High-level breakdown analysis

Inputs to the program:

Operation	Context/Entity	Inputs
Create/Update	Person	Name
		Date of birth
		Gender
		Occupation
		References
		Notes
	Adding relationships	Person 1(parent)
		Person 2(child)
	Partnering/Dissolution	Person 1(partner1)
		Person 2(partner2)
	Media	File location
		Year
		Date
		City
	Associating media with individuals	Media identifier
		List of individuals
	Tagging media	Media identifier
		Tag
Retrieval	Individual	Person name
	Media	Media name
	Individual name	personID
	Media file	fileID
	Relation	Person 1
		Person 2
	Descendants	Person
		Generations
	Ancestors	Person
		Generations
	Notes and References	Person
	Media	Tag
		Start date
		End date
	Media	Location
		Start date
		End date
	Individuals media	People
		Start date
		End date
	Biological family media	Person

Transformations to the data:

1. Date of birth can be converted from a string to the Date data type.
2. Gender can be converted from a string to the character data type.

Data processed right away:

1. Answer as soon as the input is given for the retrieval.

Data to keep longer:

Following entities could be stored in a database:

1. Person(person_id, name)
2. PersonAttribute(person_id, date_of_birth, gender, occupation)
3. Note(person_id, note)
4. Reference(person_id, reference)
5. Partner(person_id_1, person_id_2)
6. ParentChild(parent_id, child_id)
7. Media(media_id, fileLocation)
8. MediaAttribute(media_id, year, date, city)
9. PeopleMedia(media_id, person_id)
10. Tag(media_id, tag)

Output of the program:

1. Person's information
2. Media information(Tag/Location)
3. Notes and references of an individual
4. Ancestors/Descendants of an individual
5. Relationship between two individuals
6. Media associated with an individual's immediate children

Assumptions:

1. Only biological relations can exist in the family tree.
2. When a partner relationship exists and a child gets recorded for one of the partners, then the second partner also records this automatically due to symmetry.
3. The symmetry between partners gets broken after the dissolution. i.e child recorded for one of the partners does not automatically record it for the previous partner.
4. An individual cannot be added again at a different location in the family tree.

Constraints:

1. Biological family relations
2. English genealogy perspective

Strange cases:

1. When an individual is disconnected from the main tree
2. The case when dissolution happens and a child gets recorded to one of the partners

Approach:

1. Create two databases
   a. FamilyTree: Storing family relationships, Individual's information
   b. MediaManagement: Storing media metadata, tags
2. Create tables in the FamilyTree database:
   a. Person(person_id, name)
   b. PersonAttribute(person_id, date_of_birth, gender, occupation)
   c. Note(person_id, note)
   d. Reference(person_id, reference)
   e. Partner(person_id_1, person_id_2)
   f. ParentChild(parent_id, child_id)
3. Create tables in the MediaManagement database:
   a. Media(media_id, fileLocation)
   b. MediaAttribute(media_id, year, date, city)
   c. PeopleMedia(media_id, person_id)
   d. Tag(media_id, tag)
4. Create a tree data structure by querying the database to build a family tree.
5. Maintain individuals' information in a map for quick access.
6. Navigate the tree to do different operations such as finding the lowest common ancestor, finding ancestors/descendants.

Feature Development Plan

1. Create conceptual, logical and physical models for storing the family tree information and media entities.

Tasks	Entities	Tools	Planned date of completion
Design conceptual/logical model	person, person_attribute, note, reference, partner,	Draw.io	12/11/2021

Design physical model: Define tables and column constraints	parent_child, media, media_attribute, people_media, tag	MySQL Workbench	14/11/2021
Design physical model: Develop ERD, Normalize using iterative approach, Forward engineer the model			17/11/2021
Integrate Genealogy database with the java code	NA	Java, Eclipse, MySQL connector	19/11/2021

2. Work on the java classes.

Tasks	Planned date of completion
1. Create interfaces called FamilyTree, MediaManagement, and Reporting. 2. Create a class called Geneaology implementing above interfaces. 3. Implement FileIdentifier class. 4. Develop SQL queries for adding media, recording media attributes, and tagging media. 5. Implement methods for storing media information in the database: • addMediaFile • recordMediaAttributes • tagMedia	21/11/2021
1. Implement PersonIdentity class. 2. Develop SQL queries for adding person, recording person attribute, recording reference and recording note. 3. Implement methods for storing an individual's information in the database: • addPerson • recordAttributes • recordReference • recordNote 4. Implement the peopleInMedia method.	23/11/2021
1. Implement data structure for storing family tree hierarchy 2. Develop SQL queries for recording child, recording partner, and recording dissolution. 3. Implements methods for defining relationships in the database and in the memory: • recordChild	28/11/2021

• recordPartnering • recordDissolution 4. Implement methods to retrieve information from the database to build a family tree hierarchy.	
1. Implement methods to find the most recent ancestor in the family tree 2. Develop SQL queries for finding a person, finding a media file, finding name, and finding a media file. 3. Implement reporting functions: • findPerson • findMediaFile • findName • findMediaFile • findMediaByTag • findMediaByLocation • notesAndReferences	02/12/2021
1. Implement reporting functions • findRelation • descendents • ancestors • findIndividualsMedia • findBiologicalFamilyMedia	09/12/2021

Data Structures

A) PersonIdentity:
The PersonIdentity object stores the following relations:
1. parents: Stores the parent/parents related to the person in a list(List<PersonIdentity>)
2. partner: Stores the partner(PersonIdentity)
3. children: Stores all the children in a list(List<PersonIdentity>)
4. rootAncestors: Stores root ancestors of the person in a set(Set<PersonIdentity>)

B) rootAncestors:
Root ancestors represent the root in the Genealogy tree. There can be multiple root ancestors in the tree. Each individual in the tree stores the corresponding root ancestors.

C) FamilyManagement:
The FamilyManagement class stores the following data structures:
1. roots: Represents the root individuals in the genealogy tree(Set<PersonIdentity>)
2. partnered: Represents all the individuals who have a partner(Set< PersonIdentity >)

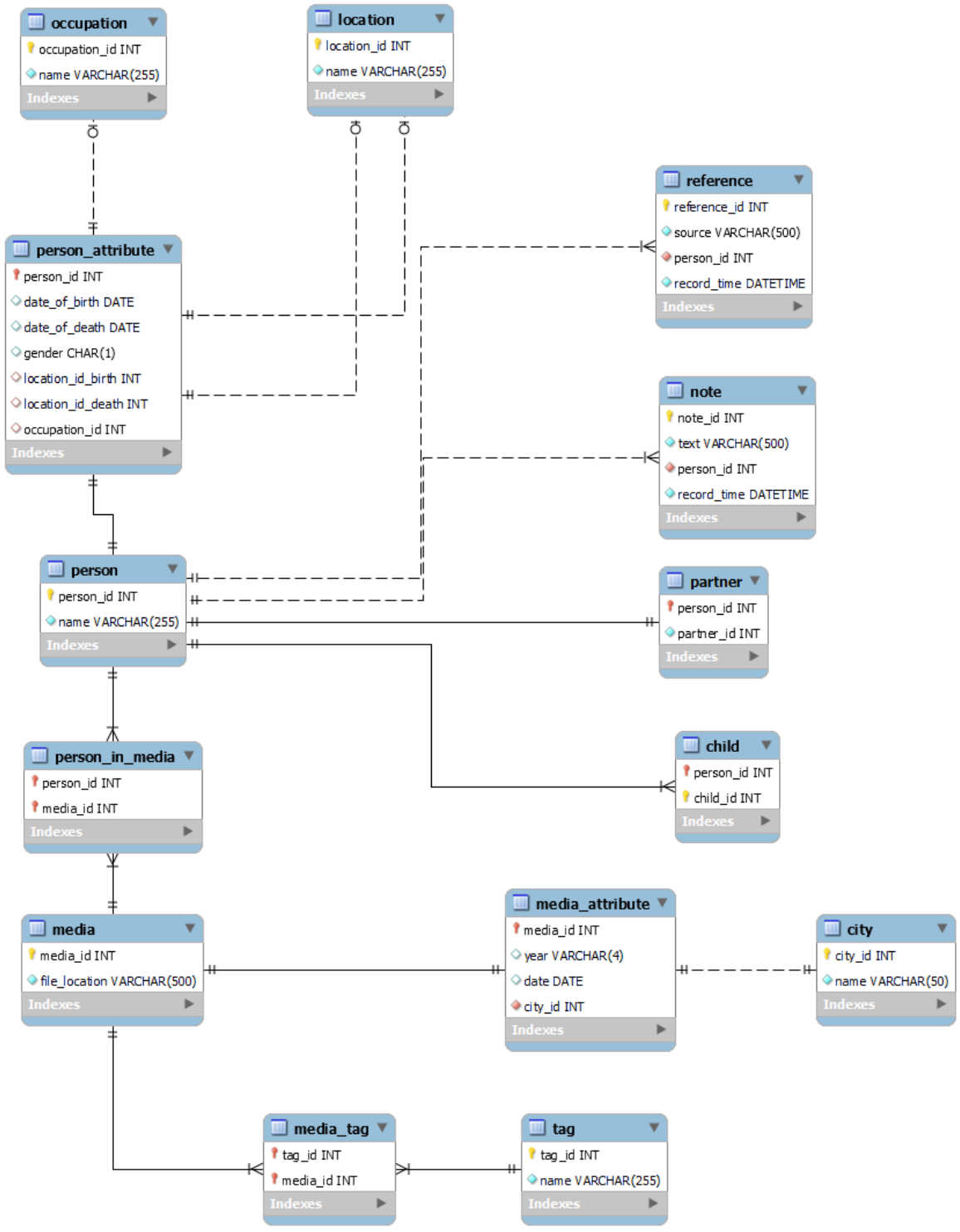
3. children: Represents all the individuals who are a child of a person(Set<PersonIdentity>)
4. personIds: A map to store the personId and PersonIdentity object(Map<Integer, PersonIdentity>)

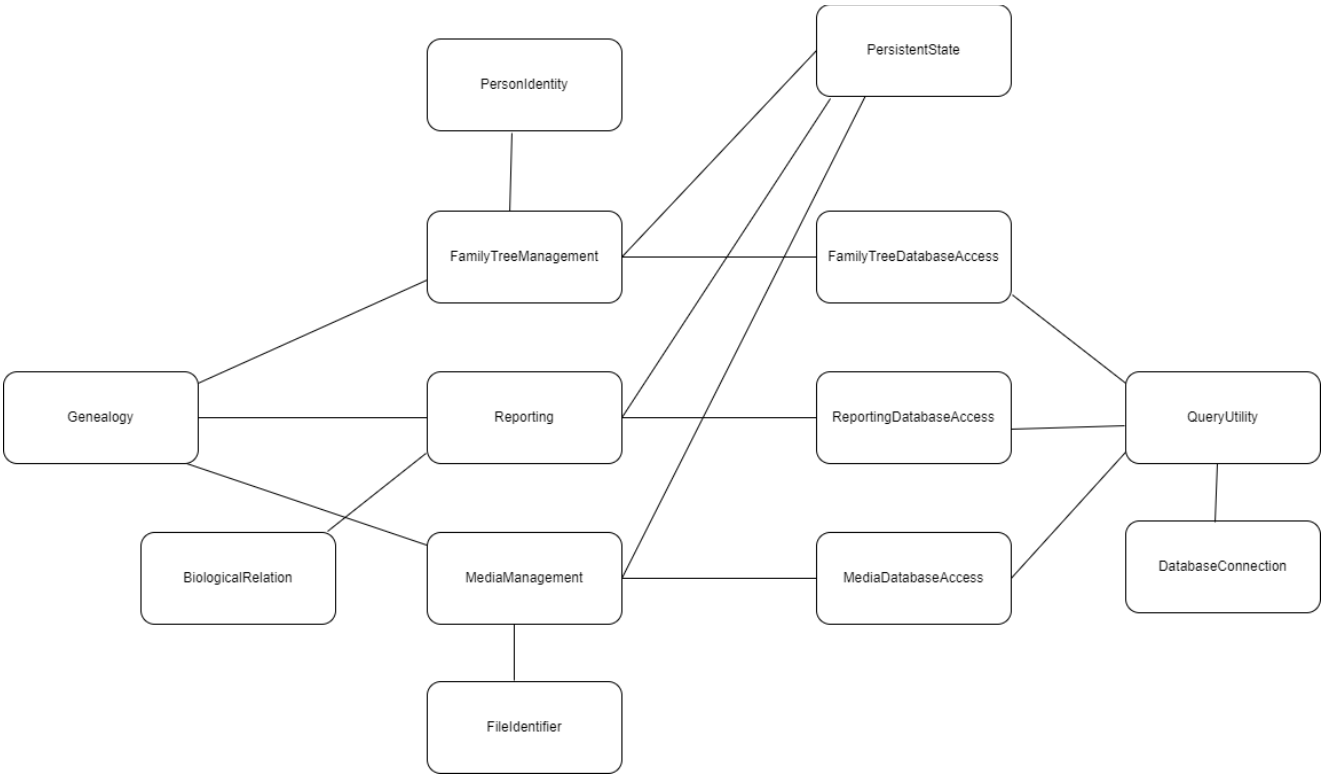
D) MediaManagement:
The media management class stores the following objects:
1. tags: Stores tag name as the key and tagId as the value(Map<String, Integer>)
2. files: Stores fileLocation as the key and FileIdentifier object as the value(Map<String, FileIdentifier>)

E) Genealogy Tree:
The fields of the PersonIdentity class discussed above automatically creates a tree like data structure. The tree can be thought of as top to bottom – In this case, the tree will be a multi-node tree. It can also be considered from bottom to top – In this case, the tree will be a binary tree(Since a person can have a maximum of two parents)

Code Design

Entity Relationship Diagram:


occupation
occupation_id INT
name VARCHAR(255)
Indexes
location
location_id INT
name VARCHAR(255)
Indexes
reference
reference_id INT
source VARCHAR(500)
person_id INT
record_time DATETIME
Indexes
person_attribute
person_id INT
date_of_birth DATE
date_of_death DATE
gender CHAR(1)
location_id_birth INT
location_id_death INT
occupation_id INT
Indexes
note
note_id INT
text VARCHAR(500)
person_id INT
record_time DATETIME
Indexes
person
person_id INT
name VARCHAR(255)
Indexes
partner
person_id INT
partner_id INT
Indexes
person_in_media
person_id INT
media_id INT
Indexes
child
person_id INT
child_id INT
Indexes
media
media_id INT
file_location VARCHAR(500)
Indexes
media_attribute
media_id INT
year VARCHAR(4)
date DATE
city_id INT
Indexes
city
city_id INT
name VARCHAR(50)
Indexes
media_tag
tag_id INT
media_id INT
Indexes
tag
tag_id INT
name VARCHAR(255)
Indexes



Files and External data:

I have distributed the classes into three different layers.

A) Input/Output layer:
The user will be able to interact with these classes while running the program.
1. Genealogy: The class which can be used by the user to invoke all the methods
2. BiologicalRelation: The class returned as the output showing relation between two persons
3. PersonIdentifier: An entity that represents a person in the genealogy tree
4. FileIdentifier: An entity for storing media related information.

B) Intermediate/Validation layer:
These classes as an intermediate layer between the input layer and database layer. The input validation is done in this layer.
1. FamilyTreeManagement: Updates/Inserts the family tree related data structures.
2. MediaManagement: Updates/Inserts the media related data structures.
3. Reporting: Retrieval of data from the tree. Traversal of the tree.

C) Database layer:
These classes hold the information related to the database tables.
1. FamilyTreeDatabaseAccess
2. ReportingDatabaseAccess

3. MediaDatabaseAccess
4. PersistentState: This class is used to retrieve and rebuild the data structure of each new session.
5. QueryUtility: Holds commonly used generic query methods.
6. DatabaseConnection: This class is used to establish a common database connection for the entire program.

Key Algorithms

1. Identifying root ancestors:
   It is important to find the root ancestor in the genealogy tree.
   a) Whenever a new person gets added to the tree, the person automatically becomes one of the roots.
   b) When a person is recorded as the child of a parent, the person loses the root status.

2. Updating root ancestors:
   The genealogy tree is a multi-root tree. To find the relationship between two person, it is necessary to narrow down a common root ancestor between two persons. The easiest way to do this is to maintain a rootAncestor set in the PersonIdentity class. The rootAncestors should be updated in two different cases.

   Scenario 1: While loading the data from the database tables
   a) Iterate over all the roots in the tree.
   b) In each iteration, recursively update the root node as the root ancestor for the descendant nodes.

   Scenario 2: While adding a child
   a) Check the root ancestors of the parent nodes.
   b) Add the root ancestors of both the parents as the root ancestors of the child node.
   c) Recursively traverse all the descendants of the child node -> remove the child node from the root ancestors of the descendants -> And add the new root ancestors of the child node to all the descendants.

3. Sequence of loading family tree information from the database:
   a) Get all individuals
   b) Get all roots
   c) Get partnering relationships
   d) Get all children

4. Finding ancestors:
   Recursively traverse all the parents of the current node till the given number of generations to find all the ancestors.

5. Finding descendants:
   Recursively traverse all the children of the current node till the given number of generations to find all the descendants.

6. Finding relation:
   a) Before finding relation, check if the two persons have any common root ancestors.
   b) If common root ancestors are not present, then they are not related.
   c) Otherwise, pick any one of the common root ancestors.
   d) Create two double ended queues for each person to find the path from the person node to the picked root ancestor node.
   e) Traverse recursively to add the nodes between the person node and the root node.
   f) After adding nodes to both the queues, remove common nodes between two queues.
   g) The size of the queues represents nX and nY.
   h) Calculate degree of cousinship and level of removal between using nX and nY.

Assumptions:

1. Only unique file locations are allowed in the system
2. Only unique named individuals are allowed.
3. Record attributes can both insert and update (the already existing attributes)
4. References does not have to be unique.
5. Note does not have to be unique for an individual.
6. A child cannot have more than two parents.
7. An individual can only be partnered with one person at a time.

Limitations:

1. Only unique named individuals are allowed by the system.
2. Does not handle add child based on birth dates.
3. Does not show relation in a textual format.

Tests:

Black Box Tests

1. Adding person

- Person name is an empty string
- Person name is made up of 1 character
- Person name is made up of maximum allowed characters
- Person name is between 2 and maximum allowed characters
- Person name is made up of more than maximum allowed characters
- Person with the same name added for the second time

2. Recording person attributes
- Record when the person does not exist
- Date of birth format is incorrect
- Date of birth is a future date
- Gender is not in the correct format
- 1 character occupation name
- Maximum allowed characters occupation name
- Occupation name is an empty string
- Occupation name is between 1 and maximum allowed characters

3. Recording reference
- Reference for that individual does not exist
- 1 reference exists for that individual
- Many references exist for that individual
- Record when the person does not exist
- Reference is an empty string
- 1 character reference string
- Maximum allowed characters reference string

4. Recording note
- Note for that individual does not exist
- 1 note exists for that individual
- Many notes exist for that individual
- Record when the person does not exist
- Note is an empty string
- 1 character note string
- Maximum allowed characters note string

5. Recording child
- Parent does not exist
- Child does not exist
- Parent is partnered
- Parent is single
- After dissolution
- Child already linked to different a parent(parents)
- Child already linked to the same parent
- Child already linked to the partner
- Child is partnered
- Child is partnered and added to a parent linked(ancestor) to the partner

6. Recording partnering
- Partner 1 does not exist

- Partner 2 does not exist
- Partner 1 already has a partner
- Partner 2 already has a partner
- Partner 1 partnering with partner 2 who had dissolution
- Partner 2 partnering with partner 1 who had dissolution
- Partnering between individuals having common ancestor
- Partnering between individuals having cousin relationship
- Partnering between individuals having ancestor/descendant relationship.

7. Recording dissolution
- Partner 1 does not exist
- Partner 2 does not exist
- Partner 1 and Partner 2 are not partnered
- Partner 1 and Partner 2 are partnered
- Dissolution twice between two individuals

8. Adding media file
- File location is an empty string
- File location is made up of 1 character
- File location is made up of maximum allowed characters
- File location is between 2 and maximum allowed characters
- File location is made up of more than maximum allowed characters
- Media with the same location added for the second time

9. Recording media attributes
- Record when the media does not exist
- Date format is incorrect
- Date is a future date
- 1 character city name
- Maximum allowed characters city name
- City name is an empty string
- City name is between 1 and maximum allowed characters

10. Recording people linked to a media
- Media does not exist
- No people passed
- One person passed
- Many people passed
- Out of many people, one person passed does not exist

11. Tagging media

- Tag does not exist
- Tag already exists
- Media does not exist
- Tag exceeding maximum allowed character limit
- Tag is already linked with a different media
- Tag is not linked with any other medias
- Tag is an empty string
- 1 character tag
- Many characters tag

12. Finding person
- Person does not exist
- Person exists

13. Finding media file
- Media does not exist
- Media exists

14. Finding an individual's name
- Person exists
- Person does not exist

15. Finding media file
- Media exists
- Media does not exist

16. Finding relation
- Person 1 does not exist
- Person 2 does not exist
- Both person 1 and person 2 exist
- Person 1 and Person 2 are partners
- Person 1 and Person 2 are related
- Person 1 and Person 2 are not related

17. Finding descendants
- Generations is 0
- Generations is a negative value
- Generations is more than the actual available generations
- Generations is 1
- Person does not exist
- Person exists
- Person has a single parent

- Person has both the parents
- After dissolution between parents
- After dissolution between parent and then one of the parents partnering with a different person

18. Finding ancestors
- Generations is 0
- Generations is a negative value
- Generations is more than the actual available generations
- Generations is 1
- Person does not exist
- Person exists
- Person has a single parent
- Person has both the parents
- After dissolution between parents
- After dissolution between parent and then one of the parents partnering with a different person

19. Finding notes and references
- Person does not exist
- Person exists
- Person has no notes associated
- Person has a single note associated
- Person has many notes associated
- Person has no references associated
- Person has a single reference associated
- Person has many references associated

20. Finding media by tag
- Tag exists
- Tag does not exist
- Tag is an empty string
- Start date is invalid
- End date is invalid
- Valid dates
- Start date is an empty string
- End date is an empty string
- Tag not associated with any media
- Tag associated with many medias

21. Find media by location
- Location exists
- Location does not exist
- Location not associated with any media

- Location associated with many medias
- Location is an empty string
- Start date is invalid
- End date is invalid
- Valid dates
- Start date is an empty string
- End date is an empty string

22. Finding individuals media
- Start date is invalid
- End date is invalid
- Valid dates
- Start date is an empty string
- End date is an empty string
- One of the persons passed does not exist
- One person passed
- Many persons passed
- Not a single person passed

23. Finding biological media
- Person does not exist
- Person is partnered
- Person is partnered after dissolution
- Person has a child
- Person has many children
- Person has no children
- Children have no media associated
- Children have media associated

Additional White Box Tests

1. addPerson():
- When the database connection is not present
- Add two persons using two different Genealogy objects
- Add person when no person is present in the database
- Add person when one person is present in the database
- Add person when many persons are present in the database

2. recordAttributes():
- For a person already present in the database
- For a person added in the same session

- Record attributes in a different Genealogy object
- Record attributes twice for the same person

3. recordReference():
- Same reference for two different individuals
- Add reference for a person added in a different session
- Add reference for a person added in the same session

4. recordNote():
- Same reference for two different individuals
- Add reference for a person added in a different session
- Add reference for a person added in the same session

5. recordChild():
- Record child for partnered individuals
- Record child after dissolution
- Record child twice for the same individual
- Record child twice for two different individuals
- Record child added in the same session as the parent
- Record child for the person added in a different session

6. recordPartnering():
- Record partner for partnered individuals
- Record partner after dissolution
- Record partner twice for the same individual
- Record partner twice for two different individuals
- Record partner added in the same session as the other partner
- Record partner for the person added in a different session

7. recordDissolution():
- Record dissolution for partnered individuals
- Record dissolution after dissolution
- Record dissolution twice for the same individual
- Record dissolution twice for two different individuals
- Record dissolution added in the same session as the other partner
- Record dissolution for the person added in a different session

8. addMediaFile():

- When no media is present in the session
- When no media is present in the database

9. recordMediaAttributes():
- For media already present in the database
- For media added in the current session
- For media added in a different genealogy object

10. peopleInMedia():
- For media already present in the database
- For media added in the current session
- For media added in a different genealogy object
- When the database connection is not present

11. tagMedia():
- For media already present in the database
- For media added in the current session
- For media added in a different genealogy object

12. findPerson():
- Person added in the same session
- Person already present in the database

13. findMediaFile():
- Media added in the same session
- Media already present in the database

14. findRelation():
- For both the persons added in the same session
- For one of the persons added in the same session
- For both the persons present in the database

15. ancestors():
- For the person added in the same session
- For the person added in a different session

16. descendants():
- For the person added in the same session
- For the person added in a different session

17. notesAndReferences():
- For the person added in the same session
- For the person added in a different session

18. findMediaByTag():
- Media added in the same session
- Media already present in the database
- Tag added in the same session
- Tag already present in the database

19. findMediaByLocation():
- Media added in the same session
- Media already present in the database

20. findIndividualsMedia():
- Media added in the same session
- Media already present in the database
- For the person added in the same session
- For the person added in a different session

21. findBiologicalMedia():
- For the person added in the same session
- For the person added in a different session

References:

[1] H. [duplicate], D. Kapatel, I. Bursov and D. Buslaev, "How can I use one database connection object in whole application?", *Stack Overflow*, 2021. [Online]. Available: https://stackoverflow.com/questions/20666658/how-can-i-use-one-database-connection-object-in-whole-application. [Accessed: December 4, 2021]