IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Online Trading Website

[1]Kshirsagar Akash, [2]Prof. Kalbande R.M., [3]Bhore Abhijeet, [4]Kolekar Rohan, [5]Maske Ashutosh
[1]Student, [2]Teacher, [3]Student, [4]Student, [5]Student
[1]Bhagwant Institute Of Technology,Barshi,
[2]Bhagwant Institute Of Technology,Barshi,
[3]Bhagwant Institute Of Technology,Barshi,
[4]Bhagwant Institute Of Technology,Barshi,
[5]Bhagwant Institute Of Technology,Barshi

CHAPTER 1

SYNOPSIS

1.1 Project Title : Online Trading Website

1.2 Internal Guide: Prof . Kalbande R.M

1.3 Technical Keywords:

- Online Trading System
- Stock Market Platform
- Real-time Data Processing
- Financial Transactions
- Trading Engine

1.4 Problem Statement

In today's fast-growing digital world, traditional methods of trading in financial markets are often slow, complex, and not easily accessible to common users. Many individuals face difficulties in understanding market trends, executing trades efficiently, and managing their investments due to lack of user-friendly platforms and real-time information.There is a need for a secure, reliable, and easy-to-use online system that allows users to perform trading activities such as buying and selling assets, tracking market data, and managing their portfolio from anywhere at any time.The problem is to design and develop an **Online**

Trading Website that provides real-time market updates, secure transactions, and an intuitive interface to help users make informed trading decisions efficiently.

1.5 Abstract

The Online Trading Website is a web-based platform designed to facilitate the buying and selling of financial assets such as stocks through the internet. The system aims to provide users with a secure, efficient, and user-friendly interface to perform trading activities in real time.This project focuses on overcoming the limitations of traditional trading methods by offering features like user registration and authentication, live market data updates, portfolio management, and transaction tracking. The platform integrates with external APIs to fetch real-time stock market information, enabling users to make informed investment decisions.The system is developed using modern web technologies such as HTML, CSS, JavaScript for the frontend, and backend technologies like Java/PHP/Node.js with a MySQL database for efficient data storage and management. Security measures such as data encryption and authentication mechanisms are implemented to ensure safe transactions.The proposed system enhances accessibility, reduces manual effort, and provides a fast and reliable trading experience. Future enhancements may include advanced analytics, AI-based recommendations, and mobile application support to further improve the system's functionality and user engagement.

1.6 Goals and Objective

Goals:

- To develop a secure and efficient **Online Trading Website** for financial transactions
- To provide users with a platform for buying and selling assets in real time
- To simplify the trading process through a user-friendly interface
- To ensure fast and reliable access to market data
- To improve accessibility so users can trade anytime and anywhere

Objectives:

- To design and implement a **user registration and authentication system**
- To integrate **real-time market data APIs** for live stock updates
- To develop a **buy and sell module** for executing trades
- To create a **portfolio management system** to track user investments
- To maintain a **transaction history** for all trading activities
- To ensure **data security** using encryption and secure login methods
- To build a **responsive frontend interface** using modern web technologies
- To implement a **database system** for storing user and trading data efficiently

1.7 Names Of Conferences/Journals where papers can be published

International Journal of Computer Applications (IJCA – Foundation of Computer Science)

- Indexed, peer-reviewed, and widely accepted for final year project publications.

International Journal of Scientific & Engineering Research (IJSER)

- Accepts system-based innovations and Android app research.

International Journal of Emerging Technologies and Innovative Research (JETIR)

- Recognized by UGC and welcomes student-led publicatio

Plan of Project Execution

The development of the Online Trading Website will follow a structured approach to ensure timely completion and proper functionality. The execution plan is divided into the following phases:

1. **Requirement Analysis**
   - Identify user needs and system requirements
   - Study existing trading platforms and features
2. **System Design**
   - Prepare system architecture, database design, and UI layout
   - Design modules like login, trading, and portfolio management
3. **Development Phase**
   - Frontend development using HTML, CSS, and JavaScript
   - Backend development using suitable technology (Java/PHP/Node.js)
   - Integration of database (MySQL) and APIs for market data
4. **Testing Phase**
   - Perform unit testing and integration testing
   - Identify and fix bugs or errors
5. **Implementation & Deployment**
   - Deploy the system on a server
   - Make the website accessible to users
6. **Maintenance & Updates**
   - Monitor system performance
   - Provide updates and add new features as required

CHAPTER 2

TECHNICAL KEYWORDS

2.1 Area of project

The Online Trading Website project falls under the domains of **Web Development** and Financial Technology. It involves creating an online platform for trading financial assets such as stocks using internet-based technologies.The project also covers areas like database management, secure transactions, real-time data processing, and user interface design, making it a combination of software development and financial services.

2.2 Technical Keywords

- Online Trading System
- Web Development
- Financial Technology (FinTech)
- Real-time Data Processing
- REST API
- Database Management System (DBMS)
- MySQL
- HTML, CSS, JavaScript
- Backend Development (Java / PHP / Node.js)

CHAPTER 3

INTRODUCTION

3.1 Introduction

The Online Trading Website is a web-based platform that enables users to buy and sell financial assets such as stocks through the internet. With the growth of digital technology, traditional methods of trading have been replaced by faster and more efficient online systems. This project is based on the domain of Financial Technology, which combines finance and modern web technologies.

The system provides features such as user registration and authentication, real-time market data, buying and selling of assets, and portfolio management. It allows users to track their investments and perform transactions from anywhere at any time.

The main aim of this project is to create a secure, user-friendly, and efficient trading platform that simplifies investment activities. By using technologies like HTML, CSS, JavaScript, backend frameworks, and databases, the system ensures smooth operation and data security.

3.2 Project Idea And Motivation

The idea behind the Online Trading Website is to create a simple, secure, and efficient platform that allows users to perform trading activities online without the need for traditional brokers. With the rapid growth of digital technology and the increasing popularity of online investments, there is a strong demand for user-friendly trading systems.

The motivation for this project comes from the need to make trading accessible to everyone, including beginners who may find existing platforms complex. By using concepts from Financial Technology, the project aims to provide real-time market information, easy navigation, and secure transactions.

This system is designed to reduce manual effort, improve speed, and help users make informed financial decisions, thereby enhancing the overall trading experience.

3.3 System Architecture

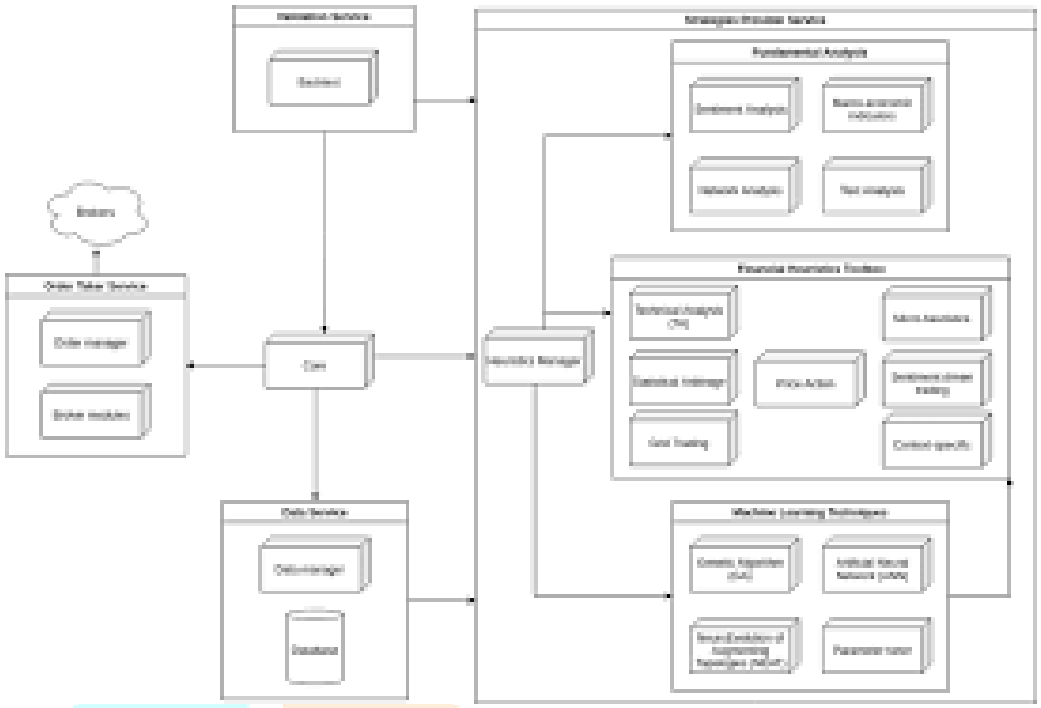


Fig.System Architecture

3.4 Literature Survey

The literature survey focuses on studying existing online trading systems and technologies used in the field of Financial Technology. Various research papers, websites, and applications were analyzed to understand their features, advantages, and limitations.

Popular trading platforms such as Zerodha, Upstox, and Groww provide functionalities like real-time market data, portfolio management, and secure transactions. These systems use modern technologies such as REST APIs, cloud computing, and advanced security mechanisms.

From the study, it is observed that while existing systems are powerful, they can be complex for beginners and may require improvements in user experience and accessibility. This project aims to overcome these limitations by designing a simple, user-friendly interface along with essential trading features.

Description –

The Online Trading Website is a web-based application designed to allow users to buy and sell financial assets such as stocks through an internet platform. It provides a simple and secure interface where users can register, log in, view real-time market data, and perform trading activities efficiently.

The system includes features like portfolio management, transaction history, and live updates using APIs. It is developed using modern web technologies and follows the principles of Financial Technology to ensure fast, reliable, and secure transactions.

Overall, the project aims to simplify trading operations, improve accessibility, and provide users with a smooth and user-friendly trading experience.

CHAPTER 4

PROBLEM DEFINITION AND SCOPE

4.1 Problem Statement

In the current financial market, traditional trading methods are often time-consuming, complex, and not easily accessible to all users. Many existing online trading platforms are difficult for beginners to understand and may lack simplicity in design and usability.

There is a need for a system that provides a secure, user-friendly, and efficient way to perform trading activities such as buying and selling financial assets with real-time data access.

The problem is to design and develop an Online Trading Website in the domain of Financial Technology that simplifies trading processes, ensures secure transactions, and allows users to manage their investments easily from anywhere at any time.

4.2 Project Goals

- To develop a secure and reliable Online Trading Website for financial transactions
- To provide a simple and user-friendly interface for all types of users
- To enable real-time buying and selling of financial assets
- To offer accurate and fast market data updates
- To allow users to manage and track their investment portfolio
- To ensure data security and privacy using modern techniques
- To make trading accessible anytime and anywhere through the internet
- To build the system using concepts from Financial Technology

4.3 Project Objectives

- To design and implement a **user registration and login system**
- To provide **secure authentication and authorization** for users
- To integrate **real-time market data** using APIs
- To develop a **buy and sell module** for trading activities
- To create a **portfolio management system** to track investments
- To maintain a **transaction history** for all user activities
- To ensure **data security** using encryption and secure protocols
- To build a **responsive and user-friendly interface**
- To efficiently store and manage data using a database system
- To apply concepts of Financial Technology for modern trading solutions

4.4 Application

- Used by individuals to buy and sell financial assets such as stocks
- Helps users in portfolio management and tracking investments
- Provides real-time market analysis for better decision-making
- Useful for beginners to learn and practice online trading
- Can be used by financial institutions for offering online trading services
- Enables secure and fast financial transactions over the internet
- Supports anytime, anywhere access to trading platforms

4.5 Hardware and Software Resources required

4.5.1 Hardware

- A computer system or laptop with at least **Intel i3 (or equivalent) processor**
- Minimum **4 GB RAM** (8 GB recommended for better performance)
- At least **500 GB hard disk or SSD storage**
- Stable **internet connection** for real-time data access
- Standard **input/output devices** (keyboard, mouse, monitor)
- Server machine (optional) for hosting the application

4.5.2 Software

- Operating System: Windows / Linux / macOS
- Frontend Technologies: HTML, CSS, JavaScript
- Backend Technologies: Java / PHP / Node.js
- Database Management System: MySQL
- Web Browser: Google Chrome / Mozilla Firefox
- Development Tools: VS Code / Eclipse / NetBeans
- API Tools: REST API / JSON for data exchange
- Version Control: Git / GitHub (optional)

CHAPTER 5

PROJECT PLAN

5.1 Project Estimation

project estimation for the Online Trading Website includes the calculation of required time, cost, and resources needed for successful development and implementation. The system is expected to be developed within a defined schedule by dividing the work into phases such as requirement analysis, design, development, testing, and deployment.

The cost estimation mainly includes software tools (mostly open-source), development environment setup, and system hosting if required. Since the project is web-based and uses technologies from Financial Technology, most of the tools are freely available, reducing overall cost.

Effort estimation is based on the complexity of modules like user authentication, trading system, database management, and API integration. Proper planning ensures efficient use of time and resources, leading to successful project completion.

1. Requirement Gathering and Analysis

Requirements were gathered through:

- Informal discussions with homeowners.
- Interaction with local service providers.
- Study of existing service platforms.
- Identifying common problems in booking home services.

2. System Design

- Designed system architecture: front-end, back-end and database modules.
- Created ER Diagrams, Data Flow Diagrams (DFDs), and system flowcharts

3. Implementation (Coding

- Developed the front-end using Python and PyQt5 for user Interface
- Implemented emotion detection using OpenCV and Deep Learning
- Integrated Spotify/YouTube API For mood-based music Recommended

4. Testing

- Conducted **unit testing for mood detection, music recommendation and login modules.**
- Performed **integration testing** to ensure seamless interaction between frontend, backened.
- Validated system accuracy by simulating different emotional inputs and analyzing recommended song output

5. Deployment

- Deployed the Mood Tune Web Based application local build generation
- Ensured compatibility across the user with real time music synchronization
- Prepared user documentation explaining mood selection, playlist generation and account management features

6. Maintenance

- Monitored application performance and resolved bugs reported by users after deployment
- Optimized recommendation logic to improve song accuracy and response time
- Planned feature enhancements such as AI-based mood detection, Spotify integration, dark mood UI, personalized playlist.

5.2 Project Resources

1. Hardware Resources

- Computer/Laptop with minimum required configuration
- Stable internet connection
- Server system (for deployment, if required)

2. **Software Resources**

- Operating System: Windows / Linux
- Frontend: HTML, CSS, JavaScript
- Backend: Java / PHP / Node.js
- Database: MySQL
- Tools: VS Code / Eclipse / NetBeans
- Web Browser: Chrome / Firefox

3. **Human Resources**

- Project Developer(s)
- Database Designer
- UI/UX Designer (optional)
- Tester/Debugging support

5.3 Risk Management w.r.t NP Hard analysis

5.3.1 Risk Identification

In systems or projects where NP-hard problems are involved (such as scheduling, routing, optimization, or resource allocation), risk identification focuses on recognizing uncertainties and limitations arising from computational complexity, approximation constraints, and scalability issues.
Key risks include:

1.Computational Infeasibility Risk
NP-hard problems do not have known polynomial-time solutions. As input size grows, exact solutions may become impractical due to exponential time complexity, leading to unacceptable processing delays or system timeouts.

2. Suboptimal Solution Risk
Because exact optimization may be infeasible, heuristic or approximation methods are often used. These approaches may produce solutions that are not optimal, potentially impacting system efficiency, cost, or performance.

3.Scalability Risk
Algorithms that perform adequately on small datasets may fail to scale effectively when data volume increases, leading to degraded performance or system instability.

4.Resource Exhaustion Risk
High computational demands can result in excessive CPU usage, memory consumption, or energy usage, potentially affecting other system processes or leading to infrastructure overload.

5. Algorithm Selection Risk
Choosing an inappropriate heuristic, metaheuristic, or approximation algorithm may result in poor convergence, instability, or inconsistent outputs.

6.Data Sensitivity Risk
NP-hard solutions can be highly sensitive to input data changes. Minor variations in data may significantly alter outputs, reducing reliability.

7.Time Constraint Risk
In real-time or near-real-time applications, NP-hard problem solving may exceed acceptable time limits, making the solution unusable in operational settings.

8. Validation and Verification Risk
Since optimal solutions are often unknown, validating correctness and quality of heuristic outputs becomes challenging.

5.3.2 Risk Impact Definition

Risk impact definition describes the potential consequences that may arise if the identified risks in NP-hard problem contexts actually occur. Since NP-hard problems are computationally intensive and often rely on approximations, the impacts typically affect performance, accuracy, scalability, and system reliability.

The impacts can be defined as follows:

1.Operational Delay Impact
If computational infeasibility or time constraint risks occur, the system may fail to produce results within required deadlines, leading to delays in decision-making or process execution.

2. Solution Quality Degradation Impact
When suboptimal or heuristic solutions are used, the resulting outputs may deviate significantly from the optimal solution, reducing overall efficiency, increasing cost, or lowering performance of the system.

3. System Performance Degradation Impact
High resource consumption due to complex computations may slow down the system, affect parallel processes, or cause bottlenecks in shared computing environments.

4. Scalability Limitation Impact
As problem size increases, inability to scale effectively may restrict system usability, making it unsuitable for real-world or large-scale deployment scenarios.

5. Resource Overutilization Impact
Excessive CPU, memory, or energy usage may increase operational costs, reduce hardware lifespan, or lead to system crashes in extreme cases.

6. Unreliable Decision-Making Impact
Due to sensitivity in input data and approximation-based outputs, decisions derived from NP-hard solutions may become inconsistent or unreliable, affecting downstream processes.

7. System Instability Impact
Poor algorithm selection or poorly tuned heuristics may cause unpredictable behavior, including convergence failures or fluctuating results.

8. Compliance and Business Impact
In regulated or business-critical environments, delays or inaccurate outputs may lead to non-compliance, financial loss, or reduced stakeholder trust

5.4 Overview of Risk Management, Mitigation, monitoring

5.4.1 Risk Management Overview

Risk management refers to the systematic process of identifying, analyzing, and controlling risks that arise while solving NP-hard problems. These risks primarily relate to computational complexity, scalability limitations, and solution accuracy.

The key objectives of risk management include:

- Ensuring acceptable solution quality within practical time limits
- Maintaining system scalability for increasing input sizes
- Optimizing resource utilization (CPU, memory, and storage)
- Reducing uncertainty in heuristic or approximate solutions
- Supporting stable and predictable system behavior

In NP-hard environments, risk management is continuous rather than one-time due to dynamic input conditions and evolving computational demands.

5.4.2 Risk Mitigation Strategies

Risk mitigation involves applying methods to reduce the likelihood or impact of identified risks. In NP-hard problem scenarios, mitigation typically relies on algorithmic optimization and system-level controls.

Common mitigation strategies include:

- **Heuristic and Metaheuristic Algorithms**
  Using approaches such as genetic algorithms, simulated annealing, or greedy methods to obtain near-optimal solutions within feasible time limits.
- **Approximation Algorithms**
  Applying mathematically bounded approaches that guarantee solutions within a known factor of optimal.
- **Problem Decomposition**
  Breaking large NP-hard problems into smaller subproblems to reduce complexity.
- **Preprocessing and Data Reduction**
  Eliminating redundant or irrelevant data to reduce computational load.
- **Parallel and Distributed Computing**
  Using multiple processors or systems to reduce execution time.
- **Parameter Tuning and Optimization**
  Adjusting algorithm parameters to improve convergence and performance.
- **Fallback Mechanisms**
  Providing simpler heuristic solutions when computation exceeds time or resource thresholds.

5.4.3 Risk Monitoring

Risk monitoring ensures that risks remain under control during system operation and that mitigation strategies continue to be effective.

Key monitoring practices include:

- **Performance Monitoring**
  Tracking execution time, memory usage, and CPU utilization during algorithm execution.

- **Solution Quality Monitoring**
  Comparing heuristic outputs against benchmarks or historical best-known solutions.

- **Scalability Tracking**
  Observing system behavior as input size increases to detect degradation trends.

- **Threshold-Based Alerts**
  Setting limits for time, memory, or error deviation to trigger warnings or fallback mechanisms.

- **Logging and Audit Trails**
  Recording algorithm decisions and outputs for analysis and debugging.

- **Continuous Evaluation**
  Periodically reassessing algorithm effectiveness as data patterns or system requirements change.

5.5 Project Schedule

5.5.1 Project Task Set

The project task set defines the structured breakdown of activities required to successfully complete the project, particularly focusing on handling NP-hard problem analysis, risk management, and solution development. The tasks are organized in a logical sequence to ensuresmooth progression from requirement gathering to deployment and evaluation.

1. Requirement Analysis

- Identify problem domain involving NP-hard characteristics
- Collect functional and non-functional requirements
- Define constraints such as time, memory, and accuracy requirements
- Understand real-world use cases and input data characteristics

2. Problem Definition and Scope Finalization

- Clearly define the NP-hard problem to be addressed
- Set boundaries for solution feasibility
- Identify expected outputs and performance goals

3. Literature Review and Study

- Study existing NP-hard solutions and optimization techniques
- Analyze heuristic, approximation, and metaheuristic approaches
- Compare strengths and limitations of existing methods

4. System Design

- Design architecture for solving the NP-hard problem
- Define modules such as input processing, optimization engine, and output generation
- Select appropriate algorithms and data structures

5. Algorithm Development and Implementation

- Implement selected heuristic/approximation algorithms
- Develop optimization logic for improving solution quality
- Ensure modular and scalable code design

6. Testing and Validation

- Test algorithm on different dataset sizes
- Validate performance against expected constraints
- Compare outputs with baseline or known benchmarks

7. Risk Analysis and Optimization

- Identify performance bottlenecks
- Tune parameters for better efficiency and accuracy
- Apply mitigation strategies such as parallelization or decomposition

8. Performance Evaluation

- Measure time complexity behavior and resource usage
- Evaluate solution quality and scalability
- Document trade-offs between accuracy and computational cost

9. Documentation

- Prepare technical documentation of design and implementation
- Record risk management strategies and results
- Compile project report and user guidelines

10. Deployment and Final Review

- Deploy the final solution in the target environment
- Conduct final validation and system review
- Collect feedback for future improvements

5.6 Timeline Chart

- Weeks 1–2: Focus on requirement analysis and defining NP-hard problem scope.
- Weeks 2–3: Literature review to understand existing approaches.
- Weeks 3–4: System design and architecture finalization.
- Weeks 4–6: Core algorithm development and implementation.

- Weeks 5–7: Testing, validation, and iterative improvements.
- Weeks 6–7: Risk analysis and optimization of performance.
- Weeks 7–8: Performance evaluation and benchmarking.
- Weeks 8–10: Documentation, deployment, and final review.

5.6.1 Team Organization

Akash Kshirsagar	Developing frontend and backend, and implementing trading features.
Rohan Kolekar	Planning the project, assigning tasks, and managing overall progress.
Ashutosh Maske	Handling user support, maintaining system, and monitoring daily activities.
Abhijeet Bhore	Testing the system, finding bugs, and ensuring security of data and transactions.

5.6.2 Team Structure

The team structure defines the roles and responsibilities of individuals involved in the project. For a project dealing with NP-hard problem analysis and risk-managed optimization, a well-organized team is essential to ensure efficient development, testing, and evaluation of computational approaches.

Proposed Team Structure

1. Project Manager
- Oversees overall project planning and execution
- Ensures timely completion of milestones
- Coordinates between all team members
- Manages risk tracking and mitigation alignment

2. System Analyst
- Analyzes NP-hard problem requirements and constraints
- Defines problem scope and system objectives
- Works on feasibility study and requirement specification
- Identifies performance and scalability expectations

3. Algorithm Developer / Researcher
- Designs and implements heuristic, approximation, or metaheuristic algorithms
- Studies existing NP-hard solution techniques
- Optimizes algorithm performance for accuracy and speed
- Works closely with research literature and model improvemen

4. Software Developer / Implementer
- Converts algorithm designs into working code
- Develops system modules and integrates components
- Ensures code efficiency and modular structure
- Handles debugging and implementation issues

5. Tester / Quality Analyst

- Performs testing on different dataset sizes
- Validates correctness and performance of solutions
- Identifies bugs, bottlenecks, and inefficiencies
- Ensures system meets functional and non-functional requirements

6. Risk and Performance Analyst

- Identifies computational and operational risks
- Monitors system performance and resource usage
- Evaluates solution quality vs optimal benchmarks

CHAPTER 6

SYSTEM REQUIREMENT SPECIFICATION

6.1 Introduction

The System Requirement Specification (SRS) document provides a comprehensive description of the functional and non-functional requirements of the proposed system. The system is designed to address computationally intensive problems, particularly those classified as NP-hard, where finding exact solutions is often impractical due to exponential time complexity. As a result, the system focuses on efficient approximation, heuristic-based optimization, and risk-aware decision-making.

The primary purpose of this system is to provide a structured and scalable approach for solving complex optimization problems within acceptable time and resource constraints. It aims to balance solution quality with computational feasibility by applying advanced algorithms and risk management strategies.

This SRS serves as a reference for developers, testers, project managers, and stakeholders to ensure a clear understanding of system objectives, scope, constraints, and expected performance.

The system will support:

- Efficient handling of large-scale input data
- Application of heuristic and approximation algorithms
- Performance optimization under resource constraints
- Monitoring and mitigation of computational risks
- Generation of near-optimal solutions within practical time limits
    - The document further outlines functional requirements, non-functional requirements, system constraints, and assumptions necessary for successful implementation and deployment of the system.

6.1.1 Purpose and Scope of Document

Purpose

The purpose of this document is to provide a clear understanding of the online trading website project, including its features, functionality, system requirements, and development objectives. It serves as a reference for developers, designers, testers, stakeholders, and users involved in the project.

Scope

The scope of the document includes the overall design and operation of the online trading platform. It covers:

- User registration and authentication
- Trading dashboard and market data display
- Buy and sell order management
- Portfolio and transaction history
- Admin panel and user management

- Security features and payment integration

6.1.2 Overview of Responsibilities of Developer

The developer plays a central role in the design, implementation, testing, and maintenance of the system, especially in projects involving NP-hard problem solving where efficiency, optimization, and correctness are critical. The responsibilities extend beyond coding to include analysis, collaboration, and continuous improvement of system performance.

1. Requirement Understanding
- Study and interpret system requirements defined in the SRS document
- Understand NP-hard problem constraints, objectives, and expected outputs
- Clarify ambiguities with analysts or stakeholders before implementation

2. System Design Implementation
- Translate system architecture and algorithm designs into working modules

- Design efficient data structures suitable for large-scale computation
- Ensure modularity and scalability in code design

3. Algorithm Development
- Implement heuristic, approximation, or metaheuristic algorithms
- Optimize algorithms for time and space complexity
- Experiment with different approaches to improve solution quality

4. Integration of System Components
- Combine different modules such as input processing, optimization engine, and output generation
- Ensure smooth interaction between system components
- Resolve integration issues and compatibility problems

5. Testing and Debugging Support
- Perform unit testing of individual modules
- Identify and fix bugs, logical errors, and performance issues
- Support system testing and validation with testers

6. Performance Optimization
- Analyze computational bottlenecks in NP-hard problem solving
- Improve execution time and memory usage
- Apply optimization techniques such as caching, pruning, or parallel processing

7. Documentation Contribution
- Document code structure, logic, and algorithmic approaches
- Maintain technical comments and developer notes
- Support preparation of technical documentation and reports

8. Maintenance and Enhancement
- Update system based on new requirements or feedback
- Improve existing algorithms for better efficiency or accuracy

6.2 Overview of Responsibilities of Developer

The developer plays a central role in the design, implementation, testing, and maintenance of the system,

especially in projects involving NP-hard problem solving where efficiency, optimization, and correctness are critical. The responsibilities extend beyond coding to include ana

6.3 Model and Description

6.3.1 Data Description

The data description section defines the nature, structure, and characteristics of the data used in the system. Since the system deals with NP-hard problems, the data plays a critical role in determining computational complexity, solution quality, and overall system performance. The system processes structured and semi-structured input data depending on the problem domain (e.g., scheduling, routing, allocation, or optimization tasks).

1.Input Data Description

The input data represents the raw information provided to the system for processing and optimization.

- **Problem Parameters:** Define the constraints and objectives (e.g., cost, time, distance, capacity).
- **Entities/Nodes:** Represent elements such as tasks, locations, jobs, or resources.
- **Relationships:** Define dependencies or connections between entities (e.g., precedence constraints, graph edges).
- **Constraints Data:** Includes limits such as deadlines, budgets, capacities, or resource availability.

2. Data Characteristics

The system data typically exhibits the following characteristics:

- **Large Scale:** Data size may increase rapidly, affecting computational complexity
- **Highly Interconnected:** Relationships between data elements often form graphs or networks
- **Dynamic Nature:** Input values may change frequently, requiring adaptable solutions
- **Constraint-Driven:** Data is heavily influenced by rules and limitations

3. Output Data Description

The output data represents the optimized or near-optimal solution generated by the system.

- **Solution Set:** Selected configuration of tasks, routes, or allocations
- **Performance Metrics:** Includes cost, time, efficiency, or utilization values
- **Constraint Satisfaction Status:** Indicates whether constraints are fully or partially satisfied
- **Quality Score:** Measures how close the solution is to the optimal result

4. Data Format

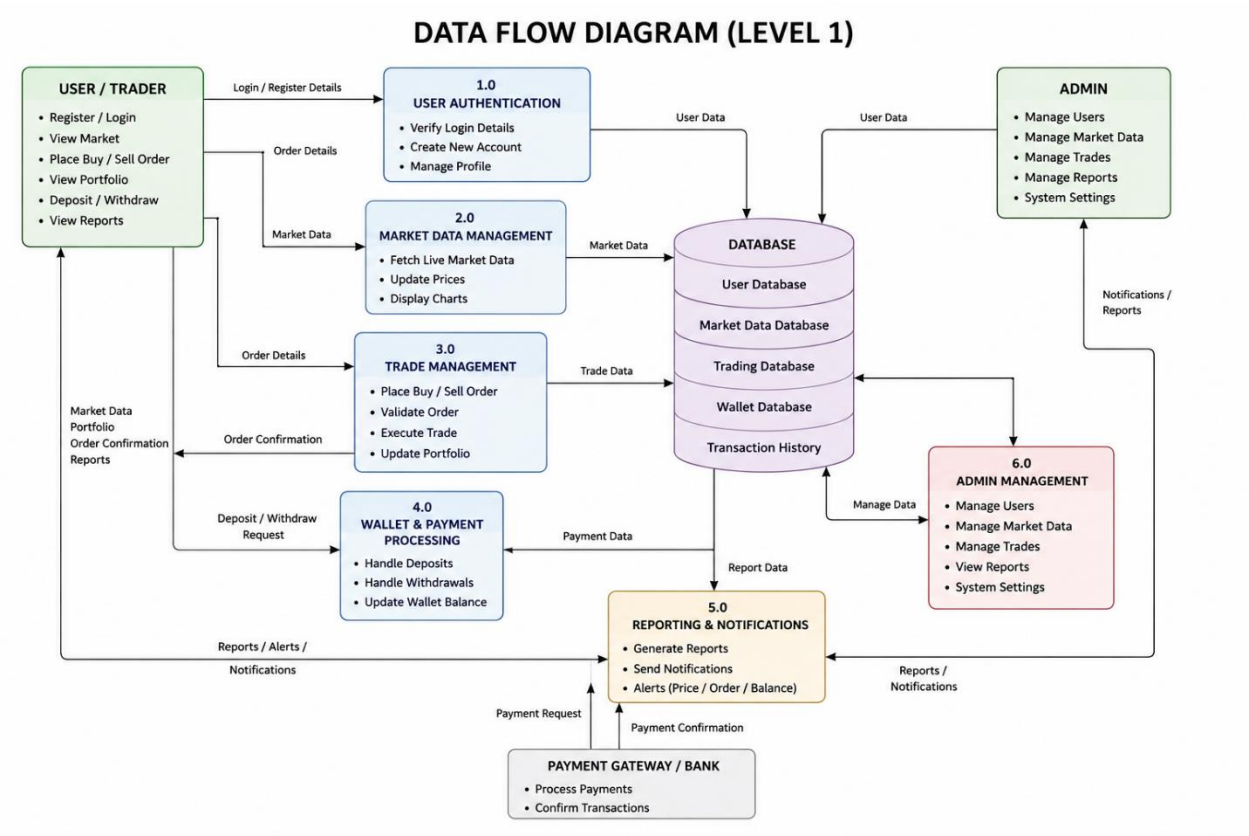
- Structured formats such as tables, matrices, or graphs are commonly used
- Input/output may also be represented in JSON, CSV, or database records depending on implementation

- Graph-based representations are frequently used for NP-hard problems like routing or scheduling

5. Data Constraints

- Data must be consistent and validated before processing
- Missing or incorrect values can significantly impact solution quality
- Input size may be limited based on system computational capacity
- Constraints must be strictly enforced during processing

6.3.2 Data flow Diagram



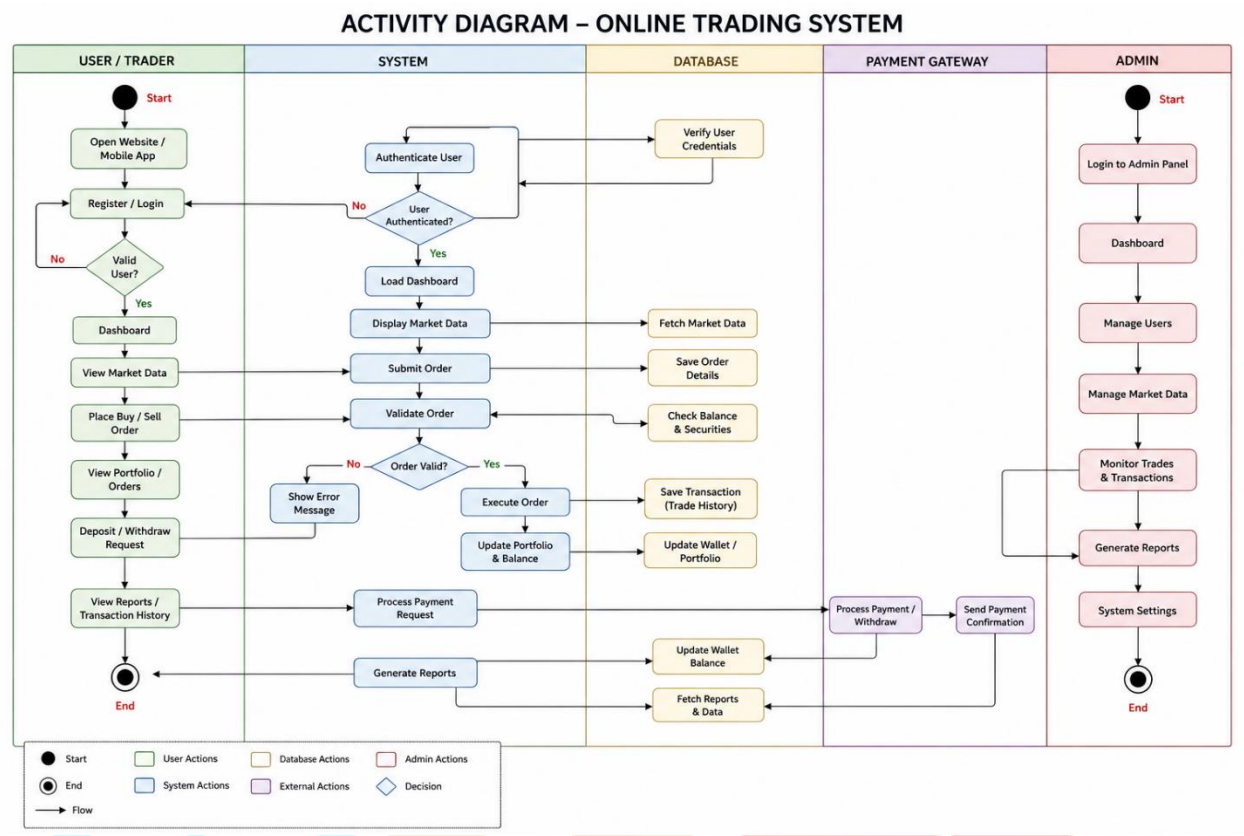
Data flow Diagram

6.3.3 Description of Function

The system performs a set of core functions to handle and solve NP-hard problems efficiently by converting input data into optimized solutions.

- Input Processing: Collects and validates raw data such as tasks, constraints, and parameters.
- Problem Formulation: Converts real-world problems into mathematical/NP-hard models with defined objectives and constraints.
- Optimization: Applies heuristic or approximation algorithms to generate near-optimal solutions.
- Solution Evaluation: Measures solution quality and checks constraint satisfaction.
- Risk Handling: Detects issues like high computation time or resource overload and applies fallback strategies.
- Output Generation: Presents final optimized results in user-friendly format.
- Monitoring: Tracks performance and logs execution details for analysis.

6.3.4 Activity Diagram :



6.3.5 Sequence Diagram:

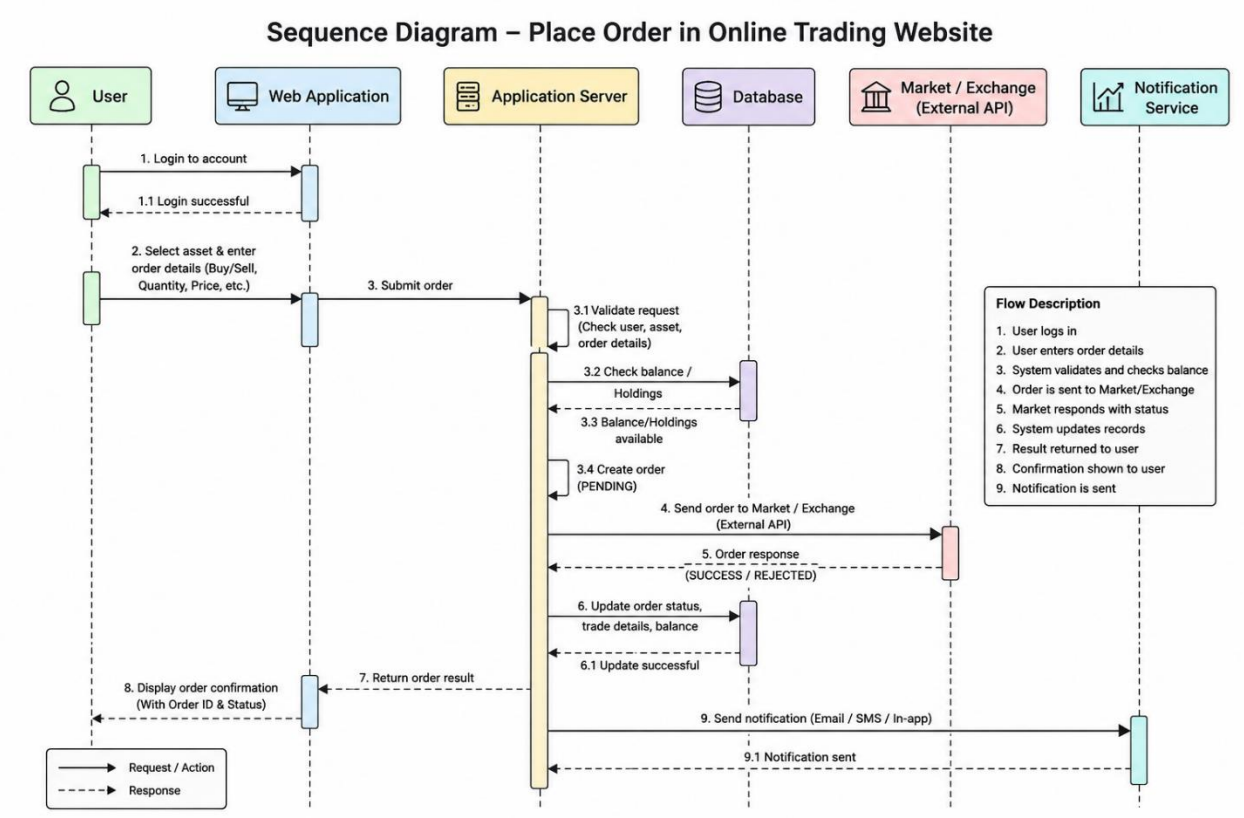


Fig. Sequence Diagram

6.3.6 Non Functional Requirements

Non-functional requirements define the quality attributes and system constraints of the NP hard problem-solving system. These requirements ensure that the system performs efficiently, reliably, and securely under various operating conditions.

1.Performance Requirements

- The system should generate near-optimal solutions within acceptable time limits.
- It must handle large datasets efficiently without significant performance degradation.
- Response time should be minimized even for complex NP-hard computations.

2. Scalability Requirements

- The system should support increasing input size without major redesign.
- It must handle growth in number of tasks, nodes, or constraints effectively.
- Performance should remain stable as problem complexity increases.

3. Reliability Requirements

- The system should consistently produce correct and stable outputs.
- It must handle unexpected inputs without crashing.
- Recovery mechanisms should be available in case of failure.

4. Efficiency Requirements

- Optimal use of CPU, memory, and storage resources is required.
- Algorithms should be optimized for time and space complexity.
- Unnecessary computations should be minimized.

5. Usability Requirements

- The system should have a simple and user-friendly interface.
- Outputs should be clearly presented and easy to interpret.
- Minimal technical knowledge should be required to operate the system.

6. Maintainability Requirements

- The system should be modular and easy to update or modify.
- Code should be well-documented for future enhancements.
- Bug fixing and improvements should be easy to implement.

7. Portability Requirements

- The system should be deployable across different environments with minimal changes.
- It should support standard operating systems and platforms.

8. Security Requirements

- Input data should be validated to prevent errors or misuse.
- Unauthorized access to system components should be restricted.
- Data integrity must be maintained during processing.

6.3.7 Hardware Requirements

Hardware requirements define the minimum and recommended physical system specifications needed to efficiently run the NP-hard problem-solving system. Since NP-hard computations can be resource-intensive, adequate hardware is necessary to ensure acceptable performance and stability.

1. Processor (CPU)

- **Minimum:** Intel Core i3 / equivalent
- **Recommended:** Intel Core i5/i7 or AMD Ryzen 5/7 and above
- Multi-core processor preferred for parallel execution of algorithms

2. Memory (RAM)

- **Minimum:** 4 GB RAM
- **Recommended:** 8 GB RAM or higher
- Higher memory is required for large datasets and complex computations

3. Storage

- **Minimum:** 10 GB free disk space
- **Recommended:** 50 GB SSD storage
- SSD is preferred for faster data access and processing speed

4. Input/Output Devices

- Standard keyboard and mouse for user interaction
- Monitor with minimum resolution of 1366 × 768 pixels
- Optional external storage devices for data backup

5. Network Requirements (if applicable)

- Stable internet connection for data transfer or cloud-based processing
- Minimum speed: 5 Mbps recommended

6. Optional High-Performance Setup

- Multi-core CPU (8+ cores) for parallel processing
- 16 GB or more RAM for large-scale NP-hard datasets
- GPU support for parallel optimization algorithms (if used)

6.3.8 Software Requirement

Software requirements define the programs, tools, and platforms needed to develop, run, and maintain the NP-hard problem-solving system. These ensure proper execution of algorithms, efficient computation, and smooth system operation.

1. Operating System

- Windows 10/11 or Linux (Ubuntu preferred for development)
- macOS (optional support depending on implementation)

2. Programming Language

- Python / Java / C++ (based on system design choice)
- Python is preferred for heuristic and optimization algorithms due to rich libraries

3. Development Tools / IDE

- Visual Studio Code / PyCharm / Eclipse
- GCC/G++ compiler for C++ implementations
- JDK for Java-based development

4. Libraries and Frameworks

- NumPy, Pandas (data handling and computation)
- SciPy (scientific and optimization functions)
- NetworkX (graph-based NP-hard problems)
- Matplotlib / Seaborn (data visualization)

5. Database Management System (if required)
- MySQL / PostgreSQL for structured data storage
- SQLite for lightweight applications

6. Version Control System
- Git for source code management
- GitHub / GitLab for collaboration and repository hosting

7. Testing Tools
- Unit testing frameworks (PyTest / JUnit)
- Debugging tools integrated within IDEs

8. Optional Tools
- Jupyter Notebook for experimentation and analysis
- Docker for deployment and environment consistency
- Cloud platforms (AWS / Azure / GCP) for large-scale computation

6.3.9 Communication Interface

1. User Interface Communication

User Interface Communication

The User Interface (UI) communication defines how users interact with the NP-hard problem-solving system and how information is exchanged between the user and the system.

1. Input Communication
- The system accepts input through a graphical or command-based interface.
- Users provide problem data such as tasks, constraints, and parameters via forms, text input, or file uploads (e.g., CSV/JSON).
- Input validation is performed immediately to ensure correctness and completeness.

2. Output Communication
- The system displays results such as optimized solutions, schedules, routes, or allocations on the screen.
- Outputs are presented in structured formats like tables, graphs, or dashboards for easy interpretation.
- Key performance metrics (time, cost, efficiency) are also shown to the user.

3. Interaction Mechanism
- Communication between user and system is interactive and event-driven (button clicks, submissions, or commands).
- The system provides feedback messages for successful operations, errors, or invalid inputs.

4. Visualization Support

- Graphical representation of solutions may be provided using charts or network graphs.
- Helps users understand complex NP-hard problem outputs more easily.

5. Error and Alert Communication

- The interface displays error messages for invalid inputs or system limitations.
- Warnings are shown when computational limits (time/memory) are reached.

- **Method:**

Weekly sync-up calls for progress review

Task distribution using shared Google Sheets

GitHub (if used): for version control and code collaboration

6.4 Performance Requirement

Performance requirements define how efficiently the system should execute NP-hard problem-solving tasks while maintaining acceptable response time, accuracy, and resource utilization. Since NP-hard problems are computationally intensive, performance optimization is a critical aspect of the system.

1. Response Time

- The system should generate results within a reasonable time limit depending on input size.
- Small datasets should be processed in seconds, while larger datasets should still remain within practical time bounds.

2. Throughput

- The system should be able to process multiple problem instances efficiently.
- Batch processing of inputs should be supported without major performance degradation.

3. Computational Efficiency

- Algorithms should be optimized to reduce time complexity wherever possible.
- Heuristic or approximation methods should be used to ensure faster execution.

4. Resource Utilization

- CPU and memory usage should be kept within optimal limits.
- The system should avoid unnecessary memory consumption and redundant computations.

5. Scalability

- The system should maintain acceptable performance even when input size increases.
- It should support larger datasets without system failure or major slowdown.

6. Accuracy vs Performance Trade-off

- A balance should be maintained between solution accuracy and computation time.
- Near-optimal solutions should be acceptable when exact solutions are computationally infeasible.

7. Stability Under Load

- The system should remain stable under high computational load.
- It should not crash or freeze during intensive processing tasks.

6.5 Software Interface Description

The Software Interface Description defines how the NP-hard problem-solving system interacts with other software components, tools, libraries, and external systems. These interfaces ensure smooth communication, data exchange, and integration between different software modules.

1. Operating System Interface

- The system runs on standard operating systems such as Windows and Linux.
- It uses OS-level services for memory management, file handling, and process execution.
- System calls are used indirectly through the programming language runtime.

2. Programming Language Interface

- The system is implemented using languages such as Python, Java, or C++.
- It interacts with language-specific runtime environments for execution of algorithms.
- External libraries are integrated through language package managers (e.g., pip, Maven).

3. Database Interface (if applicable)

- The system communicates with databases like MySQL, PostgreSQL, or SQLite.
- SQL queries are used for storing and retrieving problem data and results.
- ORM tools (e.g., SQLAlchemy, Hibernate) may be used for easier integration.

4. Library and Framework Interface

- Scientific and optimization libraries such as NumPy, SciPy, and NetworkX are used.
- These libraries provide pre-built functions for mathematical computation and graph processing.
- Visualization libraries like Matplotlib may be used for result representation.

5. File System Interface

- The system reads input data from files such as CSV, JSON, or text formats.
- Output results may also be stored in external files for reporting and analysis.
- File handling operations ensure secure and structured data access.

6. External Tool Interface (Optional)

- Integration with tools like Git for version control.
- Docker may be used for containerized deployment.
- Cloud platforms (AWS, Azure, GCP) may be used for large-scale processing.

CHAPTER 7

SYSTEM DESIGN

7.1 Introduction

The System Design phase defines the overall architecture and structure of the NP-hard problem-solving system. It provides a blueprint for how different components of the system interact to process complex inputs, apply optimization techniques, and generate efficient solutions within practical time and resource constraints.

Since NP-hard problems are computationally intensive, the system design emphasizes the use of heuristic, approximation, or metaheuristic approaches rather than exact algorithms. This helps in achieving near-optimal solutions while maintaining performance and scalability.

The main objective of the system design is to:

- Transform the requirements defined in the SRS into a structured architectural model
- Define system components and their interactions clearly
- Ensure scalability, efficiency, and modularity of the system
- Support effective risk handling and performance optimization

The system is typically divided into modules such as input processing, problem formulation, optimization engine, evaluation, and output generation. Each module works collaboratively to ensure smooth execution of NP-hard computations.

7.2 Architecture

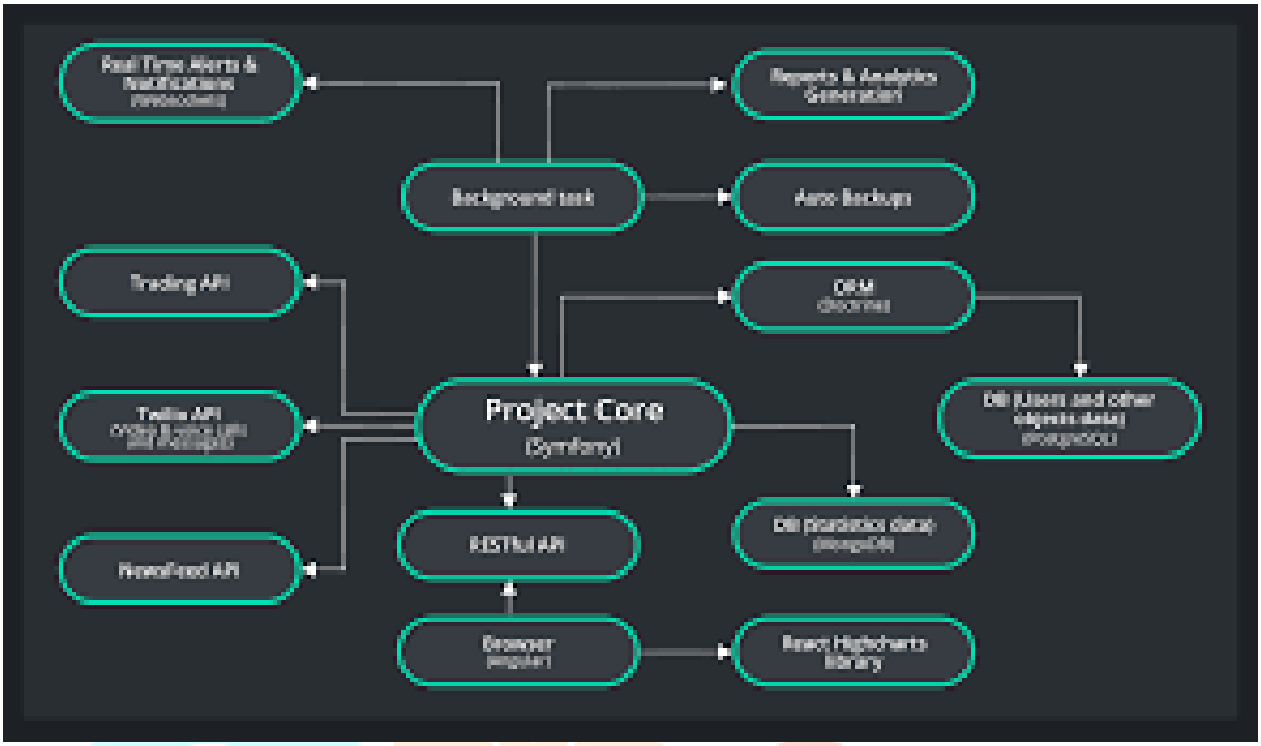


Figure 7.2: Architectural Design

CHAPTER 8

8.1 Source Code & Output

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />

<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

<title>React App</title>
</head>
<body>
<noscript>You need to enable JavaScript to run this app.</noscript>
<div id="root"></div>

</body>
</html>
```

CSS

```
body {
```

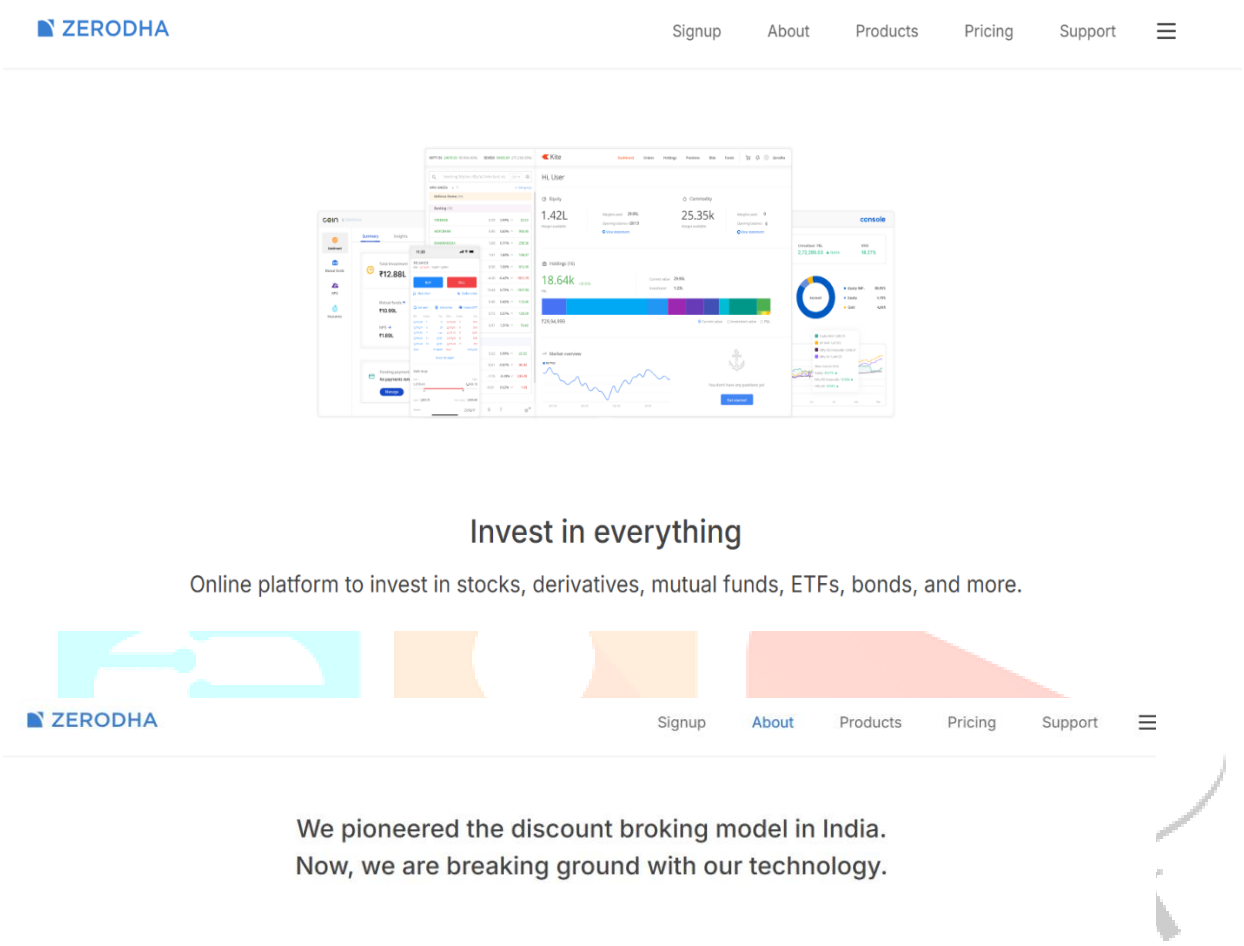
```
margin: 0;
font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

code {
font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
monospace;
}

.nav-link{
margin-left: 40px;
}

#supportHero{
background-color: rgb(56, 126, 209);
color: white;
}

#supportHero a{
color: white;
}

#supportWrapper{
display: flex;
justify-content: space-between;
margin: 0 150px;
}

#supportHero input{
padding: 20px 50px;
width: 100%;
font-size: 20px;
border-radius: 10px;
border: none;
```

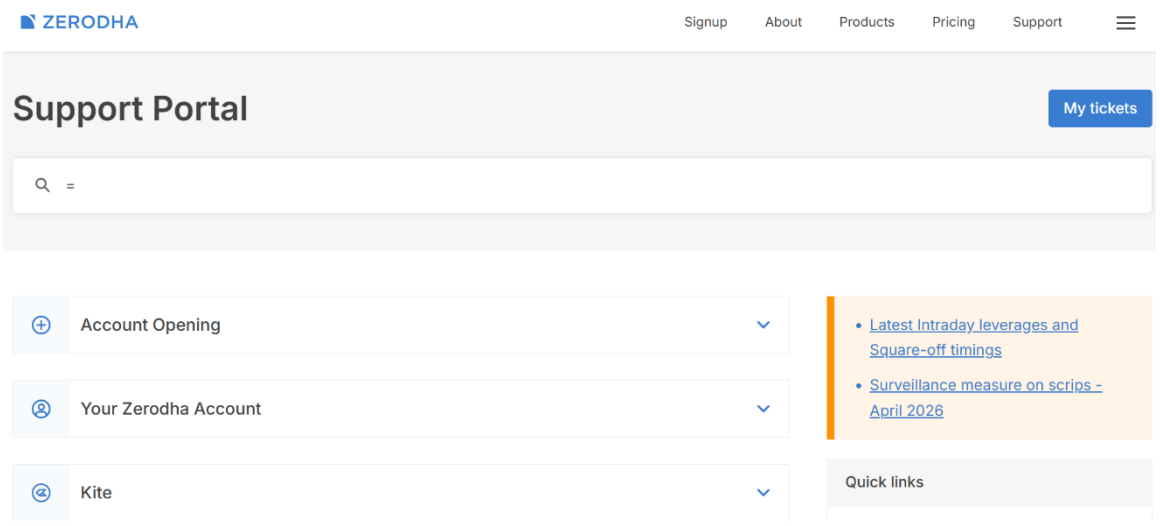
```
}
```

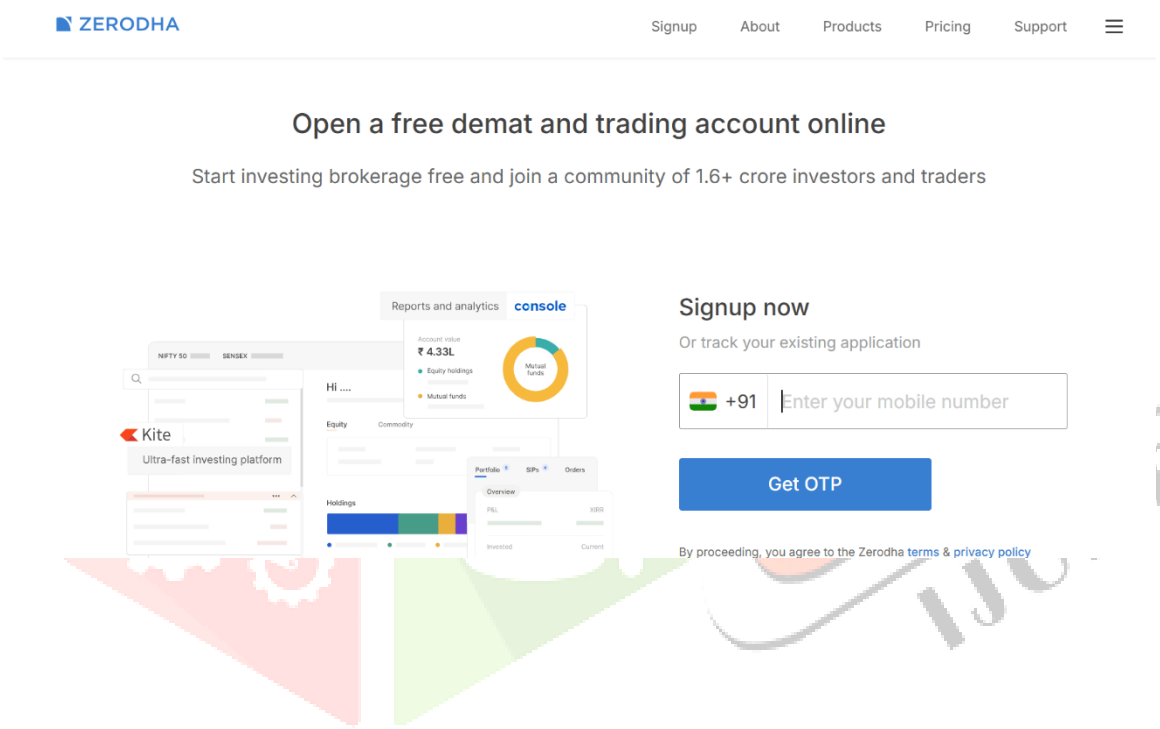
JS

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import "./index.css";

import HomePage from "./landing_page/home/HomePage";
import Signup from "./landing_page/signup/Signup";
import AboutPage from "./landing_page/about/AboutPage";
import ProductPage from "./landing_page/products/ProductsPage";
import PricingPage from "./landing_page/pricing/PricingPage";
import SupportPage from "./landing_page/support/SupportPage";

import NotFound from "./landing_page/NotFound";
import Navbar from "./landing_page/Navbar";
import Footer from "./landing_page/Footer";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
<BrowserRouter>
<Navbar />
<Routes>
<Route path="/" element={<HomePage />} />
<Route path="/signup" element={<Signup />} />
<Route path="/about" element={<AboutPage />} />
<Route path="/product" element={<ProductPage />} />
<Route path="/pricing" element={<PricingPage />} />
<Route path="/support" element={<SupportPage />} />
<Route path="*" element={<NotFound />} />
</Routes>
<Footer />
</BrowserRouter>
);
```

OUTPUT :


ZERODHA
Signup
About
Products
Pricing
Support
Invest in everything
Online platform to invest in stocks, derivatives, mutual funds, ETFs, bonds, and more.
ZERODHA
Signup
About
Products
Pricing
Support
We pioneered the discount broking model in India.
Now, we are breaking ground with our technology.
We kick-started operations on the 15th of August, 2010 with the goal of breaking all barriers that traders and investors face in India in terms of cost, support, and technology. We named the company Zerodha, a combination of Zero and "Rodha", the Sanskrit word for barrier.
In addition, we run a number of popular open online educational and community initiatives to empower retail traders and investors.
Rainmatter, our fintech fund and incubator, has invested in several fintech startups with the goal of growing the Indian capital markets.





CHAPTER 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

The Online Trading System is designed to provide a modern, secure, reliable, and efficient platform for users to perform trading activities through the internet. The system enables users to buy and sell financial assets such as stocks, cryptocurrencies, forex, or commodities without visiting a physical trading office. By using digital technologies and automated processes, the platform simplifies trading operations and improves overall efficiency.

The project successfully integrates various important modules including user registration and authentication, market data management, trade execution, portfolio management, wallet and payment processing, reporting, and administration. Each module works together to ensure smooth operation of the trading platform and provide a better experience for both users and administrators.

One of the major advantages of the system is its ability to provide real-time market information and instant

transaction processing. Users can monitor market trends, place orders, manage investments, and track transaction history from anywhere at any time. This improves accessibility, saves time, and increases user convenience.

The system also focuses heavily on security and data protection. Features such as encrypted login systems, secure payment processing, user verification, and transaction monitoring help protect user information and financial data from unauthorized access and cyber threats. The admin panel allows administrators to manage users, monitor trading activities, generate reports, and maintain system performance effectively.

The Data Flow Diagram (DFD) and Activity Diagram developed for the project clearly explain the movement of data and workflow of the system. These diagrams help developers, designers, and stakeholders understand the system structure and functionality more easily. Proper system analysis and design improve project planning, reduce development errors, and support future system expansion.

Furthermore, the proposed trading platform reduces manual work and paperwork by automating major trading operations. Automated order processing, balance updates, report generation, and notification systems increase operational accuracy and minimize human errors. The platform also supports scalability, meaning additional features such as AI-based trading, advanced analytics, mobile applications, and multi-currency support can be added in future versions.

9.2 Future Scope

The Online Trading System has significant future growth potential due to the rapid advancement of digital technologies and increasing demand for online financial services. Although the current system provides essential trading functionalities, several advanced features and improvements can be implemented in future versions to enhance performance, security, and user experience.

1. Mobile Application Development

In the future, dedicated mobile applications for Android and iOS platforms can be developed. Mobile apps will allow users to trade anytime and anywhere with better accessibility, faster notifications, and improved user convenience.

2. Artificial Intelligence and Machine Learning

AI and Machine Learning technologies can be integrated into the system for:

- Predicting market trends
- Automated trading suggestions
- Risk analysis
- Fraud detection
- Personalized investment recommendations

These technologies can help users make smarter trading decisions.

3. Real-Time Analytics and Advanced Charts

Future versions can include advanced trading charts, technical indicators, and analytical tools for better market analysis. Interactive dashboards and real-time reporting can improve decision-making for traders and investors.

4. Multi-Currency and Global Trading Support

The system can be expanded to support:

- International currencies

- Global stock markets
- Cryptocurrency exchanges
- Forex trading

This will allow users to perform worldwide trading from a single platform.

CHAPTER 10

REFERENCES:

- TradingView – Market charts and trading analysis tools.
- Binance API Documentation – Cryptocurrency trading API reference.
- Alpaca Markets API – Stock trading and brokerage API services.
- React Documentation – Frontend web application development.
- Node.js Documentation – Backend server-side development.
- PostgreSQL Documentation – Database management system reference.
- Sommerville, Ian. *Software Engineering*. Pearson Education.
- Pressman, Roger S. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
- Elmasri, Ramez & Navathe, Shamkant B. *Fundamentals of Database Systems*. Pearson.
- Oracle Documentation – Database and enterprise software documentation.
- Mozilla Developer Network (MDN) – Web development technologies and programming references.
- AWS Documentation – Cloud deployment and infrastructure services.
- Online research articles, tutorials, and technical blogs related to online trading systems, cybersecurity, and financial technology.