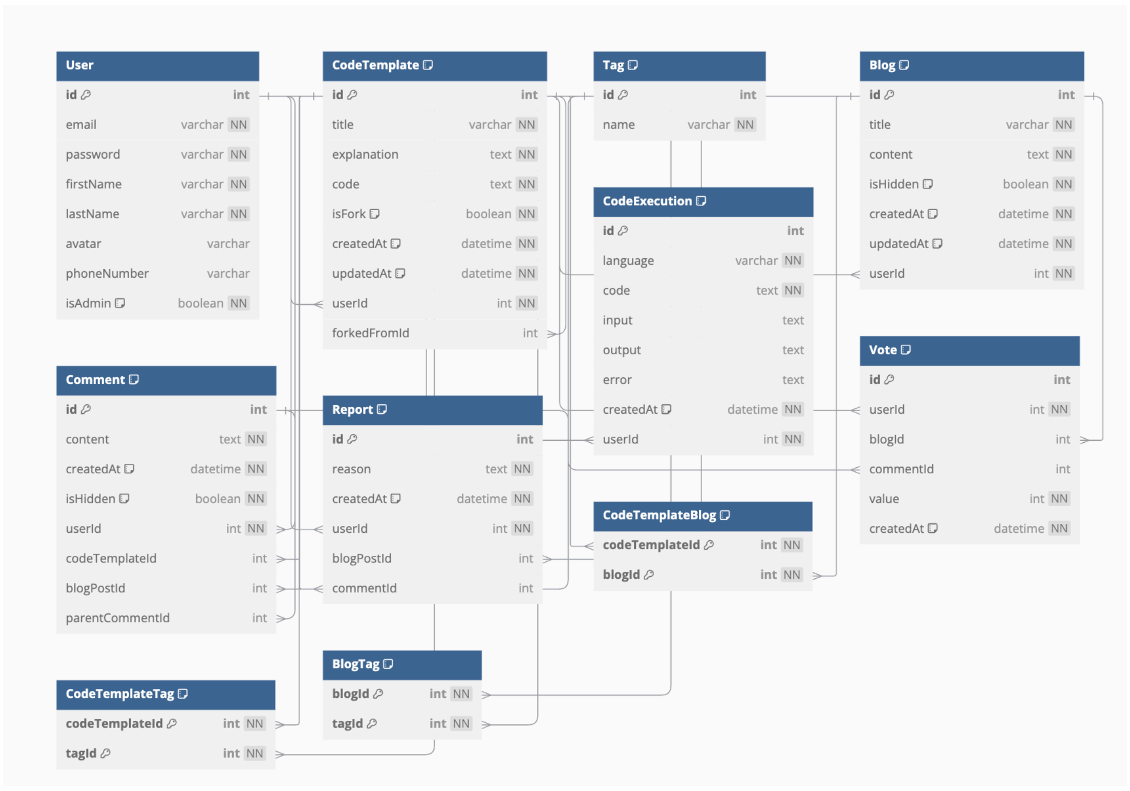
```
chmod +x startup.sh
chmod +x run.sh
```

Execute the Script:

```
./startup.sh
./run.sh
```

The application will start in development mode and can be accessed at http://localhost:3000.

ER Diagram



API Endpoints

Users

- **Sign Up**
    - **Endpoint**: `POST /api/users/signup`
    - **Description**: Registers a new user.

Request Body:
```
{
  "email": "user@example.com",
  "password": "Password123!",
  "firstName": "John",
  "lastName": "Doe",
  "avatar": "avatar.jpg",
  "phoneNumber": "123-456-7890",
  "isAdmin”: true
}
```

- **Login**
    - **Endpoint**: `POST /api/users/login`
    - **Description**: Authenticates a user and returns a JWT token.

Request Body:
```
{
  "email": "user@example.com",
  "password": "Password123!"
}
```

- **Edit user**
    - **Endpoint**: `POST /api/users/[id]`
    - **Description**: Updates user information

Request Body:
```
{
  "email": "user@example.com",
  "password": "Password123!"
}
```

- **Logout user**
    - **Endpoint**: `POST /api/users/logout`
    - **Description**: Invalidates user token

Request Body:
```
{
      "refreshToken": "{{accessToken}}"
}
```

Blogs

- **Create blog**
    - **Endpoint**: `POST /api/blogs`
    - **Description**: Creates new blog
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "title": "New Blog",
    "content": "Hello world, this is a newly created blog!"
}
```

- **Update blog**
    - **Endpoint**: `POST /api/blogs/[id]`
    - **Description**: Updates blog
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "title": "Title of newly updated blog",
    "content": "Newly updated content!"
}
```

- **Delete blog**
    - **Endpoint**: `DELETE /api/blogs/[id]`
    - **Description**: Deletes blog
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

- **View blogs sorted**
    - **Endpoint**: `GET /api/blogs`
    - **Description**: View blogs sorted by votes

Comments

- **Create comment**
    - **Endpoint**: `POST /api/comments`
    - **Description**: Creates new comments. BlogId for comment, commentId for reply
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "content": "Hello world, this is a newly created comment!",
    “blogId or commentId”: 1
}
```

- **Update comment**
    - **Endpoint**: `POST /api/comment/[id]`
    - **Description**: Updates comment
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "comment": "Updated comment",
    "blogId or commentId": 1
}
```

- **Delete comment**
    - **Endpoint**: `DELETE /api/comment/[id]`
    - **Description**: Deletes comment
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Reports

- **Create/edit report**
    - **Endpoint**: `POST /api/blogsORcomments/[id]/report`
    - **Description**: Reports blog or comment. Create if DNE, edit if exists
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "blogIdORcommentId": 1,
    "reason": "This is a test report on a bad blog!"
}
```

- **Delete report**
    - **Endpoint**: `DELETE /api/blogsORcomments/[id]/report`
    - **Description**: Deletes comment
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Votes

- **Create/edit votes**
    - **Endpoint**: `POST /api/blogsORcomments/[id]/vote`
    - **Description**: Vote on blog or comment. Create if DNE, edit if exists
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "blogIdORcommentId": 1,
    "voteValue": 1
}
```

- **Delete vote**
    - **Endpoint**: `DELETE /api/blogsORcomments/[id]/vote`
    - **Description**: Deletes vote
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Admin

- **View reported content**
    - **Endpoint**: `GET /api/admin`
    - **Description**: View blogs and comments with reports (sorted)
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

- **Hide blog/comment**

    - **Endpoint**: `POST /api/blogsORcomments/[id]`
    - **Description**: Hides blog/comment
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
    "isHidden": true
}
```

Code Templates

- **Create a Template**
    - **Endpoint**: `POST /api/templates`
    - **Description**: Creates a new code template.
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:
```
{
  "title": "Sample",
  "explanation": "sample code template",
  "code": "console.log('Hello, World!');",
  "tags": ["sample", "test"],
  "isFork": false
}
```

    -
- **Get All Templates**
    - **Endpoint**: `GET /api/templates`
    - **Description**: Retrieves all code templates with optional filters.
    - **Query Parameters**:
        - `search`: Search term.
        - `tags`: Comma-separated tags.
        - `page`: Page number.
        - `limit`: Items per page.
- **Get a Template by ID**
    - **Endpoint**: `GET /api/templates/{id}`
    - **Description**: Retrieves a single code template by ID.
- **Update a Template**
    - **Endpoint**: `POST /api/templates/{id}`
    - **Description**: Updates an existing code template.
    - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:

```
{
  "title": "Updated Title",
  "explanation": "Updated explanation.",
  "code": "console.log('Updated code');",
  "tags": ["updated", "example"]
}
```

- **Delete a Template**
  - **Endpoint**: `DELETE /api/templates/{id}`
  - **Description**: Deletes a code template.
  - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`
- **Fork a Template**
  - **Endpoint**: `POST /api/templates/{id}/fork`
  - **Description**: Forks an existing code template.
  - **Headers**: `Authorization: Bearer YOUR_JWT_TOKEN`

Request Body:

```
{
  "title": "Forked Template Title",
  "explanation": "This is a forked version.",
  "code": "// Modified code...",
  "tags": ["forked", "example"]
}
```

Code Execution

- **Execute Code**
  - **Endpoint**: `POST /api/execute`
  - **Description**: Executes code in a specified programming language.

Request Body:

```
{
  "language": "javascript",
  "code": "console.log('Hello, World!');",
  "input": ""
}
```

  -
  - **Supported Languages**: `javascript`, `python`, `java`, `c`, `cpp`

Testing the API

Use a tool like Postman to test the API endpoints. Ensure you include the `Authorization` header with the JWT token for protected routes.

Example: Testing Code Execution

- **Endpoint**: `POST /api/execute`

Request Body:

```
{
  "language": "python",
  "code": "print('Hello, World!')"
}
```

-

Expected Response:

```
{
  "output": "Hello, World!\n"
}
```
