Rushi Samirbhai Patel RS397441@dal.ca
B00886157

Final Project

Genealogy Media Integration System

Milestone 3

External Documentation:

Overview:
The development process for Genealogy Media Integration System (GMIS) requires data structures, code design and key algorithms which are discussed in this document:

- **Database Connection process and Design pattern:**

  The most important and crucial aspect of this project is the schema developed for the entire database system which would be utilized in this project. While connecting to the database, I have connected to my local server where I have created a schema 'finalproject'. Here I have created the tables FamilyTree and MediaArchive used as our primary databases.

  My code follows a singleton design pattern, i.e., only one instance of connection will be created. This will ease my database operations and will not hamper my workbench connection to the localhost. I have made a database util class where I have created separate functions to read or update data from the database. The code would be more modular and easy to access while running sql queries in this manner. The methods can take any sql query as an input and execute the query in a more generalized way.

  I am following interface implementation to achieve abstraction in my code. Since an individual can have multiple notes, multiple references, and multiple children, I have also created tables for the same which would map notes, references and children (in separate tables) for an individual.

  I am also focusing on using getters and setters to store the information of an individual and media in their particular objects. This can help in fetching and operating on the data present in the system. This also keeps the code protected and encapsulated.

Rushi Samirbhai Patel
B00886157
RS397441@dal.ca

- **Data Structures and Key Algorithm:**

  I will be using List and Map as abstract data types along with ArrayList and Hashmap as data structures to achieve the goal. I would be using ArrayList to store the objects of people added in the system and to store objects of the media added to the system. Hashmaps are required to store and update the database with the help of key value pairs (key being the attribute name and value being the attribute's value that need to be stored). Since the individual name and the filename are two mandatory elements in both our tables, I have kept both these fields as not null. This would make sure that these fields are mandatory and no other attributes or columns can have values unless the mandatory attributes are present in the tables. I have also used a column id in my table which would make each row distinct and in case of duplicates, I can use the id's to fetch the unique element/individual/media for my operations.

  A tree would also be required to better organize the family structure and operate on the particular data of an individual. The key algorithm that I would follow would be to store the information in a database as well as in a tree (to calculate degree of removal, cousinship, ancestors and descendents).

  I am hoping to develop a schema in such a way that by using sql queries I am able to operate and report back relations. This can be achieved by joining one or more tables and fetching the desired data.
  With the help of DFS, I would traverse the tree to find out the required data for the remaining reporting functions.

  While recording data - adding a person, attributes, references, notes, children, partnering, dissolution, I will be using insert/update queries to store the information recorded in the database tables (for Family management) which could be used in the future sessions as well for the entire system. Likewise, to record media files, media attributes, people in media, and tags in media, I will be using insert/update queries to store this information in the database tables (for Media Archive management) which can also be used in future sessions for the entire system.

  To report the data back to the user (genealogists), I will have to retrieve values from different tables. This can be achieved by using different types of join on tables that are required for a particular reporting function. Once the data is

fetched in a resultset, I can display the result using logic specific to the reporting functions.

Additional White Box Tests:

- Database connection must be established before firing queries from the code.
- Tables which are used to fetch and update data for GMIS must be created before firing queries from the code.
- To dissolve a relationship, it should be partnered first.
- Attributes for an individual are passed as a string in the key of hashmap which does not have the same name as the database column name of that particular attribute.
- Attributes for an media are passed as a string in the key of hashmap which does not have the same name as the database column name of that particular attribute.
- Date is passed as a string having a different format which is not compatible with the date data type in our database.

Additional Boundary Cases:

- An Individual having multiple partners.
- While recording attributes for a person if the person is not added to the system.
- While recording reference for a person if the person is not added to the system.
- While recording a note for a person if the person is not added to the system.
- While recording a child for a person if the person or the child is not added to the system.
- While recording partnering if both the partners are not added to the system.
- While recording dissolution if both the partners are not added to the system.
- While recording dissolution if both partners had not recorded partnering relation.