CS4400 Database Project:
Georgia Tech Tutors

Summer 2014

Project Overview

The purpose of this project is to analyze, specify, design, implement, document, and demonstrate a database management information system called Georgia Tech Tutors (GTT). The project will proceed in three phases as outlined in the Classical Methodology for Database Development: Analysis & Specification, Design, and Implementation & Testing. The system should be implemented using a Database Management System (DBMS) that supports standard SQL queries. Class administrators will provide you with information about how to access a college-managed MySQL server in order to implement your database. Your professor must approve alternative implementations. Under no circumstances may you use a tool that automatically generates SQL or automatically maps programming objects into the database.

The three phases of the project cover the following work-processes from the Classical Methodology for Database Development. The due dates for each phase appear in the table below.

Phase	Phase Description	Due Date
I	Analysis & Specification	June 3
II	Design	June 24
III	Implementation & Testing	July 22
	Demonstration	July 23-25

Groups

Project groups must have 2 or 3 members. A group may remove a member from further participation in the group when phase I is turned in or when phase II is turned in. A written (email) notification must be provided to the professor at that time.

Phase I – Analysis & Specification

The phase I deliverables include:

1. A cover page including the name, class section, email address, and T-Square username of each group member.

2. A list of contributions of each team member.

3. An Information Flow diagram showing the primary documents and tasks of the system and the flow of data among them. An example of this deliverable can be found in **Appendix B – Sample EER Diagram**.

4. The data model for your system in the form of an Enhanced Entity Relationship (EER) diagram. The EER diagram must capture the constraints of the system as fully as possible. For example, all relationship types should have cardinality constraints, all keys should be identified, and total participation constraints should appear where applicable. An example of this deliverable can be found in **Appendix B – Sample EER Diagram**.

5. A list of business constraints that will be enforced. Do not include any constraints that can be shown in the EER diagram, but rather business logic related constraints that cannot be expressed in EER. The constraints should be written in terms of the Entity Types, Relationship Types and Attributes of your EER diagram. An example of this deliverable can be found in **Appendix C – Sample Constraints**.

6. A list of any assumptions made, including explanations. You are allowed to make up additional reasonable assumptions and constraints as long as they do not conflict with the specified constraints and requirements. If possible, those additional assumptions and constraints should be included in the EER diagram.

Phase II – Design

The phase II deliverables include:

1. A cover page including the name, class section, email address, and T-Square username of each group member.

2. A list of contributions of each team member.

3. An updated copy of the EER diagram and Information Flow diagram including any corrections made since phase I. Alternatively, the group may choose to adopt the EER diagram and Information Flow diagram from the sample solution.

4. A relational schema diagram with primary and foreign keys identified and referential integrity shown by arrows. An example of this deliverable can be found in **Appendix D – Sample Relational Model Diagram**.

5. `CREATE TABLE` SQL statements, including domain constraints, integrity constraints, primary keys, and foreign keys. An example of this deliverable can be found in **Appendix E – Sample CREATE TABLE Statements**.

6. SQL statements and abstract code for each task. An example of this deliverable can be found in **Appendix F – Example Abstract Code + SQL**.

   Multiple SQL statements may be required in order to complete one task. However, in such cases, the last SQL statement should show the output according to the specification as much as possible. If mentioned, the returned tuples must be ordered according to the specification. Views and nested queries may be used to support the tasks. Complex queries should be broken down into views to make the query more readable.

Phase III – Implementation & Testing

There are two options for phase III: lightweight and heavyweight. The *lightweight* option requires a demonstration of working SQL statements that could be used to perform all tasks specified in the project description (i.e., the set of tasks identified in the Information Flow diagram). The queries will be executed against a database created by each group and pre-populated with sample data. A query interface such as phpMyAdmin or the MySQL Query Browser will be used to execute your queries.

The *heavyweight* option is to implement a working application including a graphical user interface (GUI) with all functionality described in the project description. Under this option, the SQL statements will be embedded in a host language, such as PHP or Python.

For both lightweight and heavyweight options, you will need to generate sample data for your database, which will be used for your demo.

The deliverables for phase III include:

1. A coverpage including the name, class section, email address, and T-Square username of each group member.

2. A list of contributions of each team member.

3. A copy of the relational schema diagram from phase II, including any revisions.

4. A copy of the `CREATE TABLE` statements from phase II, including any revisions.

5. A relational database pre-populated with sample data.

6. A set of working SQL statements for all tasks (*lightweight option*).

7. A functional application with embedded SQL statements (*heavyweight option*).

Grading

Phase I and Phase II of the project are each worth 10% of your final grade. Credit for phase III depends on the implementation option you choose: the heavyweight option counts for 20% of your final grade and the lightweight option counts for 5%.

In addition, when phase 3 is submitted, each group member will need to sign and submit the following   statement:

I certify that I have contributed XX% of effort to the overall project and I agree to be graded accordingly.
> ***Signed:_____***

The XX value should ideally be the same for all members of the project, however if one or more of you do not pull your weight during the project you should list an accordingly smaller percentage.

Overview of Georgia Tech Tutors (GTT)

Georgia Tech Tutors is a free, semester long appointment-based tutoring program offered to all Georgia Tech undergraduate students. Students sign up for a one hour long tutoring session per week per course for the entire semester. All tutoring sessions are held on campus. Tutoring is available in many courses and varies based on availability of student tutors. All tutors are successful undergraduate or graduate students who have made a grade of "A" in the course(s) they tutor and have a minimum overall GPA of 3.0. They are available for at least 5 hours per week during the day. Our tutors also have recommendations from one or more Georgia Tech faculty. Our database system should contain data for fall 2013, spring 2014 and summer 2014 semesters (i.e., the current academic year).

The following sections contain a functional description of the GTT system along with some screen mockups. We should note that we will not implement a complete real-world system for this application but rather a subset of the system that is described in this document. *The user interfaces depicted in this project description merely serve as examples to guide your thinking.* Your project's interface may look different and that is fine—even encouraged! For example, you might choose to split up some interfaces we have shown on a single screen into multiple screens. You might choose to use popup windows instead of refreshing the page. A complete reorganization of the user interface is acceptable as long as your application supports the same functionality as described below. If you choose the heavyweight option, you may implement the project as a traditional standalone application (e.g., using Python or Java GUIs) or as a web application (e.g., using a web scripting language like PHP). Your project

is not graded on its aesthetic appeal, but on its functionality as it relates to the application requirements.

Logging In

The GTT login screen is shown in Figure 1. All users are uniquely identified by his or her Georgia Tech ID number.   A user can be a student who needs a tutor, or a student who is/wants to be a tutor, or a professor who wants to recommend a tutor, or a system administrator.   A user enters a valid Georgia Tech ID number and password combination for logging into the system and selects the **OK button**. Users obtain a password by contacting the GTT Administrators offline at which time the type of user access is determined by the administrators.   In addition other information about a user such as name, email and telephone number is input into the system by the administrators. In addition, for tutors, transcripts would be required but we will not store them in our system. If invalid login credentials are input then an error message should be displayed and the user should be asked to retry.
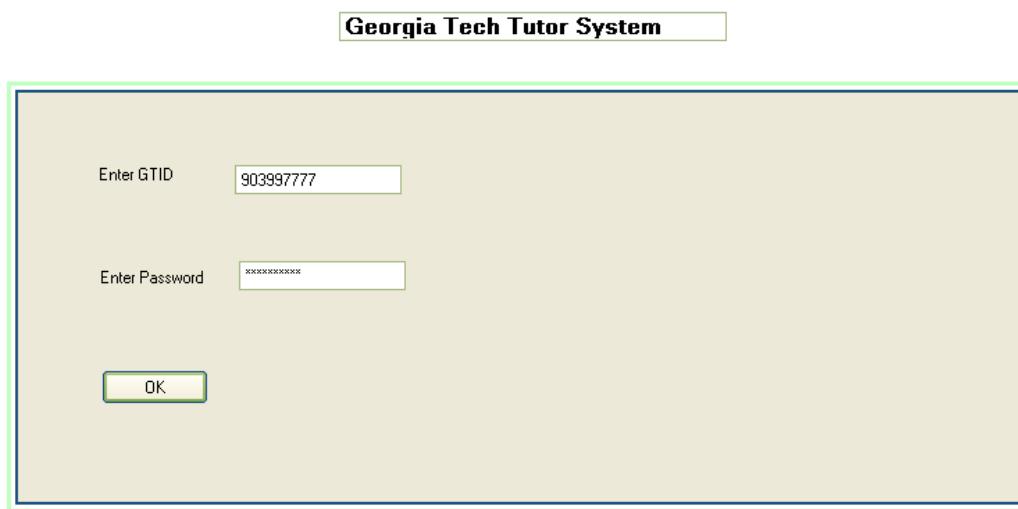
Figure 1 – GTT Login Screen

Once a user has successfully logged in, then a main menu of options is displayed as in Figure 2. Options on the main menu are restricted to the type of user. For example, a user who is only a student type can only search for a tutor, schedule a tutor and rate a tutor.   A user would select a button corresponding to the task they wish to perform. A user may also exit the GTT system by selecting the **Exit button**.
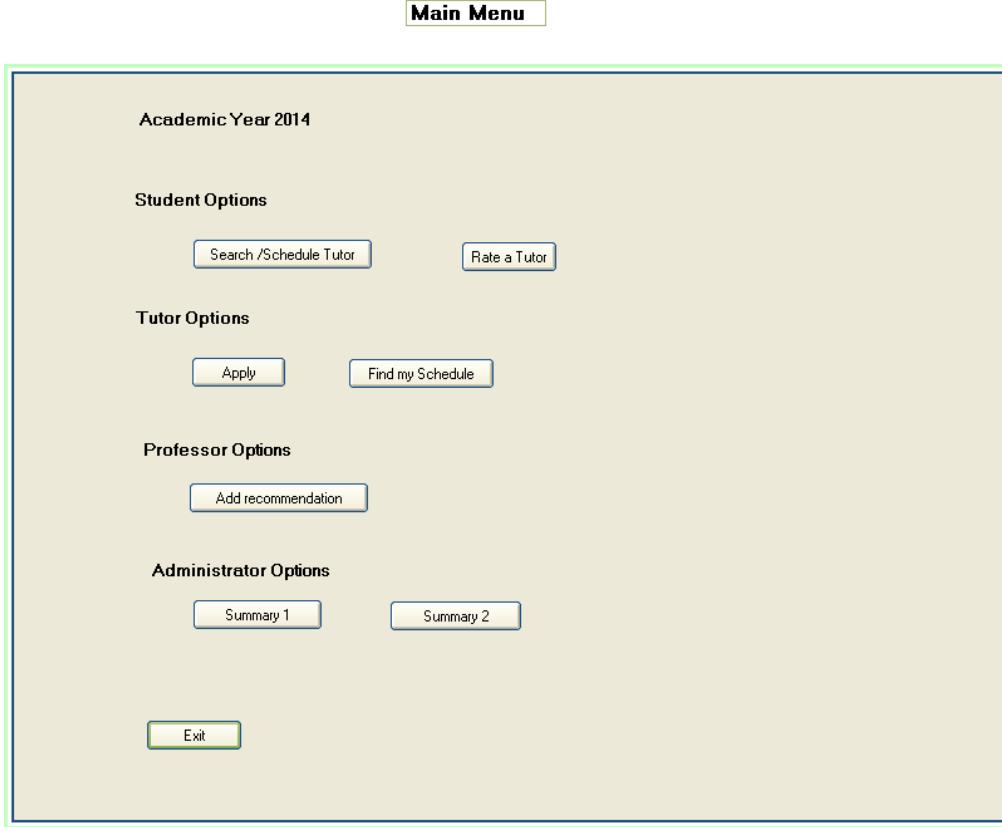
Figure 2 – GTT Main Menu Screen

At this point we will describe the individual tasks that may be performed by the different types of users: **Student**, **Tutor**, **Professor**, **Administrator**.

Student User

The Student user has 3 tasks that may be performed.   The first is to search for a tutor for a particular course for particular days/times.   This is shown in Figure 3. As input, the student would enter the course (school and number) and a set of days/times when they are free to be tutored. The course information should be provided by the system via a drop down box. To execute the search, the user selects the **OK button**. Note, that a student may only receive one hour of tutoring per week per course. The output of this search (i.e., information about the available tutors) is displayed on the screen. The output is ordered by average student rating in descending order. At this point a student may decide to return to the main menu by selecting the **cancel button** or they may decide to schedule one of the available tutors by selecting the **schedule a tutor button**.

Figure 3 – Student Search Screen

If the student chooses to schedule one of these tutors then the screen in Figure 4 will appear. This screen contains some of the tutor information from the previous screen along with the days and times the tutor is available. There is a check box next to each tutor. The student can check only one box and then select the **OK button** to schedule that particular tutor.   After that, information is stored in the database and the system will return the user to the main menu. If the student chooses the **Cancel button** then the user is returned to the main menu without selecting a tutor.

The last task a student user may perform is to rate a tutor. A student can only rate a tutor who they actually use as a tutor during the current semester. The Rate a Tutor screen is shown in Figure 5. The student would fill in **all** the information on the screen and then select the **OK button**. The course information is provided by the system via a drop down box. The

evaluation would be stored in the database. If the **OK button** is selected without filling in the required information then the user would be returned to the main menu without any changes being made.

Schedule a Tutor for a Course

Select your Tutor for CS 4400

First Name	Last Name	Email	Day	Time	Select
Matt	Schofield	MS2014@gatech.edu	M	11	☐
			W	11	☐
Mary	Smith	Msmith@gatech.edu	M	9	☐
			F	4	☑
Joe	Black	jb111@gatech.edu	F	4	☐

NOTE: Only 1 box under the Select column may be checked.

OK   Cancel

Figure 4 – Schedule a Tutor Screen

Tutor Evaluation by Student

Course	School	Number		Tutor Name	Matt Schofield
	CS	4400			

Descriptive Evaluation

Matt did a great job explaining the details of SQL programming.   It made it much easier for me to do the class project in cs 4400.

Numeric Evaluation

☑ 4 Highly recommend
☐ 3 Recommend
☐ 2 Recommend with reservations
☐ 1 Do Not Recommend

OK

Figure 5 – Rate a Tutor Screen

Tutor User

The Tutor user has 2 tasks that may be performed.   The first is to apply for a tutor position. This is shown in Figure 6.   As input, the prospective tutor would enter three types of information: basic student information, course information and availability information. The check box next to each course indicates whether the student had been a graduate TA for the course.   This applies only to graduate students.   Once the information is entered on the screen the user would select the **OK button**. The entered information would be inserted into the database and the user would be returned to the main menu.   If the **OK button** is selected without entering the required information then the user is simply returned to the main menu without inserting anything into the database.

The second task, a tutor may do is to retrieve his/her tutor schedule. This is shown in Figure 7.   Although a tutor may only see his/her own schedule, we still require the tutor to enter their Georgia Tech ID number.   The reason for this, is that in the future we plan to use this interface for administrators, so that they can look up specific tutor schedules. Selecting the associated **OK button** would execute the search in the database.   The result of the query would is displayed.   Selecting the second **OK button** returns the user to the main menu.

Professor User

The Professor user has a single task that may be performed.   The professor enters a recommendation for a single tutor. The necessary information is shown in Figure 8.   The professor selects the **OK button** once the information has been entered. This inserts the information into the database and then returns the user to the main menu.   If nothing is entered, selecting the **OK button** returns the user to the main menu without inserting anything into the database. Note that a professor may only enter a recommendation for a GT ID number of a registered tutor.

Georgia Tech Tutor Application

Student Information

Georgia tech ID	903997777		
First Name	Matt	Last Name	Schofield
Email	MS2014x@gatech.edu	Telephone	404-777-1111
GPA	3.9		☐ Undergraduate ☑ Graduate

Courses for Tutoring

School	Number	GTA
CS	4400	☑
CS	2200	☑
MATH	3012	☐
MUSI	4630	☐

Check the GTA box if you have been a graduate TA for the course

Available Days/Times

Monday
☐ 9am ☐ 10am ☑ 11am ☑ 12pm ☑ 1pm ☐ 2pm ☐ 3pm ☐ 4pm

Tuesday
☐ 9am ☐ 10am ☐ 11am ☐ 12pm ☐ 1pm ☐ 2pm ☐ 3pm ☐ 4pm

Wednesday
☐ 9am ☐ 10am ☑ 11am ☑ 12pm ☑ 1pm ☐ 2pm ☐ 3pm ☐ 4pm

Thursday
☑ 9am ☑ 10am ☑ 11am ☑ 12pm ☑ 1pm ☑ 2pm ☑ 3pm ☑ 4pm

Friday
☐ 9am ☐ 10am ☐ 11am ☐ 12pm ☑ 1pm ☑ 2pm ☑ 3pm ☑ 4pm

[OK]

Figure 6 – Apply for Tutor Position Screen

Find Tutor Schedule

Enter Tutor GTID   903997777   OK

Tutor Schedule for Matt Schofield

Day	Time	First Name	Last Name	Email	Course	
M	12	John	Towns	jb111@gatech.edu	cs 4400	
M	1	Jill	Green	jill2014@gatech.edu	math 3012	
Th	9	Dave	Brown	dbrown@gatech.edu	cs 4400	

OK

Figure 7 – Display Tutor Schedule Screen

Professor Recommendation

Student GTID   903997777

Descriptive Evaluation

Matt has a great grasp of the fundamentals of programming and database design. He has helped his fellow students on a number of occasions with program debugging.

Numeric Evaluation

☑ 4 Highly recommend
☐ 3 Recommend
☐ 2 Recommend with reservations
☐ 1 Do Not Recommend

OK

Figure 8 – Professor Recommendation Screen

Administrator User

The Administrator user has a two tasks that may be performed. These two tasks produce summary reports about the tutoring activity per course for a particular semester or combination of semesters for the current academic year. Both reports require the administrator to select the semesters of interest for the current academic year. The output of both reports is ordered by course and the calendar ordering of semester (ie., fall, spring, summer). In Figure 9, we show the first summary report. It lists, for each course, the number of students that used tutors and the number of tutors that met with those students. Selecting the first **OK button** triggers the query on the selected semesters. Selecting the second **OK button** returns the user to the main menu. In Figure 10, we show the second summary report. It lists, for each course, the number of graduate student tutors who were/were not previous graduate TAs for each course and their average student ratings. Selecting the first **OK button** triggers the query on the selected semesters. Selecting the second **OK button** returns the user to the main menu.

List Courses with Student/Tutor Summary Data for current Academic Year

Academic Year 2014 ☑ Fall   ☑ Spring   ☑ Summer   [OK]

Course	Semester	# Students	# Tutors
cs 4400	fall	5	2
	spring	3	1
	summer	10	3
	Total	18	6
math 3012	fall	12	4
	spring	10	4
	Total	22	8
phys 2211	fall	25	8
	spring	20	8
	summer	12	5
	Total	57	21
	Grand Total	97	35

[OK]

Figure 9 – Administrator Summary Report #1 Screen

Tutor Summary Data for Grad TAs /nonTAs current Academic Year

Academic Year 2014 ☑ Fall   ☑ Spring   ☐ Summer   [OK]

Course	Semester	TA	Avg Rating	non TA	Avg Rating
cs 4400	fall	1	4	2	3
	spring	1	4	3	3
	Avg		4		3
math 3012	fall	3	3.3	4	3.5
	spring	3	3	8	3
	Avg		3.15		3.1
phys 2211	fall	5	3.6	2	3.5
	spring	4	3.5	2	3
	Avg		3.55		3.22

[OK]

Figure 10 – Administrator Summary Report #2 Screen

Appendix A – Sample Information Flow Diagram

Registration Form

Log In Form

New Item Form

Search Criteria

Register

List Item

Log In

User Report

Search Results

View User Report

Search Items

Category Report

View Category Report

Database

View Item

Calculate Winner

Item Description

Edit Item Description

View Auction Results

Bid/Get It Now

Auction Results

Delete Rating

Add Rating

View Ratings

Ratings
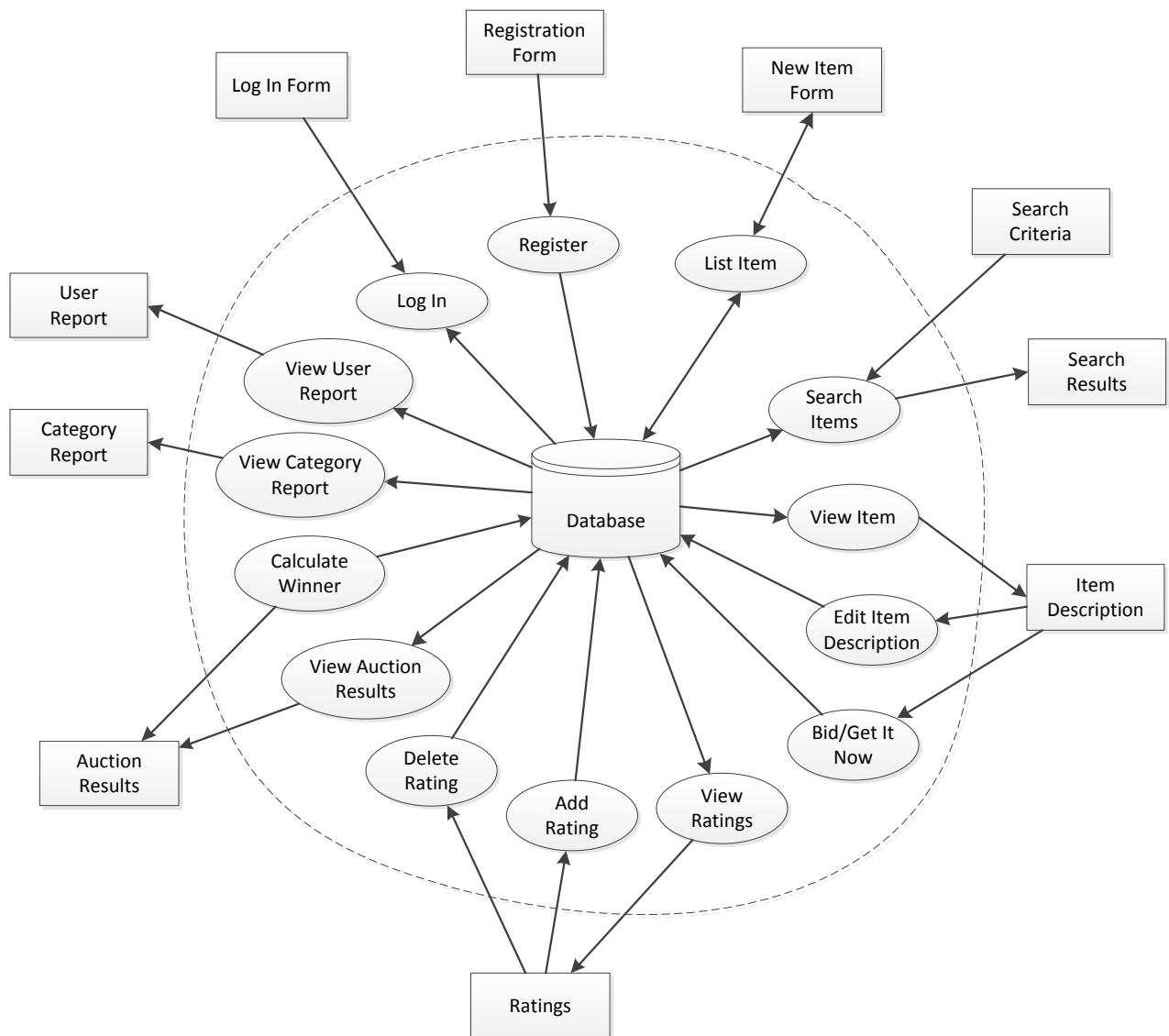
Appendix B – Sample EER Diagram
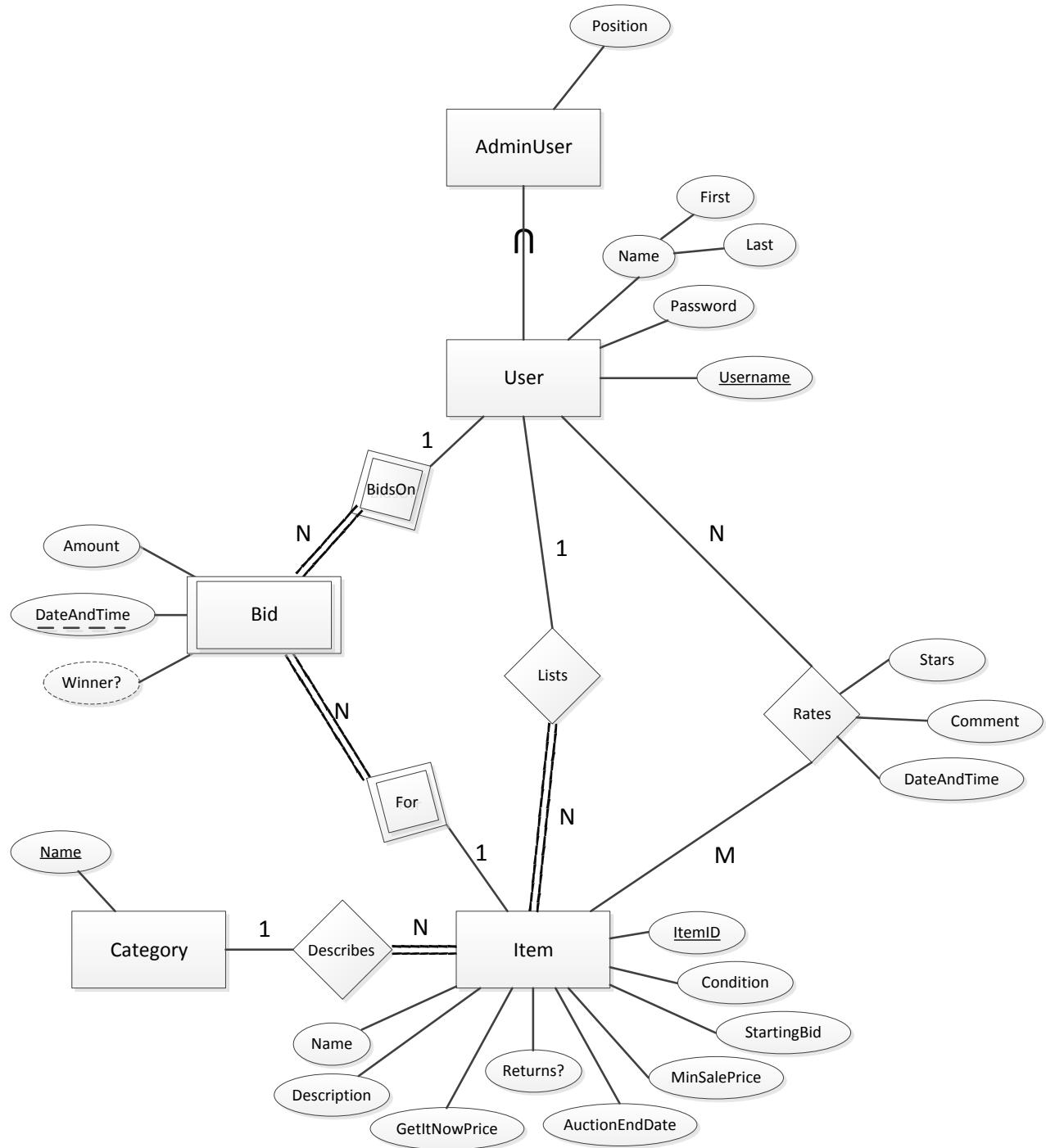
Appendix C – Sample Constraints

No.	Entity Types, Relationship Types, and Attributes involved	Constraint Definition[1]
1	Item.StartingBid[2], Item.GetItNowPrice	Item.StartingBid < Item.GetItNowPrice
2	Lists, User, Item, BidsOn, Bid, For	The user who lists an item cannot also bid on the same item.
3	Bid.Amount, Bid.DateAndTime	For any two Bids B1 and B2 on the same item, if B1.DateAndTime > B2.DateAndTime then B1.Amount > B2.Amount (Later bids must be for a greater amount than prior bids.)
4	Rates.Stars	1 <= Rates.Stars <= 5
5	Lists, User, Rates, Item	A user cannot rate an item that he or she lists.

---

[1] Constraint definition may be described in plain English or written using basic logic and mathematical operators.

[2] The notation XXX.YYY means that YYY is an Attribute of the Entity Type XXX or Relationship Type XXX.

Appendix D – Sample Relational Model Diagram

User

Username	Password	FirstName	LastName

AdminUser

Username	Position

Item

ItemID	Username	Name	Description	Condition	StartingBid	MinSalePrice	GetItNowPrice	AuctionEndDate	Returns?	CatName

Category

CategoryName

Bid

Username	ItemID	DateAndTime	Amount

Rating

Username	ItemID	DateAndTime	Comment	Stars
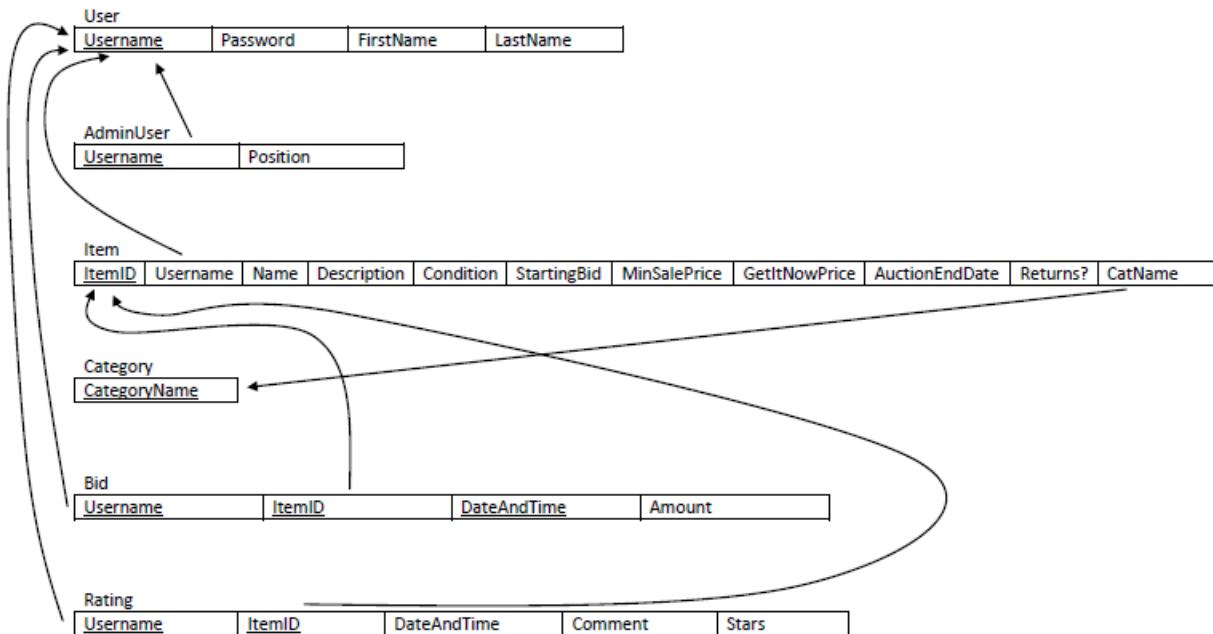
Appendix E – Sample CREATE TABLE Statements

```
CREATE TABLE User (
      Username VARCHAR(30) PRIMARY KEY,
      Password VARCHAR(30) NOT NULL,
      FirstName VARCHAR(50) NOT NULL,
      LastName VARCHAR(50) NOT NULL
)

CREATE TABLE AdminUser (
      Username VARCHAR(30) PRIMARY KEY,
      Position VARCHAR(50) NOT NULL,
      FOREIGN KEY (Username) REFERENCES User(Username)
)

CREATE TABLE Category (
      CategoryName VARCHAR(30) PRIMARY KEY
)

CREATE TABLE Item (
      ItemID INT AUTO_INCREMENT PRIMARY KEY,
      Username VARCHAR(30) NOT NULL,
      Name VARCHAR(250) NOT NULL,
      Description VARCHAR(4000) NOT NULL,
      Condition INT NOT NULL,
      StartingBid DECIMAL(10,2) NOT NULL,
      MinSalePrice DECIMAL(10,2) NOT NULL,
      GetItNowPrice DECIMAL(10,2),
      AuctionEndDate DATETIME NOT NULL,
      Returns? BIT NOT NULL,
      CatName VARCHAR(30) NOT NULL,
      FOREIGN KEY (Username) REFERENCES User(Username),
      FOREIGN KEY (CatName) REFERENCES Category(CategoryName)
)

CREATE TABLE Bid (
      Username VARCHAR(30) NOT NULL,
      ItemID INT NOT NULL,
      DateAndTime DATETIME NOT NULL,
      Amount DECIMAL(10,2) NOT NULL,
      PRIMARY KEY (Username, ItemID, DateAndTime),
      FOREIGN KEY Username REFERENCES User(Username),
      FOREIGN KEY ItemID REFERENCES Item(ItemID)
)

CREATE TABLE Rating (
      Username VARCHAR(30) NOT NULL,
      ItemID INT NOT NULL,
      DateAndTime DATETIME NOT NULL,
      Comment VARCHAR(4000) NOT NULL,
      Stars INT NOT NULL,
      PRIMARY KEY (Username, ItemID),
      FOREIGN KEY Username REFERENCES User(Username),
      FOREIGN KEY ItemID REFERENCES Item(ItemID)
)
```

Appendix F – Example Abstract Code + SQL

Task: List New Item

```
//show category listing on new item form
SELECT CategoryName FROM Category ORDER BY CategoryName

//read in parameters from form
$ItemName = read("ItemName")
$ItemDescription = read("ItemDescription")
$Category = read("Category")
$Condition = read("Condition")
$StartingBid = read("StartingBid")
$MinSalePrice = read("MinSalePrice")
$AuctionDuration = read("AuctionDuration")
$GetItNowPrice = read("GetItNowPrice")
$Returns = read("Returns?")

//validate form data
if ($StartingBid > $GetItNowPrice) then
      error("Starting Bid must be lower than Get It Now price.")
      return to form
end if

if ($MinSalePrice > $GetItNowPrice) then
      error("Minimum Sale Price must be lower than Get It Now price.")
      return to form
end if

//calculate end date
$AuctionEndDate = Now() + $AuctionDuration

//insert new item
INSERT INTO Item (Username, Name, Description, Condition, StartingBid,
      MinSalePrice, GetItNowPrice, AuctionEndDate, Returns?, CatName)
VALUES ($Username, $ItemName, $ItemDescription, $Condition,
            $StartingBid, $MinSalePrice, $GetItNowPrice, $AuctionEndDate,
            $Returns, $Category)
```

Document Version Info

Version	Notes	Date
1.0	Original version	5/19/2014