**I**

|  |  |  |
| --- | --- | --- |
|  | **NAAN MUDHALVAN** |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Stocker: Cloud-Based Stock Trading Platform With Flask On Aws Ec2 And Rds**

**Project Created by:**

**Project Created Date: 21/Nov/2024**

**College Code: 1106**

**College Name: Indira institute of engineering and technology**

**Team Name:**

***BONAFIDE CERTIFICATE***

**Certified that this Naan Mudhalvan project report “FreshBasket: Scalable E-commerce Platform Deployment with Flask on AWS EC2 and RDS” is the bonafide work of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ who carried out the project work under my supervision.**

**SIGNATURE SIGNATURE**

**Project Coordinator SPoC**

**Naan Mudhalvan Naan Mudhalvan**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**Introduction**

**Background:**

The stock market has limitations, including geographical constraints, limited accessibility, inefficient trading processes, and lack of real-time data. Cloud-based stock trading platforms can address these issues by providing:

* Global accessibility
* Real-time data
* Efficient trading processes
* Scalability and reliability

Our project, Stocker, aims to design and develop a cloud-based stock trading platform using Flask on AWS EC2 and RDS, providing a secure, efficient, and accessible trading environment for global investors.

1. **Problem Statement:**

Traditional stock trading systems face challenges such as:

* Limited accessibility and geographical constraints
* Inefficient trading processes and delayed market data
* Lack of scalability and reliability

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

1. **Project Objective:**

Design and develop a cloud-based stock trading platform, “Stocker”, using Flask on AWS EC2 and RDS, 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. It aims to provide a secure, efficient, and accessible trading environment for global investors, enabling real-time trading, market data analysis, and portfolio management.

1. **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:**

* **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

1. **Backend Implementation:**

* **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

1. **Database Implementation:**

* **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

1. **Infrastructure Implementation:**

* **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 to Assurance**

1. **Testing Strategy:**

* **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.

1. **Test Cases;**

* **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)

1. **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:**

* **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.

1. **Maintenance Strategy:**

* **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.

**Conclusion**

The “Stocker” project aims to provide a comprehensive stock management platform. With a robust frontend, backend, database, infrastructure, testing, quality assurance, deployment, and maintenance strategy in place, the project is well-equipped to handle user demands and provide a seamless experience.