Deliverable 4

Title: Final Project - Deliverable 4
Date: October 30th, 2024
Participating: Homawoo, Elorm. K., Garcia, Jose. C., Vanderhoff, Will. R.
User IDs: ehoma2, jgarc435, wvand,

1. Introduction to the Deliverable

Objective: Identify and outline the key elements and intentions of the semester project for CSC 472.

This deliverable aims to identify a semester project for CSC 472, highlight key elements of the project, the group's intentions, and the historical basis for the project. Below is an overview:

1. This group has selected to design a DBMS for a grading and feedback system, as outlined in Project 9.7 of the provided Course Project list.
2. The project will consist of two primary components:
   (a) The first component will satisfy the course requirements.
   (b) The second component will extend beyond the scope of the course to aid us in future job hunts by enhancing our resumes.
3. This group aims to submit all required deliverables and documentation in a timely manner throughout the semester. This includes, but is not limited to:
   (a) The overall and complete design of the database for a grading and feedback system, pursuant to Project 9.7 of the Course Project Selection List.
4. Additionally, some group members may choose to implement our design after all required submissions are met. This will be done to boost our portfolios, though we understand it will not impact our course grade.
5. We will use Discord for meetings and communication, and GitHub to track project iterations. This applies to both the mandatory submissions and any optional implementations.

2. Identify the Project

Objective: Explain what business or process that is to be addressed in the design of the Database Management System.

The project is a Course Performance Information Management System, designed to record and manage student performance data in various courses. Specifically, the system allows:

- Input and storage of marks for each student across multiple assignments, quizzes, or exams in a course.

- The ability to add new assessments (assignments/exams) without predefined limits.
- Calculation of a weighted sum of marks to derive total course marks.
- A grading system that allows for the specification of cutoff values for various letter grades (e.g., A, B, C, D, F).

3. Reason for Analysis

Objective: Explain the motivation for addressing the business situation, and why it was chosen.

This project addresses the need for a streamlined, flexible system in educational institutions to manage student performance data efficiently. Traditional methods of maintaining student marks, such as using spreadsheets or paper records, can become cumbersome, especially when dealing with a large number of students, assignments, and grade calculations.
Motivation:

- *Efficiency:* Automating the process of recording, updating, and calculating grades saves significant time for instructors, allowing them to focus on more critical tasks like teaching and student engagement.
- *Flexibility:* Instructors often need to add new assignments or assessments during the course. The system allows for dynamic addition of assignments/exams without predefined limits, ensuring scalability and flexibility.
- *Accuracy:* Manual calculation of weighted totals and final grades is prone to errors, especially in large classes. A system that automatically computes grades based on preset weights ensures accurate grade calculation and fair assessments.
- *Customization:* Every course might have different grading structures, weights for assignments, and grade cutoffs. This system allows instructors to set their specific course structure and grading criteria, accommodating diverse educational needs.

By addressing these issues, the system improves the academic workflow, reduces errors, and enhances both student and instructor experience.

4. Benefit of Analysis

Objective: Explain what is to be expected from improving the business process.

While an overnight success is not promised through the improvement of said business process, we guarantee that data will be easy to access, understandable to the layman, and thoroughly insightful. Here are a few benefits:

1. Whereas the reason, identification, and the overview of the project point to feedback systems, it is also important to highlight other contexts where such a project may exist. There are two forms of existence that this project can benefit:
   (a) The first is the improvement of an existing feedback/grading system with no DBMS.

   (b) The second is the upgrade to a dedicated feedback/grading system where no such digital system exists (e.g., paper copies are returned to employers/students as the main feedback).

2. Scenario 1: There exists a system with no adequate DBMS.
   (a) The main improvement here would be that of convenience and redundancy over the existing system. Imagine a professor tracks grades manually in a notebook. A proper system with a dedicated DBMS would offer:
      i. **Flexibility:** The professor can easily add another assignment/relation for each student without worrying about space.
      ii. **Redundancy/Security:** It would be much more difficult for a student to tamper with grades compared to simply stealing a grade notebook.
      iii. **Convenience:** Both the professor and student would have access to the grades at any time, making feedback and responses more streamlined.
      iv. **Efficiency:** The professor can provide feedback on past grades, enabling a more informed and positive learning environment.
3. Scenario 2: There exists no system and no DBMS.
   (a) For example, in performance reviews for employees, feedback is often on paper, making it easy to lose or misplace. Improvements would include:
      i. All the benefits from Scenario 1.
      ii. **More Data:** Having more data on performance and feedback can lead to better decision-making and tracking, providing valuable insights.

5. Project Overview

Objective: Outline the project steps.

In the following list, we attempt to show what the project creation timeline will be:

1. Brainstorm:
   Identify all the elements needed to allow the automation of the project.
2. Table Creation:
   Create tables through categorization and organization of elements decided upon in the brainstorming process.
3. Normalization:
   Identify and eliminate redundant data, verify data dependencies.
4. Schema Design:
   Identify the storage and space needs for the storage of the data model. Create a data model.
5. Project Review Report:
   Report on efforts, methodologies and educational benefits of the project.

This timeline is modeled after the deliverables required to complete the project. There may be additional steps added or taken at any point in time depending on the needs of the project and team.

6. Documents Needed

Objective: Explain the role and importance of documentation in the design and implementation of a DBMS.

Documentation is crucial not only for the initial success of the system but also for maintaining its long-term functionality. Below is an outline of the documentation that is both needed and already exists:

1. Documentation to be developed:
   (a) **Overview of DBMS Design:** This aligns with the 3rd to 5th deliverable, as it covers the assembly of project components.
   (b) **Reasoning Behind Design Choices:** For instance, why was Table A included but not Table C?
   (c) **Iterative Changes:** What modifications were made from Design K to Design P? What steps were taken, and was this an improvement?
   (d) **Project Timeline and Updates:** Keeping a project journal is beneficial, especially for projects built from scratch.
2. Documentation that already exists and is needed:
   (a) Course book.
   (b) Geek-for-Geeks articles.
   (c) Research papers on feedback to distinguish essential features from unnecessary components in the design.

7. Brainstorming

Objective: Identify the elements needed for the database.

To successfully design and implement a system that tracks course performance and allows for the addition of assignments or exams dynamically, we need to identify all the necessary components and features. Here's a comprehensive list of elements and considerations that will help build this system.

Entity Sets:

- **Instructor:** with attributes *(<u>ID</u>, name, email, department)*.
  - Relation: Relates and assigns domain over a course's administration.
  - Error Handling: Professors must have a login in order to administer feedback to assigned courses. A professor who can assign feedback must have a tuple

in Course. Additionally, professor must belong to a department that exists in the department table.

- **Student:** with attributes *(ID, name, email, GPA).*
  - Relation: Student will have assigned tuples in various grading relations (still being fleshed out for optimum design).
  - Error Handling: GPA is to be calculated from all student outcomes from assignments, exams, and overall reported grades in other relations such as Assignment, Exam, and Cutoff.
- **Course:** with attributes *(course_id, instructor_id, title, section, semester, year, enrollment).*
  - Relation: Course will hold instructor tuples for whom have access to course administration. Course will also hold possible courses that a student is/have taking/taken.
  - Error Handling: Students will have only grades from courses that exist in courses, and instructors may only preside over courses that a tuple exists for.
- **Assignment:** with attributes *(course_id, assignment_id, assignment_name, max_score, weight, due_date).*
  - Relation: Relates to course in which course an instructor has domain over to issue feedback. Grants access to assignments in a course.
  - Error Handling: Assignments must pertain to a course that exists in the course id.
- **Exam:** with attributes *(course_id, exam_id, exam_name, max_score, weight, due_date).*
  - Relation: One of possible 3 grading relations, still fleshing out based on attributes on other tables. Group consensus is this would hold examination grades.
  - Error Handling: Exam must belong to a course. Instructors and students would only have access to exams scores they are entitled to.
- **Cutoff:** with attributes *(course_id, instructor_id,grade_letter, min_score).*
  - Relation: An individual grading cutoff for each course, unique to each set by instructor.
  - Error Handling: Cutoff must belong to one course and instructor may only set if they have a tuple in the course relation.

Relationship Sets:

- **Takes:** with attributes *(student_id, course_id, semester, year, grade).*
- **Teaches:** with attributes *(instructor_id, course_id, semester, year).*

8a. Tables

Objective: Organize and categorize each element of information into groups that will become tables based on the content reference. Your submission will be a document including a list of each of your tables as well as each field in the table and the data type for each field. You will also identify keys (primary, foreign, etc).

Entity Sets:

- **Instructor:** with attributes *(ID, name, email, department, phone, role).*
  - Relation: Relates to the courses being taught and administered by each instructor.
  - Error Handling: Professors must have a login in order to administer feedback to assigned courses. A professor who can assign feedback must have a tuple in *Course*. The professor must also belong to a department that exists in the *Department* table.
- **Student:** with attributes *(ID, name, email, GPA, department, year, enrollment_date, graduation_date, phone, address).*
  - Relation: Students will have assigned tuples in various grading relations (still being fleshed out for optimum design).
  - Error Handling: GPA is to be calculated from all student outcomes from assignments, exams, and overall reported grades in other relations such as *Assignment*, *Exam*, and *Cutoff.*
- **Course:** with attributes *(course_id, instructor_id, title, section, semester, year, enrollment, credits, schedule, prerequisites, course_description).*
  - Relation: Course will hold instructor tuples for those who have access to course administration. Course will also hold possible courses that a student is/has taking/taken.
  - Error Handling: Students will only have grades from courses that exist in *Course*, and instructors may only preside over courses that have a tuple.
- **Assignment:** with attributes *(course_id, assignment_id, assignment_name, max_score, weight, due_date, description, submission_type, submission_status, grading_rubric).*
  - Relation: Relates to the course where the instructor has domain over issuing feedback. Grants access to assignments in a course.
  - Error Handling: Assignments must pertain to a course that exists in the *Course* table.
- **Exam:** with attributes *(course_id, exam_id, exam_name, max_score, weight, due_date, exam_type, duration, location, open_closed_book).*
  - Relation: One of the possible grading relations. This table will hold examination grades.

- Error Handling: Exams must belong to a course. Instructors and students will only have access to the exams they are entitled to.

- **Cutoff:** with attributes *(course_id, instructor_id, grade_letter, min_score, max_score, grading_scheme_version).*
  - Relation: Each course will have a unique grading cutoff set by the instructor.
  - Error Handling: Cutoff must belong to a course, and instructors may only set if they have a tuple in the *Course* relation.

- **Feedback:** with attributes *(feedback_id, student_id, assignment_id, comments, submission_date).*
  - Relation: Links feedback given by instructors to specific assignments for each student.
  - Error Handling: Feedback must relate to an existing student and assignment.

Relationship Sets:

- **Takes:** with attributes *(student_id, course_id, semester, year, grade, completion_status, attendance_record).*
  - Relation: Links students to courses they've taken in a given semester and year. Tracks final grades and optional attendance data.
  - Error Handling: Students can only be linked to existing courses.

- **Teaches:** with attributes *(instructor_id, course_id, semester, year, primary_instructor, office_hours).*
  - Relation: Links instructors to the courses they are teaching in a given semester and year.
  - Error Handling: Instructors can only be assigned to teach courses that exist in the *Course* table.

Additional Tables:

- **Audit Logs:** with attributes *(log_id, action_performed, performed_by, timestamp).*
  - Relation: Tracks actions taken in the system for auditing purposes (e.g., grade changes, assignment creation).
  - Error Handling: Each action must be linked to a valid system user (instructor/student/admin).

8b. Organizing and Categorizing Elements

The goal of this section is to organize and categorize each element of information into groups that will become tables. Each table will represent an entity or a relationship in the system, structured according to the content reference of our brainstorming session.

Entity Tables

- **Instructor Table**
  - Attributes:
    - *ID* (Primary Key): CHAR(9) - A unique identifier for each instructor.
    - *Name*: VARCHAR(255) - The name of the instructor.
    - *Email*: VARCHAR(255) - The instructor's email address.
    - *Department*: VARCHAR(20) - The department the instructor belongs to.
    - *Phone*: VARCHAR(15) - The contact number of the instructor.
    - *Role*: VARCHAR(20) - The role of the instructor (e.g., professor, teaching assistant).
- **Student Table**
  - Attributes:
    - *ID* (Primary Key): CHAR(9) - A unique identifier for each student.
    - *Name*: VARCHAR(255) - The student's name.
    - *Email*: VARCHAR(255) - The student's email address.
    - *GPA*: DECIMAL(1,2) The student's current GPA, calculated from all courses.
    - *Department*: VARCHAR(20) - The student's field of study.
    - *Year*: YEAR - The current year of study (e.g., freshman, sophomore).
    - *Enrollment_Date*: DATE - The date the student was enrolled.
    - *Graduation_Date*: DATE - The expected date of graduation.
    - *Phone*: VARCHAR(15) - The contact number of the student.
    - *Address*: VARCHAR(255) - The student's home or campus address.

- **Course Table**
  - Attributes:
    - *Course_ID* (Primary Key): VARCHAR(10) - A unique identifier for each course.
    - *Instructor_ID* (Foreign Key): CHAR(9) - A reference to the instructor teaching the course.
    - *Title*: VARCHAR(255) - The name of the course.
    - *Section*: VARCHAR(10) - The specific section number for the course.
    - *Semester*: VARCHAR(20) - The semester in which the course is offered.
    - *Year*: YEAR - The year in which the course is offered.
    - *Enrollment*: INT(10) - The number of students enrolled.
    - *Credits*: DECIMAL(5, 0) - The number of credits the course carries.
    - *Prerequisites*: VARCHAR(10) - Any prerequisite courses needed for enrollment.
    - *Course_Description*: VARCHAR(255) - A short description of the course content.
- **Assignment Table**
  - Attributes:
    - *Course_ID* (Foreign Key): CHAR(9) - A reference to the course to which the assignment belongs.
    - *Assignment_ID* (Primary Key): CHAR(9) - A unique identifier for each assignment.
    - *Assignment_Name*: VARCHAR(255) - The name of the assignment.
    - *Max_Score*: INT(10) - The maximum score possible for the assignment.
    - *Weight*: DECIMAL(5, 0) - The weight of the assignment in the final grade.
    - *Due_Date*: DATE - The due date for the assignment.
    - *Description*: VARCHAR(255) - A brief description of the assignment.
    - *Submission_Type*: VARCHAR(20) - The type of submission (online, physical, etc.).
    - *Submission_Status*: VARCHAR(20) - The current status of the submission (submitted, pending, etc.).
    - *Grading_Rubric*: VARCHAR(255) - The rubric used for grading the assignment.

- **Exam Table**
  - Attributes:
    - *Exam_ID* (Primary Key): CHAR(15) - A unique identifier for each exam.
    - *Course_ID* (Foreign Key): CHAR(10) - A reference to the course to which the exam belongs.
    - *Exam_Name*: VARCHAR(20) - The name of the exam.
    - *Max_Score*: DECIMAL(10,2) - The maximum score possible for the exam.
    - *Weight*: DECIMAL(10, 2) - The weight of the exam in the final grade.
    - *Due_Date*: DATE - The due date for the exam.
    - *Exam_Type*: VARCHAR(20) - The type of exam (midterm, final, quiz).
    - *Open_Closed_Book*: VARCHAR(20) - Indicates whether the exam is open or closed book.
- **Cutoff Table**
  - Attributes:
    - *Instructor_ID* (Foreign Key): CHAR(9) - A reference to the instructor.
    - *Course_ID* (Foreign Key): VARCHAR(10) - A reference to the course.
    - *Grade_Letter*: CHAR(2) - The letter grade (A, B, C, etc.).
    - *Min_Score*: DECIMAL(10, 2) - The minimum score required to achieve the grade.
    - *Max_Score*: DECIMAL(10, 2) - The maximum score for the grade range.
- **Feedback Table**
  - Attributes:
    - *Feedback_ID* (Primary Key): CHAR(9) - A unique identifier for each feedback entry.
    - *Student_ID* (Foreign Key): CHAR(9) - A reference to the student receiving the feedback.
    - *Assignment_ID* (Foreign Key): CHAR(9) - A reference to the assignment for which feedback is given.
    - *Instructor_ID* (Foreign Key): CHAR(9) - A reference to the instructor giving the feedback.
    - *Comments*: VARCHAR(256) - Textual feedback or comments provided by the instructor.
    - *Date*: DATE - The date on which the feedback was given.

Relationship Tables

- **Takes Table**
  - Attributes:
    - *Student_ID* (Foreign Key): CHAR(9) - A reference to the student taking the course.
    - *Course_ID* (Foreign Key): CHAR(9) - A reference to the course.
    - *Semester*: VARCHAR(20) - The semester when the course was taken.
    - *Year*: YEAR - The year when the course was taken.
    - *Grade*: DECIMAL(256,2) - The final grade received.
    - *Completion_Status*: VARCHAR(256) - Status of course completion (completed, withdrawn, etc.).
- **Teaches Table**
  - Attributes:
    - *Instructor_ID* (Foreign Key): CHAR(9) - A reference to the instructor teaching the course.
    - *Course_ID* (Foreign Key): CHAR(9) - A reference to the course being taught.
    - *Semester*: VARCHAR(20) - The semester when the course is taught.
    - *Year*: YEAR - The year when the course is taught.
    - *Primary_Instructor*: BOOL - Boolean to indicate whether this is the primary instructor for the course.

Additional Tables

- **Course Evaluation Table**
  - Attributes:
    - *Evaluation_ID* (Primary Key): CHAR(9) - A unique identifier for each course evaluation.
    - *Student_ID* (Foreign Key): CHAR(9) - A reference to the student submitting the evaluation.
    - *Course_ID* (Foreign Key): CHAR(9) - A reference to the course being evaluated.
    - *Instructor_ID* (Foreign Key): CHAR(9) - A reference to the instructor being evaluated.
    - *Rating*: INT - A numerical rating given by the student.
    - *Comments*: VARCHAR(256) - Additional comments or feedback provided by the student.

∗ *Date*: DATE - The date the evaluation was submitted.

Section 9: Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. This section outlines the normalization steps taken for each table created in Section 8.

1. First Normal Form (1NF)

Table Name	**1NF Compliance**	**Proposed Changes**	**Reason**
Instructor Table	Yes	None	All attributes contain only single values.
Student Table	Yes	None	All attributes contain only single values.
Course Table	Yes	None	All attributes contain only single values.
Assignment Table	Yes	None	All attributes contain only single values.
Exam Table	Yes	None	All attributes contain only single values.
Cutoff Table	Yes	None	All attributes contain only single values.
Feedback Table	Yes	None	All attributes contain only single values.
Takes Table	Yes	None	All attributes contain only single values.
Teaches Table	Yes	None	All attributes contain only single values.

Table 1: 1NF Compliance Check for Tables

2. Second Normal Form (2NF)

Table Name	**2NF Compliance**	**Proposed Changes**	**Reason**
Instructor Table	Yes	None	All non-key attributes depend on the primary key.
Student Table	Yes	None	All non-key attributes depend on the primary key.
Course Table	Yes	None	All non-key attributes depend on the primary key.
Assignment Table	Yes	None	All non-key attributes depend on the primary key.
Exam Table	Yes	None	All non-key attributes depend on the primary key.
Cutoff Table	Yes	None	All non-key attributes depend on the primary key.
Feedback Table	Yes	None	All non-key attributes depend on the primary key.
Takes Table	Yes	None	All non-key attributes depend on the primary key.
Teaches Table	Yes	None	All non-key attributes depend on the primary key.

Table 2: 2NF Compliance Check for Tables

3. Third Normal Form (3NF)

Table Name	**3NF Compliance**	**Proposed Changes**	**Reason**
Instructor Table	Yes	Change Department to Department_ID (Foreign Key)	Remove transitive dependency on Department Name.
Student Table	Yes	Change Department to Department_ID (Foreign Key)	Remove transitive dependency on Department Name.
Course Table	Yes	None	No transitive dependencies exist.
Assignment Table	Yes	None	No transitive dependencies exist.
Exam Table	Yes	None	No transitive dependencies exist.
Cutoff Table	Yes	None	No transitive dependencies exist.
Feedback Table	Yes	None	No transitive dependencies exist.
Takes Table	Yes	None	No transitive dependencies exist.
Teaches Table	Yes	None	No transitive dependencies exist.
Department Table	No	Create a Department Table	Normalize to eliminate redundancy.

Table 3: 3NF Compliance Check for Tables

4. Boyce-Codd Normal Form (BCNF)

Table Name	**BCNF Compliance**	**Proposed Changes**	**Reason**
Instructor Table	Yes	None	All functional dependencies comply.
Student Table	Yes	None	All functional dependencies comply.
Course Table	No	Adjust functional dependencies (semester, year, instructor)	Non-superkey attributes determine another attribute.
Assignment Table	Yes	None	All functional dependencies comply.
Exam Table	Yes	None	All functional dependencies comply.
Cutoff Table	Yes	None	All functional dependencies comply.
Feedback Table	Yes	None	All functional dependencies comply.
Takes Table	Yes	None	All functional dependencies comply.
Teaches Table	Yes	None	All functional dependencies comply.
Department Table	Yes	None	All functional dependencies comply.

Table 4: BCNF Compliance Check for Tables

5. Final Product after Normalization (after Conclusion due to Latex)

Conclusion

There were a few instances of overlooking while we were building out our tables in the first few weeks of the project. This is evident, for example, in the exclusion of a dedicated Department table that is present in the book. There were minor tweaks to our tables, however, for the most part - our baked-in normalization process when the group was developing tables stood strong.

Table Name	**Attributes (with Changes)**	**Changes During Normalization**
Instructor Table	ID (Primary Key), Name, Email, Department_ID (Foreign Key), Phone, Role	Changed Department to Department_ID (Foreign Key).
Student Table	ID (Primary Key), Name, Email, GPA, Department_ID (Foreign Key), Year, Enrollment Date, Graduation Date, Phone, Address	Changed Department to Department_ID (Foreign Key).
Course Table	Course ID (Primary Key), Instructor ID (Foreign Key), Title, Section, Semester, Year, Enrollment, Credits, Prerequisites, Course Description	None.
Assignment Table	Assignment ID (Primary Key), Course ID (Foreign Key), Assignment Name, Max Score, Weight, Due Date, Description, Submission Type, Submission Status, Grading Rubric	None.
Exam Table	Exam ID (Primary Key), Course ID (Foreign Key), Exam Name, Max Score, Weight, Due Date, Exam Type, Open Closed Book	None.
Cutoff Table	Instructor ID (Foreign Key), Course ID (Foreign Key), Grade Letter (Primary Key), Min Score, Max Score	None.
Feedback Table	Feedback ID (Primary Key), Student ID (Foreign Key), Assignment ID (Foreign Key), Instructor ID (Foreign Key), Comments, Date	None.
Takes Table	Student ID (Foreign Key), Course ID (Foreign Key), Semester, Year, Grade, Completion Status	None.
Teaches Table	Instructor ID (Foreign Key), Course ID (Foreign Key), Semester, Year, Primary Instructor	None.
Department Table	Department ID (Primary Key), Department Name, Department Head	Created new Department Table to eliminate redundancy.

Table 5: Final Structure After Normalization