**Scenario: Online Bookstore Management System**

**Scenario Overview:** You are tasked with developing an online bookstore management system that allows customers to browse books, make purchases, and manage their orders. The system should also enable the store admin to manage inventory, process orders, and generate sales reports. The project will involve both frontend and backend design, utilizing AWS for cloud services and an Applet for specific user interactions.

**Tasks:**

1. **Frontend Design:**
   * **Customer Interface:**
     + Design a user-friendly web interface where customers can search for books by title, author, or genre.
     + Implement a shopping cart feature where customers can add books and view their total cost.
     + Create a secure login and registration system for customers using Applet.
     + Design a payment gateway interface for customers to make purchases.
   * **Admin Interface:**
     + Develop a dashboard for the admin to manage book inventory (add, update, delete books).
     + Implement features to process orders, view pending orders, and track shipment.
     + Create a reporting module where the admin can generate sales reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the bookstore’s database, storing information on books, customers, orders, and payments.
     + Implement CRUD operations to handle inventory management, customer information, and order processing.
   * **Order Processing:**
     + Create backend logic to handle order placement, payment processing, and order confirmation.
     + Implement email notifications to customers for order confirmation and shipment tracking.
   * **Reporting:**
     + Design backend logic to generate sales reports based on various criteria (daily, monthly sales).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing book cover images and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the web application using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online bookstore with both customer-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing inventory, orders, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and integration with cloud services.

**1. Scenario: Online Grocery Store Management System**

**Scenario Overview:**  
You are tasked with developing an online grocery store management system that allows customers to browse products, add them to a shopping cart, and make purchases. The system should also enable the store admin to manage product inventory, process orders, and generate reports on sales and inventory levels.

**Tasks:**

1. **Frontend Design:**
   * **Customer Interface:**
     + Design a user-friendly interface where customers can search for products by name, category, or brand.
     + Implement a shopping cart feature with a running total of selected items.
     + Create a secure login and registration system for customers using Applet.
     + Design a payment gateway interface to handle customer transactions.
   * **Admin Interface:**
     + Develop a dashboard for the admin to manage product inventory (add, update, delete products).
     + Implement features to process orders, manage shipments, and view inventory levels.
     + Create a reporting module for generating sales and inventory reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the grocery store’s database, storing product, customer, order, and payment information.
     + Implement CRUD operations for managing inventory, customer data, and order processing.
   * **Order Processing:**
     + Create backend logic to handle order placement, payment processing, and order confirmation.
     + Implement email notifications for customers on order status and shipment tracking.
   * **Reporting:**
     + Design backend logic to generate reports on sales and inventory based on various criteria (daily, weekly, monthly).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing product images and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the web application using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online grocery store with customer-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing inventory, orders, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**2. Scenario: Online Learning Management System (LMS)**

**Scenario Overview:**  
You are tasked with developing an online Learning Management System (LMS) that allows students to enroll in courses, access course materials, and submit assignments. The system should also allow instructors to manage courses, grade submissions, and generate performance reports.

**Tasks:**

1. **Frontend Design:**
   * **Student Interface:**
     + Design a user-friendly interface for students to browse courses, enroll, and access course materials.
     + Implement a secure login and registration system using Applet.
     + Create an assignment submission interface with notifications for deadlines.
   * **Instructor Interface:**
     + Develop a dashboard for instructors to create, update, and manage courses.
     + Implement features for grading assignments and providing feedback.
     + Create a reporting module where instructors can generate student performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the LMS database, storing information on courses, students, instructors, and grades.
     + Implement CRUD operations for managing course content, student data, and assignment submissions.
   * **Assignment Handling:**
     + Create backend logic to manage assignment submissions, grading, and feedback.
     + Implement email notifications to students for assignment deadlines and grades.
   * **Reporting:**
     + Design backend logic to generate performance reports based on various criteria (course, student, assignment).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing course materials, videos, and assignments.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the LMS using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional LMS with student and instructor interfaces.
* Secure and efficient backend operations for managing courses, assignments, and grades.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**3. Scenario: Online Movie Ticket Booking System**

**Scenario Overview:**  
You are tasked with developing an online movie ticket booking system that allows customers to browse movies, select showtimes, and book tickets. The system should also allow theater admins to manage movie listings, track bookings, and generate sales reports.

**Tasks:**

1. **Frontend Design:**
   * **Customer Interface:**
     + Design a user-friendly interface where customers can browse movies, select showtimes, and book tickets.
     + Implement a secure login and registration system using Applet.
     + Create a payment gateway interface for ticket purchases.
   * **Admin Interface:**
     + Develop a dashboard for admins to manage movie listings (add, update, delete movies).
     + Implement features for tracking bookings and managing showtimes.
     + Create a reporting module for generating sales and booking reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the movie ticket booking database, storing information on movies, showtimes, bookings, and payments.
     + Implement CRUD operations for managing movie listings, customer data, and bookings.
   * **Booking Processing:**
     + Create backend logic to handle ticket booking, payment processing, and booking confirmation.
     + Implement email notifications for customers on booking confirmation and reminders.
   * **Reporting:**
     + Design backend logic to generate sales reports based on various criteria (movie, showtime, date).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing movie posters and promotional media.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the ticket booking system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional movie ticket booking system with customer-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing movie listings, bookings, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**4. Scenario: Online Food Delivery System**

**Scenario Overview:**  
You are tasked with developing an online food delivery system that allows customers to browse menus, place orders, and track deliveries. The system should also allow restaurant admins to manage menus, process orders, and generate sales reports.

**Tasks:**

1. **Frontend Design:**
   * **Customer Interface:**
     + Design a user-friendly interface where customers can browse menus, place orders, and track deliveries.
     + Implement a secure login and registration system using Applet.
     + Create a payment gateway interface for handling transactions.
   * **Admin Interface:**
     + Develop a dashboard for restaurant admins to manage menus (add, update, delete dishes).
     + Implement features for processing orders and managing delivery logistics.
     + Create a reporting module for generating sales and order reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the food delivery system’s database, storing information on menus, customers, orders, and payments.
     + Implement CRUD operations for managing menu items, customer data, and order processing.
   * **Order Processing:**
     + Create backend logic to handle order placement, payment processing, and delivery tracking.
     + Implement email and SMS notifications for order confirmations and delivery updates.
   * **Reporting:**
     + Design backend logic to generate reports on sales and order fulfillment based on various criteria (restaurant, dish, date).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing menu images and promotional content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the food delivery system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional food delivery system with customer-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing menus, orders, and deliveries.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**5. Scenario: Online Event Management System**

**Scenario Overview:**  
You are tasked with developing an online event management system that allows users to browse upcoming events, book tickets, and manage their bookings. The system should also allow event organizers to create events, manage ticket sales, and generate reports on event performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a user-friendly interface where users can search for events by date, category, or location.
     + Implement a ticket booking feature with a user-friendly calendar view.
     + Create a secure login and registration system using Applet.
     + Design a payment gateway interface for ticket purchases.
   * **Organizer Interface:**
     + Develop a dashboard for organizers to create, update, and manage events.
     + Implement features for tracking ticket sales and managing attendee lists.
     + Create a reporting module for generating event performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the event management system’s database, storing information on events, users, ticket sales, and payments.
     + Implement CRUD operations for managing event details, user data, and bookings.
   * **Ticket Management:**
     + Create backend logic to handle ticket booking, payment processing, and booking confirmation.
     + Implement email notifications for users on booking confirmation and event reminders.
   * **Reporting:**
     + Design backend logic to generate reports on ticket sales and event performance based on various criteria (event, category, date).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing event images and promotional media.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the event management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online event management system with user-facing and organizer-facing interfaces.
* Secure and efficient backend operations for managing events, ticket sales, and attendee data.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**6. Scenario: Online Job Portal**

**Scenario Overview:**  
You are tasked with developing an online job portal that connects job seekers with employers. The system should allow job seekers to search for jobs, submit applications, and track their application status. Employers should be able to post job openings, manage applications, and generate reports on recruitment metrics.

**Tasks:**

1. **Frontend Design:**
   * **Job Seeker Interface:**
     + Design a user-friendly interface where job seekers can search for jobs by title, company, or location.
     + Implement a resume upload feature and allow job seekers to apply for jobs online.
     + Create a secure login and registration system using Applet.
     + Design a dashboard where job seekers can track the status of their applications.
   * **Employer Interface:**
     + Develop a dashboard for employers to post job openings and manage applications.
     + Implement features for viewing candidate profiles and scheduling interviews.
     + Create a reporting module for generating recruitment metrics (applications received, interviews scheduled).
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the job portal’s database, storing information on jobs, job seekers, employers, and applications.
     + Implement CRUD operations for managing job postings, candidate data, and application processing.
   * **Application Management:**
     + Create backend logic to handle job applications, status updates, and interview scheduling.
     + Implement email notifications for job seekers on application status and interview invitations.
   * **Reporting:**
     + Design backend logic to generate reports on recruitment metrics based on various criteria (job, company, date).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing resumes and company logos.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the job portal using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online job portal with interfaces for job seekers and employers.
* Secure and efficient backend operations for managing job postings, applications, and candidate data.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**7. Scenario: Online Fitness Coaching Platform**

**Scenario Overview:**  
You are tasked with developing an online fitness coaching platform that allows users to connect with fitness coaches, follow workout plans, and track their progress. The platform should also allow coaches to create workout plans, monitor client progress, and generate performance reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a user-friendly interface where users can browse fitness coaches, sign up for coaching plans, and track their workouts.
     + Implement a secure login and registration system using Applet.
     + Create a dashboard for users to track their workout progress and achievements.
     + Design a payment gateway interface for users to pay for coaching services.
   * **Coach Interface:**
     + Develop a dashboard for coaches to create and manage workout plans for clients.
     + Implement features for tracking client progress and providing feedback.
     + Create a reporting module for generating client performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the fitness coaching platform’s database, storing information on users, coaches, workout plans, and payments.
     + Implement CRUD operations for managing client data, workout plans, and progress tracking.
   * **Progress Tracking:**
     + Create backend logic to handle workout tracking, progress updates, and coach feedback.
     + Implement email notifications for users on workout reminders and progress milestones.
   * **Reporting:**
     + Design backend logic to generate reports on client performance based on various criteria (workout plan, progress, time frame).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing workout videos, images, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the fitness coaching platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online fitness coaching platform with interfaces for users and coaches.
* Secure and efficient backend operations for managing workout plans, progress tracking, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the platform’s features, architecture, and cloud integration.

**8. Scenario: Real Estate Property Management System**

**Scenario Overview:**  
You are tasked with developing a real estate property management system that allows users to search for properties, schedule visits, and manage their inquiries. The system should also allow property managers to list properties, track inquiries, and generate reports on property performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a user-friendly interface where users can search for properties by location, price, or type.
     + Implement a secure login and registration system using Applet.
     + Create a dashboard where users can manage their inquiries and schedule property visits.
     + Design a payment gateway interface for property-related transactions.
   * **Property Manager Interface:**
     + Develop a dashboard for property managers to list and manage properties.
     + Implement features for tracking inquiries, scheduling visits, and managing tenant information.
     + Create a reporting module for generating property performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the property management system’s database, storing information on properties, users, inquiries, and transactions.
     + Implement CRUD operations for managing property listings, user data, and inquiry tracking.
   * **Inquiry Management:**
     + Create backend logic to handle property inquiries, visit scheduling, and inquiry follow-ups.
     + Implement email notifications for users on inquiry status and visit reminders.
   * **Reporting:**
     + Design backend logic to generate reports on property performance based on various criteria (location, price, availability).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing property images, floor plans, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the property management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional real estate property management system with user-facing and manager-facing interfaces.
* Secure and efficient backend operations for managing property listings, inquiries, and transactions.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**9. Scenario: Online Travel Booking System**

**Scenario Overview:**  
You are tasked with developing an online travel booking system that allows users to search for flights, hotels, and rental cars, and book them online. The system should also allow travel agents to manage bookings, track customer preferences, and generate reports on booking trends.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a user-friendly interface where users can search for flights, hotels, and rental cars by date, location, or price.
     + Implement a secure login and registration system using Applet.
     + Create a dashboard where users can manage their bookings and view their travel itinerary.
     + Design a payment gateway interface for travel bookings.
   * **Travel Agent Interface:**
     + Develop a dashboard for travel agents to manage customer bookings and preferences.
     + Implement features for tracking customer travel history and providing personalized travel recommendations.
     + Create a reporting module for generating booking trend reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the travel booking system’s database, storing information on flights, hotels, rental cars, users, and bookings.
     + Implement CRUD operations for managing travel options, user data, and booking processing.
   * **Booking Management:**
     + Create backend logic to handle travel bookings, itinerary generation, and booking confirmation.
     + Implement email notifications for users on booking confirmation and travel reminders.
   * **Reporting:**
     + Design backend logic to generate reports on booking trends based on various criteria (travel destination, booking date, customer demographics).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing travel brochures, booking confirmations, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the travel booking system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online travel booking system with user-facing and travel agent-facing interfaces.
* Secure and efficient backend operations for managing travel options, bookings, and customer preferences.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**10. Scenario: Online Food Ordering and Delivery System**

**Scenario Overview:**  
You are tasked with developing an online food ordering and delivery system that allows users to browse menus, place orders, and track deliveries in real-time. The system should also allow restaurant owners to manage their menus, track orders, and generate sales reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse restaurant menus, select items, and place orders.
     + Implement a real-time order tracking feature, where users can track their delivery status.
     + Create a secure login and registration system using Applet.
     + Design a payment gateway for online payments.
   * **Restaurant Owner Interface:**
     + Develop a dashboard for restaurant owners to manage their menus and track orders.
     + Implement features for updating menu items, prices, and availability.
     + Create a reporting module for generating sales reports based on orders received.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the food ordering system’s database, storing information on restaurants, menus, users, orders, and payments.
     + Implement CRUD operations for managing menu items, order processing, and user data.
   * **Order Management:**
     + Create backend logic to handle order placements, processing, and delivery tracking.
     + Implement push notifications for users on order status and delivery updates.
   * **Reporting:**
     + Design backend logic to generate sales reports and insights based on various criteria (restaurant, date, item).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing menu images, promotional banners, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the food ordering system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online food ordering and delivery system with user-facing and restaurant-facing interfaces.
* Secure and efficient backend operations for managing orders, menu items, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**11. Scenario: Online Learning Management System (LMS)**

**Scenario Overview:**  
You are tasked with developing an online learning management system that allows educators to create courses, upload content, and manage student progress. The system should also allow students to enroll in courses, access learning materials, and track their progress.

**Tasks:**

1. **Frontend Design:**
   * **Student Interface:**
     + Design a user-friendly interface where students can browse and enroll in courses.
     + Implement features for accessing course content, submitting assignments, and tracking progress.
     + Create a secure login and registration system using Applet.
   * **Educator Interface:**
     + Develop a dashboard for educators to create and manage courses.
     + Implement features for uploading course materials, setting assignments, and grading submissions.
     + Design a reporting module for tracking student performance and course analytics.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the LMS database, storing information on courses, students, educators, assignments, and grades.
     + Implement CRUD operations for managing course content, student data, and assignment submissions.
   * **Assignment Management:**
     + Create backend logic to handle assignment submissions, grading, and feedback.
     + Implement email notifications for students on assignment deadlines and grades.
   * **Reporting:**
     + Design backend logic to generate course and student performance reports based on various criteria (course, student, grade).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing course materials, videos, and assignments.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the LMS using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online learning management system with student-facing and educator-facing interfaces.
* Secure and efficient backend operations for managing courses, assignments, and student data.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**12. Scenario: E-commerce Platform with Product Recommendation Engine**

**Scenario Overview:**  
You are tasked with developing an e-commerce platform that allows users to browse products, place orders, and receive personalized product recommendations. The system should also allow vendors to manage their product listings, track sales, and generate reports on sales performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse products by category, price, or brand.
     + Implement a recommendation engine that suggests products based on user browsing history and preferences.
     + Create a secure login and registration system using Applet.
     + Design a shopping cart and payment gateway for users to place orders.
   * **Vendor Interface:**
     + Develop a dashboard for vendors to list and manage their products.
     + Implement features for tracking sales, managing inventory, and processing orders.
     + Create a reporting module for generating sales performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the e-commerce platform’s database, storing information on products, users, vendors, orders, and payments.
     + Implement CRUD operations for managing product listings, user data, and order processing.
   * **Recommendation Engine:**
     + Create backend logic to analyze user data and generate personalized product recommendations.
     + Implement machine learning algorithms for improving the accuracy of recommendations over time.
   * **Order Management:**
     + Design backend logic to handle order placements, inventory updates, and payment processing.
     + Implement email notifications for users on order confirmation and shipping status.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing product images, promotional banners, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Machine Learning:**
     + Integrate AWS SageMaker for developing and deploying the product recommendation engine.
   * **Hosting:**
     + Deploy the e-commerce platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional e-commerce platform with user-facing and vendor-facing interfaces.
* Secure and efficient backend operations for managing products, orders, and payments.
* An intelligent recommendation engine that improves user experience and increases sales.
* Seamless integration with AWS services for cloud storage, database management, and machine learning.
* A final project presentation showcasing the platform’s features, architecture, and cloud integration.

**13. Scenario: Vehicle Rental System**

**Scenario Overview:**  
You are tasked with developing a vehicle rental system that allows users to browse available vehicles, make reservations, and manage their rentals online. The system should also allow rental companies to manage their fleet, track rentals, and generate reports on vehicle utilization.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a user-friendly interface where users can search for available vehicles by type, location, or price.
     + Implement features for making reservations, viewing rental history, and managing rental details.
     + Create a secure login and registration system using Applet.
     + Design a payment gateway for processing rental payments.
   * **Rental Company Interface:**
     + Develop a dashboard for rental companies to manage their vehicle fleet and track rentals.
     + Implement features for updating vehicle availability, tracking vehicle utilization, and processing returns.
     + Create a reporting module for generating utilization and revenue reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the vehicle rental system’s database, storing information on vehicles, users, rentals, and payments.
     + Implement CRUD operations for managing vehicle listings, rental reservations, and user data.
   * **Rental Management:**
     + Create backend logic to handle vehicle reservations, rental processing, and return management.
     + Implement email notifications for users on reservation confirmation and return reminders.
   * **Reporting:**
     + Design backend logic to generate reports on vehicle utilization and revenue based on various criteria (vehicle type, location, rental duration).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing vehicle images, rental contracts, and other media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the vehicle rental system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional vehicle rental system with user-facing and rental company-facing interfaces.
* Secure and efficient backend operations for managing vehicle listings, rentals, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**14. Scenario: Online Event Management System**

**Scenario Overview:**  
You are tasked with developing an online event management system that allows users to browse and register for events, manage their registrations, and receive notifications. Event organizers should be able to create and manage events, track registrations, and generate reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse events by category, date, or location.
     + Implement features for event registration, viewing registration history, and receiving event notifications.
     + Create a secure login and registration system using Applet.
     + Design a payment gateway for event tickets.
   * **Event Organizer Interface:**
     + Develop a dashboard for organizers to create and manage events.
     + Implement features for tracking registrations, managing ticket sales, and updating event details.
     + Create a reporting module for generating event attendance and sales reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the event management system’s database, storing information on events, users, registrations, and payments.
     + Implement CRUD operations for managing event details, user registrations, and ticket processing.
   * **Registration Management:**
     + Create backend logic to handle event registrations, ticket sales, and confirmation notifications.
     + Implement email notifications for users on registration confirmation and event reminders.
   * **Reporting:**
     + Design backend logic to generate reports on event attendance and sales based on various criteria (event, date, location).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing event images, promotional materials, and registration data.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the event management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional event management system with user-facing and organizer-facing interfaces.
* Secure and efficient backend operations for managing events, registrations, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**15. Scenario: Personal Finance Management System**

**Scenario Overview:**  
You are tasked with developing a personal finance management system that allows users to track their income, expenses, and investments. The system should also provide financial insights and generate reports on spending and savings.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can input and categorize their income and expenses.
     + Implement features for tracking investments and generating financial insights.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for visualizing financial data through charts and graphs.
   * **Reporting:**
     + Develop a module for generating financial reports based on user data (monthly, yearly, category-wise).
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the finance management system’s database, storing information on transactions, investments, and user profiles.
     + Implement CRUD operations for managing financial data and generating reports.
   * **Financial Insights:**
     + Create backend logic to analyze user data and provide insights on spending habits, savings, and investment performance.
     + Implement email notifications for financial summaries and insights.
   * **Reporting:**
     + Design backend logic to generate detailed financial reports and summaries based on user data.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing financial reports, charts, and user data backups.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the finance management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional personal finance management system with user-facing interfaces for tracking and analyzing finances.
* Secure and efficient backend operations for managing financial data and generating reports.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**16. Scenario: Real Estate Property Management System**

**Scenario Overview:**  
You are tasked with developing a real estate property management system that allows users to search for properties, view details, and make inquiries. Property managers should be able to list properties, manage inquiries, and generate reports on property performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can search for properties based on location, price, and type.
     + Implement features for viewing property details, scheduling viewings, and making inquiries.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their property searches and inquiries.
   * **Property Manager Interface:**
     + Develop a dashboard for property managers to list and manage properties.
     + Implement features for tracking inquiries, scheduling viewings, and updating property details.
     + Create a reporting module for generating property performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the property management system’s database, storing information on properties, users, inquiries, and viewings.
     + Implement CRUD operations for managing property listings, user inquiries, and viewing schedules.
   * **Inquiry Management:**
     + Create backend logic to handle property inquiries, schedule viewings, and manage communication between users and property managers.
     + Implement email notifications for users on inquiry status and viewing confirmations.
   * **Reporting:**
     + Design backend logic to generate reports on property performance and inquiry trends based on various criteria (property type, location, time).
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing property images, documents, and media content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the property management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional real estate property management system with user-facing and property manager-facing interfaces.
* Secure and efficient backend operations for managing properties, inquiries, and viewings.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**17. Scenario: Online Auction System**

**Scenario Overview:**  
You are tasked with developing an online auction system where users can list items for auction, place bids, and track auction progress. Auction administrators should be able to manage auctions, verify bids, and generate reports on auction activities.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse auction items, place bids, and view auction details.
     + Implement features for tracking auction progress and receiving notifications on bid status.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their bids and auction history.
   * **Auction Administrator Interface:**
     + Develop a dashboard for administrators to manage ongoing auctions and verify bids.
     + Implement features for updating auction details, managing item listings, and handling disputes.
     + Create a reporting module for generating auction activity and revenue reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the auction system’s database, storing information on auction items, bids, users, and transactions.
     + Implement CRUD operations for managing auctions, bids, and user data.
   * **Bid Management:**
     + Create backend logic to handle bid placements, auction status updates, and item verification.
     + Implement email notifications for users on bid status, auction endings, and winning bids.
   * **Reporting:**
     + Design backend logic to generate reports on auction activities, item performance, and revenue.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing auction item images, bid history, and related documents.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the auction system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online auction system with user-facing and administrator-facing interfaces.
* Secure and efficient backend operations for managing auctions, bids, and user data.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**18. Scenario: Online Learning Management System**

**Scenario Overview:**  
You are tasked with developing an online learning management system (LMS) where students can enroll in courses, complete assignments, and track their progress. Instructors should be able to create and manage courses, track student progress, and provide feedback.

**Tasks:**

1. **Frontend Design:**
   * **Student Interface:**
     + Design a user-friendly interface where students can browse courses, enroll, and access course materials.
     + Implement features for submitting assignments, tracking progress, and viewing grades.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for students to manage their courses and assignments.
   * **Instructor Interface:**
     + Develop a dashboard for instructors to create and manage courses, assignments, and quizzes.
     + Implement features for tracking student submissions, grading, and providing feedback.
     + Create a reporting module for instructors to view course performance and student progress.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the LMS database, storing information on courses, students, instructors, assignments, and grades.
     + Implement CRUD operations for course management, student data, and assignment submissions.
   * **Assignment and Grading:**
     + Create backend logic to handle assignment submissions, grading, and feedback.
     + Implement email notifications for students on assignment deadlines and grade updates.
   * **Reporting:**
     + Design backend logic to generate reports on course performance, student progress, and assignment statistics.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing course materials, assignments, and multimedia content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the LMS using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional LMS with student-facing and instructor-facing interfaces.
* Secure and efficient backend operations for managing courses, assignments, and grades.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**19. Scenario: Online Fitness Coaching Platform**

**Scenario Overview:**  
You are tasked with developing an online fitness coaching platform where users can access workout plans, track their progress, and communicate with coaches. Coaches should be able to create workout plans, monitor user progress, and provide feedback.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can view workout plans, track their exercises, and log their progress.
     + Implement features for accessing video tutorials, setting fitness goals, and communicating with coaches.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their fitness activities and progress.
   * **Coach Interface:**
     + Develop a dashboard for coaches to create and manage workout plans, track user progress, and provide feedback.
     + Implement features for monitoring user activity, updating workout plans, and sending notifications.
     + Create a reporting module for coaches to view user progress and performance metrics.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the fitness platform’s database, storing information on workout plans, users, progress logs, and feedback.
     + Implement CRUD operations for managing workout plans, user data, and progress tracking.
   * **Progress Tracking:**
     + Create backend logic to handle progress tracking, goal setting, and feedback.
     + Implement email notifications for users on progress updates and coach feedback.
   * **Reporting:**
     + Design backend logic to generate reports on user progress, workout plan effectiveness, and coaching performance.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing workout videos, user data, and progress logs.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the fitness platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional fitness coaching platform with user-facing and coach-facing interfaces.
* Secure and efficient backend operations for managing workout plans, progress, and feedback.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**20. Scenario: Online Customer Support System**

**Scenario Overview:**  
You are tasked with developing an online customer support system where users can submit support tickets, track their status, and receive responses. Support agents should be able to manage tickets, provide solutions, and generate reports on ticket resolution.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can submit support tickets, view ticket status, and receive responses.
     + Implement features for tracking ticket history and accessing FAQs.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their support tickets and communication with agents.
   * **Support Agent Interface:**
     + Develop a dashboard for agents to view, manage, and respond to support tickets.
     + Implement features for updating ticket status, providing solutions, and handling escalations.
     + Create a reporting module for agents to view ticket resolution statistics and performance metrics.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the customer support system’s database, storing information on support tickets, users, agents, and solutions.
     + Implement CRUD operations for managing tickets, user data, and agent responses.
   * **Ticket Management:**
     + Create backend logic to handle ticket submission, status updates, and agent responses.
     + Implement email notifications for users on ticket status updates and agent responses.
   * **Reporting:**
     + Design backend logic to generate reports on ticket resolution times, agent performance, and support metrics.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing support ticket attachments, user data, and communication logs.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the customer support system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional customer support system with user-facing and agent-facing interfaces.
* Secure and efficient backend operations for managing support tickets, responses, and reporting.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**21. Scenario: Online Ticket Booking System**

**Scenario Overview:**  
You are tasked with developing an online ticket booking system for various events (movies, concerts, etc.) where users can browse events, book tickets, and manage their bookings. Event organizers should be able to list events, manage bookings, and generate reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse events, view details, and book tickets.
     + Implement features for managing bookings, viewing ticket history, and receiving booking confirmations.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their bookings and receive notifications.
   * **Event Organizer Interface:**
     + Develop a dashboard for organizers to list and manage events, track bookings, and update event details.
     + Implement features for managing ticket availability, processing refunds, and generating reports.
     + Create a reporting module for generating reports on event bookings and revenue.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the ticket booking system’s database, storing information on events, users, bookings, and payments.
     + Implement CRUD operations for managing events, bookings, and user data.
   * **Booking Management:**
     + Create backend logic to handle booking processing, ticket availability, and booking confirmations.
     + Implement email notifications for users on booking confirmations and event updates.
   * **Reporting:**
     + Design backend logic to generate reports on event bookings, ticket sales, and revenue.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing event images, booking data, and ticket information.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the ticket booking system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional ticket booking system with user-facing and event organizer-facing interfaces.
* Secure and efficient backend operations for managing events, bookings, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**22. Scenario: Smart Home Automation System**

**Scenario Overview:**  
You are tasked with developing a smart home automation system that allows users to control various devices in their homes (lights, thermostat, security cameras) from a central interface. The system should support scheduling, real-time control, and notifications.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can view and control home devices such as lights, thermostat, and security cameras.
     + Implement features for scheduling device operations and setting automation rules.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for monitoring device status and receiving notifications.
   * **Admin Interface:**
     + Develop a dashboard for managing device configurations, user accounts, and system settings.
     + Implement features for viewing system logs, generating usage reports, and handling user support requests.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the smart home system’s database, storing information on devices, user settings, schedules, and notifications.
     + Implement CRUD operations for managing devices, user data, and schedules.
   * **Device Control:**
     + Create backend logic to handle real-time device control, scheduling, and automation.
     + Implement email or SMS notifications for system alerts and status updates.
   * **Reporting:**
     + Design backend logic to generate reports on device usage, system performance, and user activity.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing device configurations, system logs, and media files.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the smart home system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional smart home automation system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing devices, schedules, and notifications.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**23. Scenario: Virtual Event Platform**

**Scenario Overview:**  
You are tasked with developing a virtual event platform where users can attend and participate in online events (webinars, workshops). The platform should allow event organizers to create events, manage participants, and facilitate live interactions.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse and register for virtual events, view event details, and participate in live sessions.
     + Implement features for interacting with speakers, participating in Q&A sessions, and accessing event materials.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their event registrations and view event schedules.
   * **Organizer Interface:**
     + Develop a dashboard for organizers to create and manage events, view participant lists, and handle live interactions.
     + Implement features for sending invitations, moderating live sessions, and generating event analytics.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the virtual event platform’s database, storing information on events, users, registrations, and interactions.
     + Implement CRUD operations for managing events, user data, and participant interactions.
   * **Live Interaction:**
     + Create backend logic to handle live streaming, participant interactions, and session management.
     + Implement email notifications for event reminders, updates, and post-event follow-ups.
   * **Analytics:**
     + Design backend logic to generate reports on event attendance, participant engagement, and session performance.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing event materials, video recordings, and media files.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the virtual event platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional virtual event platform with user-facing and organizer-facing interfaces.
* Secure and efficient backend operations for managing events, interactions, and analytics.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**24. Scenario: Online Travel Booking Platform**

**Scenario Overview:**  
You are tasked with developing an online travel booking platform where users can search for travel options (flights, hotels), book reservations, and manage their itineraries. Travel agents should be able to list and manage travel options, handle bookings, and generate reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can search for and book flights, hotels, and other travel options.
     + Implement features for managing bookings, viewing itineraries, and accessing travel documents.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for users to manage their travel plans and view booking details.
   * **Agent Interface:**
     + Develop a dashboard for agents to list and manage travel options, handle bookings, and update availability.
     + Implement features for processing payments, managing customer inquiries, and generating reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the travel booking platform’s database, storing information on travel options, users, bookings, and payments.
     + Implement CRUD operations for managing travel options, user data, and bookings.
   * **Booking Management:**
     + Create backend logic to handle booking processing, payment handling, and itinerary management.
     + Implement email notifications for booking confirmations, payment receipts, and itinerary updates.
   * **Reporting:**
     + Design backend logic to generate reports on booking trends, revenue, and agent performance.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing travel documents, booking details, and media files.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the travel booking platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional travel booking platform with user-facing and agent-facing interfaces.
* Secure and efficient backend operations for managing travel options, bookings, and payments.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**25. Scenario: Online Job Portal**

**Scenario Overview:**  
You are tasked with developing an online job portal where job seekers can search for job listings, apply for positions, and track their application status. Employers should be able to post job openings, review applications, and manage their recruitment process.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where job seekers can search for job listings, view details, and apply for positions.
     + Implement features for creating and managing resumes, tracking application status, and receiving notifications.
     + Create a secure login and registration system using Applet.
     + Design a dashboard for job seekers to manage their job search and applications.
   * **Employer Interface:**
     + Develop a dashboard for employers to post job openings, view applications, and manage recruitment.
     + Implement features for reviewing applications, scheduling interviews, and generating recruitment reports.
     + Create a reporting module for employers to view application statistics and hiring trends.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the job portal’s database, storing information on job listings, users, applications, and employer data.
     + Implement CRUD operations for managing job listings, user profiles, and application status.
   * **Application Management:**
     + Create backend logic to handle job applications, resume management, and application tracking.
     + Implement email notifications for job seekers on application status and employer responses.
   * **Reporting:**
     + Design backend logic to generate reports on job application trends, employer activity, and recruitment metrics.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing resumes, job listings, and media files.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the job portal using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional job portal with user-facing and employer-facing interfaces.
* Secure and efficient backend operations for managing job listings, applications, and recruitment.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**26. Scenario: Personal Finance Management System**

**Scenario Overview:**  
You are tasked with developing a personal finance management system that helps users track their expenses, budget their income, and generate financial reports. The system should also provide insights into spending habits and financial goals.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a dashboard where users can input and categorize their expenses, set budgets, and view financial summaries.
     + Implement features for visualizing financial data through charts and graphs.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing user accounts, reviewing financial data, and generating reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the system’s database, storing information on transactions, budgets, and user profiles.
     + Implement CRUD operations for managing expenses, budgets, and user data.
   * **Financial Analysis:**
     + Create backend logic to analyze spending patterns, budget adherence, and financial health.
     + Implement email notifications for budget limits and financial insights.
   * **Reporting:**
     + Design backend logic to generate detailed financial reports and summaries.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing financial data backups and user-generated reports.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the finance management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional personal finance management system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing finances, budgets, and reports.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**27. Scenario: Online Learning Platform**

**Scenario Overview:**  
You are tasked with developing an online learning platform where users can access courses, participate in discussions, and track their progress. Instructors should be able to create and manage courses, and monitor student performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a course catalog where users can browse and enroll in courses.
     + Implement features for accessing course materials, participating in forums, and tracking progress.
     + Create a secure login and registration system using Applet.
   * **Instructor Interface:**
     + Develop a dashboard for instructors to create and manage courses, grade assignments, and interact with students.
     + Implement features for generating course analytics and monitoring student progress.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the platform’s database, storing information on courses, users, enrollments, and progress.
     + Implement CRUD operations for managing courses, user data, and assignments.
   * **Course Management:**
     + Create backend logic to handle course enrollment, material access, and progress tracking.
     + Implement email notifications for course updates, assignment deadlines, and feedback.
   * **Reporting:**
     + Design backend logic to generate reports on course performance, student engagement, and overall progress.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing course materials, videos, and assignments.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the learning platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online learning platform with user-facing and instructor-facing interfaces.
* Secure and efficient backend operations for managing courses, user progress, and analytics.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**28. Scenario: Health and Wellness App**

**Scenario Overview:**  
You are tasked with developing a health and wellness app that tracks user fitness activities, dietary habits, and overall wellness. The app should provide personalized recommendations and support goal setting.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a dashboard where users can log their fitness activities, meals, and health metrics.
     + Implement features for tracking progress, setting goals, and viewing personalized recommendations.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing user data, viewing overall health trends, and providing support.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the app’s database, storing information on user activities, meals, and health metrics.
     + Implement CRUD operations for managing user data, activities, and recommendations.
   * **Health Analysis:**
     + Create backend logic to analyze user data, provide personalized recommendations, and track progress towards goals.
     + Implement email notifications for goal reminders, health tips, and progress updates.
   * **Reporting:**
     + Design backend logic to generate reports on user activity, dietary habits, and health improvements.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing user-generated data, activity logs, and health metrics.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the health and wellness app using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional health and wellness app with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing health data, recommendations, and progress tracking.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**29. Scenario: Subscription-Based Media Streaming Service**

**Scenario Overview:**  
You are tasked with developing a subscription-based media streaming service that allows users to stream videos, manage their subscriptions, and view content recommendations. Content providers should be able to upload and manage media content.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can browse and stream media content, manage their subscriptions, and view recommendations.
     + Implement features for content search, playlists, and viewing history.
     + Create a secure login and registration system using Applet.
   * **Content Provider Interface:**
     + Develop a dashboard for content providers to upload and manage media content, view analytics, and handle subscriptions.
     + Implement features for monitoring viewer engagement and managing content metadata.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the streaming service’s database, storing information on media content, user subscriptions, and viewing history.
     + Implement CRUD operations for managing content, user data, and subscriptions.
   * **Content Streaming:**
     + Create backend logic to handle media streaming, subscription management, and content recommendations.
     + Implement email notifications for subscription updates, content releases, and recommendations.
   * **Reporting:**
     + Design backend logic to generate reports on user engagement, subscription trends, and content performance.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing media files and content metadata.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the streaming service using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional media streaming service with user-facing and content provider-facing interfaces.
* Secure and efficient backend operations for managing media content, subscriptions, and streaming.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**30. Scenario: Social Media Management System**

**Scenario Overview:**  
You are tasked with developing a social media management system that allows users to create and manage posts, follow other users, and interact through likes and comments. The system should also provide analytics for tracking user engagement.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a profile page where users can create and view posts, follow/unfollow other users, and interact with posts.
     + Implement features for liking, commenting, and sharing posts.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing user accounts, monitoring interactions, and generating engagement reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the system’s database, storing information on users, posts, comments, and interactions.
     + Implement CRUD operations for managing user profiles, posts, and interactions.
   * **Interaction Management:**
     + Create backend logic to handle likes, comments, and sharing of posts.
     + Implement email notifications for new comments, likes, and follower requests.
   * **Reporting:**
     + Design backend logic to generate reports on user engagement, post popularity, and interaction trends.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing user profile images, post media, and other content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the social media management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional social media management system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing posts, interactions, and user engagement.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**31. Scenario: Automated Customer Support System**

**Scenario Overview:**  
You are tasked with developing an automated customer support system that provides users with answers to common queries using chatbots and ticketing. The system should also include a management interface for tracking and resolving support tickets.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a chatbot interface where users can ask questions and receive automated responses.
     + Implement a ticket submission form for users to report issues or request support.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing and tracking support tickets, viewing chatbot analytics, and configuring automated responses.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the database, storing information on tickets, user queries, and chatbot responses.
     + Implement CRUD operations for managing support tickets and user data.
   * **Chatbot Logic:**
     + Create backend logic for handling automated responses based on user queries.
     + Implement ticketing logic for routing and managing support requests.
   * **Reporting:**
     + Design backend logic to generate reports on ticket resolution times, chatbot performance, and user satisfaction.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing chatbot logs, support ticket records, and other relevant data.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the support system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional automated customer support system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing tickets, queries, and automated responses.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**32. Scenario: Online Job Portal**

**Scenario Overview:**  
You are tasked with developing an online job portal where job seekers can search for job openings, apply for positions, and manage their applications. Employers should be able to post job listings, review applications, and track hiring progress.

**Tasks:**

1. **Frontend Design:**
   * **Job Seeker Interface:**
     + Design a job search interface where users can browse job listings, apply for positions, and view application status.
     + Implement features for managing resumes, cover letters, and application history.
     + Create a secure login and registration system using Applet.
   * **Employer Interface:**
     + Develop a dashboard for posting job listings, reviewing applications, and tracking hiring progress.
     + Implement features for managing job listings, scheduling interviews, and communicating with candidates.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the database, storing information on job listings, user profiles, applications, and hiring data.
     + Implement CRUD operations for managing job listings, user profiles, and applications.
   * **Application Management:**
     + Create backend logic for handling job applications, tracking application status, and managing resumes.
     + Implement email notifications for application updates, job alerts, and interview scheduling.
   * **Reporting:**
     + Design backend logic to generate reports on job listings, application statistics, and hiring trends.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing resumes, cover letters, and job listing details.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the job portal using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online job portal with job seeker and employer interfaces.
* Secure and efficient backend operations for managing job listings, applications, and hiring processes.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**33. Scenario: Digital Library System**

**Scenario Overview:**  
You are tasked with developing a digital library system that allows users to browse and borrow digital books, access multimedia content, and manage their borrowing history. Library staff should be able to manage book inventories and user accounts.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design a catalog where users can browse, search for, and borrow digital books and multimedia content.
     + Implement features for managing borrowing history and viewing due dates.
     + Create a secure login and registration system using Applet.
   * **Library Staff Interface:**
     + Develop a dashboard for managing book inventories, processing borrow/return requests, and tracking user activities.
     + Implement features for updating book details, managing multimedia content, and generating usage reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the digital library’s database, storing information on books, users, borrowing history, and multimedia content.
     + Implement CRUD operations for managing book inventories, user accounts, and borrowing transactions.
   * **Borrowing Management:**
     + Create backend logic to handle borrowing requests, return processing, and due date tracking.
     + Implement email notifications for due dates, borrowing confirmations, and library updates.
   * **Reporting:**
     + Design backend logic to generate reports on book usage, borrowing trends, and user activities.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing digital books, multimedia content, and user-generated data.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the digital library system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional digital library system with user-facing and staff-facing interfaces.
* Secure and efficient backend operations for managing digital books, multimedia content, and borrowing history.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**34. Scenario: Inventory Management System**

**Scenario Overview:**  
You are tasked with developing an inventory management system for a retail store. The system should track stock levels, manage supplier information, and handle purchase orders. It should also allow users to generate reports on inventory status and supplier performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for viewing and managing inventory items, including adding, updating, and deleting stock entries.
     + Implement features for managing supplier information and creating purchase orders.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for monitoring inventory levels, processing purchase orders, and generating inventory reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the inventory database, storing information on stock items, suppliers, and purchase orders.
     + Implement CRUD operations for managing inventory and supplier data.
   * **Order Processing:**
     + Create backend logic to handle purchase orders, track order status, and update inventory levels.
     + Implement email notifications for order confirmations and stock alerts.
   * **Reporting:**
     + Design backend logic to generate reports on inventory levels, supplier performance, and purchase order history.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing images of inventory items and other relevant documents.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the inventory management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A functional inventory management system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing inventory and purchase orders.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**35. Scenario: Online Learning Platform**

**Scenario Overview:**  
You are tasked with developing an online learning platform that allows users to enroll in courses, track their progress, and interact with instructors. The platform should also enable instructors to create and manage courses and view student performance.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where students can browse and enroll in courses, view course materials, and track their progress.
     + Implement features for interacting with instructors through forums or messaging.
     + Create a secure login and registration system using Applet.
   * **Instructor Interface:**
     + Develop a dashboard for instructors to create and manage courses, upload materials, and view student performance.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the platform’s database, storing information on courses, students, instructors, and progress.
     + Implement CRUD operations for managing course content and student data.
   * **Course Management:**
     + Create backend logic to handle course enrollment, progress tracking, and material delivery.
     + Implement email notifications for course updates, progress reports, and instructor communications.
   * **Reporting:**
     + Design backend logic to generate reports on course enrollment, student performance, and instructor activity.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing course materials, videos, and other content.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the learning platform using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online learning platform with student and instructor interfaces.
* Secure and efficient backend operations for managing courses and tracking student progress.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**36. Scenario: Event Management System**

**Scenario Overview:**  
You are tasked with developing an event management system that allows users to create, manage, and register for events. The system should also provide tools for tracking attendance and generating event-related reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for users to browse, create, and register for events, including viewing event details and managing registrations.
     + Implement features for tracking event attendance and sending reminders.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing event details, tracking attendance, and generating event reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the event database, storing information on events, users, registrations, and attendance.
     + Implement CRUD operations for managing event details and user registrations.
   * **Event Management:**
     + Create backend logic to handle event creation, registration, and attendance tracking.
     + Implement email notifications for event reminders, confirmations, and updates.
   * **Reporting:**
     + Design backend logic to generate reports on event attendance, registration trends, and user participation.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing event-related documents, images, and other media.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the event management system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional event management system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing events and registrations.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**37. Scenario: Online Auction System**

**Scenario Overview:**  
You are tasked with developing an online auction system that allows users to bid on items, track auction status, and manage winning bids. The system should also provide tools for sellers to list items and manage auctions.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface where users can view auction items, place bids, and track auction status.
     + Implement features for managing user profiles and viewing bidding history.
     + Create a secure login and registration system using Applet.
   * **Seller Interface:**
     + Develop a dashboard for sellers to list items, set auction parameters, and manage bids.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the auction database, storing information on auction items, bids, users, and transactions.
     + Implement CRUD operations for managing auction listings and bid data.
   * **Auction Management:**
     + Create backend logic to handle bid placement, auction status updates, and transaction processing.
     + Implement email notifications for bid confirmations, auction results, and winning notifications.
   * **Reporting:**
     + Design backend logic to generate reports on auction performance, bidding trends, and seller activity.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing auction item images, documents, and other media.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the auction system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional online auction system with user-facing and seller-facing interfaces.
* Secure and efficient backend operations for managing auctions and bids.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**38. Scenario: Customer Relationship Management (CRM) System**

**Scenario Overview:**  
You are tasked with developing a CRM system for a business to manage customer interactions, track sales, and analyze customer data. The system should support functionalities for tracking customer interactions, managing sales leads, and generating reports.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for tracking customer interactions, viewing customer profiles, and managing sales leads.
     + Implement features for adding, updating, and viewing customer information and sales activities.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing customer data, tracking sales performance, and generating reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the CRM database, storing information on customers, interactions, sales leads, and transactions.
     + Implement CRUD operations for managing customer data and sales activities.
   * **Sales Management:**
     + Create backend logic to handle customer interactions, sales tracking, and lead management.
     + Implement email notifications for follow-up reminders and sales updates.
   * **Reporting:**
     + Design backend logic to generate reports on customer interactions, sales performance, and lead conversion rates.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing customer documents, interaction logs, and other relevant data.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the CRM system using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional CRM system with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing customer interactions and sales.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**39. Scenario: Online Subscription Service**

**Scenario Overview:**  
You are tasked with developing an online subscription service that allows users to subscribe to various plans, manage their subscriptions, and view billing history. The system should support different subscription tiers and payment processing.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for users to browse subscription plans, subscribe to plans, and view their billing history.
     + Implement features for managing subscription details and upgrading or downgrading plans.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing subscription plans, viewing user subscriptions, and processing payments.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the subscription database, storing information on subscription plans, user subscriptions, and billing details.
     + Implement CRUD operations for managing subscription plans and user data.
   * **Subscription Management:**
     + Create backend logic to handle subscription creation, upgrades, and downgrades.
     + Implement payment processing and billing updates.
   * **Reporting:**
     + Design backend logic to generate reports on subscription metrics, user activity, and payment history.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing subscription-related documents and media.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the subscription service using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional subscription service with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing subscriptions and billing.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**40. Scenario: Social Media Management Tool**

**Scenario Overview:**  
You are tasked with developing a social media management tool that allows users to schedule posts, track engagement metrics, and analyze social media performance. The system should support multiple social media platforms and provide insights into user interactions.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for scheduling posts, tracking engagement metrics, and viewing performance analytics.
     + Implement features for managing multiple social media accounts and viewing post history.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for managing social media accounts, tracking engagement, and generating performance reports.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the database, storing information on social media accounts, posts, engagement metrics, and analytics.
     + Implement CRUD operations for managing social media data and user interactions.
   * **Post Scheduling:**
     + Create backend logic to handle post scheduling, tracking engagement, and updating performance metrics.
     + Implement email notifications for post reminders and engagement alerts.
   * **Reporting:**
     + Design backend logic to generate reports on social media performance, engagement metrics, and account activity.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing media files, post images, and analytics data.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the social media management tool using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional social media management tool with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing posts, tracking engagement, and analyzing performance.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**41. Scenario: Real-Time Chat Application**

**Scenario Overview:**  
You are tasked with developing a real-time chat application that allows users to send and receive messages instantly. The application should support one-on-one and group chats, provide notifications for new messages, and manage user presence.

**Tasks:**

1. **Frontend Design:**
   * **User Interface:**
     + Design an interface for sending and receiving messages, viewing chat history, and managing chat groups.
     + Implement real-time notifications for new messages and user activity.
     + Create a secure login and registration system using Applet.
   * **Admin Interface:**
     + Develop a dashboard for monitoring user activity, managing chat groups, and viewing chat logs.
2. **Backend Design:**
   * **Database Management:**
     + Use AWS RDS to manage the chat database, storing information on users, messages, and chat groups.
     + Implement CRUD operations for managing user data, messages, and group information.
   * **Real-Time Communication:**
     + Create backend logic to handle real-time message delivery, user presence, and group chat management.
     + Implement notification systems for new messages and user activity.
   * **Reporting:**
     + Design backend logic to generate reports on user activity, chat history, and message statistics.
3. **Integration with AWS:**
   * **Cloud Storage:**
     + Use AWS S3 for storing chat logs and media files.
   * **User Authentication:**
     + Implement user authentication using AWS Cognito for secure and scalable user management.
   * **Hosting:**
     + Deploy the chat application using AWS Elastic Beanstalk or AWS EC2 for scalable deployment.
   * **Database:**
     + Utilize AWS RDS for a scalable and secure relational database service.

**Expected Outcomes:**

* A fully functional real-time chat application with user-facing and admin-facing interfaces.
* Secure and efficient backend operations for managing messages, user presence, and notifications.
* Seamless integration with AWS services for cloud storage, database management, and user authentication.
* A final project presentation showcasing the system’s features, architecture, and cloud integration.

**Feel free to adjust the complexity or add specific requirements based on the needs of your projects or students!**

**Evaluation Criteria (K4 Blooms Level):**

1. **Understanding:**
   * Demonstration of understanding the requirements and effective implementation of both frontend and backend components.
2. **Application:**
   * Successful application of Java programming concepts, AWS services, and Applet in building the system.
3. **Analysis:**
   * Ability to analyze customer and admin needs, and translate them into a functional and user-friendly application.
4. **Synthesis:**
   * Integration of various components into a cohesive and efficient online bookstore system.
5. **Evaluation:**
   * Critical assessment of the project’s functionality, performance, and user experience.
   * Presentation and defense of design decisions and the overall architecture.

**Here are detailed scenario-based capstone project questions for various Java programming applications, including frontend and backend design tasks, integration with AWS or Applet, and expected outcomes:**