SIMATS ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CAPSTONE PROJECT REPORT

PROJECT TITLE

Online Customer Support System

REPORT SUMITTED BY

B. Venu Gopal(192210065)

REPORT SUMITTED TO

DR. Jayashakthi Velmurugan

COURSE CODE / COURSE

CSA0912 PROGRAMMING IN JAVA FOR ACCESSING DATABASE

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI - 602105

TAMIL NADU, INDIA

SAVEETHA
INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
(Declared as Deemed to be University under Section 3 of UGC Act 1956)

BONAFIDE CERTIFICATE

This is to certify that the project report entitled Online Customer Support System submitted by B. Venu Gopal(192210065) to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.K.Jayasakthi Velmurugan

COURSE FACULTY

Department of Deep Learning.

Saveetha School of Engineering,

SIMATS, Chennai - 602105

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor Dr. N M VEERAIYAN, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics Dr. DEEPAK NALLASWAMY, SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director Dr. RAMYA DEEPAK, Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to Dr. B RAMESH, Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty Dr.K.Jayasakthi Velmurugan, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

ABSTRACT:

Online Customer Support System is a scalable, cloud-based platform that enables users to easily raise, track, and resolve support queries related to various services. The platform offers secure user authentication, flexible ticket management, and automated routing, ensuring a smooth and user-friendly support 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 support tickets, resolving user issues, and generating detailed analytical reports on customer interactions. 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.This version highlights key features relevant to a customer support system while maintaining the cloud-based infrastructure and management functionalities.

INTRODUCTION:

The Online Customer Support System provides a comprehensive platform for users to access support for a wide range of digital services, including streaming, learning modules, and software tools. The platform streamlines the customer support process by offering an intuitive interface for submitting support tickets, tracking issue resolution, and managing user queries. Built on a robust cloud-based infrastructure using Amazon Web Services (AWS), the system ensures scalability, reliability, and enhanced performance to handle varying customer support demands.Security is a key focus, with the platform implementing secure authentication and communication methods to protect user interactions and data. Administrators benefit from advanced features such as ticket management, automated issue routing, and detailed report generation to track support efficiency and customer satisfaction. Designed to deliver a seamless user experience, the system also supports real-time updates, notifications, and automated ticket escalations, ensuring that both users and administrators can efficiently manage their support needs.By leveraging the flexibility of cloud computing, this platform is well-suited for businesses offering digital services and seeks to provide a scalable and secure solution for customer support management.This version emphasizes the support management aspects while maintaining the cloud-based architecture and scalability features.

PROBLEM STATEMENT:

In today's digital landscape, businesses offering customer support services face challenges in managing a growing user base, efficiently resolving issues, and providing an intuitive experience for both customers and administrators. Traditional customer support systems often lack scalability, real-time updates, and the ability to efficiently handle support for multiple services. Additionally, maintaining data security and providing seamless integration with cloud infrastructure can be complex and costly.This Online Customer Support System aims to address these issues by developing a cloud-based platform that allows users to easily raise and track support queries while ensuring secure

communication and data privacy. The platform must provide administrators with tools for managing support tickets, 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 support management, and support business growth without compromising on security or performance.This version highlights the customer support focus, incorporating the necessary features like ticket management, dispute resolution, and secure cloud-based infrastructure.

OBJECTIVE:

Secure Communication: Ensure secure interactions and data protection using encryption and authentication techniques for safeguarding user information.

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

Real-Time Ticket Updates: Provide real-time updates for ticket status, issue resolution, and notifications for both users and administrators.

Administrative Control: Equip administrators with robust tools for managing support tickets, verifying user details, resolving disputes, and generating detailed performance reports.

Automated Ticket Assignment: Implement automated processes for ticket routing and escalation to ensure quick response times and operational efficiency..

MATERIALS AND METHODS:

Frontend Design:

- Home Page: Header with navigation, key support features, and call-to-action buttons, ending with a footer.
- Support Catalog: Search bar, filters, and a list or grid of common issues with "Submit a Ticket" buttons.
- Ticket Detail Page: Detailed issue information, ticket status updates, and FAQs with a prominent "Submit a Ticket" or "Contact Support" button.
- User Dashboard: Profile, support history, ticket status, communication logs, and account settings.
- Ticket Submission Page: Summary of the issue, support category selection, secure communication options, and ticket confirmation.

Backend Design:

Database Management:

- Use AWS RDS for managing customer support data.
- Implement CRUD operations for support tickets, user data, and communication logs.

Ticket Management:

- Handle ticket creation, status updates, escalations, and resolutions.
- Integrate communication tools for real-time customer interaction.

Reporting:

- Generate reports on ticket metrics, user activity, response times, and resolution history.

AWS Integration:

AWS RDS (Relational Database Service):

- Database Management: Use RDS to manage relational databases for storing ticket data, user information, and support communication history.
- 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 attachments or support-related media, in S3 buckets.
- Data Backup: Use S3 for additional backup and archival storage of support logs.

AWS Lambda:

- Serverless Functions: Implement serverless functions for tasks like ticket creation, automated notifications, and real-time ticket status updates.
- Event-Driven Execution: Use Lambda for handling events such as ticket escalations, support resolution, or user interactions.

AWS API Gateway:

- API Management: Create and manage APIs for ticket management, support communication, and reporting.

- Integration with Lambda: Route API requests to Lambda functions for processing support interactions.

AWS SNS (Simple Notification Service):

- Notifications: Send email or SMS notifications for ticket updates, resolution confirmations, and system alerts.

AWS CloudWatch:

- Monitoring & Logging: Monitor system performance, log errors, and set up alarms for issues like delayed responses or high ticket volume.

AWS IAM (Identity and Access Management):

- Security: Manage user permissions and secure access to AWS resources for the support system and administrators.

CONCLUSION:

The development of an Online Customer Support System entails creating a user-friendly and secure platform for managing and addressing customer queries and issues. 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 ticket tracking, secure user authentication, and comprehensive reporting on customer support metrics and activities. By integrating cloud-based architecture and focusing on user-centric design, this project delivers a robust and efficient solution for managing customer support, catering to both users and administrators with a streamlined and reliable experience.This version maintains the structure and focus on AWS, scalability, and security while adapting the context to customer support management.

CODE:

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import java.sql.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;

import software.amazon.awssdk.services.s3.S3Client;

import software.amazon.awssdk.services.s3.model.PutObjectRequest;
```

```
import software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;

import java.nio.file.Paths;

public class CustomerSupportSystem extends Applet implements ActionListener {
    // Frontend UI Elements
    TextField username, ticketDetails;
    Label label1, label2;
    Button submitTicket, viewStatus;
    TextArea responseArea;
    TicketManager ticketManager;

    // AWS S3 and Cognito integration classes
    S3Storage s3Storage;
    CognitoAuth cognitoAuth;

    public void init() {
        // Initialize frontend
        setLayout(new FlowLayout());
        label1 = new Label("Username:");
        username = new TextField(20);
        label2 = new Label("Ticket Details:");
        ticketDetails = new TextField(20);
        submitTicket = new Button("Submit Ticket");
```

```
    viewStatus = new Button("View Ticket Status");
    responseArea = new TextArea(10, 30);

    add(label1);
    add(username);
    add(label2);
    add(ticketDetails);
    add(submitTicket);
    add(viewStatus);
    add(responseArea);

    submitTicket.addActionListener(this);
    viewStatus.addActionListener(this);

    // Initialize backend and AWS integration
    ticketManager = new TicketManager();
    s3Storage = new S3Storage();
    cognitoAuth = new CognitoAuth();
}

public void actionPerformed(ActionEvent e) {
    String user = username.getText();
    String details = ticketDetails.getText();

    if (e.getSource() == submitTicket) {
```

```
            ticketManager.submitTicket(user, details);
            responseArea.setText("Ticket Submitted. Ticket ID: 12345");
        } else if (e.getSource() == viewStatus) {
            String status = ticketManager.getTicketStatus(12345); // Example ticket ID
            responseArea.setText("Ticket Status: " + status);
        }
    }

    // Backend Ticket Manager for Database Operations
    class TicketManager {
        private final String DB_URL = "jdbc:mysql://your-rds-endpoint:3306/support_db";
        private final String USER = "admin";
        private final String PASS = "password";

        public Connection connect() throws Exception {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(DB_URL, USER, PASS);
        }

        public void submitTicket(String username, String details) {
            try {
                Connection conn = connect();
                String query = "INSERT INTO tickets (username, details, status) VALUES (?, ?, ?)";
                PreparedStatement stmt = conn.prepareStatement(query);
                stmt.setString(1, username);
```

```
        stmt.setString(2, details);
        stmt.setString(3, "Open");
        stmt.executeUpdate();
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public String getTicketStatus(int ticketId) {
    try {
        Connection conn = connect();
        String query = "SELECT status FROM tickets WHERE id=?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setInt(1, ticketId);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            return rs.getString("status");
        }
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "Ticket Not Found";
}
```

```
}

class S3Storage {
    private static final String BUCKET_NAME = "support-system-bucket";

    public void uploadFile(String filePath) {
        S3Client s3 = S3Client.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        PutObjectRequest request = PutObjectRequest.builder()
                .bucket(BUCKET_NAME)
                .key("support-attachments/" + Paths.get(filePath).getFileName().toString())
                .build();

        s3.putObject(request, Paths.get(filePath));
    }
}

// AWS Cognito for User Authentication
class CognitoAuth {
    private static final String CLIENT_ID = "your-cognito-client-id";

    public void signUp(String username, String password) {
        CognitoIdentityProviderClient cognitoClient = CognitoIdentityProviderClient.builder().build();
        SignUpRequest signUpRequest = SignUpRequest.builder()
```

```
                .clientId(CLIENT_ID)
                .username(username)
                .password(password)
                .build();

        cognitoClient.signUp(signUpRequest);
    }
  }
}}
```

OUTPUT:

Output

```
 OUTPUT:
SUBMIT TICKET:
Username: JohnDoe
Ticket Details: I'm unable to log in to my account.
[Button: Submit Ticket]
Response:
Ticket Submitted. Ticket ID: 12345
VIEW TICKET STSTUS:
Username: JohnDoe
Ticket ID: 12345
[Button: View Ticket Status]
Response:
Ticket Status: Open
```

REFERENCES:

❖ Zendesk Blog on Subscription Management Best Practices: Offers practical advice and strategies for efficiently managing subscription services and enhancing customer experience..

❖ How to Design a Subscription Business Model: Provides insights and guidelines for creating and implementing a successful subscription-based business model.

❖ Building a Subscription Service with Stripe: Comprehensive documentation from Stripe on integrating and managing subscription billing processes using their platform.

❖ An Introduction to Digital Rights Management (DRM): An overview of DRM practices and their significance for protecting digital content in subscription services..