RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM _ 602 105

CS23333 Object Oriented Programming Using Java

Mini Project Report

TENDER MANAGEMENT SYSTEM

A MINI-PROJECT BY:

Dayanithi V (230701064) & Darshan S (230701063)

in partial fulfillment of the award of the degree OF BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING.

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Academic Year : 2024-2025. **Semester:** THIRD sem **Branch :-** B.E CSE-A

Register No.	230701064 & 230701063

Certified that this is the bonafide record of work done by DAYANITHI V(230701064) and DARSHAN S(230701063) the above student in the CS23333 – Object Oriented Programming Using Java during the year 2024-2025.

Signature of Faculty in-charge

Abstract

The **Tender Management System** is a robust and user-friendly software application designed to digitize and automate the tender management processes within organizations. Traditional manual systems often lead to inefficiencies, delays, and errors due to their reliance on paperwork and outdated practices. This system addresses these challenges by providing a centralized platform that organizes and automates tasks such as tender creation, bid tracking, vendor management, and reporting.

Developed using **Java Swing** for the frontend and **MySQL** for the backend, the system utilizes **JDBC (Java Database Connectivity)** to enable real-time interactions between the application and the database. This ensures data consistency, accuracy, and security while offering the scalability needed to handle growing tender data.

Key Features

1. **Tender Creation and Management:** Users can easily add, update, search, and delete tenders.
2. **Vendor Database Management:** Maintains vendor details such as name, contact information, and bidding history.
3. **Bid Evaluation and Tracking:** Allows users to track bids submitted for tenders and evaluate their feasibility.
4. **Report Generation:** Provides detailed reports on tenders, bids, and vendor performance to assist in decision-making.
5. **Scalability and Security:** Designed to accommodate an increasing volume of tenders while ensuring data integrity.

This system empowers organizations to improve their operational efficiency, focus on strategic decision-making, and enhance their overall procurement processes. Its user-centric interface and backend robustness make it an ideal choice for

modern businesses.

Table of Contents (Page 3)

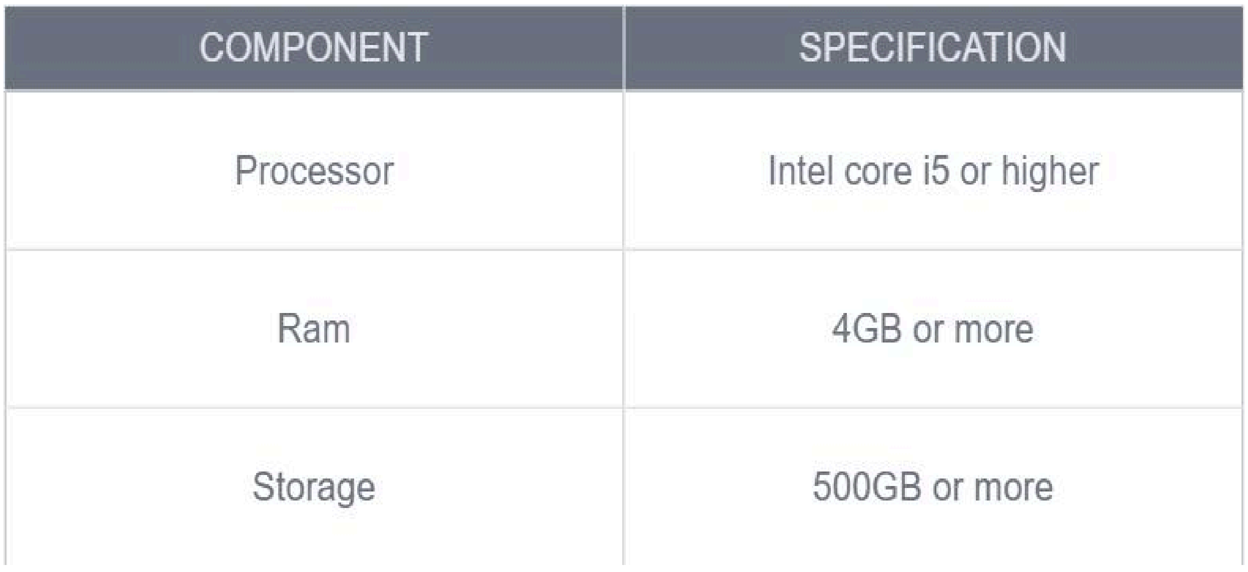
1. **Introduction**
    - 1.1 Problem Definition
    - 1.2 Need
2. **Requirements**
    - 2.1 Software Requirements
    - 2.2 Hardware Requirements
3. **Entity-Relationship Diagram**
4. **Schema Diagram**
5. **Implementation**
    - 5.1 Frontend Implementation
    - 5.2 Backend Implementation
    - 5.3 JDBC Integration and Frontend-Backend Interactions
6. **Snapshots**
7. **Conclusion**
8. **References**
9. **Appendix**

Chapter 2: Requirements

2.1 Software Requirements

- **Operating System:** Windows 10 or higher
- **Frontend Software:** Java (JDK 8, IntelliJ IDEA)
- **Backend Software:** MySQL
- **Connector:** MySQL JDBC Driver

2.2 Hardware Requirements



COMPONENT	SPECIFICATION
Processor	Intel core i5 or higher
Ram	4GB or more
Storage	500GB or more

Chapter 3: Entity-Relationship Diagram

3.1 Entity-Relationship Diagram (ERD)

The **Entity-Relationship Diagram (ERD)** visually represents the data model of the **Tender Management System**. The ERD shows how the system's entities interact with each other, providing a detailed overview of how the data is structured and related. This diagram is essential for designing and understanding the relationships within the database.

Entities and Attributes

1. **Tenders**
   - **Tender_ID**: The unique identifier for each tender. This is the **primary key**.
   - **Title**: The name or title of the tender.
   - **Description**: A detailed description of the tender, specifying the scope of work.
   - **Deadline**: The submission deadline for bids

associated with the tender.

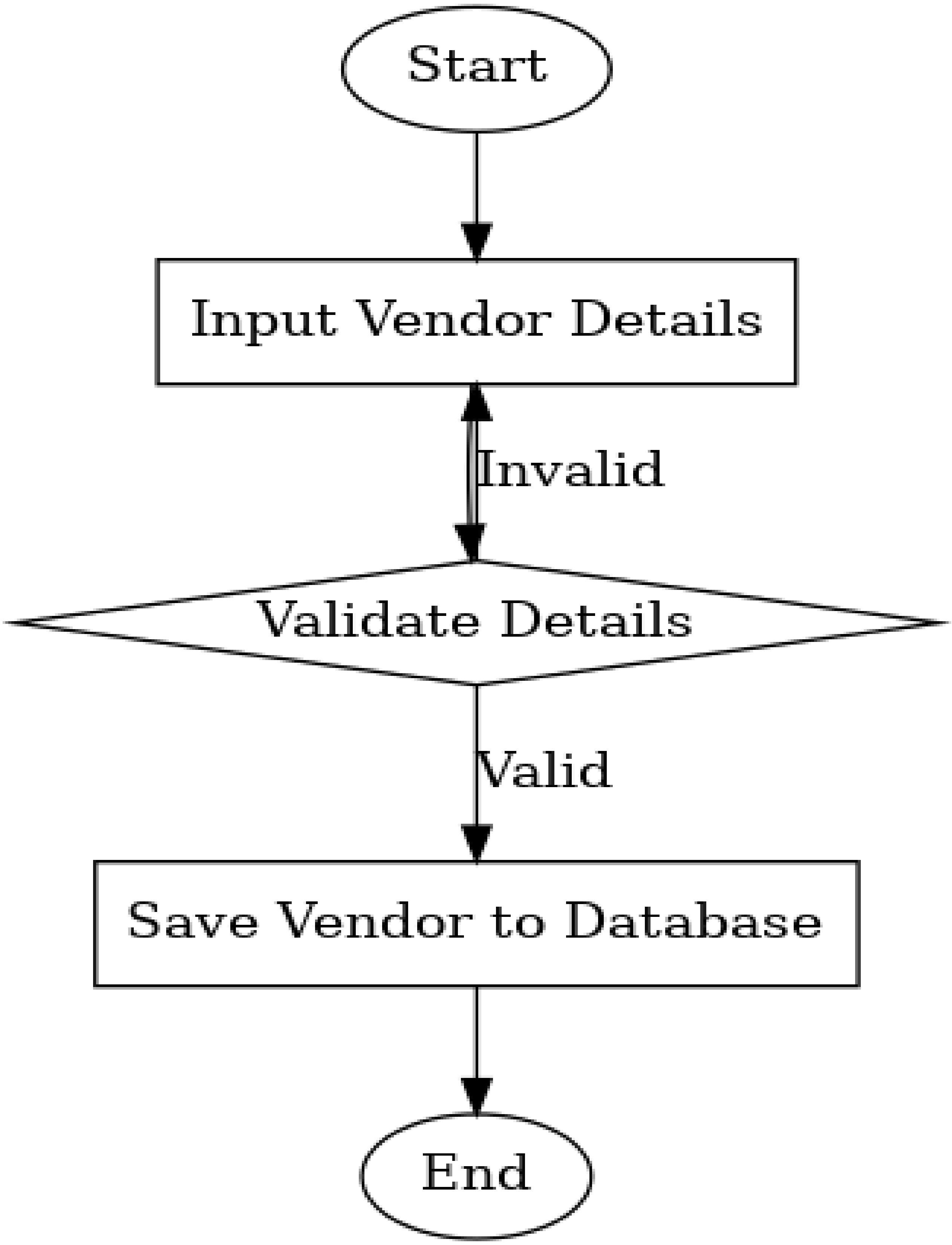
2. **Vendors**
    - **Vendor_ID**: The unique identifier for each vendor. This is the **primary key**.
    - **Name**: The name of the vendor company or individual.
    - **Contact_Info**: The contact details for the vendor (email, phone, etc.).
3. **Bids**
    - **Bid_ID**: The unique identifier for each bid. This is the **primary key**.
    - **Tender_ID**: A foreign key that links the bid to a specific tender.
    - **Vendor_ID**: A foreign key that links the bid to the vendor submitting it.
    - **Bid_Amount**: The amount proposed by the vendor for the tender.

Relationships Between Entities

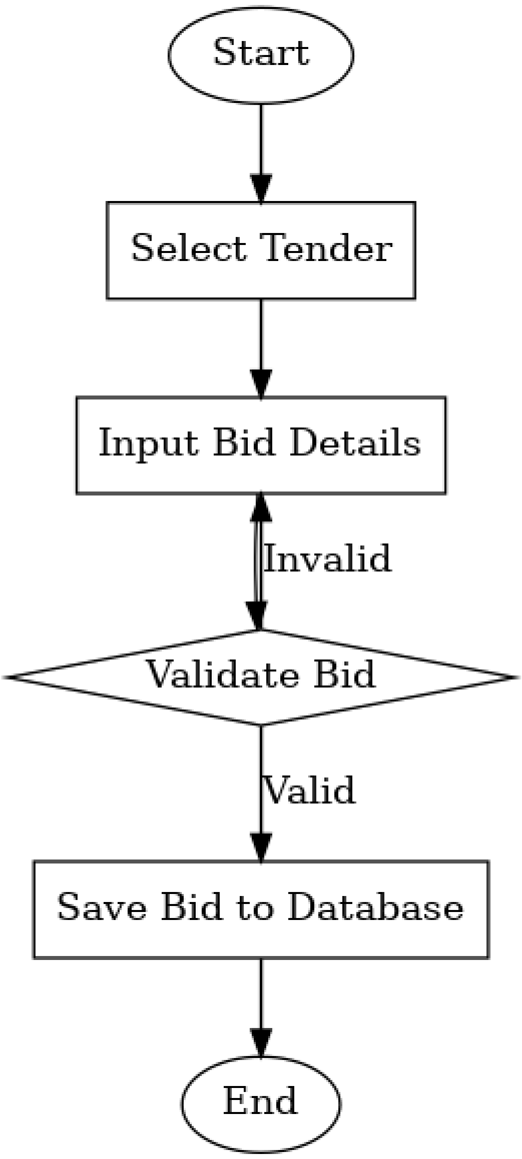
1. **Tenders to Bids**
    - **One-to-Many**: One tender can have multiple bids, but each bid is associated with only one tender. The **Tender_ID** in the **Bids** table references the **Tender_ID** in the **Tenders** table.
2. **Vendors to Bids**
    - **One-to-Many**: One vendor can submit multiple bids, but each bid is associated with only one vendor. The **Vendor_ID** in the **Bids** table references the **Vendor_ID** in the **Vendors** table.

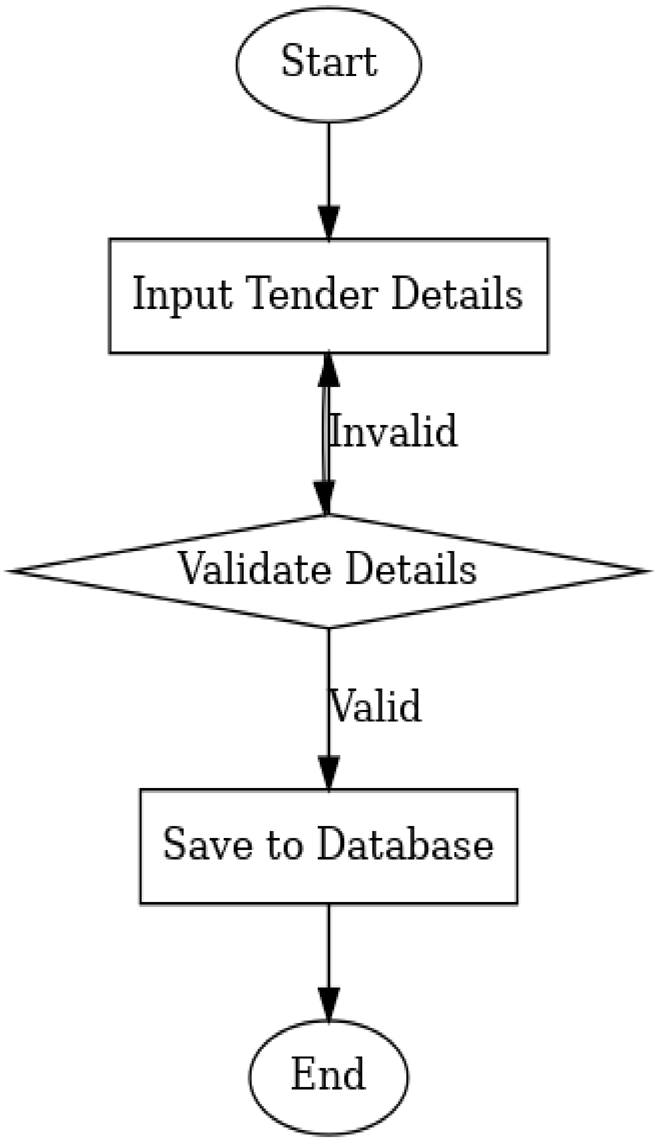
Diagram Overview

The ERD will visually depict the entities and relationships as boxes with labeled attributes and connecting lines showing the relationships between them. The relationships will be indicated as **One-to-Many**, where applicable.



TENDER CREATION FLOWCHART





BID SUBMISSION FLOWCHART

VENDOR REGISTRATION FLOWCHART

Chapter 4: Schema Diagram (Pages 10–11)

4.1 Schema Diagram

The **Schema Diagram** provides a more detailed and technical representation of the database structure. It outlines the tables in the database, their attributes, primary keys, and foreign keys. The schema diagram is essential for understanding how the data will be stored in the database and how the tables interact with one another.

Tables and Relationships:

1. **Tenders Table**
   - **Tender_ID (PK)**: Unique identifier for each tender.
   - **Title**: A short description or name for the tender.
   - **Description**: A more detailed description of the tender.
   - **Deadline**: The date and time when bids for the tender are due.
   - **Primary Key**: **Tender_ID**
2. **Vendors Table**
   - **Vendor_ID (PK)**: Unique identifier for each vendor.
   - **Name**: Name of the vendor or vendor company.
   - **Contact_Info**: Contact details like email, phone, etc.
   - **Primary Key**: **Vendor_ID**
3. **Bids Table**
   - **Bid_ID (PK)**: Unique identifier for each bid.
   - **Tender_ID (FK)**: Foreign key referencing **Tender_ID** in the **Tenders** table.
   - **Vendor_ID (FK)**: Foreign key referencing **Vendor_ID** in the **Vendors** table.
   - **Bid_Amount**: The bid price submitted by the vendor

    for the tender.
    - **Primary Key**: **Bid_ID**
    - **Foreign Keys**: **Tender_ID**, **Vendor_ID**

Relationships:

- **Tenders to Bids**: A **one-to-many** relationship where one tender can have multiple bids, and each bid is linked to a single tender.
- **Vendors to Bids**: A **one-to-many** relationship where one vendor can place multiple bids, but each bid is linked to a single vendor.

Schema Diagram Overview

The schema diagram visually represents the structure of the **Tenders**, **Vendors**, and **Bids** tables, their attributes, and their relationships through primary and foreign keys. It clearly shows the connections between these tables, supporting efficient data management and retrieval within the system.

Chapter 5: Implementation

5.1 Frontend Implementation

Code : GUI Setup

Explanation of Frontend Implementation

The frontend of the Tender Management System is responsible for providing a user-friendly interface that allows users to interact with the system. It is developed using Java Swing, a GUI toolkit for building graphical user interfaces in Java applications.

Key Components:

1. Main Frame Setup
   The main frame (`JFrame`) serves as the container for all other UI components. It is the central window that appears to the user. The main frame will contain input fields for data entry, buttons to trigger various actions, and a table to display existing tenders.
2. Input Fields
   The input fields allow users to enter tender information. These include:
   - Tender Title: A text field where the user enters the title of the tender.
   - Tender Description: A larger text area where the user provides a detailed description of the tender.
   - Deadline: A text field where the user specifies the submission deadline for the tender.
3. Buttons for Actions
   Buttons are used for performing various actions such as saving a new tender, updating existing tender details, searching for a tender, or deleting a tender. Each button

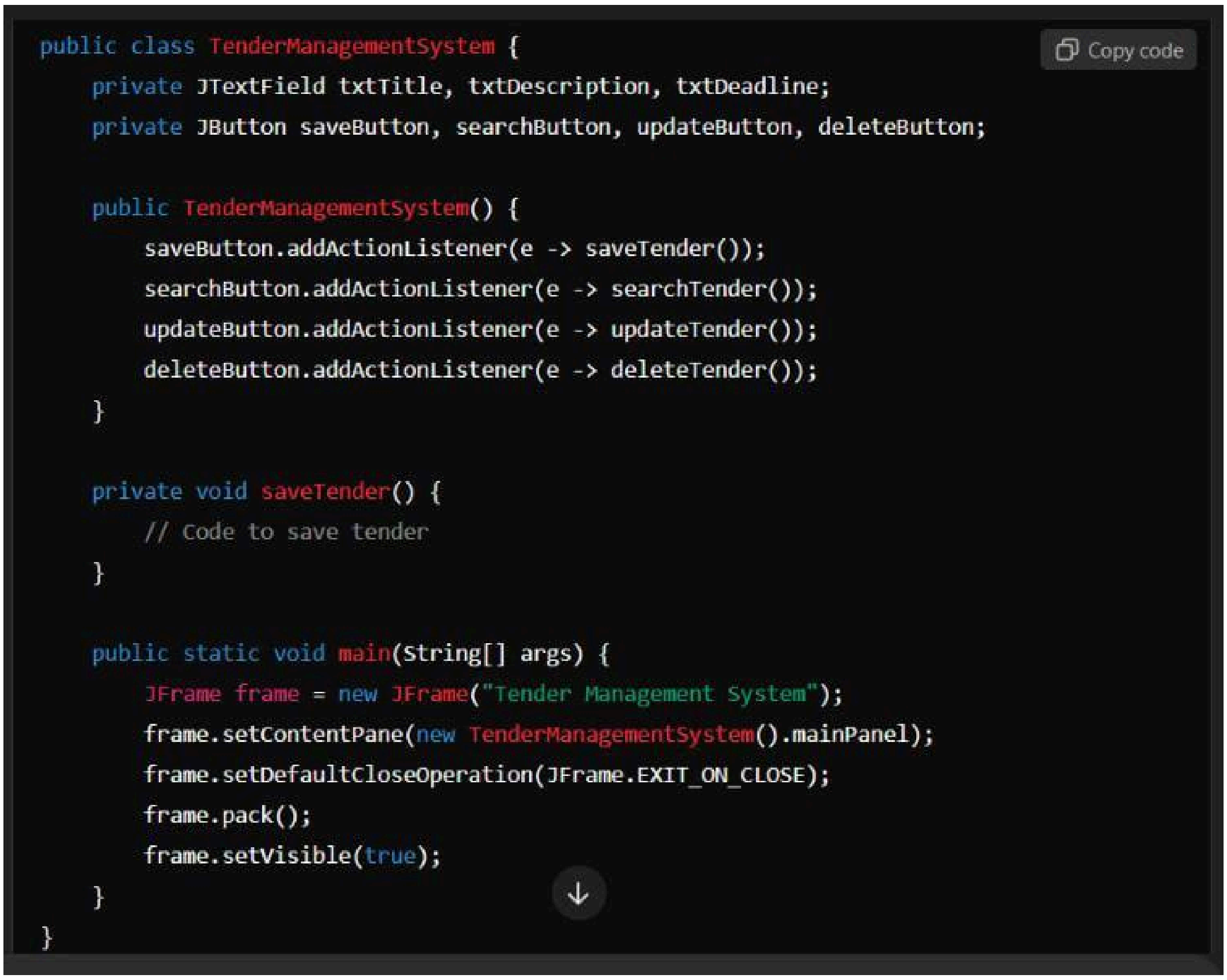
is associated with an event listener that triggers specific backend functionality:

- Save: Saves the entered tender details to the database.
- Update: Updates the details of an existing tender.
- Search: Searches for a tender based on a specific search criterion, such as Tender ID.
- Delete: Deletes a tender from the system.

4. Table for Tender Display
   A table component is used to display existing tenders. This provides users with an organized view of the tenders stored in the database, allowing them to perform actions like editing or deleting specific records.

User Interaction Flow:

When a user interacts with the frontend (e.g., entering data and clicking a button), the corresponding event handler is triggered, which calls specific backend methods to process the data and update the system.



```java
public class TenderManagementSystem {
    private JTextField txtTitle, txtDescription, txtDeadline;
    private JButton saveButton, searchButton, updateButton, deleteButton;

    public TenderManagementSystem() {
        saveButton.addActionListener(e -> saveTender());
        searchButton.addActionListener(e -> searchTender());
        updateButton.addActionListener(e -> updateTender());
        deleteButton.addActionListener(e -> deleteTender());
    }

    private void saveTender() {
        // Code to save tender
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Tender Management System");
        frame.setContentPane(new TenderManagementSystem().mainPanel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

5.2 Backend Implementation

Explanation of Backend Implementation

The **backend** of the **Tender Management System** is responsible for handling the data management logic, such as storing, retrieving, updating, and deleting tender records in the database. The backend interacts with the database using **MySQL** and **JDBC (Java Database Connectivity)**.

Key Components:

1. **Database Connection**

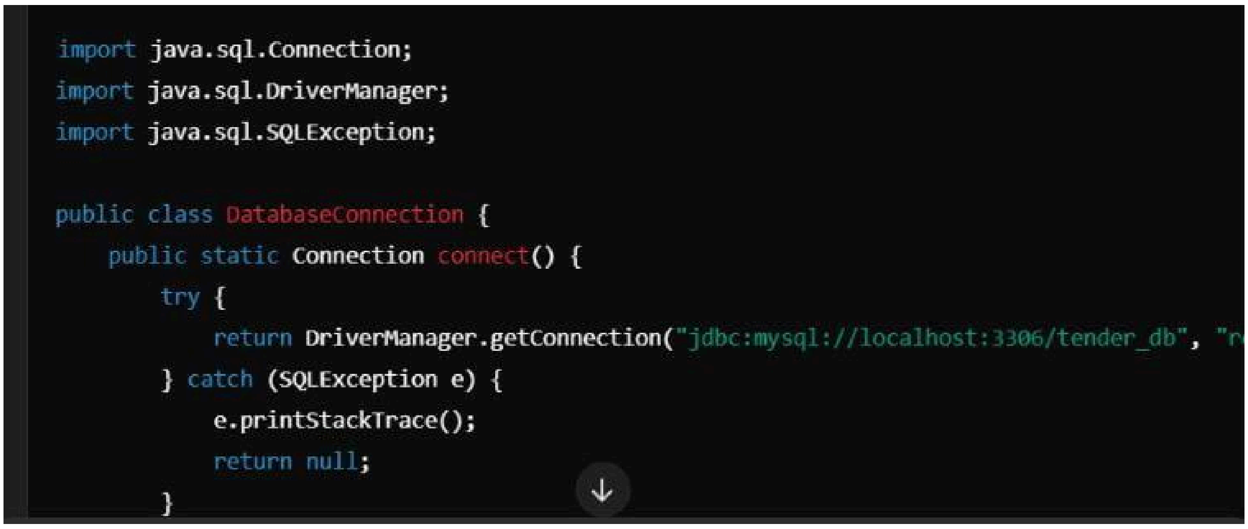
The backend must establish a connection to the **MySQL** database in order to execute SQL queries. **JDBC** is used to manage the communication between the application and the database. The database connection is created by specifying the database URL, username, and password.

2. **CRUD Operations**
   CRUD (Create, Read, Update, Delete) operations are the core functions that manage the tender data in the database. These operations are implemented using SQL queries:
   - **Create**: Adds a new tender to the database.
   - **Read**: Retrieves a specific tender or all tenders from the database.
   - **Update**: Modifies the details of an existing tender.
   - **Delete**: Removes a tender from the database.
3. **Error Handling**
   Proper error handling is critical for ensuring the system works smoothly. If an issue arises (e.g., the database connection fails or a query is invalid), the backend catches the exception and prints an appropriate error message.

Backend Logic Flow:

Whenever a user performs an action like saving or updating a tender, the frontend sends a request to the backend. The backend executes the necessary SQL query, interacts with the database, and returns the result (success or failure).

Database Connection Using JDBC



```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    public static Connection connect() {
        try {
            return DriverManager.getConnection("jdbc:mysql://localhost:3306/tender_db", "r
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
```

5.3 JDBC Integration

Explanation of JDBC Integration and Frontend-Backend Interactions

The **JDBC integration** is what allows communication between the **frontend** and **backend**. The frontend collects data from the user, and the backend uses **JDBC** to store, retrieve, and manipulate the data in the **MySQL** database.

Key Concepts:

1. **Event-Driven Architecture**
   The frontend is event-driven, meaning that user interactions (like clicking buttons) trigger specific functions in the backend. For example, when the user clicks the **Save** button, the frontend gathers the

tender data and passes it to the backend for saving in the database.

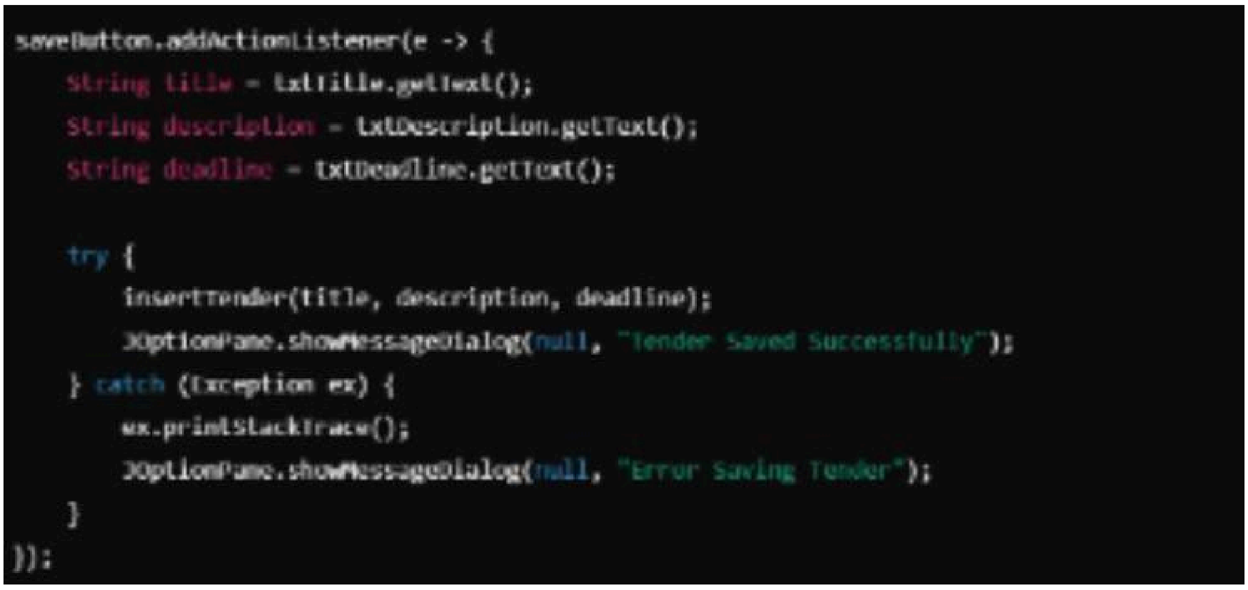
2. **Frontend to Backend Communication**
   When a user interacts with the frontend, such as entering new tender information, the system must pass that data to the backend for processing. The frontend gathers data from input fields (e.g., tender title, description, and deadline), and the backend performs operations like saving that data to the database.
   For example, when the user clicks **Save**, the following happens:
   - The frontend gathers the data from the input fields.
   - The frontend calls the backend method responsible for saving the tender to the database.
   - The backend executes an SQL `INSERT` query to store the tender information.
3. **Database Operations Using JDBC**
   The backend communicates with the database via **JDBC**. JDBC allows the backend to execute SQL queries to insert, update, or delete records. For example, when saving a tender, the backend prepares an SQL `INSERT` query and executes it to store the data in the database.
4. **Data Flow**
   The system follows a simple data flow:
   - The user interacts with the frontend (e.g., filling in a tender form).

- The frontend sends this data to the backend.
- The backend processes the data by executing the corresponding SQL query.
- The database stores or retrieves the data, and the backend sends the result back to the frontend.
- The frontend displays the result to the user (e.g., confirmation of successful data entry).

Backend to Frontend Communication

When a user performs an action like searching for a tender, the backend queries the database to retrieve the tender data. Once the data is retrieved, it is sent back to the frontend, which then updates the display with the tender's information.

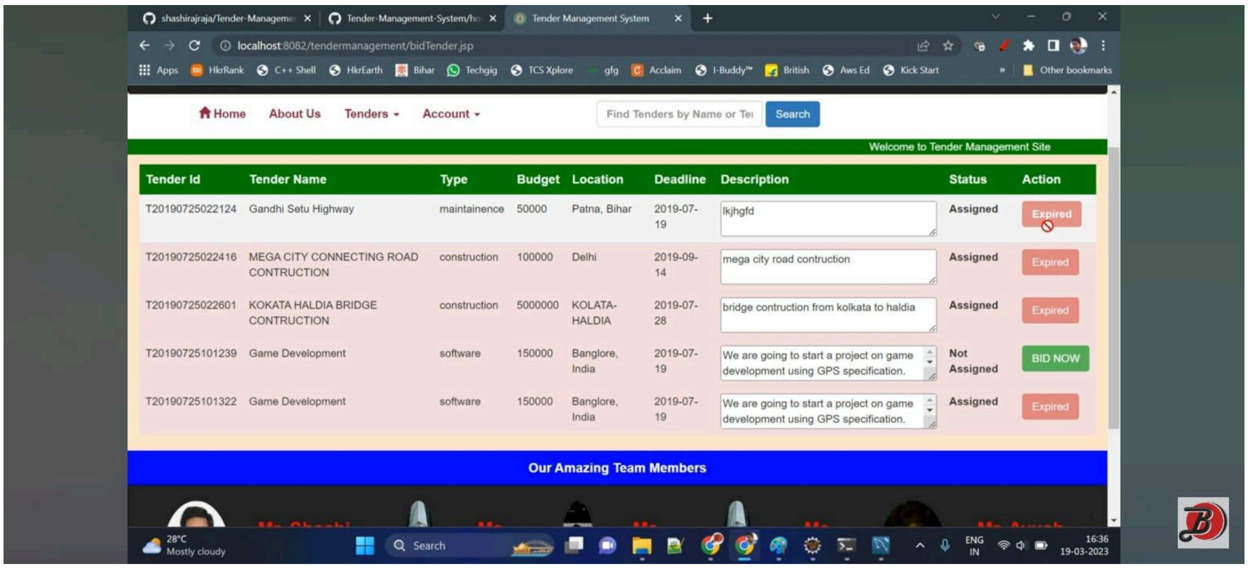
Save Tender



```
saveButton.addActionListener(e -> {
    String title = txtTitle.getText();
    String description = txtDescription.getText();
    String deadline = txtDeadline.getText();

    try {
        insertTender(title, description, deadline);
        JOptionPane.showMessageDialog(null, "Tender Saved Successfully");
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error Saving Tender");
    }
});
```

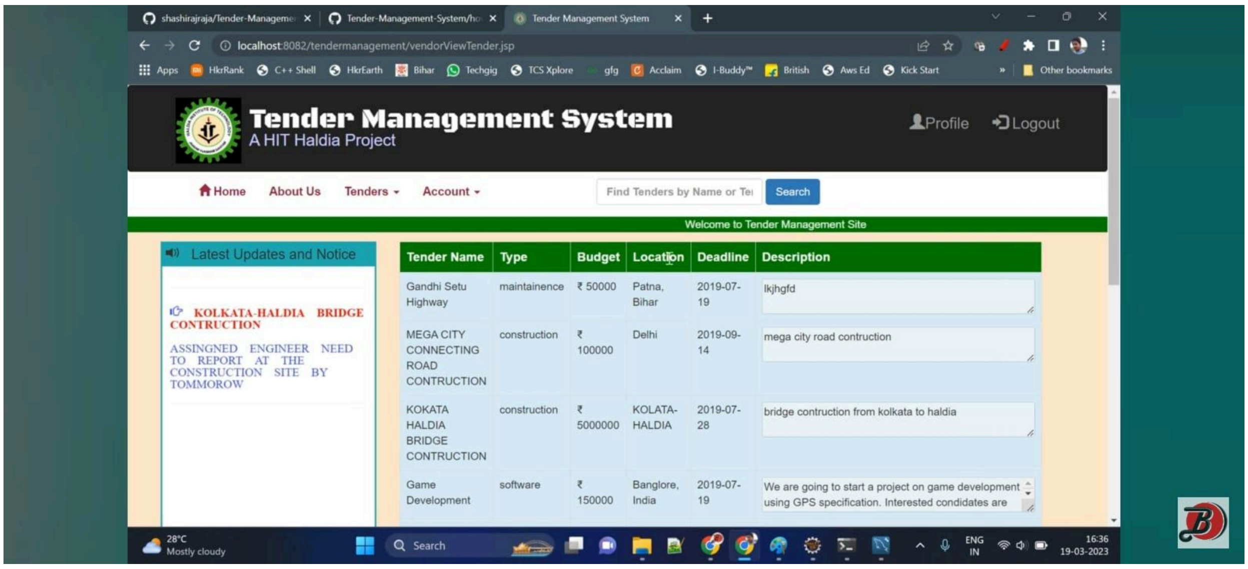
Chapter 6: Snapshots



localhost:8082/tendermanagement/bidTender.jsp
Home
About Us
Tenders
Account
Find Tenders by Name or Te
Search
Welcome to Tender Management Site
Our Amazing Team Members

Tender Id	Tender Name	Type	Budget	Location	Deadline	Description	Status	Action
T20190725022124	Gandhi Setu Highway	maintainence	50000	Patna, Bihar	2019-07-19	lkjhgfd	Assigned	Expired
T20190725022416	MEGA CITY CONNECTING ROAD CONTRUCTION	construction	100000	Delhi	2019-09-14	mega city road contruction	Assigned	Expired
T20190725022601	KOKATA HALDIA BRIDGE CONTRUCTION	construction	5000000	KOLATA-HALDIA	2019-07-28	bridge contruction from kolkata to haldia	Assigned	Expired
T20190725101239	Game Development	software	150000	Banglore, India	2019-07-19	We are going to start a project on game development using GPS specification.	Not Assigned	BID NOW
T20190725101322	Game Development	software	150000	Banglore, India	2019-07-19	We are going to start a project on game development using GPS specification.	Assigned	Expired

TENDER VIEW PAGE



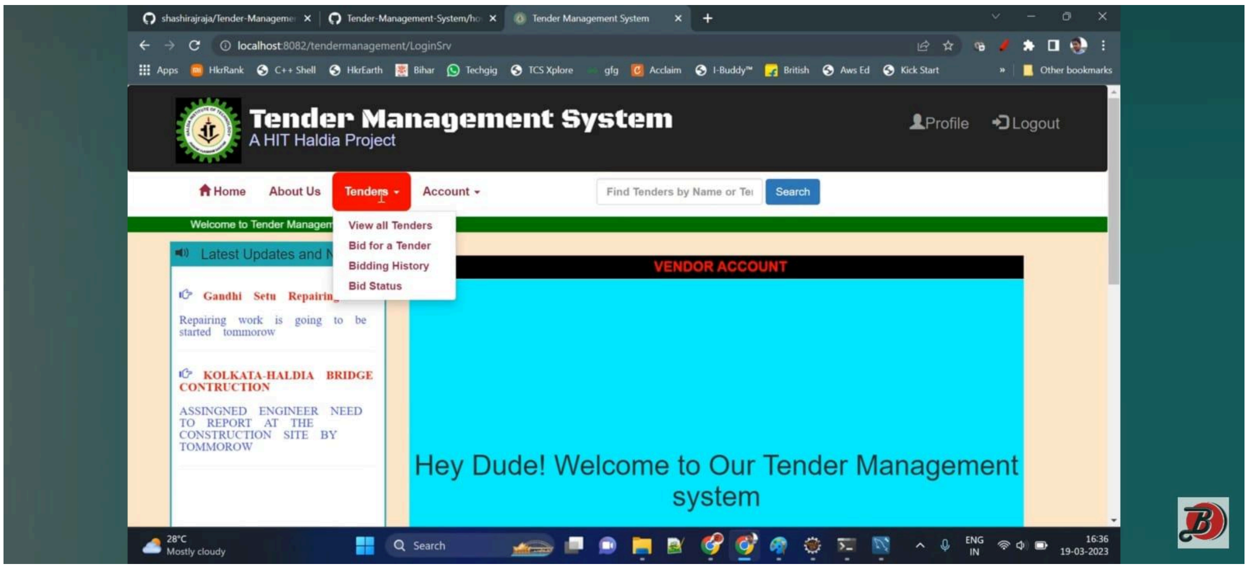
localhost:8082/tendermanagement/vendorViewTender.jsp
Tender Management System
A HIT Haldia Project
Profile
Logout
Home
About Us
Tenders
Account
Find Tenders by Name or Te
Search
Welcome to Tender Management Site
Latest Updates and Notice

KOLKATA-HALDIA BRIDGE CONTRUCTION

ASSINGNED ENGINEER NEED TO REPORT AT THE CONSTRUCTION SITE BY TOMMOROW

Tender Name	Type	Budget	Location	Deadline	Description
Gandhi Setu Highway	maintainence	₹ 50000	Patna, Bihar	2019-07-19	lkjhgfd
MEGA CITY CONNECTING ROAD CONTRUCTION	construction	₹ 100000	Delhi	2019-09-14	mega city road contruction
KOKATA HALDIA BRIDGE CONTRUCTION	construction	₹ 5000000	KOLATA-HALDIA	2019-07-28	bridge contruction from kolkata to haldia
Game Development	software	₹ 150000	Banglore, India	2019-07-19	We are going to start a project on game development using GPS specification. Interested condidates are

TENDER APPROVAL PAGE



Welcome to Tender Managem

Gandhi Setu Repairin

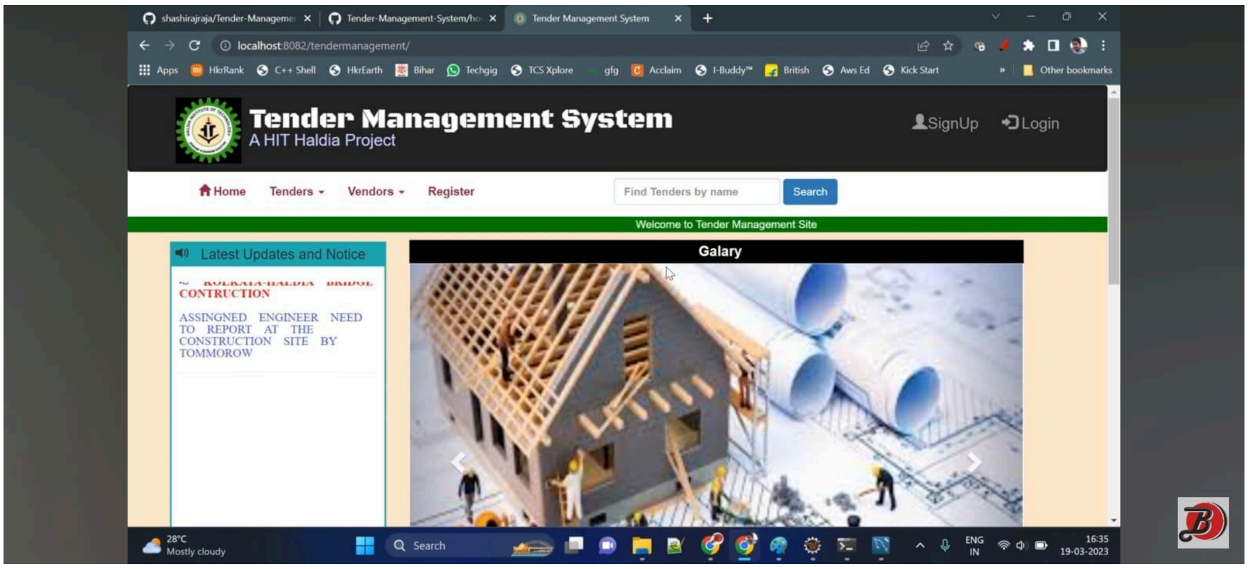
Repairing work is going to be started tommorow

KOLKATA-HALDIA BRIDGE CONTRUCTION

ASSINGNED ENGINEER NEED TO REPORT AT THE CONSTRUCTION SITE BY TOMMOROW

Hey Dude! Welcome to Our Tender Management system

WELCOME PAGE



Welcome to Tender Management Site

KOLKATA-HALDIA BRIDGE CONTRUCTION

ASSINGNED ENGINEER NEED TO REPORT AT THE CONSTRUCTION SITE BY TOMMOROW

LOGIN PAGE

Chapter 7: Conclusion

The Tender Management System has been developed to address the complexities and inefficiencies associated with the manual management of tenders within organizations. By automating key processes, this system provides a streamlined, efficient solution to manage tenders, vendor information, bids, and related data. The adoption of modern technologies such as Java Swing for the frontend, MySQL for the database, and JDBC for connecting the application to the database ensures the system is robust, scalable, and capable of meeting the growing demands of organizations.

Key Achievements

The primary achievement of this system is its ability to automate the tender management process, reducing the time spent on manual data entry and mitigating errors that are common in paper-based systems. The Tender Management System enables users to easily create, update, search, and delete tender records through an intuitive graphical user interface (GUI) designed with Java Swing. The use of a JDBC-based backend allows for seamless integration with the MySQL database, ensuring smooth data flow between the frontend and the backend.

The system's design incorporates a simple and effective relational database structure, with well-defined entities like Tenders, Vendors, and Bids, and their associated relationships. By ensuring proper data integrity through the use of foreign keys and structured queries, the system is able to maintain the quality and consistency of data. This, in turn, ensures that all users can rely on accurate and up-to-date information when managing tenders, making it an essential tool for modern organizations.

Efficiency and Productivity Gains

One of the most notable benefits of the system is the significant increase in efficiency and productivity. With the automation of tender creation, vendor management, and bid submission, administrative tasks are minimized, allowing personnel to focus on higher-value activities. The search functionality and data display tables allow users to quickly locate and interact with tender records, which speeds up decision-making processes and reduces administrative bottlenecks.

In addition to improving the overall productivity of administrators, the system also enhances the experience for vendors. By providing a centralized system for submitting bids, viewing tenders, and tracking bid statuses, the system simplifies the vendor interaction process and ensures transparency in tendering procedures.

Scalability and Flexibility

The Tender Management System is designed with scalability in mind. As the number of tenders, vendors, and bids increases, the system is able to handle larger volumes of data without compromising performance. The underlying MySQL database is capable of supporting a growing database of tender records, making the system suitable for organizations of any size. Furthermore, the system's modular design allows for future enhancements, such as the integration of advanced features like reporting tools, email notifications, or even role-based access control (RBAC).

The flexibility of the system also makes it adaptable to different industries and organizations, as it can be customized to suit specific tender management needs. Whether used by governmental organizations, private companies, or contractors, the core functionality of the system remains highly relevant and effective in any context.

Security and Data Integrity

Security is a crucial aspect of the Tender Management System, especially since it handles sensitive data related to tenders and vendors. The system's MySQL database ensures that data is stored securely, with the ability to implement user authentication and role management for controlling access. Additionally, the use of foreign keys between tables ensures that the database maintains referential integrity, preventing issues such as orphaned or inconsistent records.

As an added layer of security, the future version of the system could implement advanced encryption techniques to protect sensitive vendor information, bid amounts, and other critical data. These features would further enhance the trustworthiness of the system, making it a reliable platform for managing confidential business transactions.

Future Enhancements and Features

While the Tender Management System effectively meets the current requirements, there are several areas where additional functionality can further improve its capabilities. For instance, the system could benefit from the addition of advanced reporting features that provide actionable insights into tender performance, vendor performance, and financial trends. These reports could help administrators make more informed decisions and track the efficiency of the tendering process over time.

Additionally, the system could be enhanced with a notification feature, which would alert vendors about tender deadlines, bid status changes, or upcoming tenders. Such notifications would increase user engagement and ensure timely responses from vendors, leading to a more dynamic and responsive tendering process.

Another possible enhancement is the implementation of role-based access control (RBAC), which would allow for more granular control over user permissions. By assigning different access levels to different user roles (e.g., administrator, vendor, regular user), the system can ensure that sensitive information is only accessible by authorized users.

Conclusion Summary

In conclusion, the Tender Management System is an effective and efficient solution for modernizing the tender management process. By automating key tasks, ensuring data integrity, and providing a user-friendly interface, the system significantly improves operational efficiency, reduces human error, and enhances transparency in tendering procedures. With its ability to scale and its potential for future enhancements, the system is well-positioned to meet the needs of organizations in various industries.

The system’s design ensures that it is both flexible and secure, making it a reliable tool for managing tenders, bids, and vendors. As organizations continue to adopt this system, its value will increase, particularly as new features and improvements are added. Ultimately, the Tender Management System will continue to streamline the tendering process, contributing to better decision-making, greater efficiency, and improved organizational performance.

References

1. **Java Swing Documentation**
   Oracle. (n.d.). *Swing Overview*. Oracle. Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/
   - This documentation helped in understanding how to create graphical user interfaces using **Java Swing**, including how to use components like buttons, text fields, and tables.
2. **MySQL Documentation**
   MySQL. (n.d.). *MySQL 8.0 Reference Manual*. MySQL. Retrieved from https://dev.mysql.com/doc/
   - The official MySQL documentation guided the setup and management of the MySQL database, and was essential for constructing the database schema and executing SQL queries.
3. **Java Database Connectivity (JDBC)**
   Oracle. (n.d.). *JDBC - Java Database Connectivity*. Oracle. Retrieved from https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/
   - This guide provided crucial information on **JDBC** for connecting Java applications to MySQL, executing SQL queries, and managing database connections.
4. **MySQL Workbench Documentation**
   MySQL. (n.d.). *MySQL Workbench*. MySQL. Retrieved from https://dev.mysql.com/doc/workbench/en/
   - MySQL Workbench was used to design and manage the **Tender Management System's** database. This documentation was helpful in executing database operations like data migration and query optimization.
5. **Effective Java (3rd Edition)**
   Bloch, J. (2018). *Effective Java (3rd Edition)*. Addison-Wesley Professional. ISBN: 978-0134685991
   - This book provided insights on Java best practices, particularly on object-oriented design, which helped improve the structure and design of the **Tender Management System**.