**ACM 412 Project Ideas**

**Social Media Platform**

Build a social media platform with Django and React JS. The platform should allow users to sign up, log in, and create profiles. Users should be able to post content, follow other users, and interact with each other.

Here are some key features you could include in your social media platform:

* **User authentication:** Users should be able to sign up, log in, and log out. You can use Django's built-in authentication system to handle this.
* **Profile creation:** Users should be able to create profiles and add profile pictures and cover photos.
* **Content creation:** Users should be able to create different types of content, such as posts, images, and videos.
* **Newsfeed:** Users should see a newsfeed that displays posts from the users they follow.
* **Follow system:** Users should be able to follow other users and see their content on their newsfeed.
* **Commenting and liking:** Users should be able to comment on and like posts.
* **Notifications:** Users should receive notifications when someone likes or comments on their post, when they are followed by a new user, or when a user they follow creates new content.
* **Search functionality:** Users should be able to search for other users or specific content.
* **Security:** The application should be secure and protect against common attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

**Recomendation System**

Build a recommendation system with Django and React JS. The system should take input from the user and provide personalized recommendations for movies, music, or books based on their preferences. The system should use machine learning algorithms to learn the user's preferences and make accurate recommendations.

Here are some key features you could include in your recommendation system:

* **User authentication:** Users should be able to sign up, log in, and log out. You can use Django's built-in authentication system to handle this.
* **Input collection:** The system should collect input from the user, such as their favorite movies, music, or books.
* **Machine learning:** The system should use machine learning algorithms to learn the user's preferences and make accurate recommendations. You can use a library like scikit-learn to implement the machine learning algorithms.
* **Recommendation engine:** The system should have a recommendation engine that takes the user's input and provides personalized recommendations for movies, music, or books.
* **User profile:** The system should maintain a user profile that tracks the user's input and recommendations.
* **Integration with external APIs:** The system should integrate with external APIs to get data on movies, music, or books.
* **Security:** The application should be secure and protect against common attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

**Real-Time Chat Application**

Build a real-time chat application with Django and React JS. The application should allow users to sign up, log in, and create chat rooms. Users should be able to join existing chat rooms, send and receive messages in real-time, and receive notifications when new messages arrive.

Here are some key features you could include in your chat application:

* **User authentication:** Users should be able to sign up, log in, and log out. You can use Django's built-in authentication system to handle this.
* **Chat room creation:** Users should be able to create chat rooms and invite other users to join.
* **Direct Messaging:** User should be able to send direct messsage to other member of the system.
* **Real-time messaging**: The application should allow users to send and receive messages in real-time using websockets. You can use a library like Socket.IO to handle this.
* **Notifications:** Users should receive notifications when new messages arrive in a chat room they are a member of.
* **Message history:** The application should maintain a history of messages sent in each chat room.
* **User profile:** The application should maintain a user profile that displays the user's chat room memberships and message history.
* **Security:** The application should be secure and protect against common attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

**Adaptation of Street Animals**

Build a web application with Django and React JS that helps people find and adopt street animals. The application should allow users to search for animals based on their location, breed, age, and gender. Users should be able to view animal profiles, see their pictures, read their stories, and contact the animal shelter to adopt them.

Here are some key features you could include in your street animal adoption application:

* **Animal search:** Users should be able to search for animals based on their location, breed, age, and gender.
* **Animal profiles:** The application should display animal profiles that include pictures, stories, and information about the animal's health and behavior.
* **Contact form:** Users should be able to contact the animal shelter to ask questions and express interest in adopting an animal.
* **Admin panel:** The application should have an admin panel that allows animal shelters to manage their animal profiles and respond to user inquiries.
* **User authentication:** Users should be able to sign up, log in, and log out. You can use Django's built-in authentication system to handle this.
* **Donation system:** The application could include a donation system that allows users to contribute to the animal shelter's efforts.
* **Integration with Google Maps:** The application could integrate with Google Maps to show the user's location and the location of animal shelters.
* **Security:** The application should be secure and protect against common attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

**Application Related Natural Disasters**

Build a web application with Django and React JS that helps people prepare for, respond to, and recover from natural disasters. The application should provide users with up-to-date information on natural disasters, evacuation routes, shelter locations, and disaster recovery resources. Users should be able to search for information based on their location and the type of disaster.

Here are some key features you could include in your natural disasters application:

* **Disaster information:** The application should provide up-to-date information on natural disasters, such as hurricanes, wildfires, earthquakes, and floods. Users should be able to view the latest news, weather alerts, and evacuation orders.
* **Evacuation routes:** The application should provide users with information on evacuation routes and traffic conditions. Users should be able to view real-time traffic updates and receive directions to the nearest evacuation route.
* **Shelter locations:** The application should provide users with information on shelter locations and availability. Users should be able to search for shelters based on their location and the type of shelter they need.
* **Disaster recovery resources:** The application should provide users with information on disaster recovery resources, such as FEMA assistance, insurance claims, and community support services.
* **User authentication:** Users should be able to sign up, log in, and log out. You can use Django's built-in authentication system to handle this.
* **Integration with external APIs:** The application should integrate with external APIs to get real-time data on weather, traffic, and emergency services.
* **Security:** The application should be secure and protect against common attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

**Microservice based e-commerce application**

Build a microservice-based e-commerce application with Django and React JS. The application should consist of several RESTful microservices that communicate with each other to perform various e-commerce functions, such as product catalog management, order management, and payment processing. The application should be highly scalable and fault-tolerant.

Here are some key features you could include in your microservice-based e-commerce application:

* **Product catalog microservice:** This microservice should manage the product catalog, including adding new products, updating existing products, and retrieving product information.
* **Cart microservice:** This microservice should manage the user's shopping cart, including adding and removing items, updating quantities, and calculating the total cost.
* **Order management microservice:** This microservice should manage the user's orders, including creating new orders, updating existing orders, and retrieving order history.
* **Payment processing microservice:** This microservice should handle payment processing, including authorizing payments, capturing payments, and issuing refunds.
* **User authentication microservice:** This microservice should handle user authentication and authorization, including user registration, login, and logout.
* **API gateway:** The application should include an API gateway that sits between the frontend and the microservices, routing requests to the appropriate microservice and handling authentication and authorization.
* **Scalability:** The microservices should be designed to be highly scalable, with the ability to add or remove instances dynamically based on traffic.
* **Fault tolerance:** The microservices should be designed to be fault-tolerant, with the ability to handle errors and recover from failures.