Group Members :

ATR/6751/08 Henok Dejen

ATR/0872/08 Hiskel Kelemework

ATR/4025/08 Tewodros Yesmaw

ATR/1754/08 Mihreteab Demeke

ATR/8059/08 Dagim Amare

ATR/3529/08 Wakijira Abdisa

ATR//08 Misgan

3/23/2018

[Document title]

SDS

Contents

[**List of Tables** 3](#_Toc509431614)

[**List of Figures** 4](#_Toc509431615)

[**1** **Introduction** 5](#_Toc509431616)

[1.1 Purpose 5](#_Toc509431617)

[1.2 General Overview 5](#_Toc509431618)

[The controller 5](#_Toc509431619)

[The view 6](#_Toc509431620)

[The Model 6](#_Toc509431621)

[1.3 Development Methods & Contingencies 6](#_Toc509431622)

[**2** **System Architecture** 7](#_Toc509431623)

[2.1 Subsystem decomposition 7](#_Toc509431624)

[2.1.1 Layer 1 7](#_Toc509431625)

[2.1.2 Layer 2 7](#_Toc509431626)

[2.1.3 Layer 3 8](#_Toc509431627)

[2.2 Hardware/software mapping 8](#_Toc509431628)

[3 **Object Model** 9](#_Toc509431629)

[3.1 Class Diagram 9](#_Toc509431630)

[3.2 Sequence Diagram 10](#_Toc509431631)

[3.2.1 Create Account 10](#_Toc509431632)

[3.2.2 Search Movie 11](#_Toc509431633)

[3.2.3 Show Movie Detail 12](#_Toc509431634)

[3.2.4 Recharge Balance 13](#_Toc509431635)

[3.2.5 Show movie schedule 14](#_Toc509431636)

[3.2.6 Add News 15](#_Toc509431637)

[3.2.7 Edit Cinema 16](#_Toc509431638)

[3.2.8 Search Movie 17](#_Toc509431639)

[3.2.9 Add movies to schedule 18](#_Toc509431640)

[3.2.10 Renew Schedule 19](#_Toc509431641)

[**4** **Detailed Design** 20](#_Toc509431642)

[**Reference** 27](#_Toc509431643)

[Bibliography 27](#_Toc509431644)

# List of Tables

[Table 1 movie class 19](#_Toc509428704)

[Table 2 Attribute description of movie class 19](#_Toc509428705)

[Table 3 operation description of movie class 19](#_Toc509428706)

[Table 4 cinema class 19](#_Toc509428707)

[Table 5 Attribute description of cinema class 20](#_Toc509428708)

[Table 6 operation description of cinema class 20](#_Toc509428709)

[Table 7 News class 20](#_Toc509428710)

[Table 8 Attribute description of News class 20](#_Toc509428711)

[Table 9 operation description of news class 20](#_Toc509428712)

[Table 10 user class 21](#_Toc509428713)

[Table 11 Attribute description of user class 21](#_Toc509428714)

[Table 12 operation description of user class 22](#_Toc509428715)

[Table 13 cinemaAdmin class 22](#_Toc509428716)

[Table 14 Attribute description of cinemaAdmin class 22](#_Toc509428717)

[Table 15 operation description of cinemaAdmin class 22](#_Toc509428718)

[Table 16 system admin class 23](#_Toc509428719)

[Table 17 Attribute description of system admin class 23](#_Toc509428720)

[Table 18 operation description of system admin class 23](#_Toc509428721)

[Table 19 userManager table 23](#_Toc509428722)

[Table 20 operation description of usermanager class 24](#_Toc509428723)

[Table 21 schedule class 24](#_Toc509428724)

[Table 22 Attribute description of schedule class 24](#_Toc509428725)

[Table 23 payment class 24](#_Toc509428726)

[Table 24 Attribute description of payment class 25](#_Toc509428727)

[Table 25 card class 25](#_Toc509428728)

[Table 26 Attribute description of card class 25](#_Toc509428729)

[Table 27 Ticket class 25](#_Toc509428730)

[Table 28 Attribute description of Ticket 26](#_Toc509428731)

# List of Figures

[Figure 1 layer system architecture 6](#_Toc509431663)

[Figure 2 layer 2 system architecture of the system 6](#_Toc509431664)

[Figure 3 deployment diagram 7](#_Toc509431665)

[Figure 4 Class Diagram 8](#_Toc509431666)

[Figure 5 normal user create account sequence diagram 9](#_Toc509431667)

[Figure 6 normal user search movie sequence diagram 10](#_Toc509431668)

[Figure 7 normal user show movie detail sequence diagram 11](#_Toc509431669)

[Figure 8 normal user recharge account sequence diagram 12](#_Toc509431670)

[Figure 9 normal user show movie schedule sequence diagram 13](#_Toc509431671)

[Figure 10 System Admin add a news sequence diagram 14](#_Toc509431672)

[Figure 11 System Admin edit a cinema sequence diagram 15](#_Toc509431673)

[Figure 12 System Admin search for a movie sequence diagram 16](#_Toc509431674)

[Figure 13 cinema admin add movie sequence diagram 17](#_Toc509431675)

[Figure 14 cinema admin renew schedule sequence diagram 18](#_Toc509431676)

# Introduction

## Purpose

The purpose of this SDS document is to present the system developer with almost all the necessary information that are needed to, the upmost extent, clearly and efficiently implement the system at hand. **It should be noted that**, to the best of our abilities, all irrelevant and verbose explanations have been omitted, so the document could be precise and strait to point.

## General Overview

When taking a closer look at our system, there are two things that stand out from all the other.

1. Data centricity: the entire system is data centric. All the functional requirements of the system are based on moving data around and representing in different ways.
2. Multi-Platform subsystems: the system has components that run on different platforms. One is the mobile platform (Android in particular) intended for use by the public users. Second is one that uses the Web platform. Cinema Admins and System Admins make use of this second platform on which they run their individual systems.

Our application, being data centric and having multi-platform subsystems, favors the MVC architectural design pattern. Since MVC gives data its independence and is also great in decoupling system components to evolve independently, our multi-platform system environment could highly benefit from an architectural design pattern. Therefore, the system will be, for the most part, implemented with MVC.

The ‘M’ in MVC stands for Model, the entity that represents the data, which is the most important component in our system. Our system’s data model is contained in a persistent storage, a famous NOSQL database called MongoDB.

The controllers, ‘C’ in MVC, are written in different programming languages for the different platforms. The mobile (Android) platform implementation will have its controllers written in the Java programming language. The web platform implementations on the other hand will have their controllers written in JavaScript/Typescript.

The final component in MVC, the view (V in MVC), also varies between different platform subsystem implementations. The mobile (Android) platform will have its view written in XML and manipulated dynamically with java. The web implementations will make use of HTML/CSS.

In our system the MODEL completely resides on a central server. Although little data is stored in the mobile implementation by the subsystem itself, no other subsystems store data of their own. We also have many CONTROLLERS on the server side giving multiple services to server-internal and external (accessible to the mobile and web subsystems) components (modules) that each perform crucial individual and combined activities to provide useful services. Therefore, our system also uses the Client Server architecture.

Coming to the main functions that the components in our MVC architecture are required to carry out, an abstract view of the functionalities have been listed as follows.

### The controller

In the all the client and server-side implementations, the controller is responsible for:

* Fetching data from the server
* Sending data to the server
* Validation of data
* Handling authentication
* Managing and keeping track of user states

Client and server-side business logic is handled by controllers in the client and server respectively.

### The view

The view is present only in the client side as a means to interact with the system. It displays in friendly user interfaces, different representations of data fetched from the server and also generated from the client side. It is also through which users of the system update the system by creating new data or update/manage existing data.

### The Model

As explained before, the model completely resides in a remote server. It is represented through many management modules (components) that expose different interfaces to make working with the data stored in the database easy and strait forward. For example, the User Management module exposes an interface that provides functionalities for managing users. This includes creating, removing and authenticating users. Another such module is the Cinema Management which provides an interface for creating a cinema, editing a cinema’s profile, editing movie schedules and so on…

Detailed inner workings of the individual components will be presented late in the SDS document Detailed design section.

## Development Methods & Contingencies

The system used an object orientated approach as well as MVC architecture in development. Since the first step in OOP is to identify all the objects the programmer wants to manipulate and how they relate to each other, an exercise often known as data modelling. Once an object has been identified, it is generalized as a class of objects which defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. Objects communicate with well-defined interfaces called messages. The system used this approach in designing the overall process.

The system has also made use of UML diagrams to provide a standard way to visualize the design of a system in development. The mobile component has been programmed in Java (a very popular Object-Oriented Programming language), and the web component in JavaScript (a language that’s all about Objects). Both development languages are well suited for an Object-Oriented approach. although JavaScript doesn’t support the idea of classes in modelling real world entities, an equivalent form of object modelling called **Prototyping** will be used.

# System Architecture

## Subsystem decomposition

### Layer 1

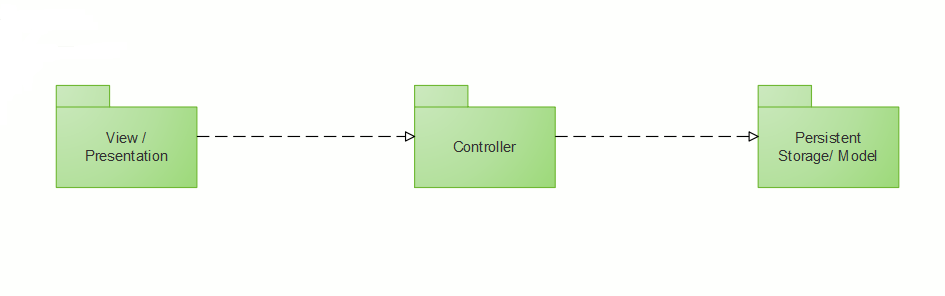


Figure 1 layer system architecture

### Layer 2

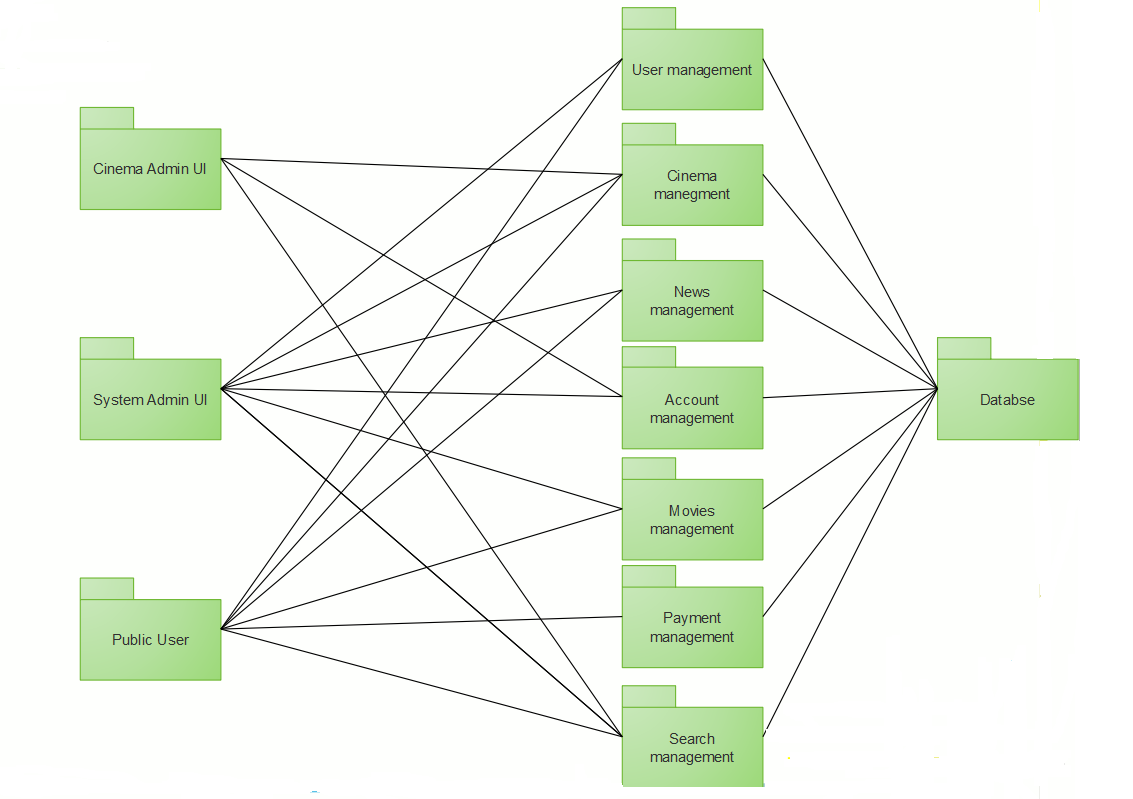


Figure 2 layer 2 system architecture of the system

### Layer 3

## Hardware/software mapping

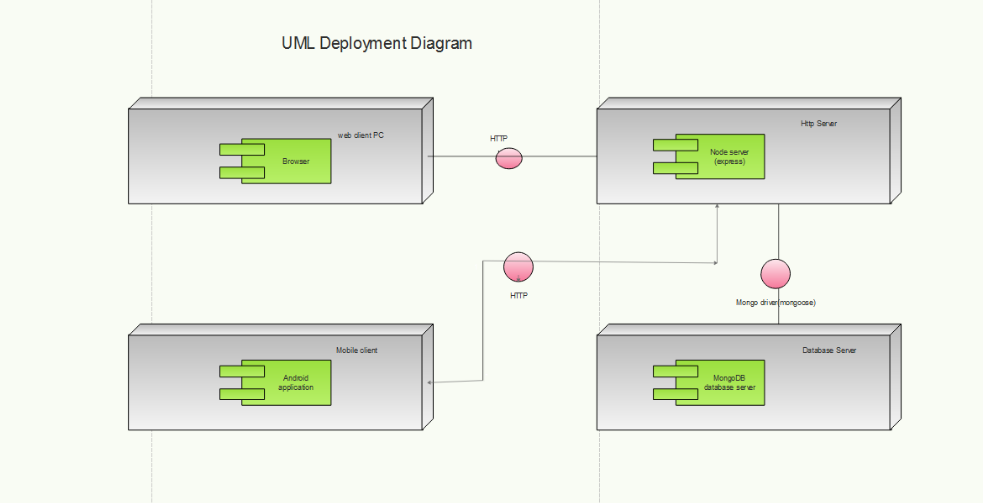


Figure 3 deployment diagram

# Object Model

## Class Diagram

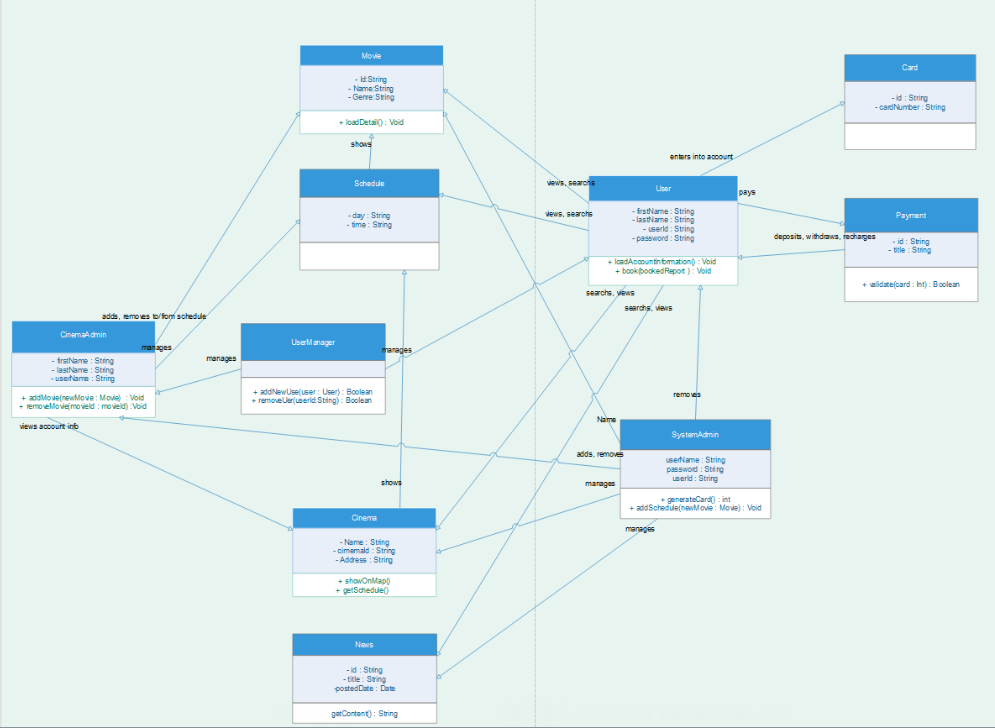


Figure 4 Class Diagram

## Sequence Diagram

### Create Account

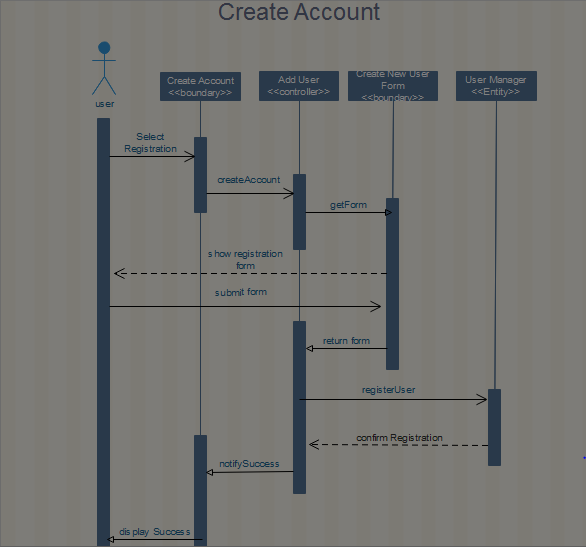


Figure 5 normal user create account sequence diagram

### Search Movie

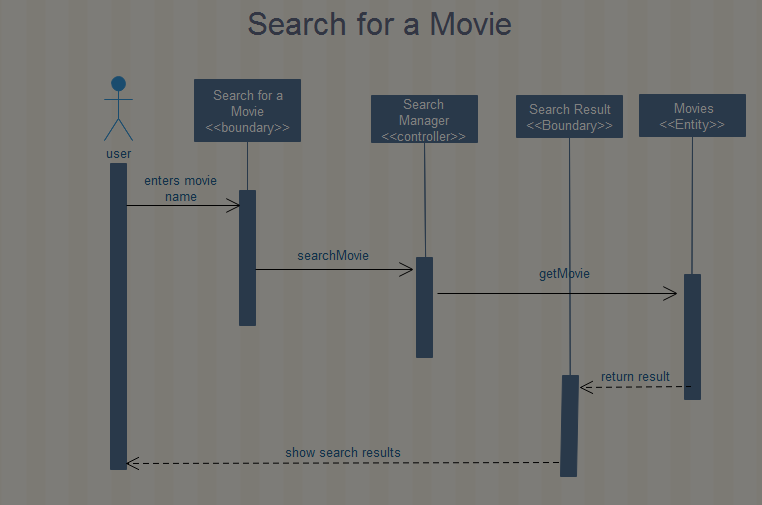


Figure 6 normal user search movie sequence diagram

### Show Movie Detail

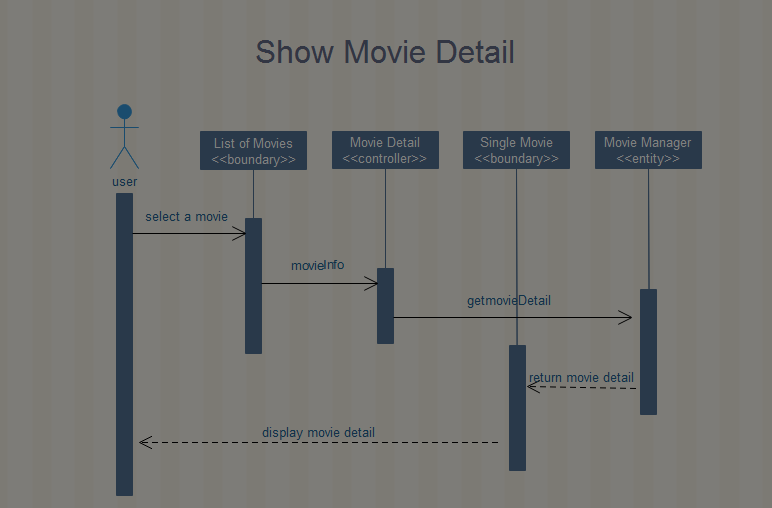


Figure 7 normal user show movie detail sequence diagram

### Recharge Balance

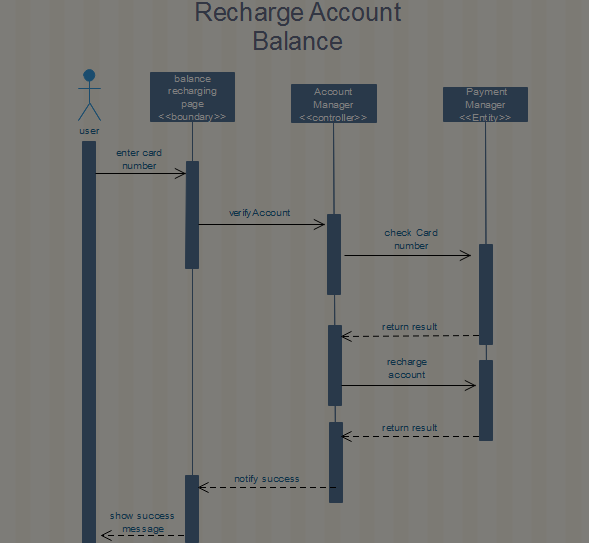


Figure 8 normal user recharge account sequence diagram

### Show movie schedule

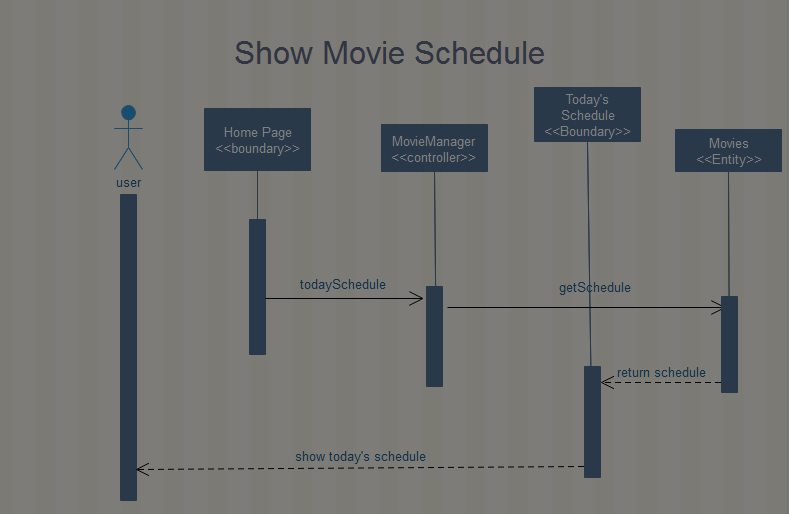


Figure 9 normal user show movie schedule sequence diagram

### Add News

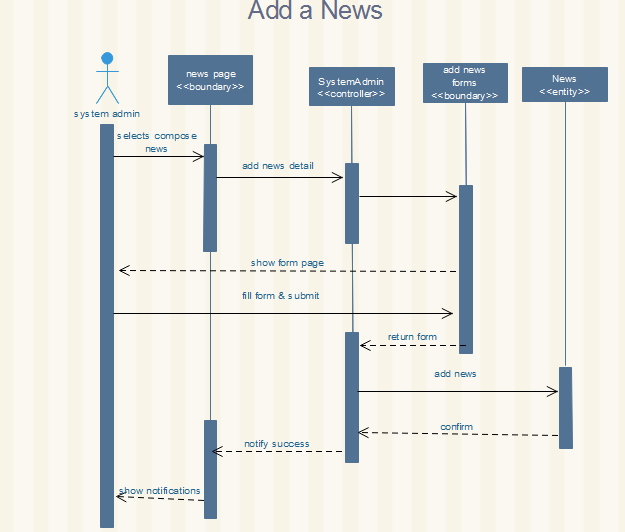


Figure 10 System Admin add a news sequence diagram

### Edit Cinema

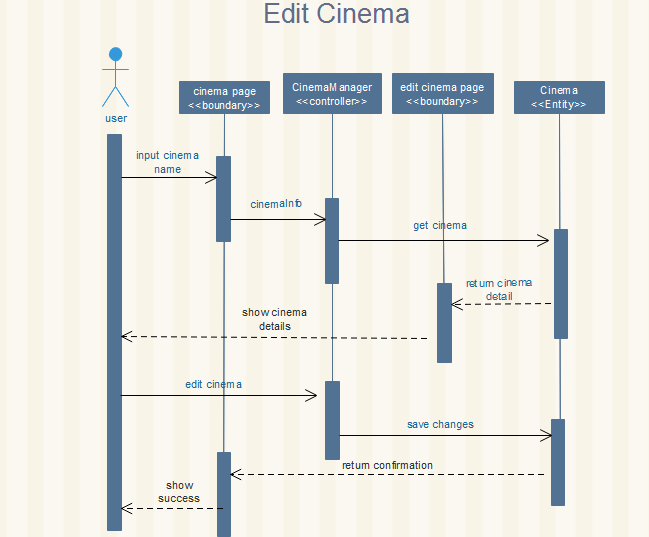


Figure 11 System Admin edit a cinema sequence diagram

### Search Movie

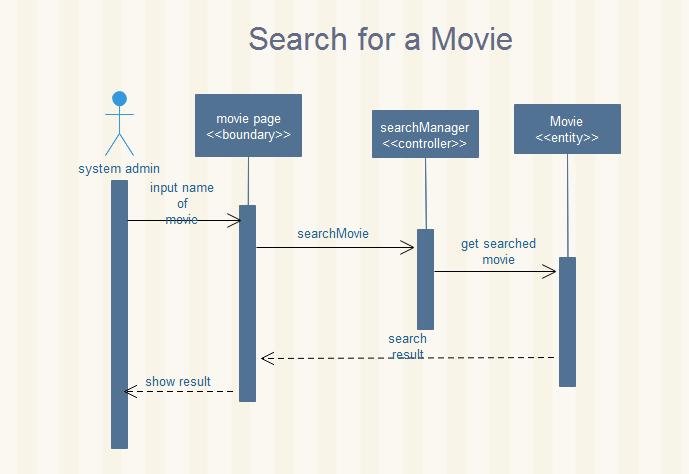


Figure 12 System Admin search for a movie sequence diagram

### Add movies to schedule

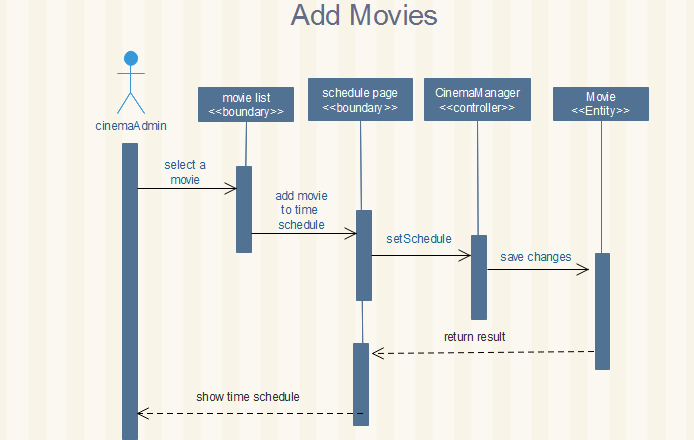


Figure 13 cinema admin add movie sequence diagram

### Renew Schedule

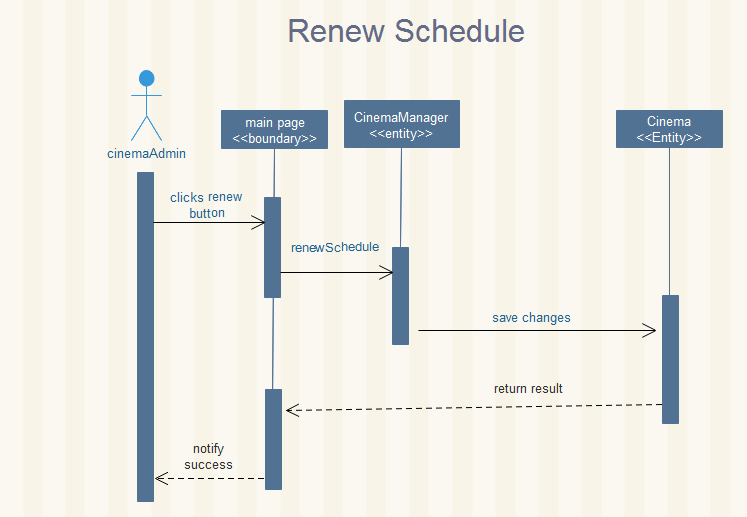


Figure 14 cinema admin renew schedule sequence diagram

# **Detailed Design**

|  |
| --- |
| **Movie** |
| * Id: String * Name: String * Genre: String * Duration: String * Cru: String[] * Story: String * Schedule: Schedule[\*] |
| + loadDetail (): void |

Table 1 movie class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| id | String | private | Can be any string but never be NULL |
| name | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| Genre | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| Duration | String | private | Can never be NULL. It should contain only numbers and colons |
| Cru | String[\*] | private | Can be string array of any size |
| Story | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| Schedule | Schedule[\*] | private | Can be schedule array of any size |

Table 2 Attribute description of movie class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| loadDetail | public | void | void | List of movies are presented | Detailed info about the movie will be loaded from server and stored in the attributes |

Table 3 operation description of movie class

|  |
| --- |
| **Cinema** |
| * Id: String * Name: String * No of seats: Integer * Address: String * Schedule: Schedule [7] * addressLatitude: String * addressLongitude: String |
| + showOnMap (): void  + getSchedules () : Schedules[] |

Table 4 cinema class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| Id | String | private | Can be any string but never be NULL |
| Name | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| No of seats | Integer | private | Should be greater than 0 |
| Address | String | private | Can be any string and NULL |
| Schedule | Schedule | private | Store schedule array of 7. Each can be NULL |
| addressLatitude | String | private | Can be any string. It can’t contain any specific character and integer |
| addressLongitude | String | private | Can be any string. It can’t contain any specific character and integer |

Table 5 Attribute description of cinema class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| showOnMap | public | void | void | Single movie detail is presented | The specified address will be presented on google map |

Table 6 operation description of cinema class

|  |
| --- |
| **News** |
| * Id: String * title: String * Content: String * Posted Date: Date |
| + loadContent(): String |

Table 7 News class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| Id | String | private | Can be any string but never be NULL |
| Title | String | private | Can be any string but never be NULL |
| Content | String | private | Can be any string and NULL |
| Posted Date | Date | private | Can be any string and NULL |

Table 8 Attribute description of News class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| loadContent | public | void | void | List of news are presented | News content will be loaded and stored to content attribute |

Table 9 operation description of news class

|  |
| --- |
| **User** |
| * FirstName: String * LastName: String * UserName: String * Password: String * UserId: String * Balance: float * Booked: Ticket [\*] * UserPhoto: Photo |
| + loadAccountInformation (): void  + book (Ticket): void  + recharge (cardNumber: String): Boolean  + updateUserInfo (FirstName: Stirng, LastName: String, Password: String): void  + deactivate(): boolean |

Table 10 user class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| FirstName | String | private | Can be any string. It can’t contain any specific character and integer |
| LastName | String | private | Can be any string. It can’t contain any specific character and integer |
| UserName | String | private | Can be any string. It can’t contain any specific character and integer |
| Password: | String | private | Can be any string and NULL |
| UserId | String | private | Can be any string and NULL |
| Balance | float | private | Should be greater than or equal to 0 |
| Booked | Ticket[\*] | private | Can be array of tickets of any size |

Table 11 Attribute description of user class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| loadAccountInformation | public | void | void | Logged in | Account information will be loaded and stored on attributes |
| Book | public | void | Ticket | Movie list available | Ticket for the movie will booked |
| recharge | public | Boolean | cardNumber | Account is presented | Account will be recharged |
| updateUserInfo | public | Void | FirstName LastName Password | Account is presented | Account information will be updated |
| deactivate | public | boolean | void | Account is presented | Account will be deactivated |

Table 12 operation description of user class

|  |
| --- |
| **CinemaAdmin** |
| * FirstName: String * LastName: String * UserName: String * Password: String * UserId: String * CinemaId: String |
| + addSchedule (newschedule: Schedule): void  + updateSchedule (newschedule: Schedule): void  + getBookings (day: String, time: String): Ticket [\*] |

Table 13 cinemaAdmin class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| FirstName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| LastName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| UserName | String | private | Can be any string but never NULL |
| Password: | String | private | Can be any string but never NULL |
| UserId | String | private | Can be any string but never NULL |
| CinemaId | String | private | Can be any string but never NULL |

Table 14 Attribute description of cinemaAdmin class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| addSchedule | public | void | newschedule |  | New schedule for the specified cinema will be added. |
| updateSchedule | public | void | newschedule |  | updated schedule for the specified cinema will be recorded. |
| getBookings | public | Ticket[\*] | Day  Time |  | Booked tickets for specified day and time will be retrieved |

Table 15 operation description of cinemaAdmin class

|  |
| --- |
| **SystemAdmin** |
| * FirstName: String * LastName: String * UserName: String * Password: String * UserId: String |
| + addMovie (newmovie: Movie): void  + removeMovie (movieId: String): void  + addNews (newNews: News): void  + removeNews (newsId: String): void |

Table 16 system admin class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| FirstName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| LastName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| UserName | String | private | Can be any string but never NULL |
| Password: | String | private | Can be any string but never NULL |
| UserId | String | private | Can be any string but never NULL |

Table 17 Attribute description of system admin class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| addMovie | public | void | newmovie |  | New movie will be added |
| removeMovie | public | void | movieId |  | Chosen movie will be removed |
| addNews | public | void | newNews |  | A new movie will be added |
| removeNews | Public | void | newsId |  | Chosen news will be removed |

Table 18 operation description of system admin class

|  |
| --- |
| **UserManager** |
|  |
| + addNormalUser (newUser: User): void  + removeNormalUser (userId: String): void  + addCinemaAdmin (newUser: CinemaAdmin): void  + removeCinemaAdmin (userId: String): void  + authenticate (UserName: String, Password: String): void |

Table 19 userManager table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| addNormalUser | public | void | newUser |  | New user will be added |
| removeNormalUser | public | void | userId |  | Chosen user will be removed |
| addCinemaAdmin | public | void | CinemaAdmin |  | A new CinemaAdmin will be added |
| removeCinemaAdmin | Public | void | newsId |  | Chosen CinemaAdmin will be removed |
| authenticate | Public | void | UserName  Password |  |  |

Table 20 operation description of usermanager class

|  |
| --- |
| **Schedule** |
| * Day: String * Time: String * CinemaName: String * CinemaId: String * MovieName: String * MovieId: String |
|  |

Table 21 schedule class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| Day | String | private | Can be any string but never NULL |
| Time | String | private | Can be any string but never NULL |
| CinemaName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| CinemaId: | String | private | Can be any string but never NULL |
| MovieName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| MovieId | String | private | Can be any string but never NULL |

Table 22 Attribute description of schedule class

|  |
| --- |
| **Payment** |
|  |
| + generateCards (): void  - validateCardNumber (): Card  + recharge (userId, Card): void  + transfer (fromId: String, toID: String, amount): boolean |

Table 23 payment class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **Visiblity** | **Return type** | **Arguments** | **Pre-condition** | **Post condition** |
| generateCards | public | void | void | Logged in | Account information will be loaded and stored on attributes |
| validateCardNumber | private | void | void | Movie list available | Ticket for the movie will booked |
| recharge | public | Boolean | userId  cardNumber | Account is presented | Account will be recharged |
| transfer | public | Void | FirstName LastName Password | Account is presented | Account information will be updated |

Table 24 Attribute description of payment class

|  |
| --- |
| **Card** |
| * Id: String * cardNumber: String * amount: integer |
|  |

Table 25 card class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| Id | String | private | Can be any string but never NULL |
| cardNumber | String | private | Can be any string but never NULL |
| amount | integer | private | Can be positive integer of multiple of 10 |

Table 26 Attribute description of card class

|  |
| --- |
| **Ticket** |
| * numberOfseats: integer * cinemaId: Integer * cinemaName: String * movieId: String * movieName: String * time: String |
|  |

Table 27 Ticket class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Invariant** |
| numberOfseats | integer | private | Can be any string but never NULL |
| cinemaId | Integer | private | Can be any string but never NULL |
| cinemaName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| movieId | String | private | Can be any string but never NULL |
| movieName | String | private | Can be any string but never be NULL. It can’t contain any specific character and integer |
| time | String | private | Can be any string but never NULL |

Table 28 Attribute description of Ticket

# **Reference**

## Bibliography

* Abel Mandefro**,** Amanuel Engeda, Biruk Adera, Helina Girmay, Nathan Tsegaye, Simon Asfaw, Yabetse Genene (May 2016), Software Design Specification: Library Management System.
* Anteneh Behailu, Essey Bisrat, Natnael Awoke, Newal Abrar, Salhadin Bilal, Yehualashet Abebe, Tabor Nekatibeb (May 2016), Software Design Specification: Selamawi