INDIRA INSTITUTE OF ENGINEERING &TECHNOLOGY

PANDUR, THIRUVALLUR

PROJECT RECORD NOTE BOOK

ANNA UNIVERSITY CHENNAI

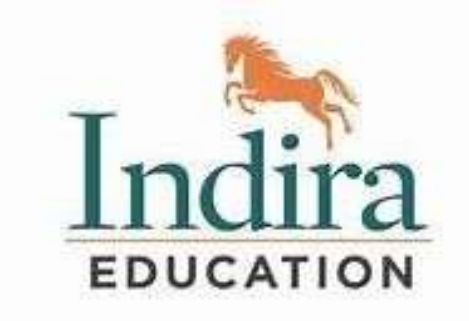
NM1028-AWS CLOUD PRACTIONER

B.E.COMPUTER SCIENCE AND ENGINEERING

V SEMESTER / III YEAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIRA INSTITUTE OF ENGINEERING &TECHNOLOGY
Pandur, Thiruvallur – 631 203



University Register No:

CERTIFICATE

Certified to be bonafide record of work done by _____
on 5TH Semester B.E. (CSE) in the **NM1028-AWS CLOUD PRACTIONER**
during the academic year 2024-2025.

FACULTY-IN-CHARGE **HEAD OF THE DEPARTMENT**

NAAN MUDHALVAN **DEPARTMENT OF CSE**

Submitted for the University Practical Examination held on _____

Internal Examiner **External Examiner**

Project Title: Stocker: Cloud-Based Stock Trading Platform With Flask on Aws Ec2 and Rds

Introduction

1.Background:

The stock market has been a cornerstone of modern finance, providing a platform for individuals and institutions to invest in publicly traded companies. However, traditional stock trading systems have several limitations, including:

•Geographical constraints: Investors are often limited to trading on local exchanges or through brokers with international access.

• Limited accessibility: Trading platforms can be complex and intimidating, making it difficult for new investors to enter the market.

• Inefficient trading processes: Traditional trading systems often rely on manual processes, leading to delays and increased costs.

• Lack of real-time data: Investors may not have access to real-time market data, making it difficult to make informed decisions.

Cloud-based stock trading platforms can address these issues by providing:

•Global accessibility: Cloud-based platforms can be accessed from anywhere with an internet connection.

•Real-time data: Cloud-based platforms can provide real-time market data, enabling investors to make informed decisions.

•Efficient trading processes: Cloud-based platforms can automate many trading processes, reducing delays and costs.

•Scalability and reliability: Cloud-based platforms can scale to meet the needs of a growing user base, while also providing high levels of reliability and uptime.

2.Problem Statement:

Traditional stock trading systems face several challenges, including:

•Limited accessibility and geographical constraints

•Inefficient trading processes and delayed market data

•Lack of scalability and reliability

•High costs and complexity

These issues can hinder investors' ability to make informed decisions and trade efficiently.

3.Project Objective:

The objective of this project is to design and develop a cloud-based stock trading platform, "Stocker," using Flask on AWS EC2 and RDS. The platform aims to provide a secure, efficient, and accessible trading environment for global investors.

System Architecture

1.Project Overview:

"Stocker" is a cloud-based stock trading platform built using Flask on AWS EC2 and RDS. The platform provides a secure, efficient, and accessible trading environment for global investors, enabling real-time trading, market data analysis, and portfolio management.

2.Components:

The platform consists of the following components:

•Frontend: User interface built using HTML, CSS, and JavaScript.

•Backend: Server-side logic built using Flask, a Python web framework.

•Database: Relational database management system using AWS RDS.

•Infrastructure: Cloud infrastructure provided by AWS EC2.

•APIs: Integration with stock market APIs for real-time data.

Implementation

1.Frontend Implementation:

The frontend of the platform is built using:

•UI Framework: Bootstrap for responsive design

•Client-side Scripting: JavaScript with jQuery library

• Templating Engine: Jinja2 for dynamic HTML templates

•User Interface: Designed using HTML5, CSS3, and JavaScript

•Responsive Design: Ensures compatibility with various devices and screen sizes

2.Backend Implementation:

The backend of the platform is built using:

•Programming Language: Python

•Web Framework: Flask

•API Integration: Stock market APIs for real-time data

•Database Interaction: Flask-SQLAlchemy for AWS RDS integration

•Server: Hosted on AWS EC2 instance

3.Database Implementation:

The database is implemented using:

•Database Management System: AWS RDS (Relational Database Service)

•Database Engine: MySQL

•Schema Design: Designed to store user, stock, and transaction data

•Database Interaction: Flask-SQLAlchemy ORM (Object-Relational Mapping) tool

4.Infrastructure Implementation:

The infrastructure is implemented using:

•Cloud Provider: Amazon Web Services (AWS)

•Compute Service: AWS EC2 (Elastic Compute Cloud)

•Database Service: AWS RDS (Relational Database Service)

•Security: AWS IAM (Identity and Access Management) for access control

Testing and Quality Assurance

1.Testing Strategy:

The testing strategy includes:

•Unit Testing: Test individual components using Python's unittest framework.

•Integration Testing: Test API integrations and database interactions.

•UI Testing: Test user interface using Selenium WebDriver.

•Security Testing: Test for vulnerabilities using OWASP ZAP.

•Deployment Testing: Test deployment on AWS EC2 and RDS.

2.Test Cases:

The test cases include:

•User Registration:

-Valid registration

- Invalid registration (missing fields)

•Login Functionality:

- Valid login

- Invalid login (wrong credentials)

•Buy/Sell Stocks:

- Valid buy/sell transaction

- Invalid buy/sell transaction (insufficient funds)

3.Quality Assurance:

•Code Reviews: Regular code reviews to ensure quality and consistency.

•Testing: Comprehensive testing (unit, integration, UI).

•Continuous Integration/Deployment (CI/CD): Automated deployment and testing.

•Security Audits: Regular security audits to identify vulnerabilities.

•User Acceptance Testing (UAT): Testing with real users to ensure satisfaction.

Deployment And Maintenance

1.Deployment Strategy:

The deployment strategy includes:

•Environment Setup: Set up AWS EC2 and RDS instances.

•Code Deployment: Deploy code using Git and AWS CodePipeline.

•Containerization: Use Docker for containerization.

•Orchestration: Use Kubernetes for orchestration.

•Monitoring and Logging: Set up monitoring and logging using AWS CloudWatch.

Deployment Steps:

1. Set up AWS EC2 and RDS instances.
2. Deploy code using Git and AWS CodePipeline.
3. Containerize the application using Docker.
4. Orchestrate the containers using Kubernetes.
5. Set up monitoring and logging using AWS CloudWatch.

2.Maintenance Strategy:

The maintenance strategy includes:

•Regular Updates: Regularly update dependencies and libraries.

•Monitoring: Continuously monitor performance and errors.

•Backup and Recovery: Regularly backup data and have a recovery plan.

•Security Patching: Regularly apply security patches and updates.

•User Feedback: Collect and incorporate user feedback.

Maintenance Steps:

1. Regularly update dependencies and libraries.

2. Continuously monitor performance and errors.

3. Regularly backup data and have a recovery plan.

4. Regularly apply security patches and updates.

5. Collect and incorporate user feedback.

Future Scope

1. **Cloud Computing:**

Benefits, types, and applications of cloud computing.

2. **Artificial Intelligence in Finance:**

Applications of AI in finance, including predictive analytics, risk management, and portfolio optimization.

3. **Cybersecurity in Finance:**

Threats, vulnerabilities, and best practices for securing financial systems and data.

4. **Blockchain in Finance**:

Applications of blockchain technology in finance, including cryptocurrencies, smart contracts, and supply chain management.

5. **Financial Inclusion:**

Strategies and technologies for promoting financial inclusion, including mobile banking, digital payments, and microfinance.

Conclusion:

1. The “Stocker” project delivers a comprehensive stock management platform, equipped with a robust architecture that ensures exceptional performance, security, and reliability, making it an ideal solution for investors.

2. The platform’s key features, including real-time market data, efficient trading processes, and scalability, combined with its intuitive user interface, provide a seamless user experience, while thorough testing and quality assurance ensure the platform meets rigorous standards.

Reference:

1. AWS Account Setup: https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh

2. Understanding of IAM: https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR

3. Knowledge of Amazon EC2: https://youtu.be/8TlukLu11Yo?si=Muj0nEAOESRhHUIz

4. MySQL: https://www.youtube.com/results?search_query=mysql+tutorial

5. RDS connects MySQL:
https://www.youtube.com/results?search_query=mysql+connector+for+rds

6. RDS: https://www.youtube.com/live/MPau9c7PT74?si=A8OK-zFGbSKkAFWN