**Abstract Review System (CAHSI)**

**Database Name:**

f17cs4342team 03

**Team Name:**

Data Miners

**Team Members:**

Joseph Gutierrez

Stephanie Loya

Andres Olivas

Adrian Gonzalez

**Table of Contents**

* 1. Title Page…………………………………………………………………………… Pg. 1
  2. Table of Contents………………………………………………………………. Pg. 2
  3. Abstract Review System Scope………………………………………….. Pg. 3-4
  4. Requirements and Assumptions………………………………………… Pg. 5-6
  5. Entity-relationship model…………………………………………………… Pg. 7
  6. Relational Model………………………………………………………………… Pg. 8
  7. Normalized schema……………………………………………………………. Pg. 9
  8. Database Schema in MySQL……………………………………………….. Pg. 10-16
  9. Database Records………………………………………………………………. Pg. 16-19
  10. **Abstract Review System Scope**

HENAAC conference planning, abstract reviews, poster competition, work with different

institutions. It need to be able to review folders that match with students and gives them

feedback about their work. The submissions must be in word. Edits can be made in word.

**Sequence of completion:**

Submission of work

Review by mentor

Feed back to the student

Edits or corrections to the work

Should be updated by student

Sent back for review

Approve

The abstract can be resubmitted only if it was not rejected. It should be able to notify

the reviewer that changes or corrections were made. It should show the deadline.

Students are going to be submitting their abstracts from more than 15 institutions.

Students are undergraduate and graduate from CS, EE, or Math. There are 2 levels of competition, graduate, and undergraduate.

It should store, last name, first name, major, classification, title of the abstract,

mentor/mentors, institution, keep track of mentor email, ethnicity, gender, and student email.

It also needs to be able to see how many abstracts have being summited, CAHSI colors,

and logo must be present. Students can submit, view, and update the abstract but they cannot

delete it. It must keep information from previous years. There are four types of users which are

Mentor, Review, Student, and Admin.

**Task to perform:**

The system must allow the reviewer to make changes and review the abstract.

Status of the system:

Approved – Ready for submission

Approved- with Corrections

Not Approved – Major Corrections (Can still submit)

Rejected (Cannot Submit again)

The system must allow the abstract to be time sensitive. Meaning that there is a time limit to upload the abstract. Posters are in editable form (PowerPoint, and Word)

Keep track of where the submission came (to make sure it’s a CHASI school)

Majors of the students

Names of participants

Classification of the students (Freshman, Sophomore, Junior, Senior)

Title of abstract

Who Is mentor(s) (Notify)

Mentor email

Ethnicity

Gender

Institution

Student email

**1.4 Requirements and Assumptions**

**Functional​ ​Requirements**

F1. The system shall only accept the abstract in a word file document.

F2. The system shall not allow a rejected abstract to be submitted again.

F3. The system shall not allow the students to delete of submissions.

F4. The system shall allow the mentors to view their student’s entries.

F5. The system shall record the administrator’s name, email, institution, and phone number.

F6. The system shall keep all reviews anonymous.

F7. The system shall keep record of students' names, emails, classifications, institutions, genders, ethnicities, and majors.

F8. The system shall keep record of mentors' and reviewers' name, email, gender, institution, title, and ethnicity.

F9. The system shall allow the administrator to set up deadlines for the reviewers and students

submissions.

F10. The system shall notify administrator of latest updates.

F11. The system shall allow the student to edit their submission and resubmit if their abstract is

not rejected.

F12. The system shall manage which mentors are supposed to review which submissions.

F13. The system shall generate reports for the administrator based on completion and status of

the abstracts.

F14. The system shall retain information from year to year to enable the administrator to contact the previous participants of new call for papers and other events.

F15. The system shall also house a review system for the posters in PowerPoint.

F16. The system shall display CAHSI logo and colors in the interface.

F17. The system shall provide separate interfaces for each type of user.

F18. The system shall be semi-reactive (able to resize).

F19. The system shall have color codes for the different statuses that an abstract may have.

**Functional Assumptions**

A1. The system will only accept posters in PowerPoint form. This will enable the program to conduct quick and easy correction mark up.

A2. The system shall save a copy of each entry and correction of the poster’s an abstract so that the administrator may, at her discretion see the progression of a submission.

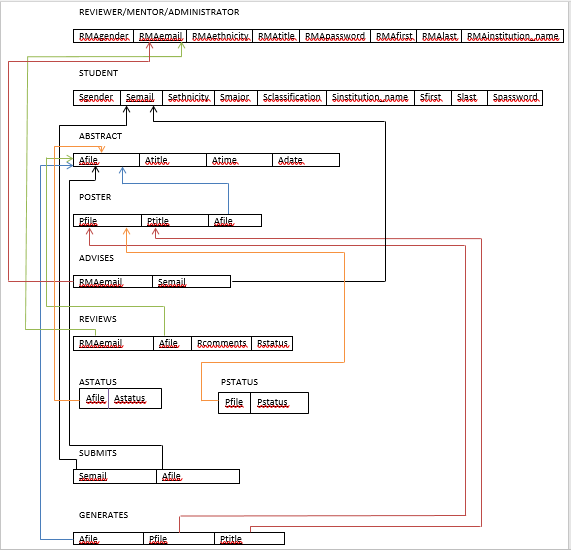
A3. The system shall timestamp each submission and ensure that it is not submitted after the time it is due.

* 1. **E/R Diagram**

**A close up of a map

Description generated with high confidence**

* 1. **Relational Model**



* 1. **Normalized Schema**A close up of a device

     Description generated with high confidence
  2. **Database Schema**

CREATE TABLE Advices

(

Uuser\_id INT(6) NOT NULL AUTO\_INCREMENT,

ADmentor INT(6) NOT NULL,

ADstudent INT(6) NOT NULL,

PRIMARY KEY (Uuser\_id),

KEY ADmentor (ADmentor),

KEY ADstudent (Adstudent)

);

CREATE TABLE User

(

Uuser\_id INT(6) NOT NULL AUTO\_INCREMENT,

Uemail VARCHAR(50) NOT NULL,

Upswd VARCHAR(50) NOT NULL,

Urole INT(1) NOT NULL,

PRIMARY KEY (Uuser\_id),

KEY Urole (Urole)

);

CREATE TABLE Role

(

Rrole\_id INT(1) NOT NULL AUTO\_INCREMENT,

Rrole VARCHAR(50) NOT NULL,

PRIMARY KEY (Rrole\_id)

);

CREATE TABLE Men\_Rev

(

MRmenrev\_id INT(6) NOT NULL AUTO\_INCREMENT,

Ggender\_id INT(6) NOT NULL,

Uuser\_id INT(6) NOT NULL,

Eethnicity\_id INT(6) NOT NULL,

Ttitle\_id INT(6) NOT NULL,

Iinst\_id INT(6) NOT NULL,

MRgender INT(6) NOT NULL,

MRuser INT(6) NOT NULL,

MRethnicity INT(6) NOT NULL,

MRtitle INT(6) NOT NULL,

MRinst\_name INT(6),

MRfirst VARCHAR(50) NOT NULL,

MRlast VARCHAR(50) NOT NULL,

PRIMARY KEY (MRmenrev\_id),

KEY MRgender (MRgender),

KEY MRuser (MRuser),

KEY MRethnicity (MRethnicity),

KEY MRtitle (MRtitle),

KEY MRinst\_name (MRinst\_name)

);

CREATE TABLE Student

(

Sstudent\_id INT(6) NOT NULL AUTO\_INCREMENT,

Ggender\_id INT(6) NOT NULL,

Uuser\_id INT(6) NOT NULL,

Eethnicity\_id INT(6) NOT NULL,

Mmajor INT(6) NOT NULL,

Cclass INT(6) NOT NULL,

Iinst\_id INT(6) NOT NULL,

Sgender INT(6) NOT NULL,

Suser INT(6) NOT NULL,

Sethnicity INT(6) NOT NULL,

Smajor INT(6) NOT NULL,

Sclass INT(6) NOT NULL,

Sinst\_name INT(6),

Sfirst VARCHAR(50) NOT NULL,

Slast VARCHAR(50) NOT NULL,

PRIMARY KEY (Sstudent\_id),

KEY Sgender (Sgender),

KEY Suser (Suser),

KEY Sethnicity (Sethnicity),

KEY Smajor (Smajor),

KEY Sclass (Sclass),

KEY Sinst\_name (Sinst\_name)

);

CREATE TABLE Gender

(

Ggender\_id INT(1) NOT NULL AUTO\_INCREMENT

Ggender VARCHAR(15) NOT NULL,

PRIMARY KEY (Ggender\_id)

);

CREATE TABLE Ethnicity

(

Eethnicity\_id INT(1) NOT NULL AUTO\_INCREMENT

Eethnicity VARCHAR(20) NOT NULL,

PRIMARY KEY (Eethnicity\_id)

);

CREATE TABLE Title

(

Ttitle\_id INT(1) NOT NULL AUTO\_INCREMENT

Ttitle VARCHAR(50) NOT NULL,

PRIMARY KEY (Ttitle\_id)

);

CREATE TABLE Institution

(

Iinst\_id INT(1) NOT NULL AUTO\_INCREMENT

Iinst VARCHAR(50) NOT NULL,

PRIMARY KEY (Iinst\_id)

);

CREATE TABLE Major

(

Mmajor\_id INT(1) NOT NULL AUTO\_INCREMENT,

Mmajor VARCHAR(50) NOT NULL,

PRIMARY KEY (Mmajor\_id)

);

CREATE TABLE Classification

(

Cclass\_id INT(1) NOT NULL AUTO\_INCREMENT,

Cclass VARCHAR(20) NOT NULL,

PRIMARY KEY (Cclass\_id)

);

CREATE TABLE Poster

(

Pposter\_id INT(6) NOT NULL AUTO\_INCREMENT,

Pfile VARCHAR(50) NOT NULL,

Ptitle VARCHAR(50) NOT NULL,

PRIMARY KEY (Pposter\_id)

);

CREATE TABLE Generates

(

Ggenerates\_id INT(6) NOT NULL AUTO\_INCREMENT,

Afile VARCHAR(50) NOT NULL,

Atitle VARCHAR(50) NOT NULL,

Pfile VARCHAR(50) NOT NULL,

Ptitle VARCHAR(50) NOT NULL,

Gab\_file INT(6) NOT NULL,

Gab\_title INT(6) NOT NULL,

Gpo\_file INT(6) NOT NULL,

Gpo\_title INT(6) NOT NULL,

PRIMARY KEY (Ggenerates\_id),

KEY Gab\_file (Gab\_file),

KEY Gab\_title (Gab\_title),

KEY Gpo\_file (Gpo\_file),

KEY Gpo\_title (Gpo\_title)

);

CREATE TABLE Submits

(

Ssubmits\_id INT(6) NOT NULL AUTO\_INCREMENT,

Uuser\_id INT(6) NOT NULL,

Afile VARCHAR(50) NOT NULL,

Atitle VARCHAR(50) NOT NULL,

SUemail VARCHAR(50) NOT NULL,

SUfile VARCHAR(50) NOT NULL,

SUtitle VARCHAR(50) NOT NULL,

PRIMARY KEY (Ssubmits\_id),

KEY SUemail (SUemail),

KEY SUfile (SUfile),

KEY SUtitle (SUtitle)

);

CREATE TABLE Abstract

(

Aabstract\_id INT(6) NOT NULL AUTO\_INCREMENT,

Afile VARCHAR(50) NOT NULL,

Atitle VARCHAR(50) NOT NULL,

Adate TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (Aabstract\_id)

);

CREATE TABLE Reviews

(

Rreviews\_id INT(6) NOT NULL AUTO\_INCREMENT,

Uuser\_id INT(6) NOT NULL,

Afile VARCHAR(50) NOT NULL,

Atitle VARCHAR(50) NOT NULL,

Sstatus\_id INT(1) NOT NULL,

RVreviewer INT(6) NOT NULL,

RVfile VARCHAR(50) NOT NULL,

RVtitle VARCHAR(50) NOT NULL,

RVstatus INT(1) NOT NULL,

RVcomments VARCHAR(100),

PRIMARY KEY (Rreviews\_id),

KEY RVreviewer (RVreviewer),

KEY RVfile (RVfile),

KEY RVtitle (RVtitle),

KEY RVstatus (RVstatus)

);

CREATE TABLE Status

(

Sstatus\_id INT(1) NOT NULL AUTO\_INCREMENT,

Sstatus VARCHAR(20) NOT NULL,

PRIMARY KEY (Sstatus\_id)

);

* 1. **Database Record**

All of the inserted Records follow this pattern in order to attain the correct look up reference numbers. This is an example provided from the student sign up page.

1. $username = mysqli\_real\_escape\_string($db,$\_POST['username']);
2. $password = mysqli\_real\_escape\_string($db,$\_POST['password']);
3. $last = mysqli\_real\_escape\_string($db,$\_POST['lname']);
4. $first = mysqli\_real\_escape\_string($db,$\_POST['fname']);
5. $gender = mysqli\_real\_escape\_string($db,$\_POST['gender']);
6. $institution = mysqli\_real\_escape\_string($db,$\_POST['institution']);
7. $ethnicity = mysqli\_real\_escape\_string($db,$\_POST['ethnicity']);
8. $major = mysqli\_real\_escape\_string($db,$\_POST['major']);
9. $class = mysqli\_real\_escape\_string($db,$\_POST['classification']);
11. //Checks if the entry exists in the lookup table
12. $checkGender = "SELECT \* FROM Gender WHERE Ggender = '$gender'";
13. $checkGenderResult = mysqli\_query($db,$checkGender);
14. $checkGenderCount = mysqli\_num\_rows($checkGenderResult);
15. if($checkGenderCount){
16. //Gets the lookup value to insert into the main table
17. $genderLookupNum = $checkGenderResult['Ggender\_id'];
18. }
19. //If it doesnt exist, the entry is added to the table and its lookup value is found.
20. else{
21. $insertGender = "INSERT INTO Gender (Ggender) VALUES ('$gender')";
22. mysqli\_query($db,$insertGender);
23. $checkGender = "SELECT \* FROM Gender WHERE Ggender = '$gender'";
24. $checkGenderResult = mysqli\_query($db,$checkGender);
25. $genderLookupNum = $checkGenderResult['Ggender\_id'];
26. }
27. //Similar process for the rest of the variables
28. $checkInstituiton = "SELECT \* FROM Instituiton WHERE Iinst\_name = '$institution'";
29. $checkInstituitonResult = mysqli\_query($db,$checkInstituiton);
30. $checkInstituitonCount = mysqli\_num\_rows($checkInstituitonResult);
31. if($checkInstituitonCount){
32. $InstitutionLookupNum = $checkInstituitonResult['Iinst\_id'];
33. }
34. else{
35. $insertInstituiton = "INSERT INTO Instituiton (Iinst\_name) VALUES ('$institution')";
36. mysqli\_query($db,$insertInstitution);
37. $checkInstituiton = "SELECT \* FROM Instituiton WHERE Iinst\_name = '$institution'";
38. $checkInstituitonResult = mysqli\_query($db,$checkInstituiton);
39. $InstitutionLookupNum = $checkInstituitonResult['Iinst\_id'];
40. }
42. $checkMajor = "SELECT \* FROM Major WHERE Mmajor = '$major'";
43. $checkMajorResult = mysqli\_query($db,$checkMajor);
44. $checkMajorCount = mysqli\_num\_rows($checkMajorResult);
45. if($checkMajorCount){
46. $majorLookupNum = $checkMajorResult['Mmajor\_id'];
47. }
48. else{
49. $insertMajor = "INSERT INTO Major (Mmajor) VALUES ('$major')";
50. mysqli\_query($db,$insertMajor);
51. $checkMajor = "SELECT \* FROM Major WHERE Mmajor = '$major'";
52. $checkMajorResult = mysqli\_query($db,$checkMajor);
53. $majorLookupNum = $checkMajorResult['Mmajor\_id'];
54. }
56. $checkEthnicity = "SELECT \* FROM Ethnicity WHERE Eethnicity = '$ethnicity'";
57. $checkEthnicityResult = mysqli\_query($db,$checkEthnicity);
58. $checkEthnicityCount = mysqli\_num\_rows($checkEthnicityResult);
59. if($checkEthnicityCount){
60. $ethnicityLookupNum = $checkEthnicityResult['Eethnicity\_id'];
61. }
62. else{
63. $insertEthnicity = "INSERT INTO Ethnicity (Eethnicity) VALUES ('$ethnicity')";
64. mysqli\_query($db,$insertEthnicity);
65. $checkEthnicity = "SELECT \* FROM Ethnicity WHERE Eethnicity = '$ethnicity'";
66. $checkEthnicityResult = mysqli\_query($db,$checkEthnicity);
67. $ethnicityLookupNum = $checkEthnicityResult['Eethnicity\_id'];
68. }
70. $checkClassification = "SELECT \* FROM Classification WHERE Cclass = '$class'";
71. $checkClassificationResult = mysqli\_query($db,$checkClassification);
72. $checkClassificationCount = mysqli\_num\_rows($checkClassificationResult);
73. if($checkClassificationCount){
74. $classLookupNum = $checkClassificationResult['Cclass\_id'];
75. }
76. else{
77. $insertClassification = "INSERT INTO Classification (Cclass) VALUES ('$class')";
78. mysqli\_query($db,$insertClassification);
79. $checkClassification = "SELECT \* FROM Classification WHERE Cclass = '$class'";
80. $checkClassificationResult = mysqli\_query($db,$checkClassification);
81. $classLookupNum = $checkClassificationResult['Cclass\_id'];
82. }
83. $checkRole = "SELECT \* FROM Role WHERE Rrole = 'student'";
84. $checkRoleResult = mysqli\_query($db,$checkRole);
85. $checkRoleCount = mysqli\_num\_rows($checkRoleResult);
86. if($checkRoleCount){
87. $studentLookupNum = $checkRoleResult['Rrole\_id'];
88. }
89. else{
90. $insertRole = "INSERT INTO Role (Rrole) VALUES ('student')";
91. mysqli\_query($db,$insertRole);
92. $checkRole = "SELECT \* FROM Role WHERE Rrole = 'student'";
93. $checkRoleResult = mysqli\_query($db,$checkRole);
94. $studentLookupNum = $checkRoleResult['Rrole\_id'];
95. }
97. $insertUser = "INSERT INTO User (Uemail, Upswd, Urole) VALUES ('$email', '$password', '$studentLookupNum')";
98. mysqli\_query($db,$insertUser);
99. $checkUser = "SELECT \* FROM User WHERE Uemail = '$username' and Upswd = '$password'";
100. $checkUserResult = mysqli\_query($db,$checkUser);
101. $userLookupNum = $checkUserResult['Uuser\_id'];


105. //Building the main table entry with the lookup values found.
106. $insertStudent = "INSERT INTO Student (Sgender, Suser, Sethnicity, Smajor, Sclass, Sinst\_name, Sfirst, Slast) VALUES ('$genderLookupNum','$userLookupNum','$ethnicityLookupNum','$majorLookupNum','$classLookupNum','$InstitutionLookupNum','$first','$last')";
107. mysqli\_query($db,$insertStudent);