Library Management System

**Project Overview:**

This project aims to develop **a comprehensive library management system that utilizes Spring Boot for the backend, ReactJS for the frontend, and MySQL as the database.** The system will facilitate the efficient management of library resources, including books, users, and borrowing activities.

**Target Users:**

The primary target users of the system are librarians, who will utilize the system to manage the library's collection, handle user accounts, and oversee borrowing and return processes. Additionally, students and faculty members will utilize the system to browse the library's catalogue, search for books, and place borrowing requests.

**Functional Requirements:**

* **User Management:**
  + Create, edit, and delete user accounts
  + Manage user roles and permissions
  + Track user borrowing history
* **Book Management:**
  + Add, edit, and delete book records
  + Categorize books using genres and subjects
  + Maintain book availability status
* **Borrowing and Returning:**
  + Allow users to search for available books
  + Facilitate borrowing requests for available books
  + Manage borrowing periods and due dates
  + Enable book returns and update book availability
* **Reporting and Analytics:**
  + Generate reports on book usage and borrowing trends
  + Analyze user borrowing patterns
  + Track overdue books and generate notifications

**Non-Functional Requirements:**

* **Performance:**
  + The system should be able to handle many concurrent users without performance degradation
  + Book search and filtering operations should be efficient and responsive
* **Security:**
  + Implement secure user authentication and authorization mechanisms
  + Protect sensitive data, such as user information and borrowing records
  + Prevent unauthorized access to library resources
* **Usability:**
  + The system should have a user-friendly interface that is easy to navigate and understand
  + Provide clear instructions and guidance for users
  + Implement consistent design patterns across the frontend and backend

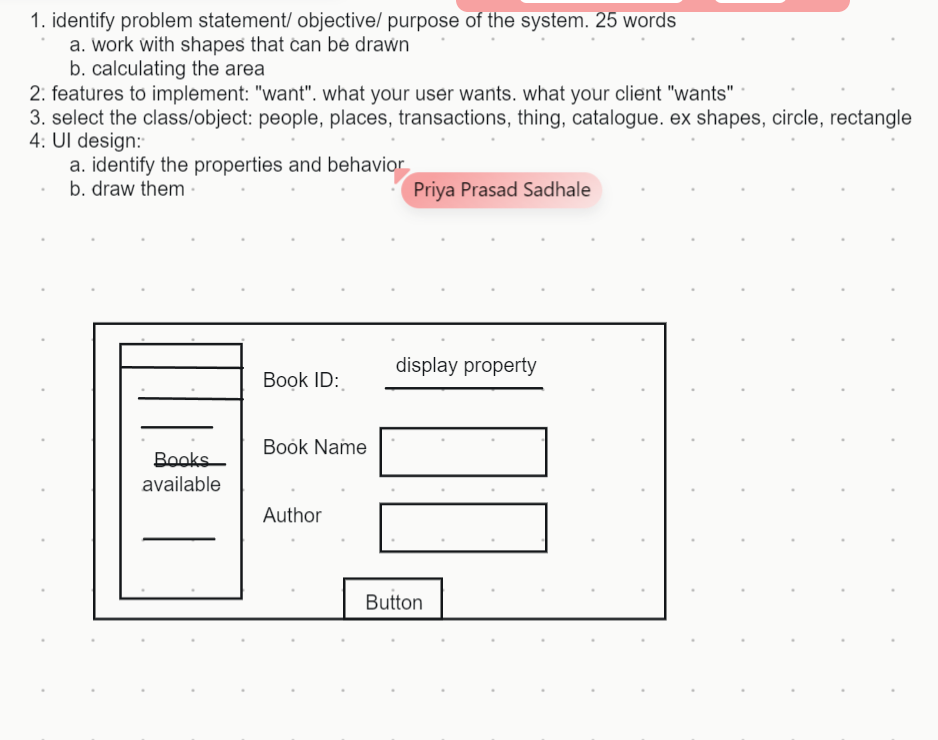
**Project Deliverables:**

* **Backend:**
  + Spring Boot application with RESTful APIs
  + Database schema and data migration scripts
  + Unit tests for backend components
* **Frontend:**
  + ReactJS application with user interface components
  + Routing and navigation between different pages
  + Integration with backend APIs for data fetching and manipulation
* **Documentation:**
* **Business Requirements Document (BRD):**
  + **This document outlines the business requirements, objectives, scope, and constraints of the project. It provides a high-level view of what the system is intended to accomplish.**
* **System Requirements Specification (SRS):**
  + **The SRS document provides detailed technical specifications, including functional and non-functional requirements. It describes the system's behavior, interfaces, data structures, and more.**
* **Design Documents:**
  + **System Architecture Design: Describes the high-level architecture of the system, including how the frontend and backend components interact, and the database structure.**
  + **UML: Use case, Class/Object diagram;**
  + **Database Design: Contains details about the database schema, including tables, relationships, and data constraints and ER Diagram.**
  + **UI/UX Design: Includes wireframes and mockups of the user interfaces in the React.js frontend.**
* **User Manuals:**
  + **User Guide: A document for end-users (librarians and administrators) explaining how to use the system. It should cover tasks like user registration, book management, and reporting.**
  + **Librarian Manual: Provides detailed instructions for librarians on managing book borrowings and returns.**

**Project Success Criteria:**

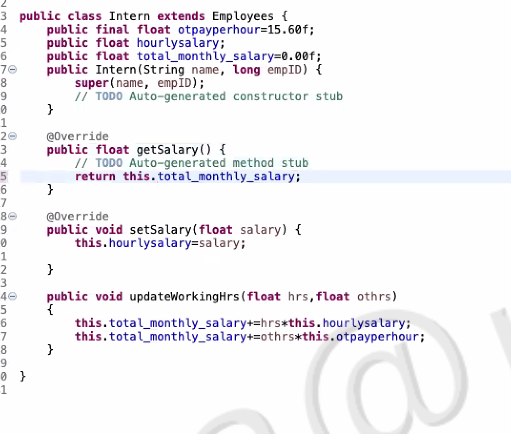
* The system meets all functional and non-functional requirements.
* The system is successfully deployed and used by librarians, students, and faculty members.
* The system receives positive feedback from users regarding its ease of use and functionality.
* The system demonstrates improved efficiency in library management operations.

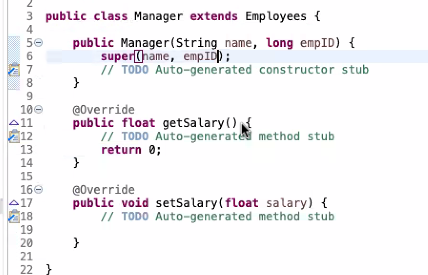
21st Dec 20231. Source code: Front-end & back-end: tables, sp2. Project Report: documentation3. PPT: final presentation of projectTimeline: 17th Nov till 20 Dec 2023

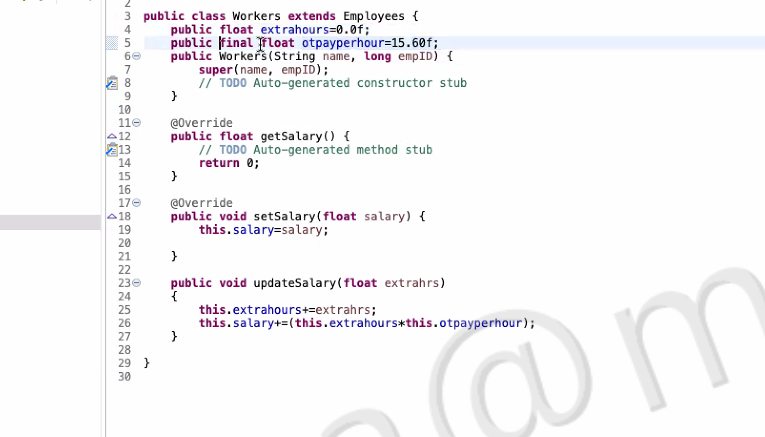


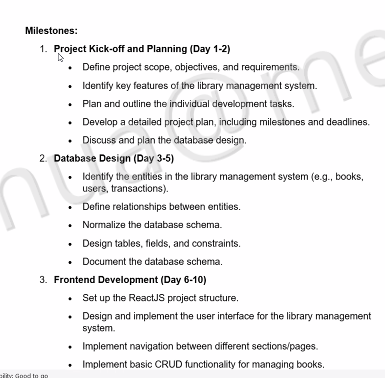
identify problem statement/ objective/ purpose of the system. 25 wordswork with shapes that can be drawncalculating the areafeatures to implement: "want". what your user wants. what your client "wants"select the class/object: people, places, transactions, thing, catalogue. ex shapes, circle, rectangleUI design:identify the properties and behaviordraw them

Scenario: Employee Hierarchy in a CompanySuppose we have various types of employees such as regular employees, managers, and interns. Each employee has common attributes like their name, employee ID, and basic salary. However, the way each employee calculates their salary and the additional responsibilities they have might differ.







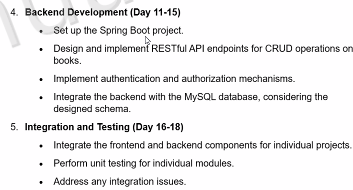


Minimum 4 tables

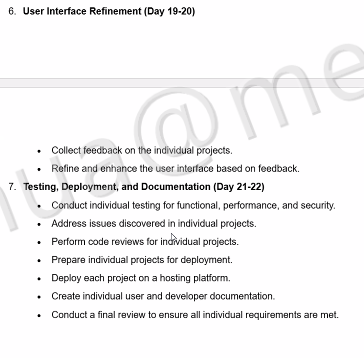
Use case, ER diagram, plant UML

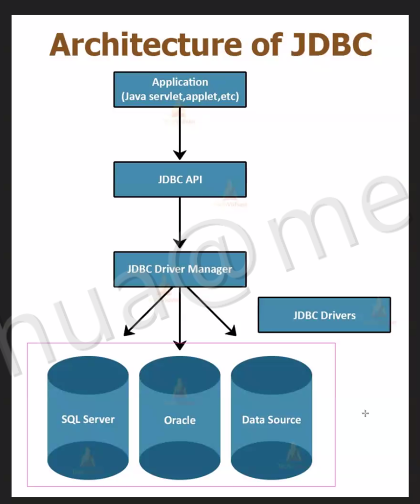
Create react app, minimum CSS is a must.

Bootstrap, tavern, for css alternatives.

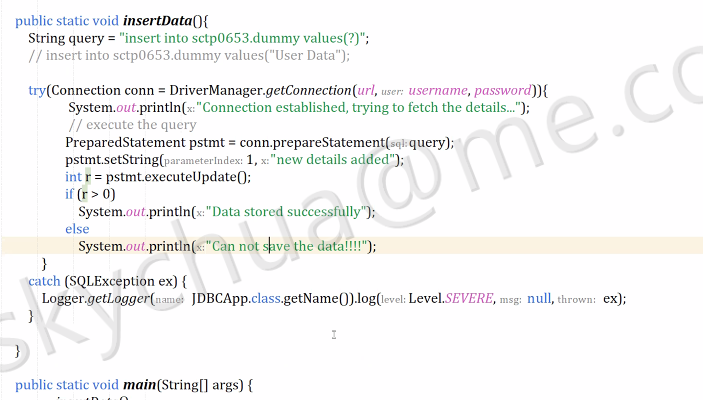


5 test cases





try(Connection conn = DriverManager.getConnection(url, username, password)){ System.out.println("Connection established, trying to fetch the details..."); // execute the query Statement stmt = conn.createStatement(); // resultset is used to store the result ResultSet rs = stmt.executeQuery("select \* from sctp0653.employees"); while(rs.next()){ System.out.println(rs.getInt("employee\_id") + " " + rs.getString(2) + " " + rs.getString("last\_name")); } }



public static void insertData(){ String query = "insert into sctp0653.dummy values(?)"; // insert into sctp0653.dummy values("User Data"); try(Connection conn = DriverManager.getConnection(url, username, password)){ System.out.println("Connection established, trying to fetch the details..."); // execute the query PreparedStatement pstmt = conn.prepareStatement(query); pstmt.setString(1, "new details added"); int r = pstmt.executeUpdate(); if (r > 0) System.out.println("Data stored successfully"); else System.out.println("Can not save the data!!!!"); } catch (SQLException ex) { Logger.getLogger(JDBCApp.class.getName()).log(Level.SEVERE, null, ex); } }