Module 3 assignment: IMDB database

Business Vision

We want to create a system to access a database for the movies on IMDB. We want to be able to access all the data, from all movies existing in the database. We also want to store new movies in the future, as well as updating titles and deleting entries.

The goal for this project is to manage database that is structured in a efficient way.

Glossary

CRUD	Create, Read, Update & Delete
Table	A file representing a table in our database
Database	The relational database used to store movie data
enumerator	Specifies the capable value of an attribute

Requirements (FURPS+)

The software is expected to take care of *CRUD* - Creating, Reading, Updating & Deleting data from our database. The software is also expected to make advanced user-friendly searches based on a set of criteria set by the user of the program. The search functioning must be optimized for large data tables to minimize processing power and time needed to finish a request.

Use Cases

Use case 1: **Create a movie**

The staff should be able to create a new table or a new line in a table. This way the staff can store data from new movies into the database.

> Staff logs in while in the menu -> clicks 'create new movie' -> goes through the *create form* -> clicks save.

Use case 2: ***Read from the database***

The users should be able to lookup a movie and see all the data, such as actors, storyline, year of publicity etc.

> User search for a movie -> selects the movie -> Information about the movie is shown.

Use case 3: ***Update***

The update part makes it possible for the staff to update existing information with new values. Thus the staff can edit ex. the actors of a given film, without running through the *create form.*

> Staff logs in -> search for a movie -> selects the movie -> navigates to the actors section -> clicks edit -> updates the information -> clicks save.

Use case 4: ***Delete***

The staff should be able to delete delete existing information. We want to be able to access the database to remove information related to the movies or whole existing movies.

> Staff logs in -> search for a movies -> selects the movie -> navigates to the wanted section -> clicks delete -> confirms -> clicks save.

Use case 5: **Search**

Use Case Section	**Comment**
Use Case Name	Search
Scope	Search engine for a imdb database
Level	User goal
Primary Actor	User
Stakeholders and Interests	User: Wants a platform that makes it possible to search for a movie and gather information about

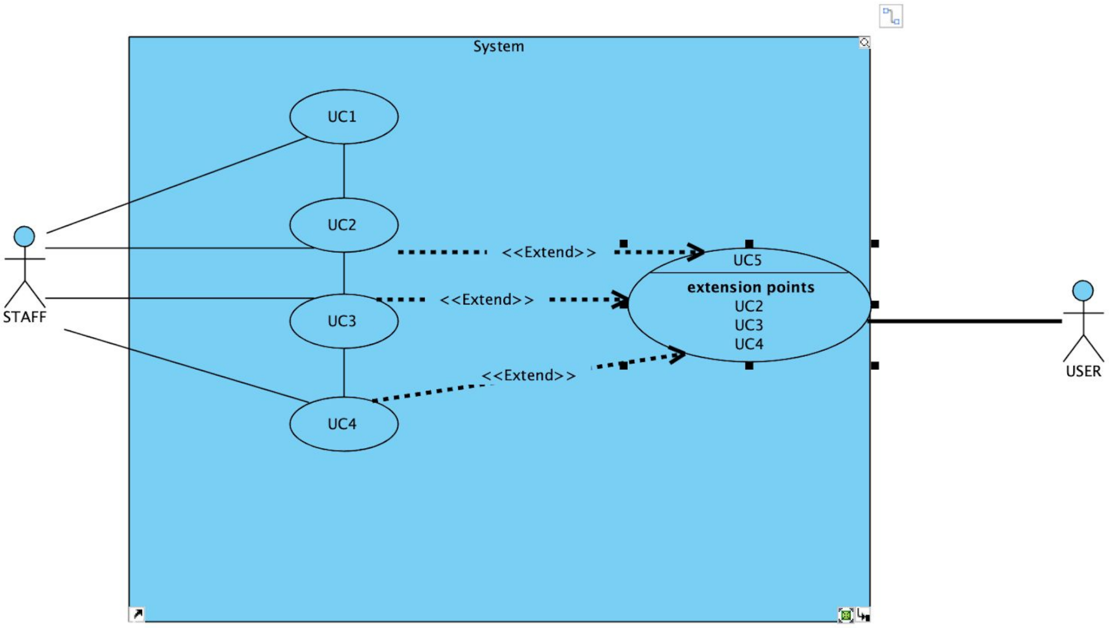
	the movie. Staff: Wants a platform to manage a movie database while also allowing read access to regular users.
Preconditions	The files containing the data used by the program must be intact and not corrupted.
Success Guarantee	The correct searched data successfully returned to the user.
Main Success Scenario	1. User enter website 2. Enters search criteria in search engine 3. Title comes up with the correct information on the movie/series
Extensions	*a. File (table) not found 1. Prompt the user to enter a different table name
Special Requirements	The user uses a device capable of running Java
Technology and Data Variations List	A server containing the data. A computer which has java installed.
Frequency of occurrence	User should be able to search multiple times
Miscellaneous	The search machine can be updated

Use case 6: ***Menu***

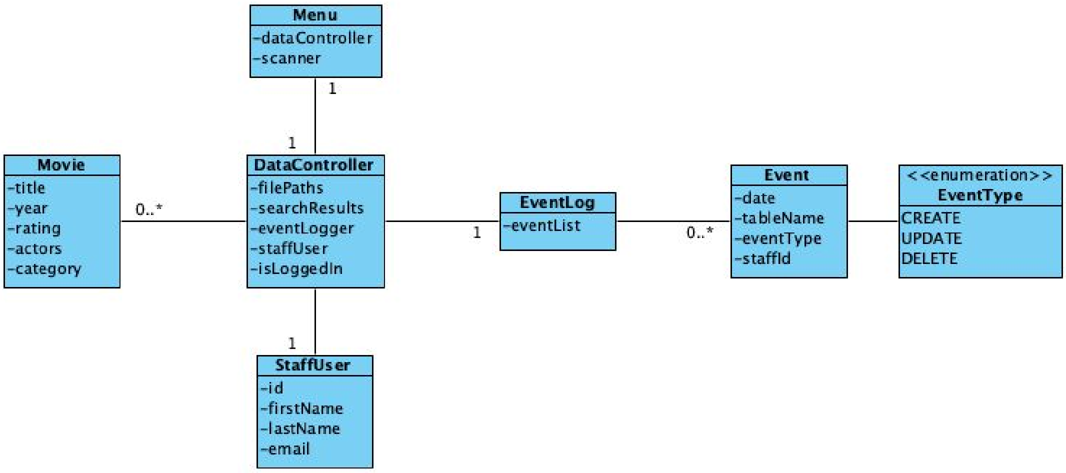
Create a accessible menu for users. The menu handles navigation, it’s going to make it easier for the user to navigate in our database.

The menu will enable the user to access the different kinds of services our program will provide.

Use Case Diagram

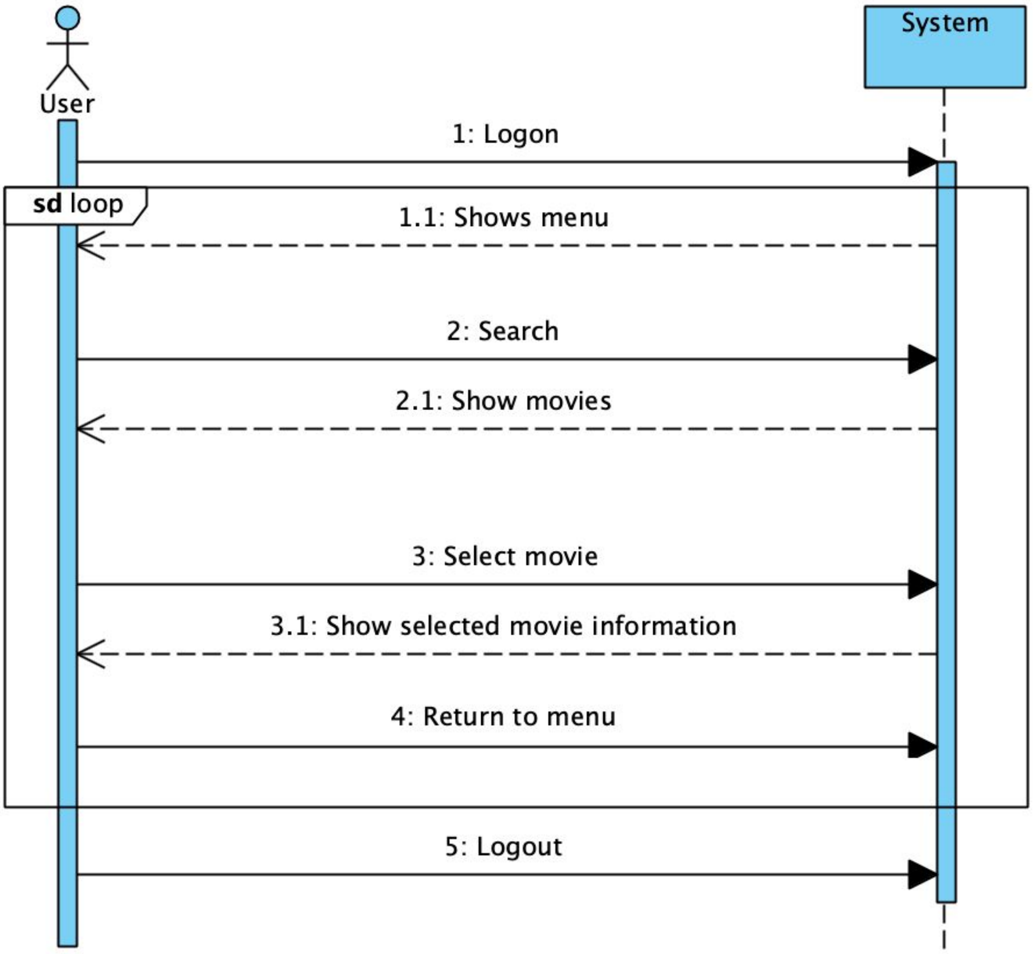


Domain Model

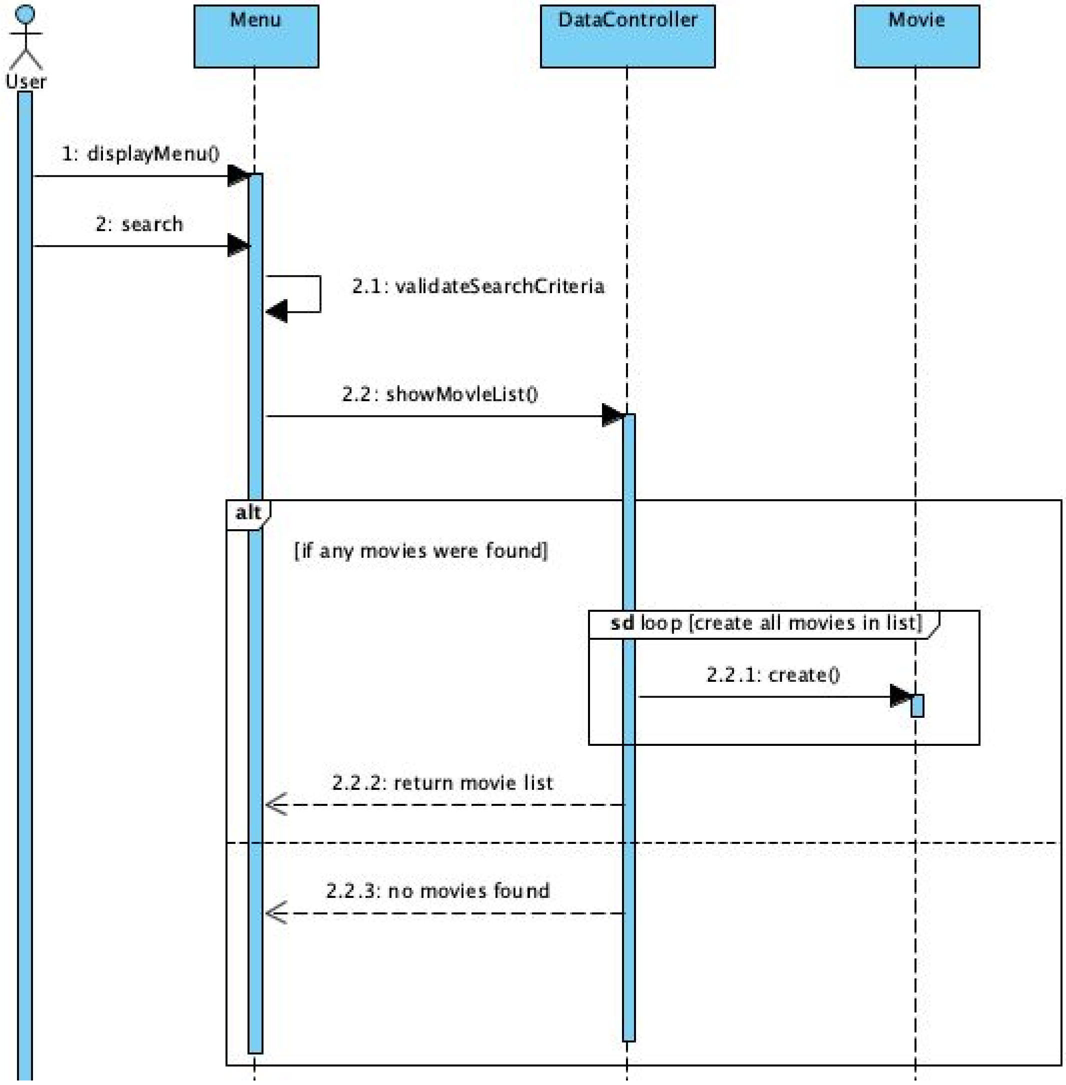


System Sequence Diagram (eventually Operational Contracts)

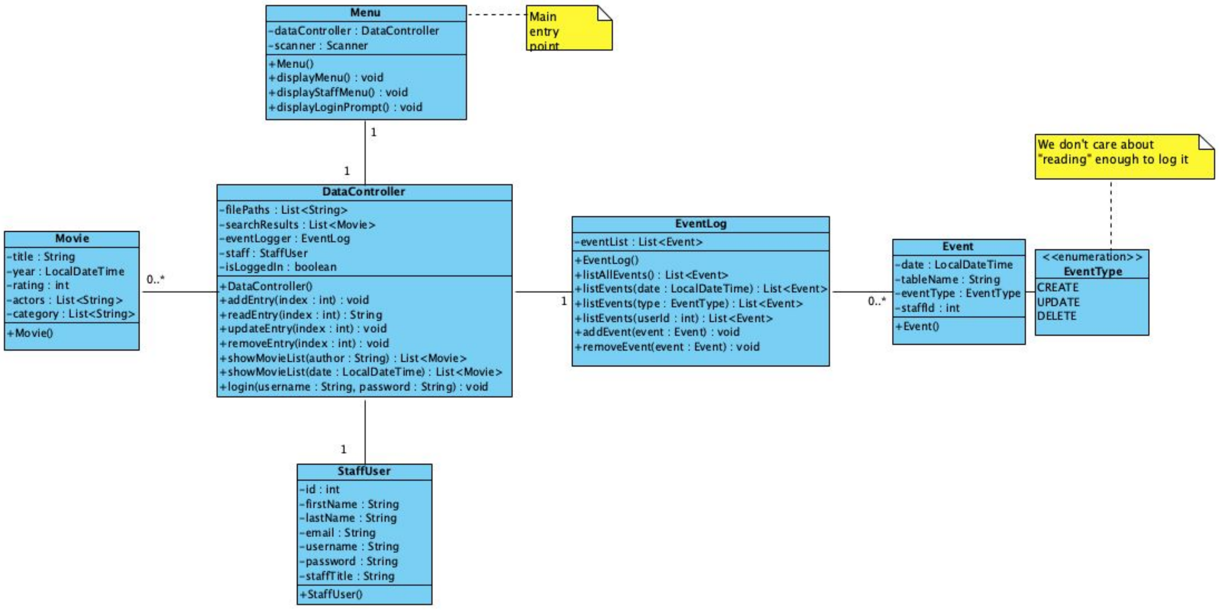
This SSD shows the users interaction with search and read from database (Use Case 5).



Sequence Diagram (eventually Operational Contracts)



Design Class Diagram


Menu
-dataController : DataController
-scanner : Scanner
+Menu()
+displayMenu() : void
+displayStaffMenu() : void
+displayLoginPrompt() : void
Main entry point
1
1
DataController
-filePaths : List<String>
-searchResults : List<Movie>
-eventLogger : EventLog
-staff : StaffUser
-isLoggedIn : boolean
+DataController()
+addEntry(index : int) : void
+readEntry(index : int) : String
+updateEntry(index : int) : void
+removeEntry(index : int) : void
+showMovieList(author : String) : List<Movie>
+showMovieList(date : LocalDateTime) : List<Movie>
+login(username : String, password : String) : void
Movie
-title : String
-year : LocalDateTime
-rating : int
-actors : List<String>
-category : List<String>
+Movie()
0..*
1
StaffUser
-id : int
-firstName : String
-lastName : String
-email : String
-username : String
-password : String
-staffTitle : String
+StaffUser()
1
EventLog
-eventList : List<Event>
+EventLog()
+listAllEvents() : List<Event>
+listEvents(date : LocalDateTime) : List<Event>
+listEvents(type : EventType) : List<Event>
+listEvents(userId : int) : List<Event>
+addEvent(event : Event) : void
+removeEvent(event : Event) : void
0..*
Event
-date : LocalDateTime
-tableName : String
-eventType : EventType
-staffId : int
+Event()
<<enumeration>>
EventType
CREATE
UPDATE
DELETE
We don't care about "reading" enough to log it