

GTMovies

CS 4400 Database Project

Summer 2005

Project Overview

In this project you will design and implement a database application using the Classical Methodology for Database Development. The design methodology will be discussed in class and is described in detail by the following document.

Classical Design Methodology:

http://www.cc.gatech.edu/classes/AY2005/cs4400_spring/methodologyFall2002.pdf

Groups

The project is designed to be completed by groups with 3 or 4 members. However, a group may choose to remove a member from further participation in the group after the phase I or phase II due dates. In this case, written notification must be provided to the professor explaining the situation.

As a group, you will decide whether to complete the lightweight or heavyweight project options. The two options are identical for phases I and II, but differ in the deliverable for phase III:

Heavyweight Option

Groups choosing this option will demo a working implementation of their project to the TA. The implementation must include a Java or web-based GUI (Graphical User Interface) that uses JDBC (Java Database Connectivity) or ODBC (Open Database Connectivity) for database access. The SQL statements you create in phase II will be embedded inside your GUI.

Lightweight Option

Groups choosing the lightweight option will submit working SQL statements for each of project tasks and demo the SQL statements to the TA. This option may be appealing to groups with little or no experience programming GUIs. A complete listing of the required SQL statements for the lightweight option will be posted after the phase II due date.

Oracle

We will provide you with access to the Oracle Database Management System on ACME. See the course webpage for further information on how to access Oracle from the ACME

command line or from a Java program. See the FAQ at the end of this document for questions about using a different DBMS for your implementation.

Project Schedule

<u>Phase</u>	<u>Due Date</u>
I – Analysis, Specification, and Psuedocode	June 15
II – Design and SQL Code	July 6
III – Implementation	July 27
Demo	July 27-29

Grading

Members of groups choosing the Heavyweight project option will *not* be allowed to take the final. Members of groups choosing the Lightweight project option will be required to take the final.

<u>Phase</u>	<u>Percent of Final Grade</u>
I – Analysis, Specification, and Psuedocode	10%
II – Design and SQL Code	10%
III – Implementation (<i>Heavyweight Option</i>)	20%
III – Implementation (<i>Lightweight Option</i>)	5%

Project Deliverables

Hard copies of the reports for each phase should be delivered in class on the given due dates. Please be sure to list the names and email addresses of all group members on the report.

Phase I

The Phase I report has the following deliverables. See the indicated page in the design methodology document for an example of each deliverable.

- A brief description of the purpose of this phase
- A brief description of technical problems and/or design decisions encountered during this phase and how the group resolved the problems
- Information Flow Diagram (p 18)
- ER Diagram (p 28)
- Data Representation (p 29)
- Task Decomposition Diagram (p 33)
- Task Forms (p 36)

Phase II

The Phase II report has the following deliverables.

- Your phase I report with TA comments
- A brief description of the purpose of this phase
- A brief description of technical problems and/or design decisions encountered during this phase and how the group resolved the problems
- A brief description of any revisions made to the phase I specification
- Relational Schema including primary and foreign keys (p 48)
- Create Table statements, including domain constraints, integrity constraints, primary keys, and foreign keys
- Updated task forms with SQL (pp 52-54)

Phase III

The Phase III report has the following deliverables.

- Your phase I and phase II reports with TA comments
- A brief description of the purpose of this phase
- A brief description of technical problems and/or design decisions encountered during this phase and how the group resolved the problems
- An updated set of Create Table statements, including justification for any revisions made since phase II
- A brief explanation of which columns you would index in your database
- A set of working SQL statements for all project tasks (*Lightweight Option*)
- A functional GUI with embedded SQL statements that accesses your database (*Heavyweight Option*)
- A listing of the contributions of each group member for all phases of the project

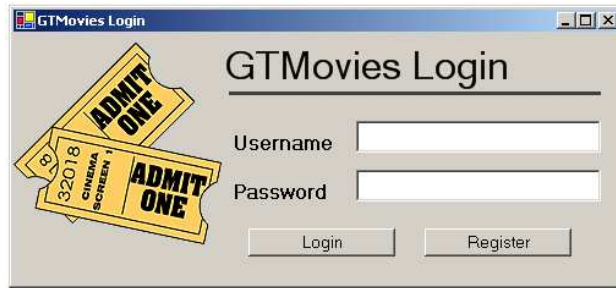
Project Description

The movie theater at the Georgia Tech Student Center has recently become so successful that the administration has decided to allow students to reserve their tickets online. The application will allow students to search for upcoming movies and then reserve tickets to be picked up at the theater. Movies are free to all students, so the system will simply be used for reserving tickets, not for taking payments.

The system supports two types of users: students and managers. Students may log in, search for movies, see movie details, reserve tickets, and see which tickets they have reserved. Managers, on the other hand, will set up the movies along with the show dates and show times. Managers will also generate a report showing which students have reserved tickets. Both managers and students will have a username and password. The password must be at least six characters long. **The Student Center would also like to track the Hire Date of each manager. There will be no interface for entering the Hire Date, but it should still be tracked by the database.**

Login Interface

Both managers and students will use the same form for logging in to the system. Depending on the type of user, the user will be taken to either the student menu or manager menu after logging in successfully. If the log in fails, display an appropriate error message and return to the log in screen.



Add New User

If the user clicks “Register” on the log in screen, he or she will be presented with a form for registering with the system. The fields required for registering are the student’s name, class (undergraduate or graduate), username, and password. The system should not allow a user to enter a username that already exists in the system.

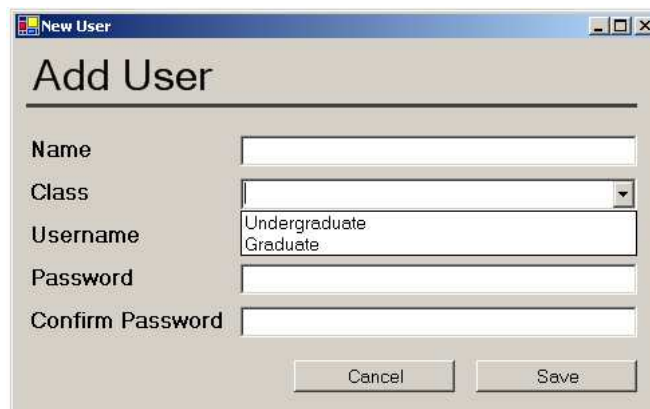


Figure 2 –Add User

Student Interface

When a student logs in, he or she will be presented with a menu with three options: search for movies, view reserved tickets, and exit the system. Clicking exit will return the user back to the log in screen.

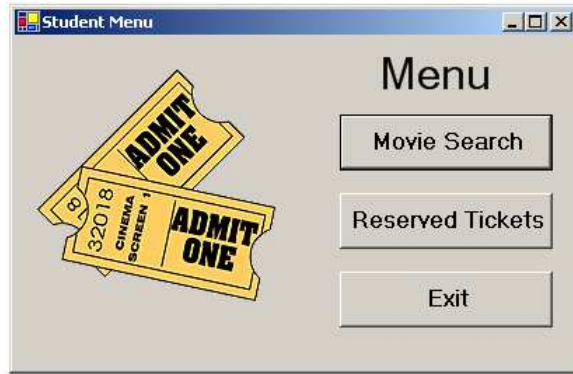


Figure 3 – Student Menu

If the student selects the “Movie Search” option, the search form will be displayed allowing the user to search for movies playing at the Student Center theater. The four search criteria are title, actor, rating, and genre. The title field allows students to search for a movie title containing the given text. For example, if the student enters “Office” then movies with the word “Office” in the title should be listed in the search results. Similarly, the actor field will search for movies with the given actor, even if only a partial name is given.

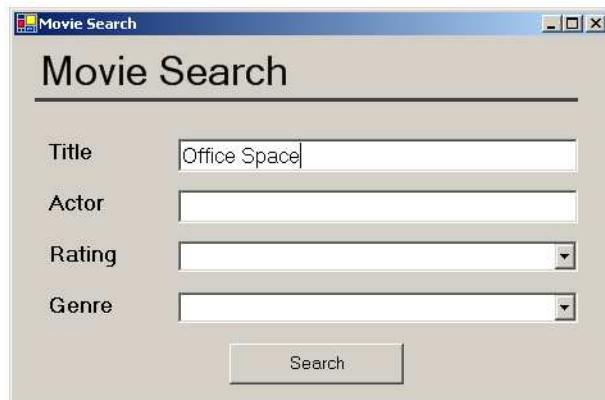


Figure 4 – Movie Search

The rating field is a drop down listing the standard MPAA movie ratings.



The screenshot shows a web browser window titled "Movie Search". The page has a header "Movie Search" and a horizontal line below it. There are four input fields: "Title", "Actor", "Rating", and "Genre". The "Rating" field is a dropdown menu that is currently open, displaying a list of MPAA ratings: "G", "PG", "PG-13", and "R". The "PG-13" option is highlighted in blue. Below the input fields is a "Search" button.

Figure 5 – Ratings

The genre field is a drop down listing the movie genres currently available in the system.



The screenshot shows the same "Movie Search" web browser window. The "Genre" field is now the dropdown menu that is open, displaying a list of movie genres: "Action", "Comedy", "Drama", "Family", and "Romance". The "Action" option is highlighted in blue. The other input fields and the "Search" button are visible but not the focus of this view.

Figure 6 – Genres

When the user clicks “Search” a results form is shown containing all of the upcoming movies that match the given criteria. All fields in the form should be ANDed together such that if a user enters multiple criteria, the resulting movies should agree with all of the entered criteria. The results form should only show current and future movies. Any movies that do not have a current or future show date should not be listed in the results so that students cannot reserve tickets for shows in the past.



Figure 7 – Search Results

The Search Results form lists the title, rating, genre, show dates, and show time of all current and future movies that match the given criteria. Because there is only a single theater in the Student Center, we assume that each movie will only be shown at one time during the day. We also assume that the show time will be the same for all dates that the movie is shown. If the user does not enter any criteria, then all current and future movies should be displayed. If no movies match the given criteria, then a message should be displayed stating that no movies match the search criteria.



Figure 8 – Movie Details

From the Search Results form, the user may choose to either view movie details or reserve tickets. A radio button appears beside the title of each movie. The user must select a particular movie before viewing movie details or reserving tickets. If the user selects a movie and clicks “View Movie Details” then a form is displayed containing details about the selected movie. The details form includes the movie title, genre, rating, a summary of the plot, a list of cast members, show dates, and the show time. After viewing the details, the user should be returned back to the search results page.

A user may also reserve tickets from the search results page. A reserve tickets form should be displayed allowing the user to select the show date and number of tickets to reserve. The name of the currently logged in student should automatically be displayed in the form. **There is also a dropdown listing the dates available for reservation, along with the number of remaining tickets. If there are no seats available for a particular date, then do not list the date as an option.** The reserve tickets form should validate all data before allowing the user to submit the reservation.

Reserve Tickets

Title Office Space
Genre Comedy
Rating R
Show Dates 5/15 - 5/20
Show Time 2:30

Reserve Tickets

Name John Doe
Select Show Date
Number of Tickets

Cancel Reserve Tickets

Figure 9a – Reserve Tickets

Select Show Date

Number of Tickets

5/15 (20 remaining)
5/16 (25 remaining)
5/17 (10 remaining)
5/18 (2 remaining)
5/19 (8 remaining)
5/20 (1 remaining)

Cancel

Figure 9b – Select Show Date Dropdown

The student's class standing determines the total number of tickets allowed per show. Undergraduates are limited to a maximum of 4 tickets per show. Graduate students are limited to a maximum of 6 tickets per show. Therefore, even if a student makes multiple reservations for the same show, the system should not allow students to reserve more tickets than they are allowed. **Currently, there is a single theater in the student center, and it has 50 seats. Therefore, the maximum number of seats that may be reserved for a single show is 50. However, the Student Center is considering expanding and adding additional theaters in the future. Therefore, you should not hardcode the maximum number of seats, instead store it in the database so that it may be changed easily. The "seats available" attribute should be specific to each theater so that if additional theaters are added later, they may have a different number of seats.** Reserved tickets are final, so the system does not have any facility for changing reserved tickets. After completing the reservation, the user should be brought back to the student menu.

The student may also review tickets that he or she has already reserved. When the user clicks "Reserved Tickets" from the student menu, a form should be presented that lists all reserved tickets, including those in the past. The history screen should show each reservation on a separate line, including the title of the movie, the show date, the show time, and the number of tickets reserved. If no reservations exist in the system for the logged in user, simply display an appropriate message.



The screenshot shows a window titled "Order History" with a table of reserved tickets. The table has four columns: Title, Show Date, Show Time, and Tickets. The data rows are:

Title	Show Date	Show Time	Tickets
Office Space	5/15	2:30	2
Office Space	5/16	2:30	2
Meet the Fockers	5/18	4:30	4

Figure 10 – Reserved Tickets

Manager Interface

If a manager logs into the system, he or she is presented with four options: Add Movie, View Reservations, View Student Report, and Exit. Clicking Exit will return the manager back to the log in screen.



Figure 11 – Manager Menu

The Add Movie option will allow the manager to enter a new movie into the system. The required fields are title, rating, start date, end date, and show time. Movie titles are not necessarily unique, so multiple movies with the same title may be entered into the system.

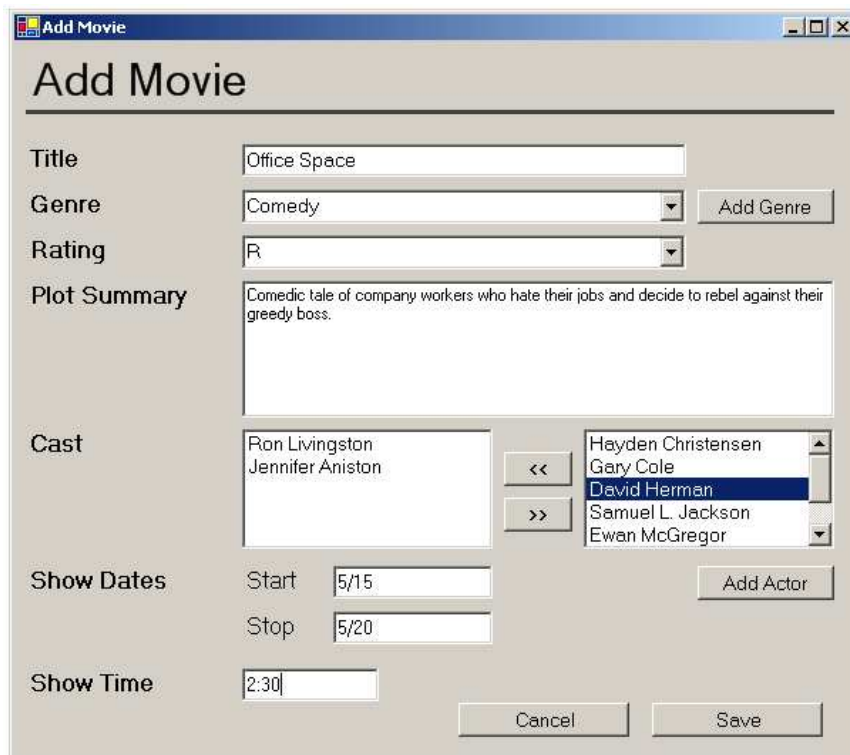


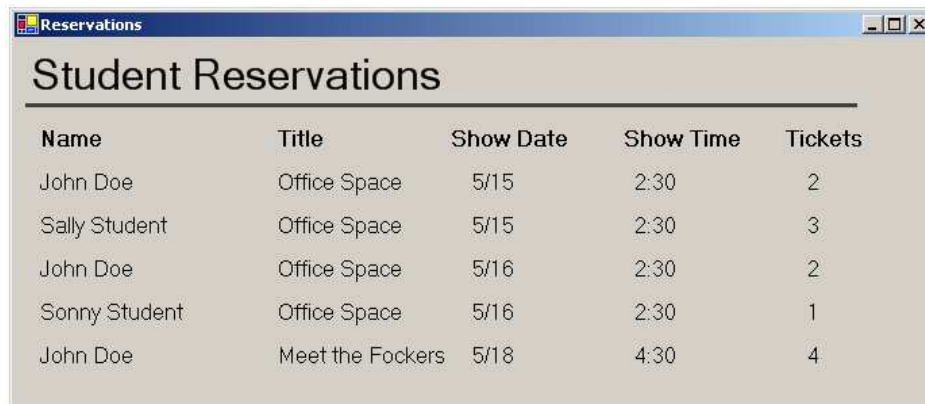
Figure 12 – Add Movie

The genre dropdown lists the current genres that have been entered into the system. If the needed genre does not yet exist, then the manager can click “Add Genre” to add a new value. Once the new genre has been entered, it should be available for selection when the next movie is added. For this system, assume that all genres are unique. If the manager

attempts to add a duplicate genre, display an appropriate error message and do not add the new genre.

The options in the rating dropdown are fixed and will not change during the life of the system. The ratings are the MPAA standard G, PG, PG-13, and R. The Student Center will likely not be showing NC-17 films, so that rating is not required.

The system must also support a way to assign cast members to the movie. A movie can have any number of cast members, including zero if the manager decides not to enter any. In the screenshot above, all current actors are listed in the box on the right, and they are sorted alphabetically by last name. The manager may then move actors between the two boxes using the buttons marked “<<” and “>>.” Also, the manager has the ability to add new actors to the system. As in the genre case, an actor only has to be added to the system one time.



The screenshot shows a window titled "Reservations" with a sub-header "Student Reservations". Below the header is a table with five columns: Name, Title, Show Date, Show Time, and Tickets. The table contains five rows of reservation data.

Name	Title	Show Date	Show Time	Tickets
John Doe	Office Space	5/15	2:30	2
Sally Student	Office Space	5/15	2:30	3
John Doe	Office Space	5/16	2:30	2
Sonny Student	Office Space	5/16	2:30	1
John Doe	Meet the Fockers	5/18	4:30	4

Figure 13 – View Reservations

The “View Reservations” option allows the manager to view the current reservation list. The reservation list should contain one row per student reservation and include the student’s name, the movie title, show date, show time, and number of tickets reserved. The list should be sorted first by date and then by time so that the next showing appears at the top of the list. Only current and future reservations should be listed.



The screenshot shows a window titled "Student Report" with a sub-header "Student Report". Below the header is a table with three columns: Name, Tickets This Month, and Total Tickets. The table contains three rows of summary data for students.

Name	Tickets This Month	Total Tickets
John Doe	4	22
Sally Student	8	16
Sonny Student	2	2

Figure 14 – Student Report

The Student Center also likes to track the number of reservations per student so they can provide promotions to the students who have reserved the most tickets. The manager has the ability to run a student report listing all student currently in the system along with the number of tickets reserved this month, and the total number of tickets reserved for the life of the system. The report should be sorted by the total number of tickets reserved in descending order.

Assumptions

The Student Center has provided the following system usage predictions based on the current ticket sales and student interest. You may use these assumptions to make educated guesses for the “frequency” section of each task form.

- Total number of student users: 4000
- Total number of managers: 5
- Average number of student logins per day: 100
- Average number of new users per month: 30
- Number of new movies added per month: 8
- Number of theaters: 1 with 50 seats (but may be changed in the future)

Frequently Asked Questions

A number of similar questions come up from semester to semester. Before you ask your TA or post a message to the newsgroup, check here to see if we have already anticipated your question.

1. Do we have to use ORACLE?

You are not required to use the ORACLE installation on ACME, although that is the only DBMS that we will support or be able to answer technical questions about. If you are comfortable using another DBMS (commercial or open source), then we will most likely allow you to use it. However, you must first confirm with your TA to make sure the particular product you have in mind is acceptable. Obviously, the DBMS must support standard SQL. Examples of DBMSs used in the past are MySQL, PostgreSQL, and SQL Server.

2. Our group would rather do a web-based GUI instead of a regular Java GUI. Is this acceptable?

A web-based GUI is acceptable. Many server-side scripting languages (such as JSP, PHP, or ASP) allow database connectivity and embedded SQL. If you decide to go this route, please notify your TA in advance to confirm that the platform you are using is acceptable.

3. Does our GUI have to look exactly like the one in the project description?

No. In fact, we encourage you to make any improvements that you would like. The only requirement is that you support all of the same functionality as stated in the description. Feel free to add additional functionality or improve the user interface in any way you see fit.

4. How will the demo work?

If you are doing the heavyweight project, then you will start up your demo either in the undergraduate computing cluster (i.e. States cluster) or on a personal laptop. If your demo requires an extensive setup, we would suggest you use a laptop that you have already configured.

If you are doing the lightweight option, then we will watch you execute all of your SQL commands using the SQLPLUS interface to ORACLE. You will start with an empty database and then run your CREATE TABLE statements. It is also suggested that you have a script to populate the database with test data.

5. For the task decomposition diagram, do all tasks need subtasks?

No. It is possible (likely, actually) that some of your tasks will not have any subtasks. In this case, just list the task by itself in the diagram without any child tasks. Do not attempt to link all tasks together by some global decomposition. That often results in a number of empty task forms that have no purpose but to group subtasks.

6. What goes in the *operation* section of the task forms?

In this section you should put pseudocode detailing the function of the task form. In particular, you should describe both program control (such as showing a menu or moving to another screen) and data manipulation (such as inserting new records, updating records, or deleting records).

There is no standard pseudocode language expected just as long as the logic is clear. However here are two suggestions for making your lives (and ours) a little easier.

- Choose a common representation for variables and use it throughout. For example you could start all variables with a colon (e.g., :username).
- Stick with typical programming language constructs, such as if statements and loops. Avoid using lengthy English prose to describe the logic of the task.

7. Just how detailed should the operation section be?

You want to shoot for *attribute-level detail*. In other words, you should describe what is happening with each attribute relevant to the task. Do not simply list the entities involved, but also state what is happening with each attribute.

Bad:

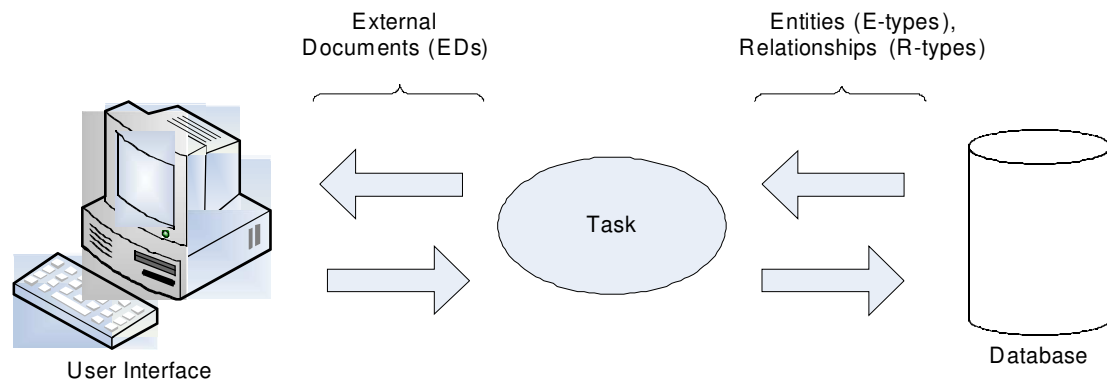
IF correct USER exists

Good:

IF USER exists SUCH THAT username = :username and password = :password

8. How do the task inputs and outputs work?

Think of each task as sitting in between the user interface and the database. Here's a diagram to help you visualize:



Inputs to the task come from either the database or the user. If the input is from the database, it is either an entity or relationship and it is being *read* by the task. Any data required by the task that is already in the database should be an input to the task. Inputs that come from the user are in the form of an external document. These inputs are used any time a user enters information into the system that is utilized by the task.

Likewise, outputs may go back to the user or into the database. Outputs going to the database are in the form of entities or relationships. Any time a task *changes the state* of the database, the changed entities or relationships should be listed as outputs. Information displayed back to the user is also an output, but in the form of an external document.

9. For phase III, do we have to implement the indexes?

No. Your phase III report should just include a description of which columns you would index and why. Note that primary key and foreign key columns are

automatically indexed by the database, so there is no need to mention them again. Instead, focus on other columns that may benefit from having an index.