UNIVERSITY OF SOUTH CAROLINA BEAUFORT

CSCI/ISAT B320
Database Management Systems I
Fall 2025

Project Documentation

Team 09

Table of Contents

Project Contributions by 09

Purpose:
Document the contributions of each team member over the course of the project.

Members and their Contact Information

Member	Email	Text
Nicholas Vickery	nvickery@email.uscb.edu	(864) 985-3553
Connor Floyd	cwfloyd@email.uscb.edu	(843) 816-3643

Overall
Relative Contribution of each member over the course of the entire project

Member	Contribution	Total Hours
Nicholas Vickery	50%	22
Connor Floyd	50%	36

Data Design (i.e., ERD Creation & Revisions)

Member	Contribution	Hours	Components
Nicholas Vickery	50%	3	Attribute Identification, 3NF, ERD design, continual ERD revisions.
Connor Floyd	50%	3	Attribute Identification, 3NF, ERD design.

Create & Populate Script: Entity Creation (i.e., Table, View, Constraint, etc.)

Member	Contribution	Hours	Components
Nicholas Vickery	50%	5	Creation of multiple tables ranging from professor to gpa history, and editing the tables
Connor Floyd	50%	2	Creation of multiple tables ranging from professor to gpa history, and editing the tables

Create & Populate Script: Entity Population (i.e., Table Inserts)

Member	Contribution	Hours	Components
Nicholas Vickery	50%	10	Created inserts for professors, subjects, student status, campus, buildings, rooms, enrollments, gpahistory, gpapoints, advisors, and term
Connor Floyd	50%	26	Created Inserts for students, courseschedule, and courses

Query Script: Query Development
Note: include here any Views created to support your queries

Member	Contribution	Hours	Components
Nicholas Vickery	50%	2	Created Queries for 1-5
Connor Floyd	50%	2	Created Queries for 6-10

Presentation Preparation

Member	Contribution	Hours	Components
Nicholas Vickery	25%	1	Ensured that database works and was presentable
Connor Floyd	75%	2	Creation of slides PowerPoint

Project Documentation & Administration

Member	Contribution	Hours	Components
Nicholas Vickery	50%	1	Updated Project Documentation
Connor Floyd	50%	1	Comments through out code and Presentation Timeline

Collaboration Tools Employed

- Github: For collaboration and version control
- LucidChart: ERD
- Visual Code: For script creation
- SSMS and SQL Developer for database

Naming Conventions

1. **Tables:**
   - Tables are named using PascalCase to represent entities (e.g., Students, Professors, Courses).
   - Singular naming is used to represent a single entity (e.g., CourseSchedule instead of CourseSchedules).
2. **Fields/Columns:**
   - Column names use camel case format (e.g., firstName, lastName, courseNumber).
   - Foreign key columns follow the naming pattern <ReferencedTableName>ID (e.g., ProfessorID, SubjectID).
   - Descriptive column names that avoid abbreviations for readability.
3. **Primary Key Constraints:**
   - Primary key fields use the table name followed by ID (e.g., StudentID, CourseID).
   - Primary key constraints are named with the prefix PK_ followed by the table name (e.g., PK_Students).

4. **Foreign Key Constraints:**
   - Foreign key constraints are named with the prefix FK_ followed by <ReferencingTable>_<ReferencedTable> (e.g., FK_CourseSchedule_Courses, FK_Enrollments_Students).
5. **Naming of Lookup and Reference Tables:**
   - Lookup/reference tables use a descriptive name (e.g., CourseType, GradePoints) indicating their purpose.
6. **Conventions for Specific Tables:**
   - CourseSchedule uses CourseTypeID to reference CourseType instead of a descriptive field for meeting type.
   - GPA-related data, when present, uses tables like GPAHistory and GradePoints for clarity and standardization.

Design Assumptions and Data Clarifications

1. Name Changes for Students and Professors:

- **Students**: The design assumes that students may change their names without losing historical records (such as enrollment history). The Students table directly updates the FirstName and LastName fields to reflect the name change. The StudentID remains consistent as the primary identifier, ensuring continuity of records.
- **Professors**: Similarly, if a professor changes their name, their ProfessorID remains the primary identifier, while the ProfessorName field can be updated without affecting historical course assignments or records.

2. Multiple Professors Assigned to a Course (Co-Teaching):

- The design addresses co-teaching by introducing a **ProfessorAssignments** table to create a many-to-many relationship between **Professors** and **CourseSchedule**. Each professor assigned to a course section has a unique entry in this join table, allowing multiple professors to be associated with a single course section.

3. Course Location or Professor Change Mid-Semester:

- **Location Changes**: Since each course section in the **CourseSchedule** table is associated with a RoomID, changing a course's location requires updating the RoomID for that specific schedule entry. This update keeps the record intact while accurately reflecting the new location.
- **Professor Changes**: For changing professors mid-semester, the **ProfessorAssignments** table is used, allowing professors to be added or removed as needed without altering historical data. This flexibility also supports cases where multiple professors might take turns covering sections throughout the semester.

Assumptions and Clarifications on Data from the Registrar and Mock Data

1. Registrar Data Assumptions:

- **Unique Identifiers**: The registrar data assumes that fields like CRN (Course Registration Number) uniquely identify each course section. Likewise, each professor, student, and room has a unique identifier in the design.
- **Consistency in Data Entries**: Assumptions were made regarding the consistency of data entries, such as standardized abbreviations for buildings and departments, which are essential for joining data accurately.
- **Room Assignments**: The design assumes that room assignments are valid only for the associated term. Historical room data is maintained by each specific **CourseSchedule** entry without overwriting records.

2. Mock Data Clarifications:

- **Standardized Names**: For simplicity, mock data was created with standardized formats (e.g., "Dr. Firstname Lastname" for professors). This ensures clarity and reduces potential mismatches during mock queries.
- **Building and Room Codes**: Unique codes and names were assumed for buildings and rooms to maintain clarity, and BuildingCode was set as a foreign key in **Rooms** to avoid redundancy.

Issues and Solutions in Analyzing Registrar's Data

1. Repetitive or Ambiguous Data Entries:

- **Resolution**: During analysis, repetitive subject or professor names required deduplication. The **Subjects** and **Professors** tables were structured to hold unique entries, preventing duplicate entries in related tables.

2. Handling Room and Building Relationships:

- **Resolution**: Initially, BuildingName was directly included in the **Rooms** table. This was revised by creating a separate **Buildings** table, where BuildingCode serves as the primary identifier linked to **Rooms** as a foreign key. This adjustment preserved normalization and reduced redundancy.

3. Complex Relationships (e.g., Co-Teaching and Room Reassignments):

- **Resolution**: Complex relationships like co-teaching and room reassignments were managed by introducing **ProfessorAssignments** and referencing BuildingCode in **Rooms**. This approach provided flexibility in representing dynamic scenarios while keeping historical data intact.

4. Missing or Inconsistent Location Data:

- **Resolution**: For cases where room details were inconsistent or missing, the design ensures that each RoomID is linked to a building with a BuildingCode in **Buildings**. Mock data was created with placeholder values where necessary, assuming standardized room codes to complete the ERD.

Bibliography

Citations for:

Naming conventions:
https://www.scholarhat.com/tutorial/sqlserver/sql-server-naming-conventions-and-standards

Reinforced learning for python script used mainly for refresher (File handling and other):
https://www.w3schools.com/python/python_file_write.asp

Acknowledgements

Specific people who helped you debug:
Dr. Erdei for modification ideas