

**CAPSTONE PROJECT REPORT**

**PROJECT TITLE**

**Online Subscription Service**

**REPORT SUMITTED BY**

**K.**Sukesh Reddy(192211863)

**REPORT SUMITTED TO**

DR. K.JayasakthiVelumurgan

**COURSE CODE / COURSE NAME**

CSA0912 PROGRAMMING IN JAVA FOR WEB DESIGING

ABSTRACT:

he online subscription service is a scalable, cloud-based platform that enables users to easily browse, subscribe to, and manage various digital services. The platform offers secure user authentication, flexible subscription management, and automated payment processing, ensuring a smooth and user-friendly experience. Built on Amazon Web Services (AWS), the system leverages cloud storage, database management, and authentication services to provide a reliable, high-performance solution. Administrators have comprehensive control over managing subscription plans, resolving disputes, and generating detailed analytical reports. The system is designed with a focus on data security, scalability, and seamless integration, catering to both users and administrators with an intuitive interface and efficient backend operations.

INTRODUCTION:

The online subscription service provides a comprehensive platform for users to subscribe to a wide range of digital services, including streaming, learning modules, and software tools. The platform streamlines the subscription process by offering an intuitive interface for browsing services, selecting subscription plans, and managing user accounts. Built on a robust cloud-based infrastructure using Amazon Web Services (AWS), the system ensures scalability, reliability, and enhanced performance to handle varying user demands.Security is a key focus, with the platform implementing secure authentication and transaction methods to protect user data. Administrators benefit from advanced features such as subscription management, dispute resolution, and report generation to track service performance and user engagement. Designed to deliver a seamless user experience, the system also supports real-time updates, automatic renewals, and notifications, ensuring that both users and administrators can efficiently manage their subscription needs.By leveraging the flexibility of cloud computing, this platform is well-suited for businesses offering recurring digital services and seeks to provide a scalable and secure subscription management solution.

PROBLEM STATEMENT:

In today's digital landscape, businesses offering subscription-based services face challenges in managing a growing user base, securely handling transactions, and providing an intuitive experience for both subscribers and administrators. Traditional subscription management systems often lack scalability, real-time updates, and the ability to efficiently handle multiple service offerings. Additionally, maintaining data security and providing seamless integration with cloud infrastructure can be complex and costly.This online subscription service aims to address these issues by developing a cloud-based platform that allows users to easily subscribe to and manage services while ensuring secure payments and data privacy. The platform must provide administrators with tools for managing subscriptions, resolving disputes, and generating reports, all within a scalable and secure environment leveraging Amazon Web Services (AWS). The solution should enhance user experience, offer real-time subscription management, and support business growth without compromising on security or performance.

OBJECTIVE:

• User-Friendly Subscription Management: Develop an intuitive platform for users to browse, subscribe to, and manage their subscriptions easily.

• Secure Transactions: Ensure secure payment processing and data protection using encryption and authentication techniques.

• Scalable Cloud Infrastructure: Leverage AWS cloud services for scalable storage, database management, and seamless system performance to handle growing user demands.

• Real-Time Subscription Updates: Provide real-time updates for subscription status, renewals, and notifications for both users and administrators.

• Administrative Control: Equip administrators with robust tools for managing subscription plans, verifying user details, resolving disputes, and generating detailed reports..

• Automated Renewal and Payment Systems: Implement automated processes for subscription renewals and recurring payments to ensure convenience for users and operational efficiency.

MATERIALS AND METHODS:

Frontend Design:

➢ Home Page: Header with navigation, service highlights, and call-to-action buttons, ending with a footer.

➢ Service Catalog: Search bar, filters, and a grid of services with "Subscribe" buttons.

➢ Service Detail Page: Detailed service info, pricing plans, and FAQs with a prominent "Subscribe" button.

➢ User Dashboard: Profile, subscription management, billing info, and settings.

➢ Checkout Page: Subscription summary, secure payment options, and confirmation.

Backend Design:

➢ Database Management:

* Use AWS RDS for managing subscription data.
* Implement CRUD operations for subscriptions and user data.

➢ Subscription Management:

* Handle creation, upgrades, downgrades, and cancellations.
* Integrate payment processing for billing.

➢ Reporting:

* Generate reports on subscription metrics, user activity, and payment history.

AWS Integration:

➢ AWS RDS (Relational Database Service):

* Database Management: Use RDS to manage relational databases for storing subscription plans, user data, and billing information.
* Scaling & Backup: Utilize RDS features for automatic scaling, backups, and high availability.

➢AWS S3 (Simple Storage Service):

* File Storage: Store static files, such as user documents or service-related media, in S3 buckets.
* Data Backup: Use S3 for additional backup and archival storage.

➢AWS Lambda:

* Serverless Functions: Implement serverless functions for tasks like subscription processing, payment notifications, and real-time updates.
* Event-Driven Execution: Use Lambda for handling events such as subscription renewals or user actions.

➢AWS API Gateway:

* API Management: Create and manage APIs for subscription management, payment processing, and reporting.
* Integration with Lambda: Route API requests to Lambda functions for processing.

➢AWS SNS (Simple Notification Service):

* Notifications: Send email or SMS notifications for subscription renewals, payment confirmations, and system alerts.

➢AWS CloudWatch:

* Monitoring & Logging: Monitor system performance, log errors, and set up alarms for issues like payment failures or high usage.

➢AWS IAM (Identity and Access Management):

* Security: Manage user permissions and secure access to AWS resources for the application and administrators.

CONCLUSION:

The development of an online subscription service entails creating a user-friendly and secure platform for managing and accessing digital subscriptions. Leveraging AWS services ensures the platform's scalability, security, and reliability, while modern web technologies provide a seamless and engaging user experience. The system supports real-time subscription management, secure user authentication, and comprehensive reporting on subscription metrics and user activities. By integrating cloud-based architecture and focusing on user-centric design, this project delivers a robust and efficient solution for managing digital subscriptions, catering to both users and administrators with a streamlined and reliable experience.

CODE:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class SubscriptionPlan {

private String planName;

private double price;

public SubscriptionPlan(String planName, double price) {

this.planName = planName;

this.price = price;

}

public String getPlanName() {

return planName;

}

public double getPrice() {

return price;

}

@Override

public String toString() {

return planName + " ($" + price + ")";

}

}

class User {

private String username;

private String email;

private SubscriptionPlan subscriptionPlan;

public User(String username, String email) {

this.username = username;

this.email = email;

this.subscriptionPlan = null; // Initially, no subscription plan is selected

}

public String getUsername() {

return username;

}

public String getEmail() {

return email;

}

public SubscriptionPlan getSubscriptionPlan() {

return subscriptionPlan;

}

public void subscribe(SubscriptionPlan plan) {

this.subscriptionPlan = plan;

System.out.println(username + " has subscribed to the " + plan.getPlanName() + " plan.");

}

public void cancelSubscription() {

this.subscriptionPlan = null;

System.out.println(username + " has canceled their subscription.");

}

@Override

public String toString() {

return "User: " + username + " (" + email + ") | Subscription: " +

(subscriptionPlan != null ? subscriptionPlan.getPlanName() : "None");

}

}

public class OnlineSubscriptionService {

private List<User> users;

private List<SubscriptionPlan> plans;

public OnlineSubscriptionService() {

users = new ArrayList<>();

plans = new ArrayList<>();

initializePlans();

}

private void initializePlans() {

plans.add(new SubscriptionPlan("Basic", 9.99));

plans.add(new SubscriptionPlan("Premium", 19.99));

plans.add(new SubscriptionPlan("Platinum", 29.99));

}

public void registerUser(String username, String email) {

users.add(new User(username, email));

System.out.println("User " + username + " registered successfully.");

}

public void subscribeUser(String username, String planName) {

User user = findUserByUsername(username);

if (user != null) {

SubscriptionPlan plan = findPlanByName(planName);

if (plan != null) {

user.subscribe(plan);

} else {

System.out.println("Plan " + planName + " not found.");

}

} else {

System.out.println("User " + username + " not found.");

}

}

public void cancelSubscription(String username) {

User user = findUserByUsername(username);

if (user != null) {

user.cancelSubscription();

} else {

System.out.println("User " + username + " not found.");

}

}

private User findUserByUsername(String username) {

for (User user : users) {

if (user.getUsername().equalsIgnoreCase(username)) {

return user;

}

}

return null;

}

private SubscriptionPlan findPlanByName(String planName) {

for (SubscriptionPlan plan : plans) {

if (plan.getPlanName().equalsIgnoreCase(planName)) {

return plan;

}

}

return null;

}

public void displayUsers() {

for (User user : users) {

System.out.println(user);

}

}

public void displayPlans() {

System.out.println("Available Subscription Plans:");

for (SubscriptionPlan plan : plans) {

System.out.println(plan);

}

}

public static void main(String[] args) {

OnlineSubscriptionService service = new OnlineSubscriptionService();

Scanner scanner = new Scanner(System.in);

System.out.println("Welcome to the Online Subscription Service!");

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Register User");

System.out.println("2. Subscribe to Plan");

System.out.println("3. Cancel Subscription");

System.out.println("4. Display Users");

System.out.println("5. Display Plans");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter email: ");

String email = scanner.nextLine();

service.registerUser(username, email);

break;

case 2:

System.out.print("Enter username: ");

String userToSubscribe = scanner.nextLine();

service.displayPlans();

System.out.print("Enter the plan name to subscribe to: ");

String planName = scanner.nextLine();

service.subscribeUser(userToSubscribe, planName);

break;

case 3:

System.out.print("Enter username to cancel subscription: ");

String userToCancel = scanner.nextLine();

service.cancelSubscription(userToCancel);

break;

case 4:

service.displayUsers();

break;

case 5:

service.displayPlans();

break;

case 6:

System.out.println("Exiting...");

scanner.close();

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please try again.");

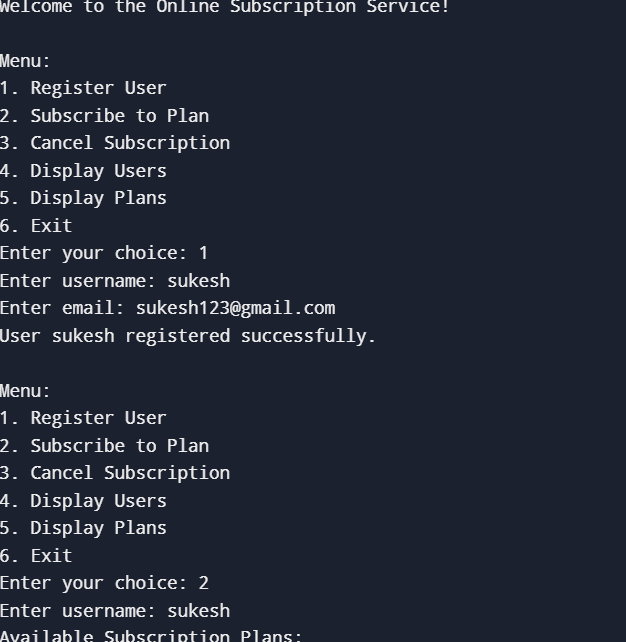
}

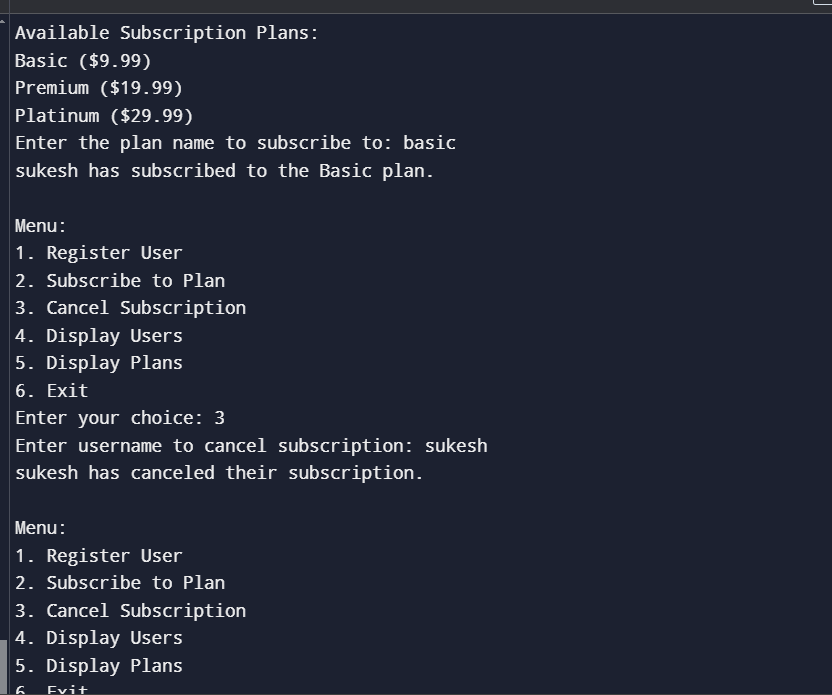
}

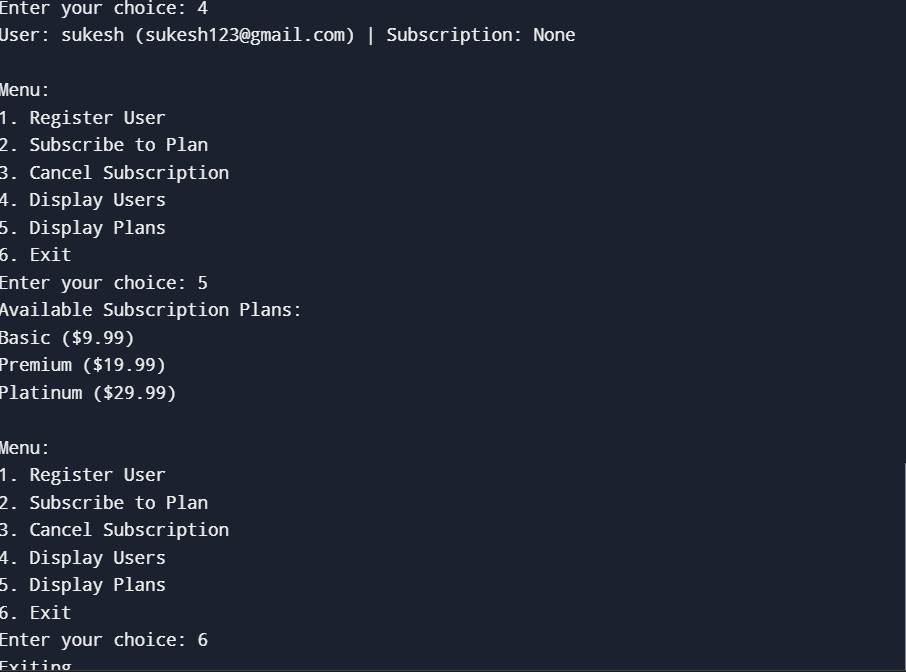
}

}

OUTPUT:







REFERENCES:

❖ Zendesk Blog on Subscription Management Best Practices: Provides tips and best practices for managing subscriptions effectively.

❖ How to Design a Subscription Business Model: Insights into designing and implementing a subscription business model.

❖ Building a Subscription Service with Stripe: Stripe’s documentation for integrating subscription billing.

Authentication and Security Best Practices: Guide on securing user accounts and authentication mechanisms.

❖ An Introduction to Digital Rights Management (DRM): Overview of DRM and its importance for content protection.