Project 1: Ticketing System

Features

1. User and Admin Accounts:
   - Users can register and log in with a username and password.
   - Admin accounts have advanced privileges to manage tickets.

2. User Capabilities:
   - Create tickets with a title and detailed description.
   - Update existing tickets as needed.
   - View the status of all tickets submitted by them.

3. Admin Capabilities:
   - Update ticket statuses (e.g., Open, In Progress, Resolved, Closed).
   - Add comments to tickets to share updates or notes.
   - Remove tickets as necessary.

Implementation Outline

- Authentication:
  - Secure registration and login.
  - Use JWT for token-based authentication and role-based access control.

- Ticket Management:
  - User dashboard for creating, updating, and viewing tickets.
  - Admin dashboard for ticket management (status updates, comments, and deletion).

- Database Schema (MongoDB Collections):
  - Users Collection:
    ```json
    {
      "_id": "ObjectId",
      "username": "string",
      "password": "string (hashed)",
      "role": "string (user/admin)"
    }
    ```
  - Tickets Collection:
    ```json
    {
      "_id": "ObjectId",
      "user_id": "ObjectId",
      "title": "string",
      "description": "string",
      "status": "string (Open/In Progress/Resolved)",
      "comments": "array of strings",
    ```

```
  "created_at": "timestamp"
}
```

Project 2: Single Seller E-Commerce Application

Features

1. User and Admin Accounts:
   - Users can register, log in, and shop for products.
   - Admins manage products and view orders.

2. User Capabilities:
   - Browse and view available products.
   - Add products to their cart and view the total price.
   - Place orders from their cart.

3. Admin Capabilities:
   - Add, edit, and delete products from the store.
   - View and manage all user orders.

Implementation Outline

- Authentication:
  - Use JWT for user authentication and role-based access control.

- Product and Order Management:
  - A user interface for browsing products, managing the cart, and placing orders.
  - Admin dashboard for managing product inventory and viewing orders.

- Database Schema (MongoDB Collections):
  - Users Collection:
    ```json
    {
      "_id": "ObjectId",
      "username": "string",
      "password": "string (hashed)",
      "role": "string (user/admin)"
    }
    ```
  - Products Collection:
    ```json
    {
      "_id": "ObjectId",
      "name": "string",
      "description": "string",
      "price": "number",
      "stock": "number",
      "created_at": "timestamp"
    }
    ```

```
- Orders Collection:
```json
{
  "_id": "ObjectId",
  "user_id": "ObjectId",
  "products": [
    {
      "product_id": "ObjectId",
      "quantity": "number"
    }
  ],
  "total_price": "number",
  "created_at": "timestamp"
}
```
```

Project 3: Reddit Clone

Features
1. User Accounts:
  - Users can register, log in, and manage their posts.

2. Post and Comment Management:
  - Users can create, edit, and delete text-based posts.
  - Users can upvote or downvote posts.
  - Users can add, edit, and delete comments on posts.

3. Engagement Metrics:
  - Posts and comments display engagement scores based on votes.

Implementation Outline
- Authentication:
  - Secure registration and login with JWT.

- Post and Comment Features:
  - Feed displaying posts, sorted by engagement metrics.
  - Post and comment editing and deletion functionality.

- Database Schema (MongoDB Collections):
  - Users Collection:
    ```json
    {
      "_id": "ObjectId",
      "username": "string",
      "password": "string (hashed)"
    }
    ```
  - Posts Collection:
    ```json
    {
      "_id": "ObjectId",
      "user_id": "ObjectId",
      "content": "string",
      "votes": "number",
      "comments": [
        {
          "comment_id": "ObjectId",
          "user_id": "ObjectId",
          "content": "string",
          "votes": "number"
        }
    ```

```
  ],
  "created_at": "timestamp"
}
```

General Implementation Tips

- Use Node.js with Express for the backend.
- Use Mongoose for MongoDB schema definitions and operations.
- Build a clean and responsive frontend with React or Vanilla JavaScript.
- Test APIs using tools like Postman to ensure smooth functionality.
- Deploy applications using Heroku, Netlify, or AWS.