**Archival Database System - Second Prototype**

**Team Members:**

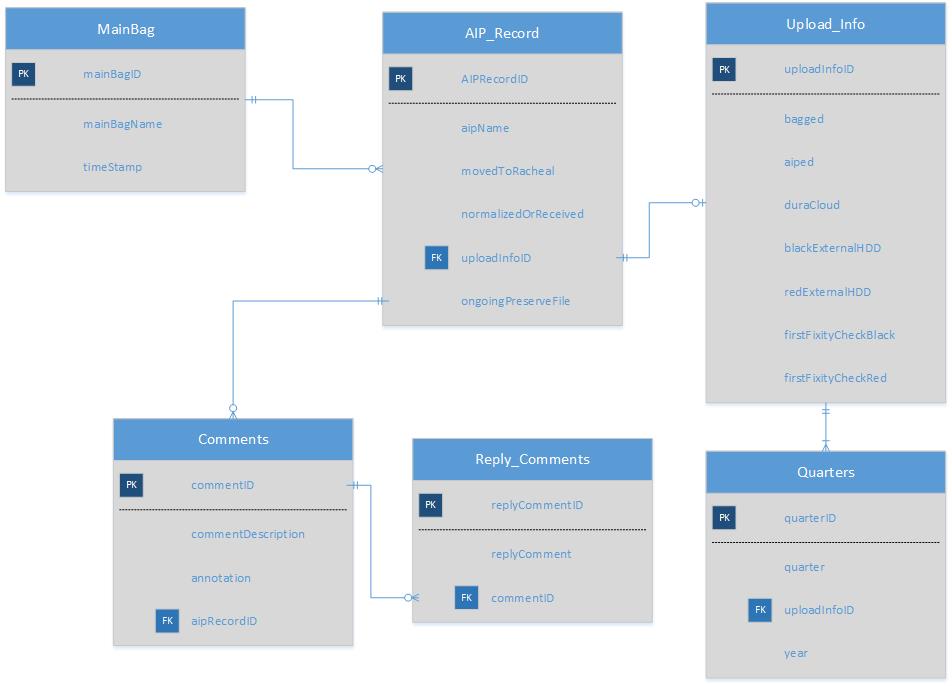
Durga Sparsha Bhagwat

Shafali Gondi

Pavan Jupally

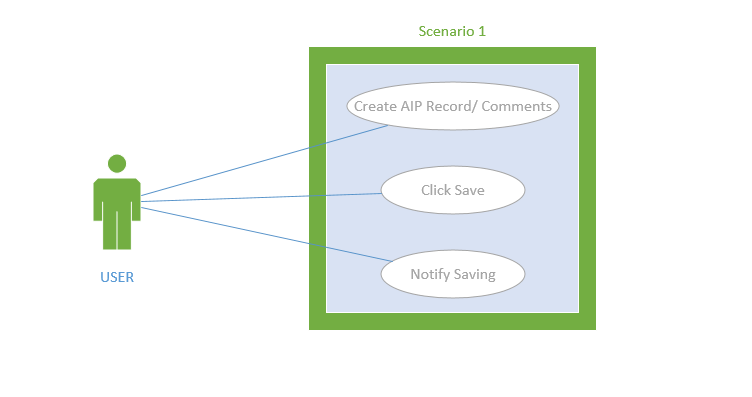
Raihan Ahmed Mohammed

**Entity-Relationship Diagram:**

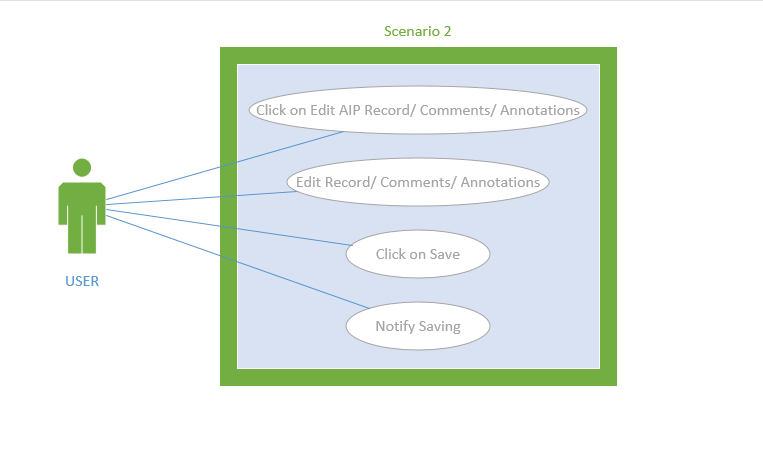
****

**Use Cases:**

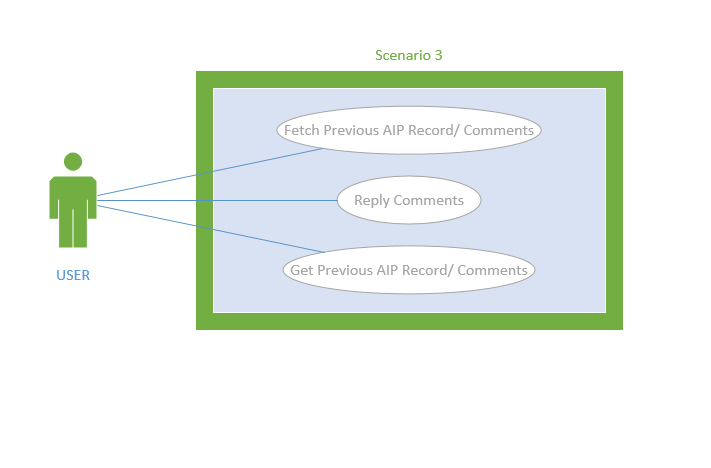
In the first scenario the user tries to create a new AIP Record/ Comment which is saved in the database and clicks on “Save” then the user is provided with the feedback that the changes are made.

****

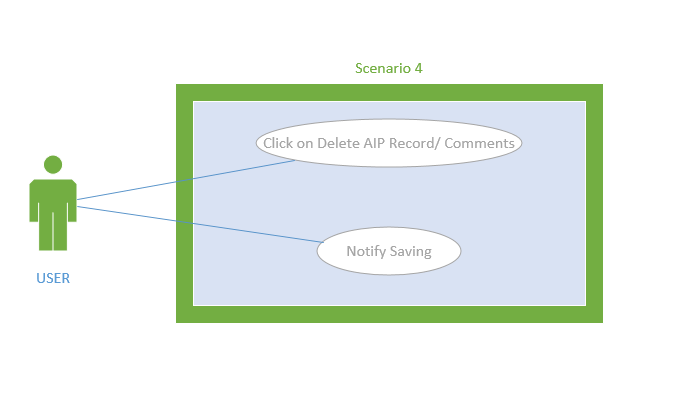
In the second scenario the user first clicks on editing the existing AIP Record/ Comments/Annotations which when in edit mode will be available for the user to edit. After the user edits the required information, he/she clicks on “Save” which saves the data to the database and the user is notified about the changes through feedback.

****

The third scenario takes into consideration that the user has already created/updated the AIP Records/Comments and now wants to retrieve the information and reply to any of the Comments, so the user fetches the AIP Records/Comments, replies to the Comments and the system gets the requested information to the user.

****

The fourth scenario is based on the face that the user may want to delete any of the AIP Record/Comment. The user Clicks on the “Delete” AIP Record/ Comments then the user is notified about the changes.

****

**API Documentation Methods:**

The following methods are used for database access:

* **createAIPRecord** **(AIPRecord aipRecordObject)**

**Syntax:** void createAIPRecord (AIPRecord aipRecordObject)

**Description:** This method takes in aipRecordObject as an argument which contains all the AIP Record values as attributes. The values are taken from the user, saved and passed as an object. When the function receives this object it retrieves all the values from this object and passes to the query to create a new record in the database.

**SQL Function:** INSERT INTO AIP\_Record VALUES (value1, value2, value3 ...);

* **editAIPRecord(int aipRecordID, var valueToEdit)**

**Syntax:** void editAIPRecord (int aipRecordID, var valueToEdit)

**Description:** This method takes in the aipRecordID corresponding to the AIP Record which is to be edited and the value that is supposed to edit and passes this to the query that updates the AIP Record.

**SQL Function:** UPDATE AIP\_Record SET column1=value1, column2=value2...

WHERE aipRecordID = recordID;

* **deleteAIPRecord(int aipRecordID)**

**Syntax:** void deleteAIPRecord (int aipRecordID)

**Description:** Deletes the AIP Record associated with the aipRecordID.

**SQL Function:** DELETE FROM AIP\_Record

WHERE aipRecordID=recordID;

* **getAIPRecord(int aipRecordID)**

**Syntax:** AIPRecord getAIPRecord (int aipRecordID)

**Description:** This method returns an object of type AIPRecord which contains the AIP Record associated with the aipRecordID which the user provides as input.

**SQL Function:** SELECT column\_name FROM AIP\_Record WHERE aipRecordID=recordID;

* **getLatestFixityReportName(DateTime timestamp)**

**Syntax:** string getLatestFixityReportName (DateTime timestamp)

**Description:** Based on given timestamp the method will get the latest fixity report. The mainBagName is concatenated with timestamp and aipRecordName and a string containing the latest fixity report name is returned.

**SQL Function:** SELECT column\_name FROM MainBag WHERE timestamp < currenttimestamp;

* **removeLatestFixityReportName()**

**Syntax:** void removeLatestFixityReportName (DateTime timestamp)

**Description:** Based on given timestamp the method will delete the fixity report.

**SQL Function:** DELETE FROM MainBag WHERE timestamp=currenttimestamp;

* **string getComment()**

**Syntax:** string getComment (int aipRecordID)

**Description:** This method returns a string that contains comment corresponding to the aipRecordID.

**SQL Function:** SELECT commentDescription FROM Comments WHERE aipRecordID=recordID;

* **postComments (int aipRecordID, String commentDescription, String annotation)**

**Syntax:** void postComments (int aipRecordID, String commentDescription, String annotation)

**Description:** This method updates the comment description in the Comments table. It takes in arguments like aipRecordID, annotation and the comment description.

**SQL Function:** INSERT INTO Comments VALUES (aipRecordID, commentDescription, annotation);

* **editComments(int aipRecordID)**

**Syntax:** void editComments (int aipRecordID)

**Description:** Edits the comments based on the aipRecordID.

**SQL Function:** UPDATE Comments SET column1=value1, column2=value2...

WHERE aipRecordID = recordID;

* **removeComments()**

**Syntax:** void removeComments (int aipRecordID)

**Description:** This method deletes the specified comment based on the aipRecordID.

**SQL Function:** DELETE FROM Comments WHERE aipRecordID=recordID;

* **getReply()**

**Syntax:** string getReply (int commentID)

**Description:** This method return a string that contains the reply of a comment based on the commentID.

**SQL Function:** SELECT column\_name FROM Comments WHERE commentID=commentID;

* **postReply(String replyCommentDescription, int commentId)**

**Syntax:** postReply (String replyCommentDescription, int commentID)

**Description:** This method saves the reply comment description for the corresponding commentID

**SQL Function:**  INSERT INTO Reply\_Comments VALUES (commentID, replyCommentDescription);

* **editReply(int replyCommentID)**

**Syntax:** void editReply (int replyCommentID)

**Description:** Edits the reply comment description for the corresponding replyCommentID.

**SQL Function:** UPDATE Reply\_Comments SET column1=value1, column2=value2...

WHERE replyCommentID =replyCommentID;

* **removeReply(int replyCommentID)**

**Syntax:** removeReply (int replyCommentID)

**Description:** The user can delete the reply Comment associated with the replyCommentID.

**SQL Function:** DELETE FROM Reply\_Comments WHERE replyCommentID = replyCommentID;