**1. Online Booking and Reservation System**

**Functional Requirements:**

1. Users should be able to register and log in with JWT-based authentication.
2. Users should be able to search for available services (hotels, appointments, flights, etc.).
3. Users should be able to book available slots for a service.
4. Admins should be able to manage bookings (CRUD operations for reservations).
5. Users should receive email confirmations after successful booking (using Node.js mailer).
6. Users should be able to view and cancel their bookings.
7. System should provide a dashboard to view booking statistics (number of bookings, types of services, etc.).

**Non-Functional Requirements:**

1. The system should handle at least 1000 concurrent users.
2. Responsive design for mobile and desktop views.
3. Data should be stored in MongoDB for bookings and users.
4. Implement CSRF protection for secure requests.
5. Integration with third-party payment systems (Stripe/PayPal).
6. API should have detailed logging for debugging and monitoring.

**2. E-Learning Platform**

**Functional Requirements:**

1. Users can register as students or instructors.
2. Instructors can create, update, and delete courses (CRUD operations).
3. Students can enroll in courses and view their progress.
4. Implement video upload for course lessons (using Firebase Storage).
5. Instructors can monitor students' progress and view course statistics.
6. System should support quizzes and assignments for courses.
7. Students should be able to review and rate courses.

**Non-Functional Requirements:**

1. The platform should be scalable to handle multiple courses and user interactions.
2. Use WebSockets (Socket.IO) for real-time notifications on course updates or assignments.
3. Store user data and course details in MongoDB.
4. Implement user authentication and access control using JWT.
5. Frontend should be fully responsive, compatible with mobile and desktop.
6. Optimize media files (videos, images) for faster loading times.

**3. Building a Social Media Dashboard with React**

**Functional Requirements:**

1. Users can register, log in, and manage their profiles.
2. Users can create, edit, and delete posts.
3. Implement a news feed that displays posts from friends/followed users.
4. Users can like, comment, and share posts.
5. Implement search functionality to find users and posts.
6. Admins can monitor user activity and manage posts.
7. System should support real-time notifications for likes, comments, and follows using Socket.IO.

**Non-Functional Requirements:**

1. Implement role-based access control (admin, user).
2. Responsive UI with a focus on a clean dashboard design.
3. Store all posts and user data in MongoDB.
4. Apply rate limiting on API requests to prevent abuse (e.g., too many posts in a short time).
5. Implement caching for frequently accessed data to improve performance.
6. Use JWT for secure authentication and CSRF protection.

**4. Restaurant Ordering and Delivery Platform**

**Functional Requirements:**

1. Users can browse the restaurant's menu and add items to the cart.
2. Users can place orders for delivery or pickup.
3. System should support real-time order tracking.
4. Admins/restaurant staff can manage orders and update their status.
5. Users should receive notifications when their order is being prepared and out for delivery.
6. Implement payment integration for users to pay online (Stripe/PayPal).
7. Users can rate and review their orders.

**Non-Functional Requirements:**

1. The platform should handle a minimum of 500 orders concurrently.
2. Use MongoDB to store menu items, orders, and user data.
3. Implement real-time updates using Socket.IO (order status).
4. Implement secure payment handling with data encryption.
5. Optimize images of food items to reduce load time.
6. Responsive design for mobile ordering.

**5. Movie Recommendation App using React**

**Functional Requirements:**

1. Users should be able to register, log in, and manage their profiles.
2. Users can browse, search, and view details of movies.
3. System should suggest movie recommendations based on user preferences and viewing history.
4. Users can create a list of their favorite movies.
5. Implement a review and rating system for movies.
6. Users can receive notifications about new movie releases and recommendations.
7. Admins can manage movie data and user reviews.
8. **Users can upload their own movies or video content** (with title, description, genre, etc.).
9. **Users can watch uploaded movies directly on the platform** (streaming functionality).
10. System should support **video categories** like genres or themes (action, comedy, etc.).
11. **Movies should be uploaded to cloud storage** (like Firebase Storage) and fetched via streaming.
12. Implement **video player controls** (play, pause, forward, backward) for watching movies.
13. Users can **like and comment on uploaded movies**.
14. Admins can **moderate uploaded content** to ensure quality and compliance with platform rules.

**Non-Functional Requirements:**

1. API integration with a third-party service (e.g., The Movie Database API) to fetch movie data and recommendations.
2. Implement **JWT authentication** for secure access to user data.
3. Store user data, preferences, and uploaded movie metadata in MongoDB.
4. Use **Socket.IO** for real-time notifications on new movie uploads and releases.
5. Responsive design optimized for **mobile and tablet viewing**.
6. **Implement video streaming** with efficient buffering and playback.
7. **Limit video file size** for uploads and implement progress bars for users during upload.
8. **Ensure proper encoding and optimization of video files** for different devices and internet speeds.
9. Use **CDNs** to optimize video delivery and minimize load times.
10. Implement **caching for frequently requested movie data** to improve performance.
11. Ensure **secure handling of video uploads** to prevent misuse (virus scans or file type validation).
12. Allow users to **report inappropriate or copyright-violating content**.