A Report on project idea “Disaster Resource Management System”

Submitted by:

Name	Roll Number
Rishi Kumar	2447031
Prashant Kumar Mishra	2447021
Satyam Bhardwaj	2447052
Mayank Shaw	2447050
Monil Mishra	2447017
Biswajit Ghadei	2447036

Submitted to :- Dr. Anand Shanker Tewari

Course Code :- CS032003

Course Title :- Object Oriented Programming using JAVA

Batch :- MCA (AI & IoT)

Semester :- 3rd

Department of computer science & engineering

National Institute of Technology Patna

Ashok Rajpath, Mahendru, Patna, Bihar 800005

Core Project Idea & Objectives:

Our project aims to design and implement a robust, text-based Disaster Resource Management System (DRMS) using a client-server architecture. This system streamlines disaster-relief operations by providing a centralized platform for real-time coordination between Field Offices (Relief Camps) and Headquarters.

- **Primary Objectives:**
  - To enable Field Offices to manage local inventory and automatically generate supply requests when stock levels fall below a defined threshold.
  - To provide Headquarters with a consolidated view of all incoming requests and live inventory across all camps for informed decision-making.
  - To establish a secure, thread-safe server that handles concurrent client connections, manages authentication, and ensures data consistency.
  - To demonstrate a practical application of core Object-Oriented Programming principles in Java.

Methodology & Technical Approach

We will adhere to a structured, object-oriented methodology to ensure code modularity, reusability, and scalability.

1. **Paradigm: Strict Object-Oriented Programming (OOP).**
2. **Core OOP Concepts Utilized:**
   - Encapsulation: All data models (e.g., Request, Inventory) will be bound with their corresponding operations.
   - Inheritance: A base Client class will be extended to create specialized FieldOfficeClient and HeadquartersClient classes.
   - Polymorphism: Will be used for handling different types of client requests and messages through a common interface.
   - Abstraction: Complex subsystems like database interaction and network communication will be hidden behind simplified interfaces.
3. **Architecture: Client-Server Model with a central server managing multiple concurrent clients using Java Sockets and multi-threading.**

Implementation Workload Distribution

the workload is distributed as follows:

- **Core System Architecture**
  - **Members:** Rishi Kumar, Prashant Kumar Mishra
  - **Responsibilities:**
    1. Design and implementation of the central Server class.
    2. Managing multi-threading to handle concurrent clients.
    3. Designing the application-level communication protocol.
    4. Core socket programming and data stream management.

- **Client-Side Logic & Data Modelling**
    - **Members:** Satyam Bhardwaj, Mayank Shaw
    - **Responsibilities:**
        1. Designing and implementing the FieldOfficeClient and HeadquartersClient classes.
        2. Creating the data model classes (Request, Inventory, User).
        3. Implementing the text-based menu-driven interface for both client types.
        4. Developing the logic for automatic request generation based on inventory thresholds.
- **Data Persistence & System Integration**
    - **Members:** Biswajit Ghadei, Monil Mishra
    - **Responsibilities:**
        1. Implementing the DBHandler class for database operations (CRUD).
        2. Designing and creating the necessary database schema (e.g., using SQLite or a file-based system).
        3. Writing utility classes for logging and input validation.
        4. Assisting with the integration of different modules and comprehensive testing.

System Requirements & Technology Stack

- **Programming Language:** Java (JDK 8 or above)
- **Core Technologies:**
    - Java Sockets & ServerSocket for network communication.
    - Java Multi-threading (Concurrency utilities).
    - Java I/O and Serialization for object transmission.
- **Data Persistence:** To be determined between:
    - **SQLite** (for a lightweight, file-based RDBMS).
    - **File-based Storage** using Java Serialization or JSON.
- **Development Tools:**
    - **IDE:** IntelliJ IDEA / Eclipse / VS Code.
    - **Build Tool:** Apache Maven (optional).
    - **Version Control:** Git & GitHub.
- **Key Java Packages:** java.net, java.io, java.util.concurrent, java.sql (if DB used).