Video Rental Software
Design Document

Name: Huming Tang
ID: 101281618
EMAIL: httang@ic.sunysb.edu
Date: 3/21/2003
Type: CSE308 technical paper

TO FULFILL THE WRITING AND ORAL REQUIREMENT FOR
COMPUTER SCIENCE
Table of Content

I. Introduction .................................................................................................................. 2

II. Product Features Design ............................................................................................. 3-4

III. Software Structure and Internal Component Design ............................................. 5-8
    A. Summary of software’s general architecture ....................................................... 5
    A.1 Database vs. Local File Storage ................................................................. 5-6
    B. Internal/Transactional Components Design ..................................................... 6-8

IV. Testing Procedures Design ....................................................................................... 9-10

V. Appendix .................................................................................................................... 10-23
    A. User Interface Design Detail .......................................................................... 11-17
        A.1 System Login Window ........................................................................... 11
        A.2 Main Menu Frame .................................................................................. 12
        A.3 Login Panel .............................................................................................. 12
        A.4 Employee Panel ....................................................................................... 13
        A.5 Member Panel ........................................................................................... 14
        A.6 Inventory Panel ....................................................................................... 15
        A.7 Transaction Panel .................................................................................... 16
        A.8 Search Panel ............................................................................................. 17
    B. System Behavior ................................................................................................. 18-23
        B.1 Object-relational model of server-client in Collaboration Diagram .......... 18-19
            collaboration diagram ................................................................................. 19
        B.2 Activity Diagram of a complete transaction (rental process) ................... 20
            activity diagram .......................................................................................... 20
        B.3 State Diagram of all possible states of a rental item ............................... 21-22
            state diagram .............................................................................................. 23
I. Introduction

This is the Design Document of the Video Rental software. All design information of this software’s functionalities and implementation will be explained in detail. This document also covers the design of the Graphical User Interface and the design of the testing procedures.

Before going into the design details, let’s have a brief description of the Video Rental software:

This software is a visual application that automates the processes in a video store. It basically performs essential functions such as renting items for the users and stores the data in a device for future access. And of course everything will be presented in a user friend graphical environment.

A centralized database is used to store data such as customer and employee information. In our case the Sybase SQL-92 server will be the storage device. We choose the Sybase server because it can provide a reliable and secure access to the stored data. The issue over whether using local file or a remote server will be debut in this document.

Other than storing common data that are necessary for a rental transaction, our program also helps to manage the store itself. It can be used to access the information of the employee and customer so the manager can search for an employee quickly. It has a report generator to produce summary about the store’s transaction that helps the manager to decide how to improve the store’s efficiency and profits (see what type of item is the most popular and the amount of profits made so far). It has a security scheme that helps the employer to limit the access to the store information.

The most desirable feature of our software is that it supports multiple stores. Our software can be used simultaneously within multiple store and access the store information at the same time. The design of this function will be explained in detail in this document.

In general, this application is a general-purposed Video Rental Management software.
II. Product Features Design

This Video Rental Software is basically a terminal that provides different functions and features that will be necessary for running a video store. During the design process, the features and functions are categorized into two groups: the user-level and the developer-level. The developer-features are features that a developer/debugger can use to change and update the software.

The user-level features are the set of functions of the program that the user has visual interactions with. The first and most important function of our software is process a rental transaction. This function allows the user to rent items and return item. Since this is a member-only video store, only registered member can have their rental processed. When this function is activated, the system will first check for the availability of the item, if the item doesn’t exist, the item cannot be rented and the user will be informed. The system then will check the renter’s information; if the renter doesn’t exist or if he is not allowed to rent due to limitations, the system will inform that the item cannot be rented. If all checks are passed, then the item will be processed. The second function is to store/update employee information. This function will store the registering person’s name, address, phone, rental information (customer only), and working information (employee only). This function is essential for the rental process to work. The third function is store/update inventory information. This function is necessary for getting new items into the store and remove items that are lost or damaged. The fourth function is the search engine for customer/employee/items. This is a very powerful function in a sense that allow the user to find what he needs fast. Since this is multi-store software, the search function will allow the user to obtain information on items in other stores (the customer may want to know if an item is available is another location). However, this function’s ability to search employee information is limited to high-level user only since those are private information. The fifth user-level feature is the multi-store capability of the software. This feature is allows the software to be used to manage multiple stores that meant they share the same server and access the same sets of data. This feature combines the information of all stores then groups them for easy access. The last user-level feature is the security management. This feature is almost invisible to the users but the user is directly interacting with it. This feature will divide the user into three different types. Each type will have a certain amount of access rights that limit the user’s available functions. For example, the top store manager will have the highest access right; he has all the functions in the store. The counter-person (employee) will have the lowest access right, he can only process rentals and view his personal information. These are the user-level features.

The developer-level features are features that benefit the user/developer during the pre and post development of the software. They are also keys to the software’s value and efficiency. They are not visible to the users at all. A very simple but useful developer-level feature is visual error handling. This means that every time the software encounters an error or exception, it will display the error message to the users but it doesn’t terminate the software. However process that causes the error will be cancelled. This feature prevents the software crashing from an unexpected error.
Another developer-level feature is the utilization of a database server instead creating a server-side application. The feature will simplify the organization of the data and will help to take care of the thread-safe problems that created by the multi-store feature.

Another developer-level feature to be implemented is a small set of functions that allows the user to manage the store more efficiently. This set of functions includes a simple report generator that generates summary on the employee/customer/inventory of a specific store or all stores and a command-line prompt that allows the user to enter view-only SQL command that list Data in the command-line prompt. The last feature for the developer is the structure design of this software. We use object-oriented/modular as the goal for the architecture of our software. The software will have a component design scheme. Each component will be designed to have minimal interaction with each other so errors can be reduced and isolated quickly. This design also allows other programmers to create additional functions in this software easily by adding extension objects without changing the core objects. The details of each object will be explained in the software component design section.

So if we summarize the features, these are the features our software provides:

- add/update employee/customer/inventory information (user level)
- processing rental transaction (user level)
- search engine (user level)
- multi-store capability (user level)
- security (user level)
- error handling/admin debug (developer level)
- object oriented design (developer level)
III. Software Structure and Internal Component Design

A. Summary of software’s general architecture

This software is a stand along program; it is implemented as a client-side application only. This software will require a designated database server to operate. This software is designed to be efficient, error-free, and user-friendly.

This software is also designed to achieve the goal of Modularity. In order to support object-oriented design, JAVA (an object-oriented language) is used as our core language; and to ensure the independency of the objects, each object is carefully designed to support a group of data that is only relevant to the object that carries them. To extend the independency of our objects, we also divide the software components into 3 model groups (transaction, GUI, internal). This way, the actions of each group are completely isolated.

In general, the individual object for each independent component are: an object for Inventory Items, an object for customers of the store, an object for employee of the store, and an object for GUI of the application.

The 3-model groups construction is basically like this: We have taken a layered approach towards building the connection from the client side to the server. We will build a core model (inventory items, etc) and put them on top of a connection model (used for communication between server and client) then we wrap them all with the GUI. The advantage of this is that we can debug the software quickly by isolating the problems to their specific area: whereas they are the GUI part, the connection part, or the core model part. Another advantage is that we can add additional functions by changing or adding individual models without changing other models so other programmers don’t needed to rewrite a lot of objects to achieve the goal. For example: if the customer wants us to change the GUI of the application, we only have to make little changes to the GUI model of our software.

For the server side, only SQL-tables and relations are defined and implemented since a JAVA library is available for accessing the data structures in the server. The software is mostly the client side application.

A.1 Database vs. Local File Storage

There are several reasons for why have we decided to use database server instead of creating a local file system to store the data. The most important factor is that database server provides us a safe and secure environment of accessing data. With the Database Server, we are able to block most illegal access and change to our data since we do not have any data remain in local physical device; although a file system may be protected through data encryption, the fact that a person can get his hand on the actual physical data locally is dangerous. With a database server, we were able to share the information of all stores so they can all be easily accessed through any terminal in any store quickly. If a
local file system is used, we need to build communication between all stores that are prone to errors; this can also be very inefficient since terminals need to create connections to multiple store to get the information in every store. The only disadvantage of the database server implementation is that if the server ever crashes, all store terminals will not be able to function. This is a very serious problem. However this may be resolved by combining database server with local file system. But due to time limit, this implementation is not possible.

B. Design of Client-Side Components and Sever Side Database

This software is composed of 2 major components, the server and the client. The client will be an application build by us, and the server is a pre-defined SQL database. Let us first go through the details of the client side application.

As described in the summary section, this software is consisted of a bunch of objects on the client side. In order to achieve the goal of modularity, we designed the objects according to three models: GUI, core objects, and connectors. Each model’s objects will have most of their interaction within the model group and only 1 or 2 objects will have to access other models.

The core object model contains 7 objects. They are: login, inventory data, employee data, customer data, transaction, search, and error. The login object is used to perform login checks; it connects to the database to check for the existence of the user. If exist then start the software, else an error message will be given. Its instance will contain the login information of the user for later access. The inventory data object is used to store the information of an item in store. The inventory object has an item’s name, type, producer, renter (if any), availability, and date of last rental/return. This object is also capable of verifying the stored information and accessing the database (for adding/updating items). The employee data object is used to temporarily store the information of an employee. This object contains the employee’s name, address, phone number, position, and access level. It also contains get and set methods for accessing the variable values, a method to verify the information, a method to receive information from database, and a method to send the data to the database. The customer data object is very similar to the employee data object except instead of having position and access level stored, it will store rental information: rental limit, borrowed item, disabled or not, and outstanding balance. The purpose of employee and customer object is to separate the data from the application so if changes were to be made, these objects are not affected. The transaction object, like its name, is used to perform rental processes. It utilizes the customer data object to verify the information of the renter and if it’s passed, it also uses the inventory data and customer data objects to update the items’ and customer’s information in the database. It contains several methods to handle exceptions and calculation of the final cost. The search object is a completely different from the previous objects. It doesn’t carry any data itself but will create instances of the customer/employee/inventory objects to pass the information found in a search. The search object contains three methods for searching the database: search employee, search customer, and search inventory. Each method will return the corresponding object that
contains the information about them. The last core object is the error object; this object is used to store the error messages incurred during the usage of the software. It's used to pass the error messages of the core objects to the GUI error display window.

The connector model has only one object: Database object. The Database object contains all the methods necessary to communicate with the Sybase SQL-92 database. There are a total of four methods: open connection, end connection, and query, and process result set. The open connection method is used to connect to the Database. The end connection method is used to end the existing connection. The query method will take a SQL query statement and send it to the database, this method will return the result set table from the Database. The process result set method will then process a result set with parameter on how to process them and return a Vector of objects that contains the information. By doing this, we can handle the database errors such as server down more efficiently.

The last model, GUI is the set of objects that have direct visual contact with the users. Each object in the GUI model is an independent window/frame that can be displayed on the screen. There are a total of 8 GUI objects. They are System Login Window, Login Panel, Main Menu Frame, Employee Panel, Member Panel, Inventory Panel, Transactional Panel, and Search Engine Window. The System Login Window is a window opened when the user starts this software. It asks the user to enter the database id, the password, and the store id to login. If any of those three is incorrect or an internal error occurred (such as server down), an error message will be prompted in an error window. The next objects the user will encounter are Login Panel and Main Menu Frame. This Login Panel will ask the user to enter his/her employee id and password. If successful, the person can proceed to the Main Menu Frame else an error message will be shown and nothing will happen. The Main Menu Frame is the background window that controls all the panels in this software. It has eight buttons. Seven of them are used to access the other panels and windows and the last one is used to logout of the system. The buttons however are not all available to all users. Different levels of users have different set of buttons he/she can access. In the center of the main menu frame, a dynamic window is used to display information regarding to the button pressed by the user. The default panel of the center image is white background with the word Welcome on it. From the Main Menu Frame, the Employee panel can be access. The Employee panel will allow the user to add/update/delete employee data. This panel will call the Employee object and use it to handle all the error checks and transactions. If an update is successful, a window will popup to show that the process is successfully done else an error message will be displayed. Only level-two access right can activate employee object. The Member Panel is similar to the employee panel except it handles updates of customer information. It will use the Customer Data object instead of the Employee Data object to process the user input. Then Inventory Panel is a little different from the previous two panels. It has fields for the user to enter the item information and it will generate item id and availability by using the Inventory Data object. And in this panel, the entire non-blocked fields set must be filled in order for the transaction to be processed. The “foundation” panel of this software is the Transaction Panel. This panel will handle all the rentals and return procedures in GUI. It requires the renter’s id and item’s id for any process to start. The Transaction panel will first initiate the Customer Data object to check the renter’s ability to rent and then create the Inventory Data object.
to check the item’s availability as well as making changes to the database. But please notice that the Transaction Panel is only a GUI component so it doesn’t contain any content that actually process the transaction. The last GUI component is build to perform searches in the database. This object contains three buttons for the user to select the search category (employee, customer, item). Each button will change the search window’s content according to the data contained by that particular object. This GUI object will utilizes the search functions of the search core object to perform searches. The results will be shown in a table. The user can double click the table content to go to the panel that contains the information of that searched object. For example, if the user searches the employee category and found John Dan, then he double clicks the person’s data in the table; the Main Menu Frame will automatically switch the panel on it into Employee panel and it will contain the information of John Dan. This is all about the GUI objects. For the detailed GUI descriptions, please see Appendix A.

The server side database is a pre-defined SQL database. We don’t need to implement an application on the server side to communicate with the client but we do create the data structure of the information contained in the server. The SQL database in the server is build corresponding to the data objects of the client application. In order to achieve modularity, we build the database in an object-oriented structure. We have a total of 14 tables: person, employee, member, language, rate, store, rental items, game category, video category, game format, video format, game, video, and type. Each table represents a unique object. The person table contains all the information of a person type object, the employee and member table contains the specific information of its type and it’s mapped to the corresponding information in person table. All the other tables are attributes of the inventory item. They were implemented in different table to prevent redundant information been stored (inventory item can have the same format, type, etc). In order to access the database, a standard communication library is used. The Sybase server provides this library. It is used for communication with java client. We decided to use this library because it’s free and it saves us from implementing a communication package. The overview of the database components is as follows:

- Person: store the information of all person type objects such as employees and customers
- Employee: store the information of all employee objects
- Member: store the information of all member (customer) objects
- Lang: store languages of the inventory items
- Rate: store the rental prices of the inventory items
- Store: store the stores that were using this program
- RentalItem: store the information of the rental items
- GameCategory: store the categories of the games in the inventory
- GameFormat: store the format of the games in the inventory
- Game: store the information of the games in stores
- VideoFormat: store the format of the video
- VideoCategory: store the category of the videos in the inventory
- Type: store the type of an inventory item (video or game)
- Video: store the information of the videos in the store

Anyway, by utilizing modular design on both the client and server, we provide a useful environment for any people who wish to change/patch this program.
IV. Testing Procedures Design

This application must be tested thoroughly before it becomes the final product. Good testing processes will eliminate possible bugs and help the developer to discover errors early. These testing processes are design to find out if the program will work or not. The main goal of our testing procedure is to find out whether our software can perform a rental process correctly. We decided to use this process because it will reference many other processes and functions of the application.

The testing processes design is consisted of the following parts:
- testing environment
- expected outcome of rental process
- possible errors
- possible sources of errors

The video rental procedure is chosen as the test case because it is one of the essential functions of the program. If the program cannot handle renting a video correctly, the program is basically useless. This function must be tested thoroughly through the setup in all necessary environments. During the test, the customer information is assumed to be correct which means that the customer database is error-free. Here is a brief description of how will we test the software step by step: we first make sure the correct data is in the SQL tables before the test (video’s existence, availability, etc). Then we open the GUI interface that is responsible for the processing the change. In this case, it will be the Transaction Panel. After that we enter the appropriate test case data that has the Environment Properties. Example: For Environment A, we simply enter an id of an existing and available video. We also enter a member id that exists in the member database that is clean. Then we hit the rental button. If the rental is successful, a success dialog will be shown on screen, else a failure dialog that display the reason the process failed will be shown. Finally, we check the database see if the update has been made CORRECTLY and then check the subtotal field to see if the item’s rental fee and tax are calculated correctly.

The rental process testing will be performed in several different environments. We have devised seven testing environment for the procedure. An environment is consisted of a list of properties that can occur when the user is processing the rental. The first environment has the following properties: the video exists, the video is available for rent, the renter is a registered member, the renter is not banned for overdue fees, and the renter has no overdue items. The correct output of the program should be that the success dialog is prompted and the new updated SQL tables have the appropriate value set (the properties of rent by who, rent date, return date, availability and etc.) If an error should occur, it should be caused by the following: the queries for SQL tables are not functioning correctly, typos in the text Fields, logical error in Transaction object, the variables were not passed correctly, or/and logical error in GUI. The second environment contains these properties: the video exists, the video is not available for rent, the renter is a registered member, the renter is not banned for overdue fees, and the renter has no overdue items. The outcome should be: the failure dialog should prompt and display the message that the item is not available for rent and the SQL tables should not be updated
because the failure occurred, the subtotal, tax and total field should stay at 0. Possible cause of error are: the queries for SQL tables are not functioning correctly, typos in the input text Fields, logical error with Transaction object, the variables were not passed correctly, and/or logical error in GUI. The third environment contains the following attributes: the video doesn’t exist (invalid video id number), the renter is a registered member, the renter is not banned for overdue fees, and the renter has no overdue items. The result of the rental process should be: the failure dialog should prompt and display that the item doesn’t exist in the inventory and the SQL tables should not be updated. The price fields values are to be remain the same. If anything goes wrong, the source of error will most likely be: the queries for SQL tables are not functioning correctly, logical error with Transaction object class, typos in the text Fields, the variables were not passed correctly, and/or logical error in GUI. The fourth environment contains these properties: the video exists, the video is not available for rent, and the renter is not a registered member (given an invalid id). The outcome of the process should be: the failure dialog should prompt and display that the member is not registered and the SQL tables should not be updated. The price fields’ values are to be remained the same, if the dialog shows that the video is not available then not ok because the member validity should be checked prior to the video’s status. Any error occurred would mostly likely to be the faulted by the following problems: the queries for SQL tables are not functioning correctly, logical error with Transaction object class, typos in the inputting text Fields, the variables were not passed correctly, logical error in GUI. The fifth environment is like this: the video exists, the video is not available for rent, the renter is a registered member, the renter is banned for overdue fees, and the renter has overdue items. The correct output would be: The failure dialog prompts and display that member id is banned and the SQL tables should not be updated. The price fields’ values are to be remained the same; again, the member status must be check prior to the checking of the videos. The cause of any error would be the same as the fourth environment. The sixth environment is defined by the following properties: the video exists, the video is not available for rent, the renter is a registered member, the renter is not banned for overdue fees, and the renter has overdue items. The correct output from the program would be: the failure dialog should prompt and display that the member has over due items and the SQL tables should not be updated. The price fields’ values are to be remained the same. When the error dialog appears, the user than can ask the customer to pay… The cause of any error will be the same as the fourth and fifth environment. The seventh environment is used to test the crash-protection of the software. It contains the following properties: either one or both of customer database and item database become unavailable after the program started (case can be tested by setting a trigger to disconnect from the database right after the queries are send). The correct state of the server and client after the transaction would be: An error message should appear immediately saying that the main server is disconnected. In this case, SQL tables SHOULD NOT be updated even if the server has received all the necessary information from the client. The possible source of errors would be: the queries for SQL tables are not functioning correctly, logical error with Transaction object class, typos in the text Fields, the variables were not passed correctly, logical error in GUI – the program should not have any internal errors (they should all be caught and handled so the user cannot see them, but logged for programmer to debug later). Our testing design ends here.
V. Appendix

A. User Interface Design Detail

*The goal of the Graphical User Interface is to provide a better experience for the user. A good graphical interface can make an application very fun to use while a bad interface will give the user bad experience during the usage of the program. The goal of this software’s GUI is not to be fancy but to present the information in a clear and precise manner.*

GUI Design

Our user interface is consisted of several Panels and Frames. The user first starts the application through a login window into the database, and then he/she must login as an employee to start to use the program. The main menu of our application is a static frame that contains buttons and logos that will be available every time the software is been used, however the availability of the buttons are determined by the security level of the user. In the center of the main menu frame, a dynamic window is used to display information regarding to the button pressed by the user.

The frames implemented in our software are as follows:

A.1 - System Login Window

| Purpose: | login to start the software |
| Buttons: | A. connect button to initiate the connection to the Database |
|          | B. cancel button to exit the program |
| Fields:  | C. input field of Database Login ID |
|          | D. password field for entering Database password |
|          | E. input field for store ID, it will determine the store where the software is used |

A.2 - Main Menu Frame
Purpose: contain buttons to direct the users to different panels for different transactions

Buttons:
A. button used to activate the Employee panel
B. button used to activate Customer panel
C. button used to activate Inventory panel
D. button used to activate the Transaction panel
E. button used to activate the Search window
F. button used to activate screen lock function
G. button used to generate report of the store
H. button used to log out of the system and return to login panel

A.3 - the Login Panel

Purpose: the panel used to log into the system

Buttons:
A. LOGIN button to initiate login

Fields:
B. password field to enter the password of the employee  
C. id field to enter the id of the employee  

A.4 - Employee Panel  

Purpose: the panel activated by the Main Menu Frame to add/delete/change employee information, this feature is only accessible by the highest level of login  

Buttons:  
A. back button to get back to the previous panel  
B. EDIT button to update an existing record  
C. Add Record button to add these information into a new record  

Fields:  
D. The input field for entering id of the employee  
E. The input field for last name  
F. The input field for first name  
G. The input field for login id  
H. The input field for password  
I. The input field for position of the employee  
J. The selection field for access level of this employee  
K. The input field for middle name  
L. The input field for address  
M. The input field for phone number  
N. The input field for when the employee registered  
O. The input field for hourly wage of this employee  

A.5 - Member Panel
Purpose: the panel activated by the Main Menu Frame to add/delete/change the information of the members of the store, it’s accessible by all login levels

Buttons:
A. Edit button for update an existing record with the following information
B. Add Record button to add the information as a new record
C. Back button to go back to previous screen

Fields:
D. Id field for input user id
E. Field for enter last name
F. Field for enter first name
G. Field shows the current amount of rental made by this customer
H. Field shows the total amount of rental allowed
I. Field shows the amount of violation made by this customer
J. Field for enter middle name
K. Field for enter address
L. Field for enter phone number
M. Field shows the registration date of the customer

A.6 - Inventory Panel
**Purpose:** the panel activated by the Main Menu Frame to add/delete/change the information of an inventory item, it’s accessible by all login levels

**Buttons:**
A. Update button to update the existing record with the following information
B. Add Record button to add the information as a new record
C. Cancel button to exit the Inventory panel

**Fields:**
D. Item Number field of the item’s unique id
E. Title field for input title of the item
F. Selection field for select item type
G. Release Date field for input when is the item will be released
H. Store Id field for input the id of the store where the item is stored
I. Store Location field for input the location of the store
J. Store Phone # field for input the phone number of the store
K. Selection field to select whether the item is available or not
L. Selection field for entering category of the item
M. Developer/Director field for input the developer/director of the item
N. Selection field for selecting the language of the item
O. Selection field for selecting the format of the item
P. Renter/Member Id field for input the id of the customer who rent this item
Q. Date Rented field for input the date when the item is rent
R. Rental duration field for input the number of days this item can be rent
S. Date Return field for input the date the item will be returned
T. Rental Date field for input the day when the item is rent
A.7 - Transaction Panel

**Purpose:** the panel activated by the Main Menu Frame to rent or return items borrowed by a member

**Buttons:**
- B. The button to validate the existence of membership
- D. The button to proceed to rent the item
- E. The button to proceed to return the item
- I. The button to calculate the money needed to rent the items that are listed in the J text field

**Fields:**
- A. Enter Member Id field to input the id of the renter
- C. Enter Item Number field to input the id of the item to be rented
- F. Subtotal field that shows the total amount of rental money before tax
- G. Tax field that shows the total tax needed
- H. Total field that displays the total money needed to rent
- J. The List field that list all items that will be rented

A.8 - Search Window
Purpose: the window activated by the Main Menu Frame to search for employee, member, and inventory item. The window can direct the user to the panel containing all the information of the searched object. The login level will determine the type of search a user can perform

Buttons:
A. The button to select the search panel to search employee records
C. The button to select the search panel to search item records
D. The button to select the search panel to search member record
F. The button to start the search
G. The button to change the panel behind the search window to a panel contains all the information corresponding to the item selected
H. The button used to delete the selected item

Fields:
B. Selection field to select whether to search the current branch or all the store data
E. Selection field to select the searching method
I. The input field for input the keyword to search
B. System Behavior

This section will present a few processes and events that might happen during the usage of this software. It will show the possible states and functionalities and the states of components of the software.

B.1 Object-relational model of server-client in Collaboration Diagram

This is a collaboration diagram of creating a new customer account in the database through the program. It contains information about the specific methods that will used by the involved objects when they are affected by the process.

**Actors** are the objects that initiate and control the process. 
**Relations** are paths between objects. 
**Objects** are objects involved in this event.

Create A New Account Collaboration Diagram:

**Actors:**
1. Renter: The renter is the actor who is going to register for a new account this renter can be registered or not as the employee will check for the renter's status through the program.
   - Associated Actors: Employee
   - Associated Objects: None
2. Employee: This is the actor that is going to process the request of the renter to create a new account, he’s the terminal between the renter and the program.
   - Associated Actors: Renter
   - Associated Objects: CustomerDataMgtPanel.

**Relations:**
1. Request() – the method that request the registration of a new member
2. Denied() – the method that indicate the denial of the request
   This happens when the registration is denied by the database and pass back to the renter who is trying to register for an account
3. addMemberInfo() – This is the general procedure call to add information to an SQL Table.
4. accept() – this method will act as the accepted signal enter back to the data panel, this signal then initiate adding information into the members SQL table
5. addInstance() – this method will try to add the entered information into the database, the database then will check for the existence of the member
6. check() - the database check for the existence of the member as it was been added into the table.

**Objects:**
1. CustomerDataMgtPanel - This is the interface that the employee will use to process the request of the renter, it's the terminal that collects input information and pass them to
2. Person(SQL Table) - this is the SQL table that manages all the data of the people related to the store and it will be used to check redundancies in the system which prevents old members to register and etc.

3. PersonalDataMgtPanel - This is the terminal that controls the constrains flow such as the accept() and denied() functions return by the database.

4. Members(SQL Table) - the SQL table that stores the member information, this is the table that will be altered only after the Person(SQL Table) return the accept() signal and passed into the PersonalDataMgtPanel.
B.2 Enter A New Item

This diagram provides information on the effects of adding inventory items over other objects. The diagram also shows how enter a new item is done step by step through the objects that will be encountered through the process.

**Enter new item data activity diagram documentation**

Employee opens our Video Rental System and in the login frame and he enters his ID and password. The system will then compare it to the entries in the database to see if an employee is valid, if not it will return to the employee logins, which the person will be able to enter the password again.

If it is valid the employee will be able to enter new data, and the data will be taken to compare to the database’s if no error occur or no identical data in the database, the data will be stored and the user will return to the data entering state so he can make another entry, if there’s error occur or there is an identical entry in the database, the customer will be able to enter it again, and the data will not be saved to the database.

If everything is done, the sequence will exit.
B.3 State Diagram of all possible states of a rental item

This diagram gives the overview of all possible states of the object Rental Item. The diagram summarizes the relations between the states as well as the conditions to get to that state. Again, all states here are of one object: rental item.

State Diagram for Instance of Rental Item

The **associated states** are the states that can be achieved from the current state. The **Events** are updates and changes performed when leaving/entering the current state.

States

**in store:**
this state shows that this rental item is currently in store and it’s available for rental. In this state, the rental item can be gone through any processes that alter its state. Basically, this is the terminal between all the states of the rental item.

**associated states:**
- Coming Soon: this is an initiation state that will create a new rental item, this state represents the rental items that haven’t arrived to store but have a release date.
- Borrowed: this is an exit state from the ‘in store’ state that represents the transitional state of the rental item. In this state, the item will be temporarily disabled (meaning availability is false), this state will stay until the item is returned.
- Delete: this state of the rental item will eventually lead to the deletion of this item in the inventory.

**Events:**
- Entry/available(): alter the availability property of the item.
- Entry/countTitle(): alter the number of available items of this Title.

**borrowed:**
this state is when the updating state trigger the event available() and setRenter() to make the item be unavailable for the other renters. The setRenter() method will set the borrower of the item. In this state, the item will be disabled. This is kind of like freezing the item. When the borrower return the item, it will be available for the renters again.

**associated states:**
- In store: this is the middle state of the inventory item during a transaction, in this state, the item will be available for rent, this state is pretty much of a terminal that transits with all the other states in the diagram.

**Events:**
- Entry/available(): alter the availability property of the item.
- Entry/setRenter(): triggered to set the borrower of the item.
- Entry/countTitle(): alter the number of available items of this Title.
Deleting:
this state is when the updating state trigger the event delete() to delete an item from the RentalItem SQL table, the rental item is now in the state to be removed from the RentalItem SQL table. This item will be set to be unavailable to all renters permanently. The actions that trigger the deleting could be selling the item or the item is lost.

associated states:
- In store: this is the middle state of the inventory item during a transaction, in this state, the item will be available for rent, this state is pretty much of a terminal that transits with all the other states in the diagram
- finish: this is the final state of the item that ends the transaction

Events:
- Entry/available(): alter the availability property of the item
- Entry/countTitle(): alter the number of available items of this Title.

coming soon:
this is the state where the item is not yet available in store however the item will arrive soon on the designated release date. The item in this state is still very unstable because the stock may not arrive on time or it may not even arrive at all. This state will be the one that create the rental item in the store. Before this state, the item doesn’t exist at all in the SQL table.

associated states:
- start: this is the starting state of the process
- In store: this is the middle state of the inventory item during a transaction, in this state, the item will be available for rent, this state is pretty much of a terminal that transits with all the other states in the diagram

Events:
- Delete the data from the rental SQL table
- Delete the data from all the associated SQL table
- Entry/available(): alter the availability property of the item
- Entry/countTitle(): alter the number of available items of this Title.

State Features
- entry/action: Entry/available(): alter the availability property of the item. Entry/setRenter(): triggered to set the borrower of the item. Entry/countTitle(): alter the number of available items of this Title.
- exit/action: upon exiting the coming soon state, properties will be changed inside that state and the item will be added to the inventory. This is the only exiting action that will create the rental item.