CS5363: DBMS II
Class Project

General Guidelines:

1. **Whatever projects you choose, please make sure to implement the following models:**
    a. **Data Modeling & Normalization:** Designing structured tables for products, categories, suppliers, and transactions with minimal redundancy.
    b. **SQL Queries & Transaction Integrity:** Ensuring accurate stock updates while preventing race conditions in concurrent transactions.
    c. **Indexing for Fast Retrieval:** Implementing indexes on frequently queried fields (e.g., product name, stock level) to enhance performance.
    d. **Triggers & Stored Procedures:** Automating inventory updates, alerts, and stock adjustments.Security & Access Control: Restricting critical operations to authorized users only.
    e. Your project design must match the real world requirements. Otherwise marks will be deducted from each of the sub-part mentioned above.
2. **Projects will be evaluated based on:**
    a. You must show the ER-Diagram, the relations, FDs, and normal forms. You must identify and justify all the functional dependencies in your setting, show how your database preserves all the FDs, and develops an E-R / schema diagram in the report. [20% Marks including viva]
    b. The database features that you make use of – such as, triggers, constraints, indexes, views, stored procedures etc. [40 % Marks including viva]
    c. The user interface designed with web technologies or other APIs. [40% Marks including viva]. I am expecting an MVC framework from your project. If you don't design MVC then 10% marks will be deducted. Some materials related to the MVC model will be shared in Google classroom.
3. **Consequences of copying online projects:**
    a. As you can see above, the maximum marks are on Database Design and Implementation. Also, an MVC model is connected with your database design. Hence, it is recommended that you do everything from scratch. If required, choose a small project and do it from scratch. Taking a larger project where the majority of the codes are downloaded will not help you.
    b. If you fail to answer any question related to your database or the GUI, marks will be deducted during viva. All group members will get the same marks for the project quality but marks may vary for viva.
    c. Whatever project you choose, make sure that

4. Other guidelines:
    - Databases can either be PostgreSQL/ MySQL.
    - There should be at least 10 relations in the database.
    - The database should be real, and consist of at least 1,000 tuples.

Some sample projects are mentioned next. Please note that, you can also choose your own project, after getting approval from me.

==

Project 1: Course Management System
For this course we are using the course-management system as Moodle. Implement a subset of the functionality of Moodle. For example, you can provide assignment submission and grading functionality, including mechanisms for students and teachers/teaching-assistants to discuss grading of a particular assignment. You could also provide polls and other mechanisms for getting feedback.

Key Functionalities:

1. Find some limitations in the current Moodle and implement it from scratch. You are not allowed to use the Moodle source code.

Project 2: Event Ticket Booking System
This system should be designed to manage tickets for small events, college fests, or workshops. By creating tables for event listings, seat categories, user bookings, and related data, you can explore how to handle concurrency when multiple users attempt to book the same spot.

Key Functionalities:

1. Event and Ticket Tables: Design tables to link seat availability, price tiers, and user information.
2. SQL Queries for Booking and Cancellations: Ensure seat counts adjust correctly after every transaction, maintaining consistency.
3. Concurrency Control: Prevent duplicate bookings or seat over-allocation by using transaction locking or isolation levels.
4. Indexing for Quick Lookups: Enable real-time queries to check seat availability in specific sections.
5. Support for multiple event listings and seat categories.
6. Online reservation and cancellation functionality.
7. Real-time seat availability queries

Project 3: Payroll Management System
A Payroll Management System is designed to handle employee salaries, deductions, and final payments while maintaining a transparent audit trail. This project is ideal for developing a deeper understanding of database integrity, especially for financial data that changes each pay cycle.

Key Functionalities:

2. Table Layout for Employee Info and Pay Structures: Consistently log salaries, bonuses, and tax deductions.
3. SQL Queries for Pay Calculations: Handle increments, overtime, and partial pay cycles.

4. Triggers and Stored Procedures: Automate updates for pay data across multiple employees.
5. Data Security and Encryption: Safeguard personal information and prevent unauthorized changes.
6. Indexing for Large Employee Sets: Quickly retrieve records to generate monthly statements.
7. Employee role and salary management.Automated deductions (e.g., taxes, provident fund).
8. Pay slips generated for each cycle.
9. Admin panel for salary revisions and approval processes.

Project 4: Inventory Management System (IMS)
The Inventory Management System (IMS) is a database-driven application designed to help users efficiently track incoming and outgoing stock across various products or categories. This system ensures real-time stock updates, reducing manual errors and improving ordering decisions. It allows businesses, student clubs, or small stores to manage their inventory efficiently while maintaining data integrity and security.

Key Functionalities:

1. Product & Supplier Management
- Register new products with details such as category, supplier, and stock level.
- Maintain supplier information and purchase history.
- Organize products into categories for easier retrieval.

2. Stock Tracking & Updates
- Monitor stock levels in real-time as purchases and sales occur.
- Automatically update stock quantity on sales or restocking.
- Set alerts for low stock items to prevent shortages.
- Integrate with supplier APIs for automatic restocking.

3. Sales & Purchase Transactions
- Record sales and purchases, ensuring transaction integrity.
- Prevent conflicting updates using SQL transactions.
- Track sales history to analyze inventory trends.
- Implement demand forecasting using past sales data.

4. User Roles & Access Control
- **Admin:** Manages users, suppliers, and stock levels.
- **Staff:** Can update stock, process transactions, and view reports.
- **Viewers:** Can check stock levels without making changes.
- **Auditor:** Can review transactions and ensure compliance.

5. Multi-Warehouse & Location-Based Inventory
- Manage stock across multiple warehouses or stores.
- Assign stock from the nearest warehouse for faster fulfillment.
- Optimize deliveries based on location-based inventory availability.

6. Reports & Analytics
- Generate reports on stock levels, sales trends, and purchase patterns.
- Identify overstocked and understocked items.
- Provide insights for optimized inventory planning.
- Implement AI-driven demand forecasting and reorder recommendations.

Project 5: Optimizing Online Sports Retail Revenue
The "Optimizing Online Sports Retail Revenue" project focuses on leveraging SQL for data-driven decision-making to enhance the profitability of an online sports retail business. By analyzing product performance, customer reviews, and sales trends, the system provides actionable insights to optimize inventory, pricing, and marketing strategies.

Key Functionalities:

1. **Product Performance Analysis**
   - Track sales data and revenue contribution per product.
   - Identify best-selling and underperforming items.
   - Monitor seasonal demand fluctuations.
2. **Customer Review Insights**
   - Analyze the impact of customer reviews on product sales.
   - Identify review-based sales trends (e.g., products with 50+ reviews performing better).
   - Provide recommendations for review collection strategies.
3. **Revenue Optimization Strategies**
   - Implement SQL-based analysis to determine pricing trends.
   - Optimize discount offers and promotional campaigns.
   - Suggest dynamic pricing strategies based on demand.
4. **Inventory Management Integration**
   - Ensure stock levels match product demand.
   - Set automatic alerts for low-stock products.
   - Prevent overstocking of slow-moving items.
5. **User Roles & Access Control**
   - **Admin:** Manage product listings, pricing, and user access.
   - **Marketing Analyst:** Access insights for targeted advertising campaigns.
   - **Sales Manager:** Track revenue trends and adjust sales strategies.
   - **Customer Support:** Monitor customer feedback and resolve complaints.

Project 6: Airline Reservation System with Dynamic Scheduling
Develop a database system to manage airline reservations, ticketing, and dynamic flight scheduling based on demand.

Key Functionalities:

1. Storage for flight schedules, passenger information, and bookings.
2. Real-time seat availability and schedule updates.
3. Multi-city routing and dynamic ticket pricing.
4. Support for cancellations and refunds.
5. **Technical Complexities:**
   a. Concurrency control to handle simultaneous bookings.
   b. Optimization for dynamic schedule updates.
   c. Transaction management to avoid double bookings.
   d. Performance tuning for high traffic scenarios.

Project 7: Cricket Tournament Management System
Develop a database system for maintaining fixtures, win loss, player details and match statistics based on match outcomes.

Key Functionalities:

1. **Tournament Management:**
    a. Storage for tournament details such as name, dates, and locations
    b. Creation of fixtures and schedules for all matches in the tournament
2. **Match Information:**
    a. Track match outcomes, scores, and key statistics
    b. Store detailed statistics for individual players, (runs, wickets, and batting/bowling averages)
    c. Support for different formats (T20, ODI, Test etc.)
3. **Team management:**
    a. Store team rosters and player assignments
    b. Keep a record of team performance, including wins, losses and rankings
4. **Dynamic Match Reporting:**
    a. Real time updates on match progress and results.
    b. Store live statistics, including player performances, overs bowled, wickets taken and runs scored.
5. **Ranking and Standings:**
    a. Automatically update team rankings based on match outcomes
    b. Calculate and display leaderboards for best performing players and teams.
6. **Technical Complexities:**
    a. Concurrency Control:
        i. Handle multiple users accessing and updating match data and statistics simultaneously
        ii. Ensure real time match updates without data inconsistencies.
    b. **Optimization for Performance**:
        i. Optimize queries for handling large amounts of match data and player statistics.
        ii. Efficiently calculate team rankings and player stats in real-time.
    c. **Data Integrity**:
        i. Maintain data consistency across match records, player details, and tournament progress.
    d. **User Access Control**:
        i. Implement role-based access for tournament organizers, team managers, and spectators.
        ii. Provide a secure interface for users to input match scores and player statistics.
    e. **Scalability**:
        i. Design the system to support large-scale tournaments with multiple teams and matches.
        ii. Allow the addition of new tournaments with minimal disruption to existing data.

Project 8: Real Estate Management System

The **Real Estate Management System** is designed to manage properties, tenants, lease agreements, rent payments, and maintenance requests. It provides tenants with the ability to view properties, submit maintenance requests, and make payments online. Admins have full control over managing property listings, lease terms, and tracking payments, ensuring a smooth management process.

Key Functionalities:

1. **Property Management**
   - Admins can add, update, or remove properties from the system.
   - Tenants can view property details such as location, type, price, and availability.
2. **Tenant Management**
   - Admins can add and update tenant information such as contact details and lease terms.
   - Tenants can view and manage their personal information, including lease agreements.
3. **Rent Payment System**
   - Tenants can securely pay rent online through integrated payment gateways.
   - The system automatically tracks payment history and sends reminders for upcoming payments.
4. **Maintenance Request System**
   - Tenants can submit maintenance requests with details of the issue.
   - Admins can assign maintenance tasks to staff and track the status of requests (pending, in-progress, completed).
5. **Lease Management**
   - Admins can create, update, and manage lease contracts for tenants.
   - Set automated reminders for lease renewal or expiration dates.

Technical Considerations:

- Tables for properties, tenants, leases, and payments are designed with normalization to minimize data redundancy.
- Use transactions to ensure secure payment processing, preventing errors in the rent payment or lease management process.
- Create indexes for frequently queried fields like tenant IDs, property addresses, and payment dates to optimize search performance.
- Use triggers to automatically update property availability when a lease ends or is renewed.
- Implement stored procedures to handle tasks such as generating invoices and updating payment status.
- Role-based access control is implemented to ensure only authorized users (admins, tenants, staff) can perform specific tasks.

Project 9: Job Portal System

The **Job Portal System** connects job seekers with employers. Job seekers can create profiles, browse job listings, apply for positions, and track their application status. Employers can post job openings, review applications, and manage candidates. The system ensures secure and efficient management of job applications and user profiles, providing a smooth recruitment process.

Key Functionalities:

1. **Job Listing Management**
    - Employers can post new job openings with detailed job descriptions and requirements.
    - Job seekers can search and filter job listings based on location, job type, and skills.
2. **User Profile Management**
    - Job seekers can create and update their profiles, including resumes, skills, and experience.
    - Employers can maintain company profiles with details like company name, industry, and location.
3. **Application Tracking System**
    - Job seekers can apply for jobs and track the status of their applications (e.g., "Applied," "Interview Scheduled," "Rejected").
    - Employers can view and manage job applications, shortlisting candidates for interviews.
4. **Job Search and Filtering**
    - Job seekers can search for jobs using filters like job type, salary, and experience level.
    - Employers can search for candidates based on skills, experience, and qualifications.
5. **Interview Scheduling and Notifications**
    - Employers can schedule interviews with shortlisted candidates and send automated interview notifications.
    - Job seekers receive email or in-app notifications for interview invitations and application updates.

Technical Considerations:

- Design normalized tables for job seekers, employers, job listings, and applications to ensure minimal redundancy.
- Use transactions to ensure that job applications are correctly linked to job seekers and job listings.
- Create indexes on frequently queried fields such as job title, location, and applicant skills for fast search results.
- Use triggers to automatically update application statuses when actions (e.g., interviews scheduled) occur.
- Implement RBAC to limit access to sensitive information, ensuring employers can only view their applications and job seekers can only view their profiles.

Project 10: Online Auction System

The **Online Auction System** is designed to facilitate the auctioning of items where sellers can list their products, buyers can place bids, and the system ensures a fair and efficient auction process. Users can browse available items, place bids, and track auction statuses. The system supports automated bidding, item status updates, and notifications to ensure users are informed about auction progress and results.

Key Functionalities:

1. **Item Listing and Management**
    - **Sellers** can list items with details such as item description, starting bid, auction start/end time, and images.
    - **Admin** can manage and approve items listed for auction, ensuring compliance with system rules.
2. **Bidding System**
    - **Buyers** can place bids on items, either manually or by setting up automated bids within a specified range.
    - **System** tracks the highest bid in real-time and ensures bid increments are followed according to auction rules.
3. **Auction Time Management**
    - Each auction has a defined start and end time. The system ensures auctions close automatically when the end time is reached.
    - **Admin** can extend or shorten auction times in case of disputes or issues.
4. **User Profile Management**
    - **Users** can create and manage profiles, including personal details, payment information, and past auction history.
    - **Buyers** can track their bidding history, and **Sellers** can view their sold items.
5. **Notifications System**
    - **Users** receive email or in-app notifications when they are outbid, a bidding limit is reached, or an auction ends.
    - **Admin** is notified of items that have not received bids or have violated system policies.

Technical Considerations:

- Design tables for items, bids, users, auctions, and transactions, ensuring that the database schema is normalized to 3NF.
- Key entities: `Item`, `Bid`, `User`, `Auction`, `Transaction`.
- Use transactions to ensure that bids are recorded accurately and auction data is updated atomically.
- Prevent race conditions during concurrent bidding by using transaction locks or isolation levels.
- Create indexes on commonly queried fields like `item_id`, `user_id`, `auction_id`, and `bid_amount` to speed up search and bid retrieval operations.
- Use **triggers** to automatically update the highest bid on an item when a new bid is placed.
- Implement **stored procedures** to handle auction closing, notification sending, and transaction finalization.
- **Role-based access control** (RBAC) ensures that sellers can manage their listings, buyers can bid, and admins have full control over item approval and system monitoring.

Project 11: Vehicle Rental System

The **Vehicle Rental System** is designed to allow users to rent vehicles, manage bookings, and track rental transactions. It provides an online platform for customers to view available vehicles, make reservations, and track payment statuses. Admins can manage vehicle inventories, customer details, rental prices, and booking records. The system ensures rental transactions, real-time availability updates, and accurate tracking of rentals.

Key Functionalities:

1. **Vehicle Management**
    - **Admins** can add, update, or remove vehicle details, including vehicle type, availability, rental price, and vehicle condition.
    - **Users** can view vehicle details like type, features, and pricing to make informed rental choices.
2. **Booking and Reservation System**
    - **Customers** can browse available vehicles and make bookings for specific rental dates.
    - The system checks for availability and confirms the booking if the vehicle is free on the requested dates.
3. **Payment System**
    - **Customers** can pay for rentals online through integrated payment gateways.
    - **Admin** can track payments, generate invoices, and apply discounts or promotions to rental charges.
4. **Rental History and Tracking**
    - **Customers** can view their rental history, including past rentals, payment status, and invoices.
    - **Admins** can monitor ongoing rentals, vehicle condition, and return schedules to ensure timely vehicle returns.
5. **Vehicle Return and Maintenance**
    - **Customers** can return vehicles, and **admins** can update vehicle status and track the condition of the vehicle (e.g., any damages).
    - Automated reminders for return due dates are sent to customers.

Technical Considerations:

- Create normalized tables for vehicles, customers, bookings, payments, and transactions to minimize redundancy.
- Key entities: `Vehicle`, `Customer`, `Booking`, `Payment`, `Transaction`.
- Ensure that transactions are atomic, ensuring secure booking and payment processes.
- Implement transactions to handle booking, payment, and inventory updates to maintain data integrity.
- Implement indexes on frequently queried fields such as `vehicle_id`, `customer_id`, `booking_date`, and `payment_status` for faster query performance.
- Use **triggers** to automatically update vehicle availability when a booking is made or returned.
- **Stored procedures** for generating invoices and processing payments securely.
- Implement role-based access control to restrict sensitive operations (like payment processing and vehicle management) to authorized users (admin, staff).