International Islamic University Chittagong (IIUC)

Department of Computer and Communication Engineering

Program: B.sc (Eng.)
Course Code: CCE–3506
Type: Lab
Segment: Lab work outline

Baizid MD Ashadzzaman
Designation: Adjunct Lecturer
Email: baizid.md.ashadzzaman@gmail.com
Phone: 8801862420119

Introduction to DBMS software:

Software		Contents	Note
Oracle Mandatory	Oracle express edition OracleXE213_Win64	Information: Oracle Database Express Edition (XE) is a free, lightweight version of the powerful Oracle Database. It's designed for developers, DBAs, educators, and anyone who wants to get started with Oracle without the cost or complexity of the enterprise editions. Key Features: • Free: No licensing costs. • Easy to use: Simple installation and configuration. • Powerful: Includes many features from the full Oracle Database, such as SQL, PL/SQL, and database management tools. • Limited resources: Suitable for smaller-scale applications with restrictions on database size (11GB), memory usage (1GB), and CPU usage (1 core). Ideal for: • Learning Oracle Database concepts and technologies • Developing and testing small-scale applications • Teaching database courses • Building prototypes Limitations: • Smaller database size limit (11GB) • Limited memory and CPU usage • No commercial support (community-based support available)	
	Sql developer by oracle Sqldeveloper-23.1.1.345.2114-x64	Information: Oracle SQL Developer is a powerful and free integrated development environment (IDE) specifically designed for working with Oracle databases. It provides a user-friendly interface for:	

		• Database development: Creating and managing database objects like tables, indexes, views, and procedures. • SQL query execution: Running SQL queries and analyzing results. • PL/SQL development: Creating, editing, and debugging PL/SQL code. • Database administration: Performing various administrative tasks like backup and recovery, performance tuning, and security management. • Data modeling: Creating and visualizing database schemas. Key Features: • Cross-platform compatibility: Runs on Windows, macOS, and Linux. • Intuitive interface: Easy to learn and use. • Rich functionality: Offers a wide range of features for database development and management. • Free: No licensing costs.	

Software		Contents	Note
Xampp Mandatory	Xampp p-v8.2 xampp-windows-x64-8.2.12-0-VS16-installer	Information: XAMPP is a free and open-source software package that provides a development environment for web applications. It's a popular choice for developers, students, and hobbyists to test their websites and web applications locally before deploying them to a live server. XAMPP stand for : • X: Cross-Platform (works on Windows, macOS, and Linux) • A: Apache HTTP Server (for serving web pages) • M: MySQL (database for storing information) • P: PHP (programming language for web development) • P: Perl (another programming language, though less commonly used) Key components of XAMPP: • Apache HTTP Server: Handles requests for web pages and delivers them to the client. • MySQL: Stores and manages data for your web applications. • PHP and Perl: Programming languages used to create dynamic web content. Why use XAMPP? • Easy to install: XAMPP simplifies the setup process for a local web server environment. • Cross-platform compatibility: Works on different operating systems. • Free: No cost to use. • Open-source: You can modify the code if needed.	

		• Ideal for development: Provides a testing ground for web applications before deployment. In essence, XAMPP is a convenient package that bundles essential tools for building and testing web applications on your local computer.	

Software		Contents	Note
Laragon Not mandatory	Laragon p-v8.2 xampp-windows-x64-8.2.12-0-VS16-installer	Information: Laragon is a popular, portable, and fast local development environment designed for web developers. It's particularly well-suited for PHP projects but also supports Node.js, Python, Java, Go, Ruby, and more. Key Features: • Portability: You can easily move Laragon to different computers without reinstalling. • Isolation: Each project runs in its own isolated environment, preventing conflicts. • Speed: Optimized for performance to boost development efficiency. • All-in-one: Includes essential tools like Apache, Nginx, MySQL, PHP, Node.js, Python, Git, and more. • Easy to use: Simple interface and straightforward setup. • Extensibility: You can add additional software and configurations. Benefits: • Faster development: Quickly set up new projects with pre-configured environments. • Improved organization: Keep projects separate and avoid conflicts. • Enhanced productivity: Focus on coding without worrying about server setup. • Consistent development environment: Ensure your code works the same locally and on the server. In essence, Laragon provides a comprehensive and efficient platform for web developers to build and test their applications locally.	

Software		Contents	Note
DB Browser Not mandatory	DB Browser (SQLite) xampp-windows-x64-8.2.12-0-VS16-installer	Information: DB Browser for SQLite is a popular, open-source graphical tool designed to help users interact with SQLite databases. It provides a user-friendly interface for creating, viewing, editing, and managing SQLite database files. Key Features: • Visual interface: Offers a spreadsheet-like view of database tables for easy navigation.	

		• SQL query support: Allows you to execute complex SQL queries. • Data editing: Enables you to modify table data directly. • Import/export: Supports importing data from CSV files and exporting data to various formats. • Table creation: Helps you define table structures and relationships. • Index management: Allows you to create and manage indexes. Who Uses It? • Developers: For testing and prototyping applications. • Data analysts: For exploring and analyzing small datasets. • Students: For learning database concepts. • Hobbyists: For personal projects involving data storage. Use Cases • Rapid prototyping: Quickly create and test database structures and data. • Data exploration: Analyze data visually and with SQL queries. • Small-scale applications: Build simple applications using SQLite as the backend. • Educational purposes: Learn database concepts and SQL. • Data migration: Transfer data between SQLite and other database systems. Limitations • Primarily designed for SQLite databases. • Not suitable for large-scale or high-performance applications. • Limited advanced features compared to enterprise-grade database management tools.	

Software		Contents	Note
Firebase Not mandatory	Firebase (NoSQL) xampp-windows-x64-8.2.12-0-VS16-installer	Information: The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. It's essentially a large JSON object that is shared across all clients. Any changes made to the data are reflected immediately on all connected devices. Key Features: • Real-time Synchronization: Data updates are instantly propagated to all connected clients. • Offline Capabilities: Data is cached locally, allowing for continued operations even without an internet connection. • Security Rules: Define who can read, write, and modify data. • Data Structure: Hierarchical JSON format for easy data organization.	

		• Scalability: Handles increasing numbers of concurrent connections. Limitations: • No SQL: Lack of complex query capabilities compared to relational databases. • Data Structure: Hierarchical structure might not be ideal for all data models. • Scalability Limits: While scalable, there are limits to the number of concurrent connections and data volume.	

Software		Contents	Note
Mongo DB Not mandatory	Mongo DB(NoSQL) xampp-windows-x64-8.2.12-0-VS16-installer	Information: MongoDB is a popular, open-source, document-oriented NoSQL database. Unlike traditional relational databases that use tables and rows, MongoDB stores data in flexible, JSON-like documents. This structure makes it highly adaptable to various data models and application requirements. Key Features: • Document Model: Data is stored as documents, similar to JSON objects, providing flexibility in data structure. • Scalability: MongoDB can handle large volumes of data and high traffic loads, making it suitable for big data applications. • High Performance: Optimized for read and write operations, offering fast query performance. • Indexing: Supports various indexing techniques to improve query efficiency. • Replication and Sharding: Ensures data durability and availability through replication and partitioning. • Aggregation Pipeline: Powerful data aggregation and analysis capabilities. • Rich Query Language: Flexible query syntax for complex data retrieval. Limitation: • Complex Relationships: MongoDB's document-oriented structure can be challenging for modeling complex relationships, such as hierarchical or many-to-many relationships. • Data Normalization: Achieving optimal data normalization can be more complex compared to relational databases. • Query Complexity: Complex queries can be less efficient than in relational databases, particularly those involving joins or aggregations over large datasets. • Compared to relational databases: MongoDB has a smaller ecosystem of tools and mature practices for data modeling, query optimization, and performance tuning.. • Complex Indexes: Creating and managing complex indexes can be more challenging than in relational databases.	

Understanding and implementing DB/CRUD operation using SQL and manual mode:

Task	Contents	Note
Database setup and operation	Information: • Introduction to database management systems (DBMS) and their role in data management. • Overview of different types of DBMS (relational, hierarchical, network, object-oriented). • Steps involved in setting up a database: a. Database creation b. Table creation c. Data types and constraints d. Indexes • Database connectivity using appropriate drivers or connectors. Note: Provide hands-on practice in creating a database, defining tables, and understanding data types and constraints.	
Table create	Information: • Importance of table design in database normalization. • Steps involved in creating a table: a. Defining table name and columns. b. Specifying data types for columns. c. Adding primary key and foreign key constraints. d. Creating indexes for efficient data retrieval. • Examples of different table structures based on real-world scenarios. Note: Guide students in designing tables for various entities and relationships, emphasizing normalization concepts.	
Insert data in table	Information: • SQL INSERT statement syntax and usage. • Inserting single and multiple rows of data. • Using different data types while inserting values. • Handling null values and default values. • Importance of data integrity and validation. Note: Provide exercises on inserting data into tables with different data types and constraints.	
Read data from table	Information: • SQL SELECT statement syntax and usage. • Retrieving data using various selection criteria (WHERE clause). • Sorting data using ORDER BY clause. • Filtering data using DISTINCT keyword. • Calculating aggregate functions (COUNT, SUM, AVG, MIN, MAX). • Grouping data using GROUP BY clause. Note: Provide exercises on retrieving data in different formats and performing calculations on data.	
Update data in table	Information: • SQL UPDATE statement syntax and usage. • Modifying existing data based on conditions. • Updating multiple rows at once. • Precautions to avoid data inconsistencies.	

	Note: Provide exercises on updating data with different conditions and ensuring data integrity	
Delete data in table	Information: • SQL DELETE statement syntax and usage. • Removing rows based on specific criteria. • Deleting all rows from a table (TRUNCATE). • Precautions to avoid accidental data loss. Note: Emphasize the importance of backups before performing delete operations.	
Extra (advance learning)	Introduction: • Advanced SQL concepts (joins, subqueries, views, stored procedures, triggers). • Database performance optimization techniques (indexing, query tuning). • Database security and access control. • Database administration tasks (backup, restore, monitoring). Note: This section can be optional or covered in subsequent labs based on the course level.	

Design and develop Database for a management system (group project):

This project requires students to work collaboratively in a group to design and implement a database for a specific management system. The goal is to apply database design principles, data modelling techniques, and SQL to create a functional and efficient database solution.

Process	Contents	Note
Phase 1	System Analysis and Requirements Gathering: • Identify the Management System: Clearly define the type of management system the database will support (e.g., library, hospital, school, inventory, HR). • Define System Scope: Determine the boundaries of the system, including the core functionalities and data to be managed. • Identify Users and Their Needs: Determine the different user roles and their specific requirements for accessing and manipulating data. • Gather Data Requirements: Identify the essential data elements to be stored in the database, considering entities, attributes, and relationships.	
Phase 2	Database Design: • Conceptual Data Modelling: Create an Entity-Relationship (ER) diagram to visually represent the entities, attributes, and relationships within the system. • Logical Data Modelling: Translate the ER diagram into a logical data model, defining tables, columns, and data types. • Normalization: Apply normalization techniques to ensure data integrity and efficiency. • Database Schema Creation: Create the database schema using a chosen DBMS (e.g., MySQL, Oracle, SQL Server).	
Phase 3	Database Implementation: • Table Creation: Create tables in the database based on the normalized design. • Data Population: Populate the database with sample data to test the system.	

	• Constraints and Indexes: Define primary keys, foreign keys, and indexes to enforce data integrity and improve query performance.	
Phase 4	Database Testing and Optimization: • Data Validation: Test the database by inserting, updating, and deleting data to ensure data integrity. • Query Performance: Evaluate query performance and optimize database structure and indexes as needed. • Error Handling: Implement error handling mechanisms to gracefully handle unexpected situations.	
Phase 5	Database Documentation: • Database Schema Documentation: Create detailed documentation of the database schema, including table structures, relationships, and constraints. • User Manual: Develop a user manual explaining how to interact with the database (if applicable).	

By following these guidelines, students will gain hands-on experience in database design and development, fostering critical thinking and problem-solving skills.

Deliverables	• A clear project proposal outlining the chosen management system and project scope. • A well-structured ER diagram. • A detailed database schema. • A populated database with sample data. • Test cases and results. • Comprehensive database documentation.	
Evaluation Criteria	• Adherence to database design principles. • Quality of the ER diagram and database schema. • Efficiency of database implementation. • Correctness of data and query results. • Completeness and clarity of documentation. • Teamwork and collaboration among group members.	
Additional Considerations	• Encourage students to explore advanced database concepts such as views, triggers, and stored procedures. • Consider incorporating user interface design for data entry and retrieval. • Emphasize the importance of data security and privacy. • Provide opportunities for students to present their project and receive feedback.	

Develop a simple portfolio using database and flask (individual project):

This individual project aims to provide students with hands-on experience in designing and implementing a web application using Python's Flask framework and a database. Students will learn to integrate database operations with web development to create a functional portfolio website.

Process / Method		Contents	Note
Personal portfolio Mandatory	UI/UX design	Information: Html, CSS, JavaScript	
	Database develop	Information: Oracle or Xampp	

	Backend development	Information: Python , Flask	

Process	Contents	Note
Phase 1	Database Design: • Define the entities and relationships for the portfolio (e.g., projects, skills, experiences). • Create an ER diagram to visualize the database structure. • Design database tables with appropriate columns and data types. • Create the database schema and populate it with sample data.	
Phase 2	Flask Application Setup: • Set up a Flask project structure. • Configure the Flask application with necessary settings (e.g., template directory, static file directory). • Establish a database connection . • Create database models corresponding to the database tables.	
Phase 3	CRUD Operations: • Implement CRUD operations for portfolio items using Flask. • Create routes for adding, editing, and deleting portfolio items. • Handle form submissions and database updates.	
Phase 4	Portfolio Website Development: • Create HTML templates for different pages (e.g., home, about, projects, contact). • Design the layout and styling of the portfolio website using CSS. • Integrate database-fetched data into the templates to display portfolio content.	
Phase 5	Testing and Deployment: • Test the application thoroughly to ensure correct functionality. • Fix any bugs or errors encountered during testing. • Deploy the application to a web hosting platform (optional).	
Evaluation Criteria	• Correctness of database design and implementation. • Efficiency of Flask application and database interactions. • User-friendliness and visual appeal of the portfolio website. • Adherence to coding standards and best practices. • Overall project quality and completeness.	
Additional Considerations	• Encourage students to explore advanced features like user authentication, content management systems, and responsive design. • Provide guidance on database optimization and performance tuning. • Emphasize the importance of security and data protection.	

Understanding how modern framework uses DB queries (advance learning):

Process / Method		Contents	Note
Laravel Not mandatory	Raw SQL query	Information: Laravel provides a robust query builder to interact with databases, making it easy to construct and execute SQL queries without writing raw SQL directly. However, there are instances where you might need to use raw SQL for complex queries, performance optimization, or specific database features that the query builder doesn't cover.	
	Query builder	Information:	

		Laravel's query builder provides a convenient and expressive way to interact with your database. It offers a fluent interface that chains methods together to construct complex queries.	
	Eloquent query	Information: Eloquent is Laravel's Object-Relational Mapper (ORM) that provides an elegant, expressive, and fluent interface for interacting with your database. It simplifies the process of querying, inserting, updating, and deleting records. Core Concepts • Models: Represent database tables as PHP classes. • Relationships: Define relationships between models (one-to-one, one-to-many, many-to-many). • Query Builder: Provides methods for constructing and executing database queries.	

Database deployment or connect with online server (advance learning):

Process	Contents	Note
Server deploy Not mandatory	• Introduction to database deployment options (local, cloud, hybrid). • Understanding the concept of database servers and clients. • Connecting to a database server using appropriate drivers or connectors. • Deploying a database application to a web server. • Database backup and restoration strategies. • Security considerations for database deployment (firewalls, authentication, encryption). • Cloud database services (e.g., AWS RDS, Azure SQL Database, Google Cloud SQL). Note: This section can be covered as a separate lab or integrated into other labs depending on the course structure. Practical exercises should involve deploying a database application to a local or cloud server and connecting to it from different clients.	
	Additional: • If cloud database services are introduced, provide hands-on experience in creating and managing databases on these platforms. • Discuss the advantages and disadvantages of different deployment options based on project requirements. • Cover topics like database scaling, high availability, and disaster recovery.	

Live Q&A Session with Industry Expert in Field of DBMS (career path):

To provide students with insights into the real-world applications of database management systems and career opportunities in the field.

Topic	Contents	Note
Q&A session Not mandatory	Format: • A live online / offline session with an industry expert. • A question-and-answer format where students can ask questions about the expert's career path, industry trends, and challenges.	
	Potential Topics:	

	• The role of DBMS in various industries (e.g., finance, healthcare, e-commerce). • Database technologies and trends (e.g., NoSQL, cloud databases, big data). • Career paths in database management (database administrator, data analyst, data scientist, data engineer). • Skills and qualifications required for a successful career in DBMS. • Challenges and opportunities in the field of database management. • Advice for students starting their career in DBMS.	
	Benefits: • Provides students with a real-world perspective on the field. • Helps students understand the importance of database management in different industries. • Offers career guidance and advice from an experienced professional. • Encourages networking and industry connections.	
	Additional Considerations: • Choose an industry expert with relevant experience and expertise. • Promote the session to students well in advance. • Encourage students to prepare questions in advance. • Record the session for future reference (with the expert's permission).	