CS364 Database Application Project

Semester-long Project

CS 364 – Fall 2025

Major Component	Minor Component	**Points**
	Partner Signup	0
Proposal		20
	Check-in 1	2
	Check-in 2	2
	Check-in 3	2
	Check-in 4	2
	Check-in 5	2
Demo		50
Report		30
	Group & Self-eval	10
Total		120
Submission		Groups of 2 (or 3)

1 Objectives

The purpose of this assignment is to practice the various database skills and concepts covered during the course through the design and development of a real database and database application. The project has several deliverables to provide timely feedback and help you structure the development of your project.

2 Deliverables

◯ **Partner Sign-up** – you will need to tell me who you are working with on the project, OR let me know you are looking for a partner. If you have a partner (or team of three), all team members MUST submit the partner sign-up with who they want to work. If you are looking for a partner, include topics for the project you are interested in and your preferred work schedule. I will do my best to match people up based on this information. I will announce the teams shortly after the sign-up period.

◯ **Proposal** – you will need to complete a project proposal. The purpose of the proposal is to describe the real world problem you are modeling, the stakeholders for the database and application, the database design, the functionality you plan to complete, and the tools you plan to use to complete the project.

◯ **Check-ins** – you will complete a weekly check-in as a group to share with me your progress on the project and ask any questions that may have come up. The purpose of this is to help you stay on track and model the software development processes that are used in industry. It is also useful for me to be able to help you with technical and design issues throughout the project. Groups that take the time to craft detailed check-ins and keep up with the project tend to do well on the project overall and enjoy the experience.

◯ **Demo** – you will demonstrate your database and database application for me. This is the bulk of the points and where you will show both the frontend application and the underlying database. I expect you to also show me your updated ER diagram and describe the purpose of your application. All members should be able to navigate and explain the application and database.

◯ **Report** – you will produce a final report about your project. While it is similar in content and structure to the proposal, this report should describe your finished product. This means it should be written in the appropriate tense, the ER diagram and functionality describes what was actually completed, and screenshots of the finished application should be include to document the finished state of the application.

◯ **Group & Self-eval** – you will evaluate your teammates' performance on the project as well as your own. I also appreciate feedback on the structure of the project and tips for the next group of students.

3 Major Components

3.1 Proposal

The proposal document will describe the project you will complete for this course. Its purpose is to describe the real world problem, your database design, your database application functionality, and technical requirements. The document should be written in a professional manner and be written for an audience that has the same technical background as you do. The document must include the names of all the team members, the name of your project, and be submitted as a PDF by the due date posted on Canvas. The content of the document should be organized as follows:

- Problem Statement – this section should succinctly describe the real world problem you are modeling, the data and functionality that will be modeled and implemented, and the stakeholders or users of the application. Be sure to clearly define what is and is not part of the scope of your project.
- Database Design – this section will include an ER diagram of your project as well as a list of the tables you intend to create based on that design and the attributes of each table. The purpose of this section is to provide the reader with an understanding of the data that will exist in the database and how it will be structured. (See the next section for a full description of the database requirements.)
- Functionality – this section will describe the functionality of your database application. The functionality of your application can be broken into two categories: CRUD and advanced queries. You will describe each functional activity and present them as a bulleted list. You can write them as user stories if you wish. (See the next section for a full description of the database application requirements.)
- Technical Requirements – this section will describe the languages, tools, machines, and services you will use to implement your project. You must include the project modality (e.g., desktop, web), the language(s) and DB connector you intend to use (e.g., Java and JDBC), the database platform you intend to use (e.g., MySQL on the class server, SQLite on a team member's computer), how you plan to share code (e.g., GitHub), and what experience with these tools your team members have. It is a good idea for some team members to have some experience with some of the tools, but it is not required for you to have prior experience with everything. You may change these before the final project, but it is best to choose carefully now a set of tools that you will be successful with.

3.2 Database and Database Application

This section describes the technical requirements for the database and database application. The technical goal of this project is to demonstrate you know how to design and implement the SQL required to setup, populate, and query a **database**, as well as design a front-end **database application** that interacts with a database in interesting ways. These should be reflected in the proposal, demo, and final report.

3.2.1 Database Requirements

You will design and implement a database with the following characteristics:

- Your database should be implemented using a relational database implementation such as MySQL, SQLite, Microsoft SQL Sever, or Postgres. NoSQL databases such as MongoDB are not acceptable. If you are unsure, check with the instructor.
- Your database is required to have at least five (5) tables, at least three (3) of which must be represented as entity types in the ER diagram.
- When describing your tables in the proposal and report, be sure to describe how relationships in the ER diagram are captured by the tables.
- The relationships in your database design should include multiple relationships that are either one-to-many or many-to-many. In other words, your database should have interesting relationships between the data to highlight the capabilities of relational database and demonstrate your understanding of how to utilize them well. You will also find it is difficult to write interesting advanced queries without this requirement.

- There must be sufficient data in the database to write interesting queries and demonstrate the usefulness of the application. All of your tables should have 10s of tuples and some should have 100 or more for most applications. Many projects end up with more data, especially if you use real data or use a tool like Mockaroo to generate data.
- When populating the tables with data make sure the data makes sense (has data integrity). In other words, the relationships are maintained when adding data and it is reasonably representative of the real world problem.
- When describing your data in your report, you must cite where you got the data from, or how you generated the data. Most students will use a mix of approaches. For example, you might get real data about books for a library application, generate some data for library patrons using mockaroo, and then manually create entries to represent patrons checking out books. You need to describe where you got data for each table. If it is from an external source, you should cite it. If it is generated by a tool, you need to mention the parameters you used to generate it and cite the tool used. For data you manually created, mention any details that explain your methodology for creating the data such that it represents the real world problem.

3.2.2 Database Application Requirements: CRUD

You will design and implement a database application with the following characteristics:

- You will implement a UI application for your database. Remember that it is important that end users do not directly access the database, so applications are created to give users access to the data but in a controlled manner. For full credit, your UI application must be a GUI (either desktop or web). Partial credit is granted for a command-line interface.
- Your application should implement CRUD for at least some of the data. CRUD stands for creation, read, update, and delete. This means that your application should allow users to add new data to the database, view data in the database, update data in the database, and delete data from the database.
- Your application should make use of prepared statements and allow users to enter data that is then used in the query to do CRUD and advanced operations.
- Think about your users and make sure the application provides a reasonably complete set of operations for the project you set out to make.
- Your application needs to implement at least three (3) advanced queries for each member of your team. Each team member should take ownership of these three advanced queries and come up with the SQL query and corresponding application code to implement the query for end users. Additional details on advanced queries are described below.

3.2.3 Database Application Requirements: Advanced Queries

You will design and implement complex SQL queries to demonstrate your knowledge of databases and how they can be used in interesting ways in real applications. **Each** team member is responsible for creating and presenting (at least) one (1) query (and corresponding application code) in **each** of the following groups. *If you have two group members, there should be six (6) total advanced queries!*

- **Group 1**: aggregate functions, `LIKE`, `GROUP BY`, `ORDER BY`, `LIMIT`
- **Group 2**: `HAVING`, `OFFSET`, outer join, joining four or more tables
- **Group 3**: subqueries, `IN`, set operators, any additional functionality outside of what was discussed in class will likely fall into this category (but please talk to me first if we did not cover this in class)
- In the **proposal** you will describe these advanced queries in words (you do not need the SQL ready yet) and what group they satisfy for each advanced query.
- In the **demo** you will demonstrate the functionality in the application for each advanced query. Most (or all) will use some user input to parameterize the query. I will also want to see the SQL code for the query in MySQL Workbench or similar.
- In the (report) you will include the SQL for all advanced queries and screenshots of some of the UI for the advanced queries.

3.3 Check-ins

Each week you will report on your progress and describe your goals for the next week. These reports model a daily scrum meeting from the Agile software development life cycle model[1]. There are lots of variations on this and different terminology, but in this class we will use the following structure for the weekly check-ins:

- **What did you do last week?** – this is where you describe the progress the team has made from the previous week. You can report what each person did or just describe your collective work.
- **What do you plan to do next week?** – this is where you plan out your tasks for the next week. Again, you can list individual team members and their tasks or just the collective work.
- **Are there any issues or questions or problems that need resolving?** – this is where you share any questions or technical problems you are having that I may be able to help with, or are just issues you are working through as a team. This is often where students will report problems with data or technologies that necessitate a change from what you proposed.

3.4 Demo

You will demonstrate your project as a group at the end of the semester. Details on how to sign up for a demo time will be posted on Canvas at least one week before demos start. Your demonstration will include a brief overview of the project, presentation of the ER diagram, brief demonstration of the database tables in an SQL tool (like MySQL Workbench), demonstration of the CRUD functionality of your application, and demonstration of the advanced queries implemented by each team member in both the application and SQL tool (like MySQL Workbench).

3.5 Final Report

The final major piece of the project is a final report that describes the final product you produced. While this is similar in structure and content to the proposal, the document needs to be written to reflect that the project is done. For example, instead of saying "In this project, we plan to use Java and JDBC to implement a desktop application." you should say "In this project, we implemented a desktop database application using Java and JDBC." Often things change as you develop a project, so any changes made to the database design, tables, functionality, and technical requirements should be included. For example, if you changed the design of the database, the ER diagram should capture the final design of the database.

In **addition to the four sections from the proposal**, you should include the following **new sections**:

- CRUD Screenshots – this section should contain screenshots capturing an example of the CRUD functionality of your application. At least one image for each of the four actions. Each screenshot should be captioned with a brief description of what is being shown. This should be a complete sentence mentioning the action being performed (create, delete, etc.) and the data being manipulated.
- Advanced Query Screenshots – this section should contain screenshots capturing the advanced queries implemented in the application and the SQL to run them in an SQL tool (like MySQL Workbench). Each screenshot should be captioned with the author of the query and the group it belongs to.
- Normalization – this section should describe what normalization level your database is in. It should be in at least 2NF. Most projects will be 3NF, however there are some cases where you may choose to break 3NF like we discuss in class. If your database is 3NF, describe how and why each table adheres to this. If your database is not 3NF, describe how it mostly adheres to 3NF but breaks it in a few specific ways.

[1]Website: `https://www.agilealliance.org/agile101/`

4 Further Notes, Checks, and Pitfalls

4.1 Tools

Although we are learning about MySQL and JDBC (associated with Java) in this class, you are free to use other relational database implementations, database connectors, and programming languages. Additionally, your team might choose to use some sort of code repository system (such as GitHub[2]) to share code.

GUIs tend to be a pain point for students, which is understandable. Websites are commonly developed by those who have some experience, but many groups choose to do desktop applications. If you are unsure, I recommend using MySQL as your database and set it up on my server, Java for the application language, and Java Swing for the GUI library.

Note that a console-based application is ok but will not earn you full points for the application. See the rubric on canvas for details.

4.2 Planning for the Semester

It can be daunting to tackle a project of this scope, especially with a partner (or two!). While you are free to organize completion of your project in any way you like (e.g., a different order than I suggest, waiting until the last week to do everything), below is a suggested ordering:

- Complete the project proposal document first! This will help you organize and scope the project. It is much easier to address issues or determine the project is not well suited for this class at this stage.
- While waiting for feedback on your document, (1) try setting up and testing the tools/technologies you plan on using, and (2) start looking for data sources for your project. For example, if you'll be using a repository manager like GitHub, make sure everyone has set up accounts and try sharing some code. If you are going to be doing a website, set up the tools you need for that. This will allow you to be productive independent from the specifics of the project. If you have a real data set to use, note that it may not have all the data you need or may need to be changed before it fits your design.
- Gathering data is very time consuming! Consider real sources, tools to generate data, and manual entry. Each has time consuming challenges. You may want to add a small amount of data first, but remember that you will need to have a good amount of data in the end for the application to be interesting.
- Designing your software and your user interface is important! Think about the CRUD interactions with the application and break them down into steps before trying to implement them. Sketch out the interfaces you will need in order to achieve the functionality outlined in your document. Maybe brush up on your GUI skills if they are a weak point, or find a tool to help you design a GUI.
- Consider how you might split up the work (e.g., interface design, database implementation) to play to people's strengths. How will you split up work such that you can work on components independently but still be able to combine them? How will you check back in weekly to make sure you're all still on track? When will you combine everything? You will want to meet at least weekly to discuss your progress and report your work for the check-ins. Consider different tools for collaboration such as email, git, zoom, or discord.

4.3 Group Work

Group work is an important part of working in industry. Some of the most valuable skills you will learn in pursuit of your degree are not graded and are not the focus of classes—communication (both within your team and with me through documents), professionalism, dividing and combining work, and so on. You should consider this project an opportunity to work on these skills. Our graduating seniors regularly report that they wish that had more opportunities for teamwork.

In the event that there is a problem within your team, I encourage you to find a time to talk to me, either during office hours or by setting up an appointment. Although grades earned for the project portion of this class should be the same across team members, I reserve the right to adjust grades for individual

[2]Website: `https://github.com/`

components of the project—up to and including the whole project—for individual members based on behavior and contribution. Note that this is reserved for extreme cases. While I do encourage you to bring problems to my attention, I may choose not to modify grades unless the offense is egregious.