CMSC 461 Project 1: Project Report

Phase A Report

This project requires a database to be built that can manage and edit information about certain entities (Trainees, Mentors, Certificates, Challenges, and Skills), can be populated with instances of the required entities, and be able to fulfil query requests about information stored in the database. This will require implementing database design techniques such as specialization and mapping cardinalities to fulfil specific requests of the database, such as limitations on the number of evaluators per challenge and recording the dates of counseling periods and challenges. Entities with appropriate attributes and relationships between those entities will need to be created and implemented in this database. The conceptual design will need to be converted to a relational data model (while continuing to meet use requirements), then a MySQL database designed based on the conceptual design. Python and SQL code will be written to both update the database and create a user interface to interact with the database. Finally, a user's guide will need to be created to instruct users on proper usage of the database.

Phase B Report

An initial E-R Diagram was constructed to lay the groundwork for the CertRUs database design. The diagram outlines the planned entities and relations, with the primary key for each entity underlined and simple attributes listed beneath the title of each entity. Some entities required different types of attributes – the contact information for each user required a composite attribute for the address, and multivalued attributes for phone numbers and emails.

Relation and entity constraints are identified in the E-R diagram using mapping cardinality, directed and undirected lines, and associated minimum and maximum cardinalities. The document implies a trainee may only undergo one counseling session per period at a time, and that counseling session has one counselor, so the ER model reflects this constraint. It also reflects the constraint that 3 mentors exactly are needed for each instance of a challenge.

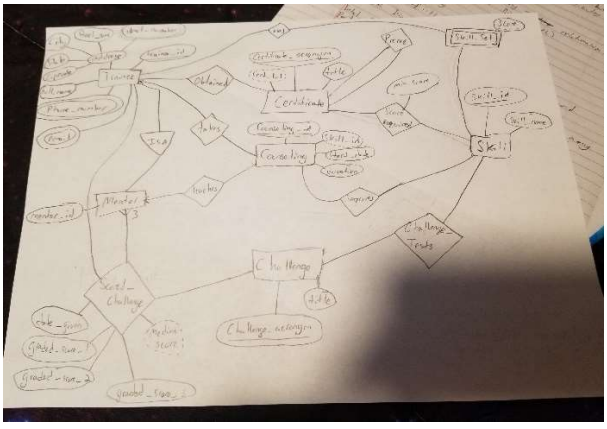
This design uses a few instances of specialization – carrying over attributes helps to avoid redundancy in the overall design, especially in the case of a challenge instance for a trainee and the general entity of a challenge itself. Certificate levels (beginner, intermediate, and advanced) are differentiated with an attribute rather than specialization of the Certificate entity for simplicity, but this may change through development as needed.

Phase C Report

The initial E-R Diagram in Phase B was altered based on feedback received for Milestone 1, then converted into a Relational Data Model. The removal of Skill and Challenge specializations were especially helpful in streamlining the conversion and safeguarding against future data loss. The exact changes made are listed below:

- Changed many unique primary key names to be more specific to their respective tables(many were “unique_id”)
- Changed contact info to a composite attribute
- Removed specialization of skill prowess from skill, retained only one entity (converted into a relation for the relational model) “Skill”.
- Created a weak entity “Skill Set” depending on “Trainee”
- Removed specialization of challenge instance from challenge, retained only one entity (converted into a relation for the relational model) “Challenge”
- Added a relationship “Scored Challenge” to track when challenges were given by mentors/taken by trainees
- Removed “evaluator score”, added attributes for teacher and graded scores as needed to “Scored Challenge” to record evaluated scores and who gave them for each instance of a challenge.
- Added a derived attribute for the median score (this does not actually show up in the relational model, though)
- Added a relation to track certifications earned by trainees
- Converted “cert_lvl” into a derived attribute (basic vs. intermediate vs. advanced) based on prerequisites required
- Changed “Counseling” to include the start date and duration in months instead of start date and end date, to make it easier to calculate future queries regarding counseling lengths

Updated E-R Model (larger picture attached separately):



Phase D Report

This phase saw the preliminary construction of the physical MySQL database using an updated version of the design from section B and the resulting relational data model from section C. Scripts were created and maintained for basic creation and deletion of tables, though deletion of tables will fail if the primary key of the table is a reference key elsewhere in the schema – this is intentional at this time to prevent accidentally breaking the database by deleting important tables. Scripts were created and used to load preliminary data to test the basic and primary storage of the current database – specific functions and tidying up constraints and input limitations are underway.

Phase E Report

This phase saw the implementation of data entries and MySQL queries into the database built in Phase D. No changes to physical database construction were made during this phase. The user interface chosen for the database was a simple text-input/output application run on Jupyter Notebooks – this was chosen for ease of coding and accessibility, and the interface itself can support and satisfying all listed functional user requirements.

Most of the code required to satisfy the function user requirements are present, though some are missing. The ability to edit tuples as well as a few functional queries are missing from both the Jupyter Notebook Python code as well as in the submitted SQL scripts, but all other functionality is present.

While most of the code is present, there are numerous places this database could improve. The code currently requires the user to manually update any changes to a trainees score, and this is not sustainable in the long term – a helper function should be added in the future to automatically check a trainees score and update it with the median of the best three scores assigned to it by different evaluators. Also, many of the user-entered fields are not checked for errors and require accuracy on the end of the user – while the database will catch most of them and throw errors when trying to submit them to the database, adding a mentor will not force the user to enter a negative value and nor will the database throw an error, which could throw off the entire database. There are also no checks to make sure information is consistent with other information in the database – a mentor could be added when they are not qualified to be one. Finally, there are no catch systems in place for python-side errors – an inappropriate entry that throws off the code will currently crash the entire application and require a reboot. These are all fixes and improvements that are reasonable to implement in the future.

Phase F Report

Data was entered without issue to populate the physical database from Phase D based on the Functional Requirements given in the project document. A simple text users guide was created and implemented within the Jupyter Notebook file to explain the text interactive application and includes information about the databases current limitations.