SPRING 2023

CSE3044 SOFTWARE ENGINEERING

TERM PROJECT GROUP 10

CineMap Design Specification Document

12.05.2023

❖ Project Team Members:

➢ 150119678 - Asaf Talha Gültekin

➢ 150119066 - Ertan Karaoğlu

➢ 150119039 - Emir Said Haliloğlu

➢ 150117032 - Fatih Akgündüz

➢ 150118015 - Hasan Fatih Başar

➢ 150118024 - Oruç Berat Turan

➢ 150119738 - Sefa Görkem Keçeci

➢ 150119858 - Yasin Çörekci

Prepared for.
CSE3044 Software Engineering Term Project
Instructor: Associate Professor Borahan TÜMER

Table of Contents

1. Introduction

Our app uses a powerful algorithm that analyzes your viewing history, rating patterns, and other factors to suggest movies that you're most likely to enjoy. If you are looking to expand your movie horizons and discover new favorites, our movie suggestion software app is the perfect tool for you.

1.1 Purpose

One of the significant advantages of a movie suggestion application is its ability to offer personalized recommendations based on an individual's viewing history and preferences. By analyzing the user's past movie choices, the application can suggest films that are more likely to be enjoyable. This feature not only saves time but also ensures that the user is exposed to movies they may not have discovered otherwise.

The primary objective of this application is to provide tailored recommendations that align with the user's movie preferences.

1.2 Statement of scope

Overview:

The purpose of this project is to recommend movies to users based on their viewing history.

Features:

The primary goal of the app is to provide users with personalized movie recommendations based on their viewing history, ratings, and genre preferences. The app will have the following features:

- User registration and profile creation
- Detailed information about movie cast and crew
- In-depth movie details
- Personalized movie suggestions based on user preferences and viewing history.

Platforms:

The Movie Suggestion App will be available on the iOS platform and compatible with a wide range of iOS devices.

Conclusion:

The main objective of the Movie Suggestion App is to provide users with a seamless and personalized movie discovery experience. With a user-friendly interface and personalized recommendations, the app aims to help users find the perfect movie that matches their preferences.

1.3 Software Context

The software of this app will ensure that users receive accurate movie recommendations based on their viewing history and preferred categories. Users will be able to search for their favorite movies, remove them if desired, and view additional details about the movies.

Overall, the Movie Suggestion App aims to provide a personalized and user-friendly experience for discovering new movies that match each user's individual preferences and viewing history.

1.4 Major Constraints

- **Device/Budget:** Some of us may not have access to a macOS-operated computer, which can hinder development efforts. When using virtual machines, we may encounter dependency issues.
- **Database:** Due to budget constraints, we are utilizing a cloud-based database and making calls to it via API keys.
- **Time:** It is imperative that we complete each document within its designated deadline, and that we finish the overall project before its deadline.

1.5 Definitions

- **User:** An individual seeking information about various movies and their respective casts or seeking recommendations for new movies to watch.
- **Swift:** Swift is a programming language utilized for development purposes.
- **The Movie Database API:** The Movie Database API is a cloud-based database that we have implemented for our project.
- **Movie:** A form of entertainment that the user is interested in learning more about.
- **Cast:** The collective group of actors who are featured in each film or stage production.

1.6 Acronyms and Abbreviations

- **SW:** Software
- **UCD:** Use Case Diagram
- **UI:** User Interface
- **API:** Application Programming Interface
- **GUI:** Graphical User Interface

1.7 References

- M. Kaya, "CSE 3044 Lecture Notes," Marmara University, Istanbul, Turkey, Accessed April 14, 2023. [Online]. Available: https://mimoza.marmara.edu.tr/~mehmet.kaya/Courses/CSE3044/lecturenotes.html
- Sommerville, "Software Engineering," 8th ed. Addison-Wesley, 2007.

2. Design Consideration

2.1 Design Assumptions and Dependencies

1) Design Assumptions:

a) **Platform:** The application is intended to be developed for use on iOS.

b) **User Interface (UI) Design:** The UI will be created with a vibrant color palette and will be designed to capture the user's attention. Additionally, it will be kept simple and easy to navigate to provide practicality for the user.

c) Accessibility: The app will be designed with accessibility in mind, following Apple's guidelines for accessibility.

2) Dependencies:

a) iOS Device Compatibility: For the application to work consistently across different iOS devices, the design of the application will adhere to the specified version.

b) App Store Eligibility: The application must comply with Apple's specified obligations to be deemed eligible for the App Store.

2.2 General Constraints

1) Platform Limitations: iOS applications must adhere to the guidelines set forth by Apple.

2) Device Compatibility: The application should be optimized for seamless use across various iOS devices.

3) Security and Privacy: iOS applications must conform to stringent security and privacy standards, including best practices for safeguarding user data and implementing proper authentication and authorization protocols.

2.3 System Environment

The mobile application has been developed on a MacOS operating system, utilizing a MacBook device with a minimum of 8GB of RAM. The application was built using the Swift programming language, version 5, and developed using the XCode integrated development environment (IDE). Our application is intended for use on iPhone and iPad devices and relies on a third-party API for information retrieval. We also employ a local database for data storage and make use of the Kingfisher Swift package.

2.4 Development Methods

We have decided to use GitHub for code hosting, version control, and collaboration purposes. The incremental model is the most suitable approach for our project, as we plan to develop it by breaking it down into smaller portions. At each iteration, we will be implementing a new feature. Our group comprises of eight members, and the iterative model is the best and quickest way to develop our project.

To work efficiently, we have created a Discord server and a WhatsApp group. Additionally, we hold weekly meetings on Discord. During the implementation phase, all team members utilize Git's branch feature. We ensure that all changes are reviewed by the entire team before merging into the main branch. If we encounter a bug, we make every effort to resolve it as a group. However, if the bug does not impede the application's functionality, we plan to deal with it later to avoid getting bogged down in minor details. Our goal is to develop an MVP quickly to obtain feedback.

3. ARCHITECTURAL AND COMPONENT-LEVEL DESIGN

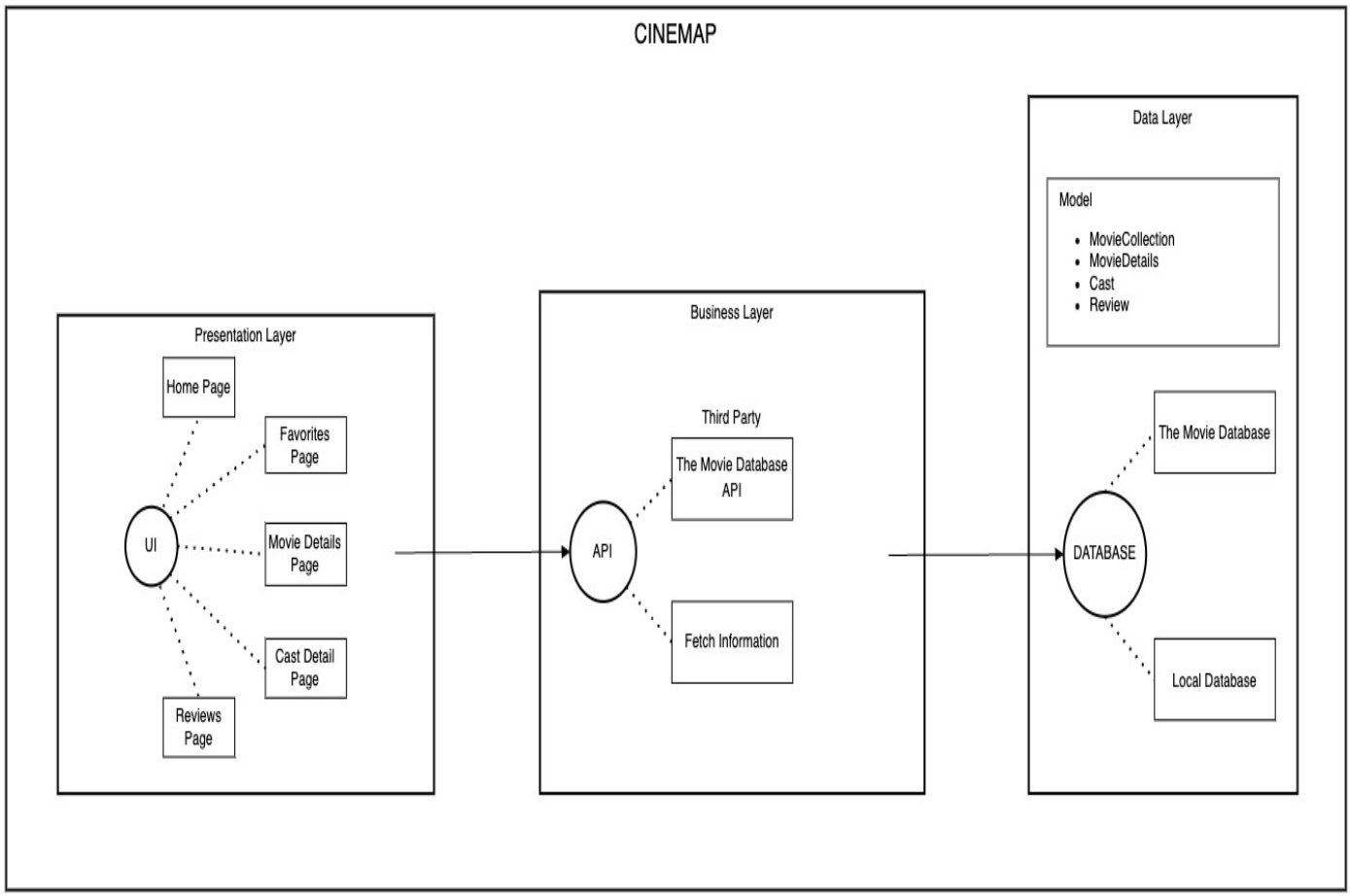
3.1 System Structure

Firstly, our software is a project developed using Swift, an iOS mobile application development language. On the homepage, popular movies will be recommended, which can be clicked on to view more details. Users will be able to search for different types of movies and individual actors from Hollywood. Movies will appear with small covers, along with their names, release dates, and IMDB ratings, which will be gathered using an API. Users will be able to mark their favorite movies, which will be saved and can be accessed at any time. Users will also be able to perform an advanced search, where movies will be categorized.

If a movie is chosen, more detailed information about the movie will be displayed, along with a cover photo. Further information such as the duration, production company, budget, revenue, and an overview of the movie will also be available. Users will also have the option to search for actors by entering their name and viewing details about them along with a small picture. The actors' information, including their birthdate, place of birth, and brief information about their life and the movies they've acted in will be available. Additionally, movies in which the actor has played a role will be recommended to the user.

I hope this revised version meets your needs. Let me know if you have any further questions or if there's anything else I can assist you with.

3.1.1 Architecture diagram



3.2 Description for Component N

3.2.1 Description of the Cast Details Screen

Identification	Cast Details
Type	Package
Purpose	The purpose of the cast details screen is to give information details about the selected actor of the movie. It contains information about actor's name, image, birthdate, death date, place of birth, biography, and movies that he/she played.
Subordinates	This screen does not contain links to the other screens.

Dependencies	The following screen links to this screen ● Movie Details Screen
Resources	API call for rendering each component's content
Processing	The process is going through with the movie details page. When user clicks on an actor, page navigates to cast details page and, in the process, it does API call and takes all information from database. After receiving information, it renders them and shows on the screen properly.
Data	The data is coming from database which is called in API.

3.2.1.1. Processing narrative (PSPEC) for component Cast Details Screen

User can view the details of the cast of the movie that he/she clicked earlier. In this screen user can see very detailed actor/actress' information. User can go back to Movie Details Page

3.2.1.2. Component Cast Details Screen Processing Detail

3.2.1.2.1. Restrictions/limitations for component

User only can see the actor/actress which belongs to the movie. Besides that, the user has no further restrictions.

3.2.1.2.2 Performance issues for component

At first, User may see a circular indicator which means that app is fetching data from API. As long as it's finished, the user will be able to see the details of the movie.

3.2.1.2.3 Design constraints for component

The design of activity pages is such that they can adapt to iOS devices and all the buttons and actions are represented by icons that have clear meanings.

3.2.1.2.4 Processing Detail for Each Operation of Component

Click on Actor/Actress: User can see Actor/Actress' info.

3.2.2 Description of the Favorites Page Screen

Identification	Favorites Page
Type	Package

Purpose	The "Favorite Page" feature on CineMap allows users to save and organize a list of their favorite movies, TV shows, and other content. This feature is designed to help users keep track of the titles they enjoy and make it easier for them to find and recommend content to others.
Subordinates	This screen contains links to the following screens. ● Saved Favorite and Popular Movie Page ● Actors & Actress personal data page ● Movie information in more detail page
Dependencies	The following screen links to this screen ● The initial screen is the Homepage
Resources	API call for fetching data.
Processing	Process is before the initial screen is displayed to the user, it fetches the data for popular movies and prepares it to display. After the user picks a movie, it redirects them to another page and shows details about the movie. Users can also decide to search for a specific movie or an individual actor. This page can also redirect them to actors detailed page. Search mode can also be cancelled by using the cancel button.
Data	The data is coming from database via an API. A local database in the system for all movies.

3.2.2.1. Processing narrative (PSPEC) for component Favorites Page Screen

User can view the movies that he/she added to favorites. Here, the user can click on the movies and check the details of the movie that he/she clicked. User can go back to homepage or where he/she last before coming to that page.

3.2.2.2. Component Favorites Page Screen Processing Detail

3.2.2.2.1. Restrictions/limitations for component

User only can see the movies that he/she added to favorites.

3.2.2.2.2 Performance issues for component

Favorites database is not fetched from API, it is local db in the machine which is kept as list in structure.

3.2.2.2.3 Design constraints for component

The design of activity pages is such that they can adapt to iOS devices and all the buttons and actions are represented by icons that have clear meanings.

3.2.4.2.4 Processing Detail for Each Operation of Component

Click on Movie: User can see movie's info.

3.2.3 Description of the Home Page Screen

Identification	Home Page
Type	Package
Purpose	The Home Page will initially display popular movies which will be recommended by the system. The user can pick any movie, it redirects the user to another link, and will show further information. The user will also be able to search for specific actors & movies.
Subordinates	This screen contains links to the following screens. ● Saved Favorite Movie Page ● Actors & Actress personal data page ● Movie information in more detail page
Dependencies	The following screen links to this screen ● The initial screen is the Homepage
Resources	API call for fetching data.

Processing	Process is before the initial screen is displayed to the user, it fetches the data for popular movies and prepares it to display. After the user picks a movie, it redirects them to another page and shows details about the movie. Users also can decide to search for a specific movie or an individual actor. This page can also redirect them to actors detailed page. Search mode can also be cancelled by using the cancel button.
Data	The data is coming from database via an API. A local database in the system for the saved favorite movies.

3.2.3.1. Processing narrative (PSPEC) for component Home Page Screen

Users can view a search bar on the top, below the bar user can see various movies and actors. Right below that it has 2 blocks, one for movie and the other one for person to search. In the footer user can see 2 icons which are popular and favorites icon.

3.2.3.2. Component HomePage Screen Processing Detail

3.2.3.2.1. Restrictions/limitations for component

When user clicks on movie and search, he/she only see the movies that is matched with the search keywords. So do for Person. Beside that searching, user will get recommendations as long as user slide to down.

3.2.3.2.2 Performance issues for component

At first, User may see a circular indicator which means that app is fetching data from API. As long as it's finished, the user will be able to see the details of movie.

3.2.3.2.3 Design constraints for component

The design of activity pages is such that they can adapt to iOS devices and all the buttons and actions are represented by icons that have clear meanings.

3.2.3.2.4 Processing Detail for Each Operation of Component

Search in the bar on top: User can search movies, actors etc. in this bar.

Click on Movie: User can view the details of the movie.

Click on Actor/Actress: User can see Actor/Actress' info.

Click on Favorites Icon: User can view the favorite movies by him/herself.

3.2.4 Description of the Movie Details Screen

Identification	Movie Details
Type	Package
Purpose	The main purpose of movie details screen is to give information about the movie such as release date, cast crew, budget, revenue, IMDB rate, production companies, home page navigator URL, overview. It shows all the cast as well. Besides all these there is a side purpose that it has movie recommendation area around the bottom.
Subordinates	This screen contains links to the following screens. ● Home page of the movie ● Each actor/actress' information page ● Movie details page of recommended movies.
Dependencies	The following screen links to this screen ● Home page Screen
Interfaces	The links are contained in the bottom half of the screen.
Resources	API call for rendering each component's content
Processing	The process is going through with the home page. When a user clicks on a movie, the homepage navigates to Movie Details page and in the process, it does API call and takes all information from database. After receiving information, it renders them and shows on the screen properly
Data	The data is coming from a database which is called API. Beside this, in the favorites page we created a local database to not to tire system by doing API call all the time.

3.2.4.1. Processing narrative (PSPEC) for component Movie Details Screen

User can view the details of movie that he/she clicked. In this screen user can see actor/actress' information and when user slides down, he/she can see movie recommendations. In this screen user go back to homepage.

3.2.4.2. Component Movie Details Screen Processing Detail

3.2.4.2.1. Restrictions/limitations for component

User has no restrictions, he/she will get recommendations if user slide to side.

3.2.4.2.2 Performance issues for component

At first, User may see a circular indicator which means that app is fetching data from API. If it's finished, the user will be able to see the details of the movie.

3.2.4.2.3 Design constraints for component

The design of activity pages is such that they can adapt to iOS devices and all the buttons and actions are represented by icons that have clear meanings.

3.2.4.2.4 Processing Detail for Each Operation of Component

Click on Actor/Actress: User can see Actor/Actress' info.

Swipe down and slide side the recommendations: User can see various recommendations for different movies.

3.2.5 Description of the Reviews Page Screen

Identification	Reviews Page
Type	Package
Purpose	The purpose of the Reviews Page screen is that it displays user reviews for a movie. When the screen loads, it fetches the reviews data and displays it in a table view format. The user can scroll through the reviews and read them.
Subordinates	This screen does not contain any link.
Dependencies	The following screen links to this screen ● Movie details Screen
Resources	API call for fetching data.

Processing	The process starts on the movie details page. When user clicks on the “see reviews button, movie details navigate to reviews page and in the process, it does API call and takes all information from database. After receiving information, it renders them and shows on the screen properly
Data	The data is coming from database via an API. Besides, in the favorites page we created a local database to not to overload system by doing API call all the time.

3.2.6 Description of Model Package Screen

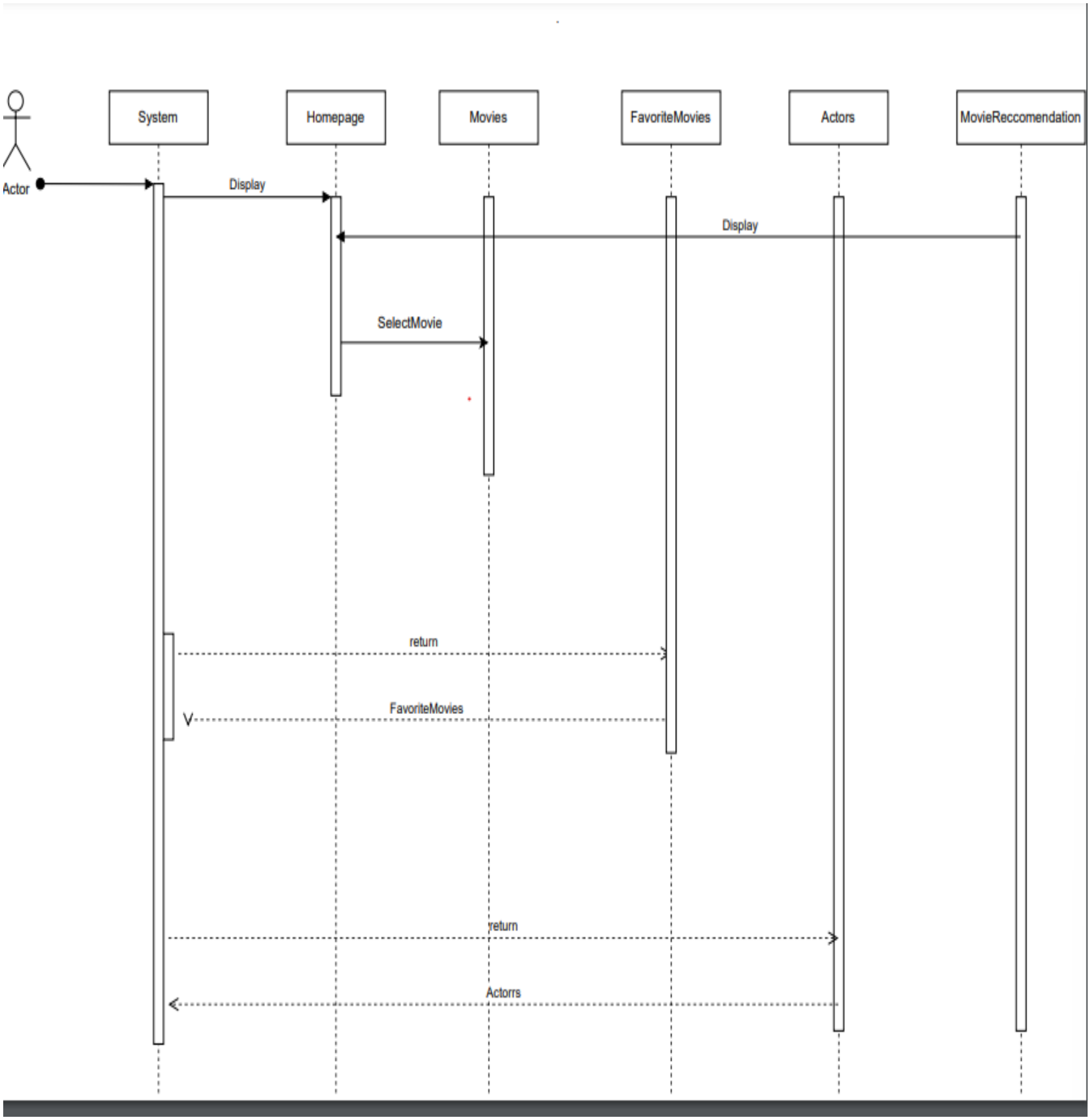
Identification	Model
Type	Package
Purpose	The purpose of this package is to keep data of casts and movies. It also keeps movie reviews simultaneously and matches them with movies.
Subordinates	This package contains. ● Cast ● MovieCollection ● MovieDetails ● Review
Dependencies	None
Resources	API call for rendering each component’s content
Processing	The code defines three Swift structs that conform to the Codable protocol, which allows them to be encoded and decoded to and from JSON format. The ReviewCollection structure represents a collection of reviews and has properties for the ID, page number, total pages, total results, and an array of Review objects. The CodingKeys Enum is used to specify custom keys for encoding and decoding properties that have different names in the JSON data. The package already has movies and cast data. Data is updatable.
Data	The data is coming from database which is called in API.

3.3 Dynamic Behavior for Component N

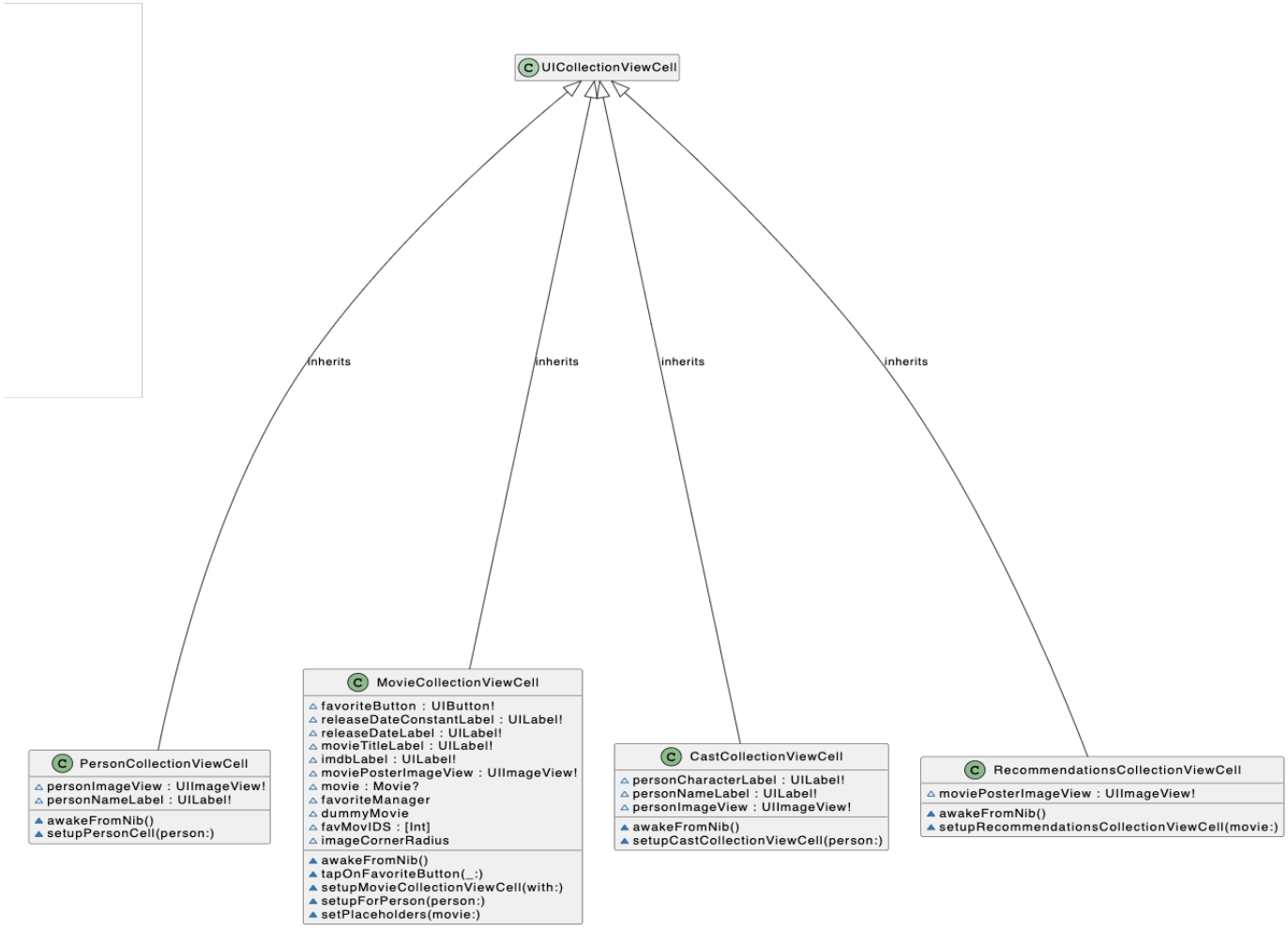
- The interaction between the classes is described.

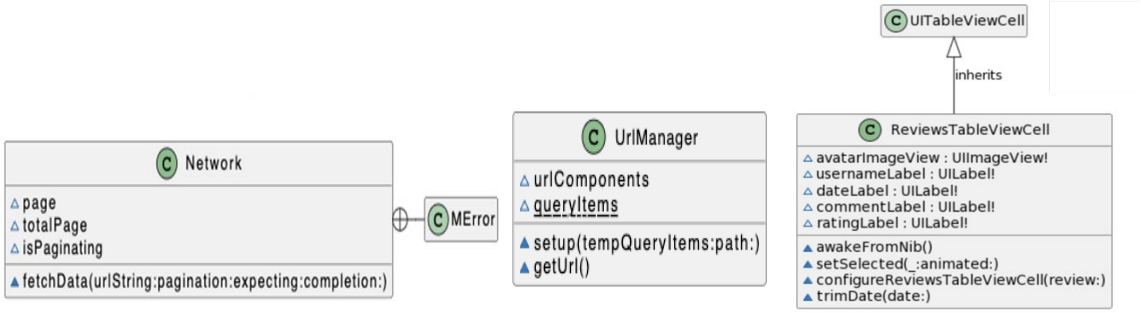
3.3.1 Interaction Diagrams

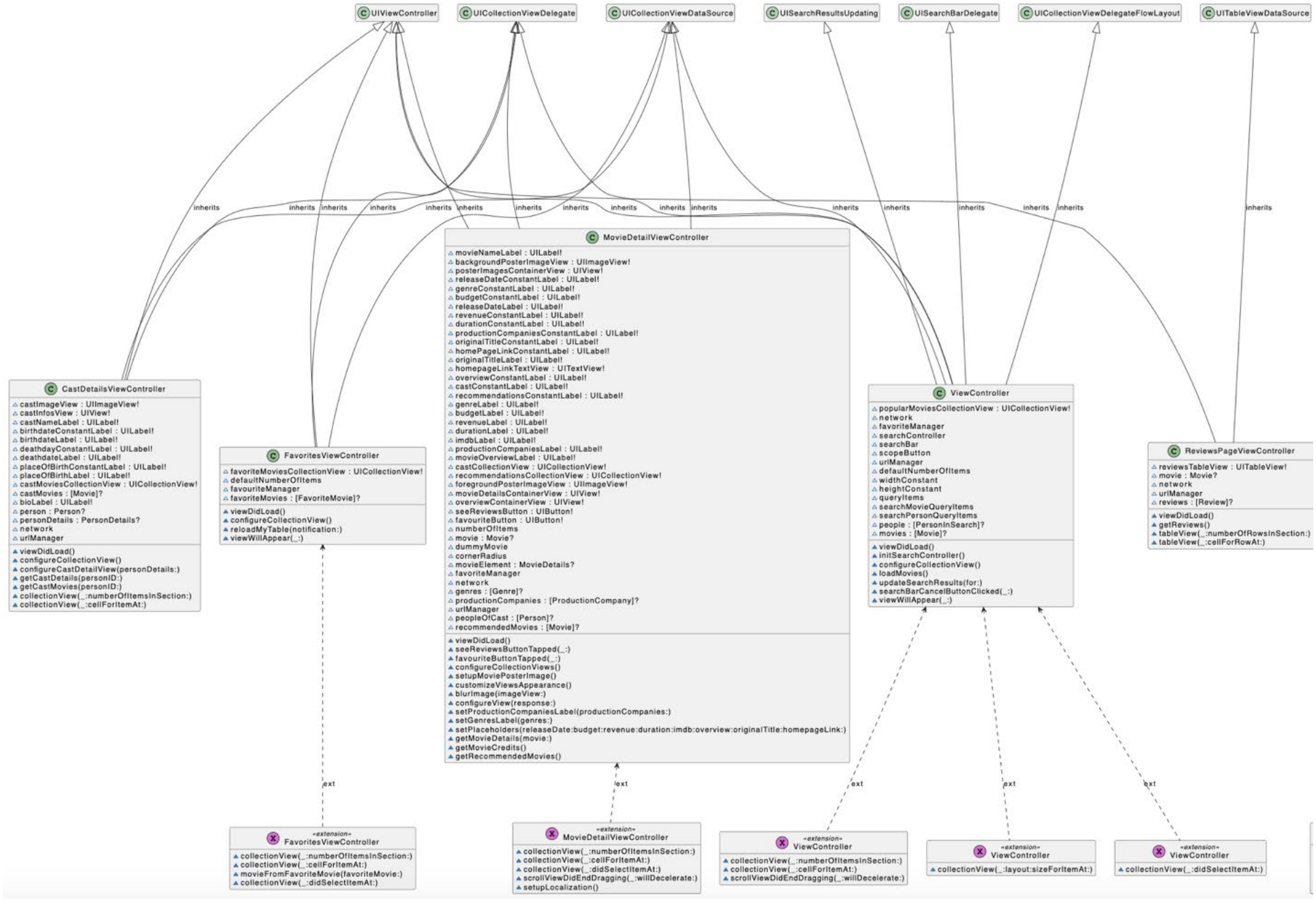
- **System Sequence Diagram (SSD):**



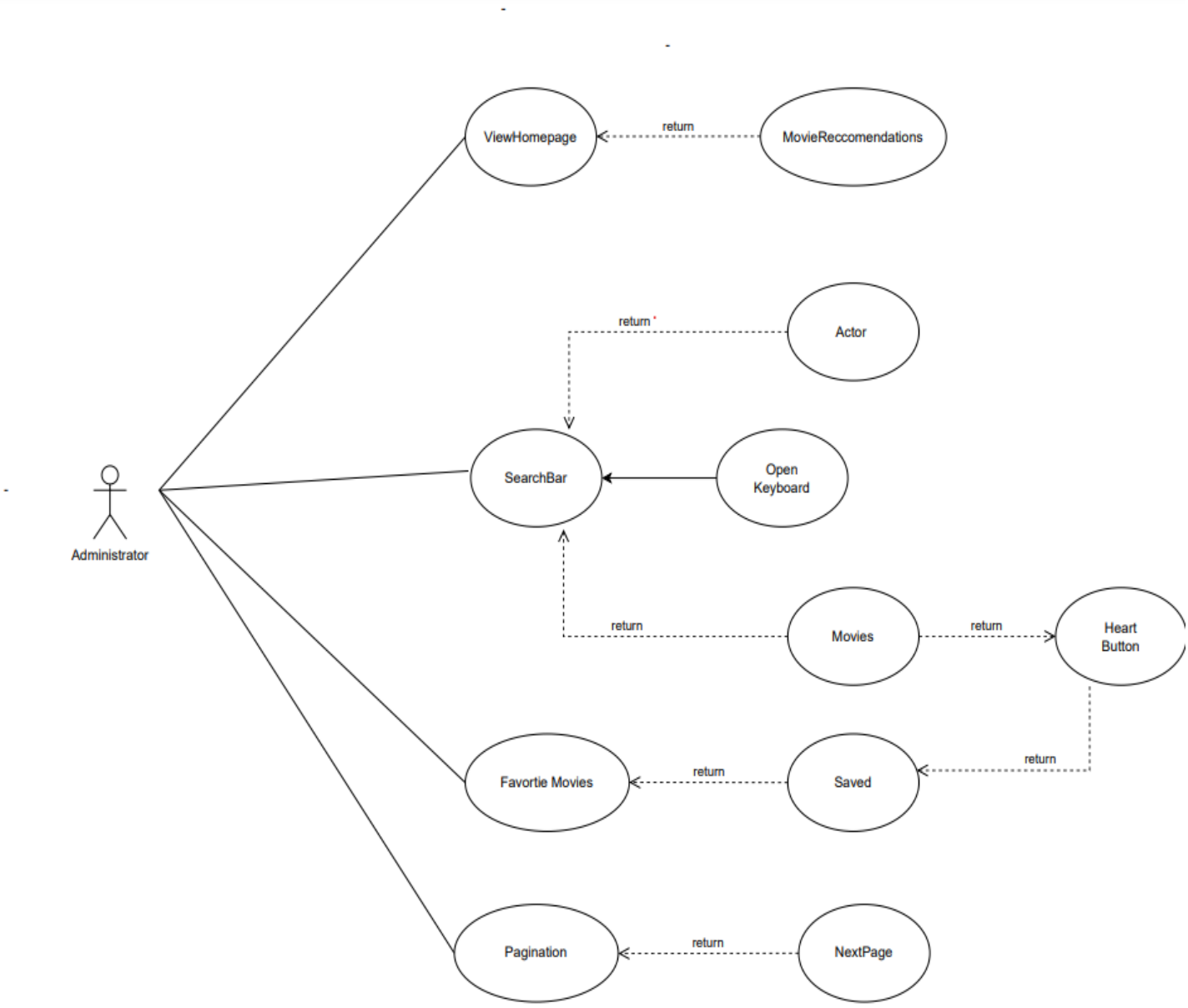
- **UML Diagram:**


UICollectionViewCell
inherits
inherits
inherits
inherits
PersonCollectionViewCell
△ personImageView : UIImageView!
△ personNameLabel : UILabel!
▲ awakeFromNib()
▲ setupPersonCell(person:)
MovieCollectionViewCell
△ favoriteButton : UIButton!
△ releaseDateConstantLabel : UILabel!
△ releaseDateLabel : UILabel!
△ movieTitleLabel : UILabel!
△ imdbLabel : UILabel!
△ moviePosterImageView : UIImageView!
△ movie : Movie?
△ favoriteManager
△ dummyMovie
△ favMovIDS : [Int]
△ imageCornerRadius
▲ awakeFromNib()
▲ tapOnFavoriteButton(_:)
▲ setupMovieCollectionViewCell(with:)
▲ setupForPerson(person:)
▲ setPlaceholders(movie:)
CastCollectionViewCell
△ personCharacterLabel : UILabel!
△ personNameLabel : UILabel!
△ personImageView : UIImageView!
▲ awakeFromNib()
▲ setupCastCollectionViewCell(person:)
RecommendationsCollectionViewCell
△ moviePosterImageView : UIImageView!
▲ awakeFromNib()
▲ setupRecommendationsCollectionViewCell(movie:)

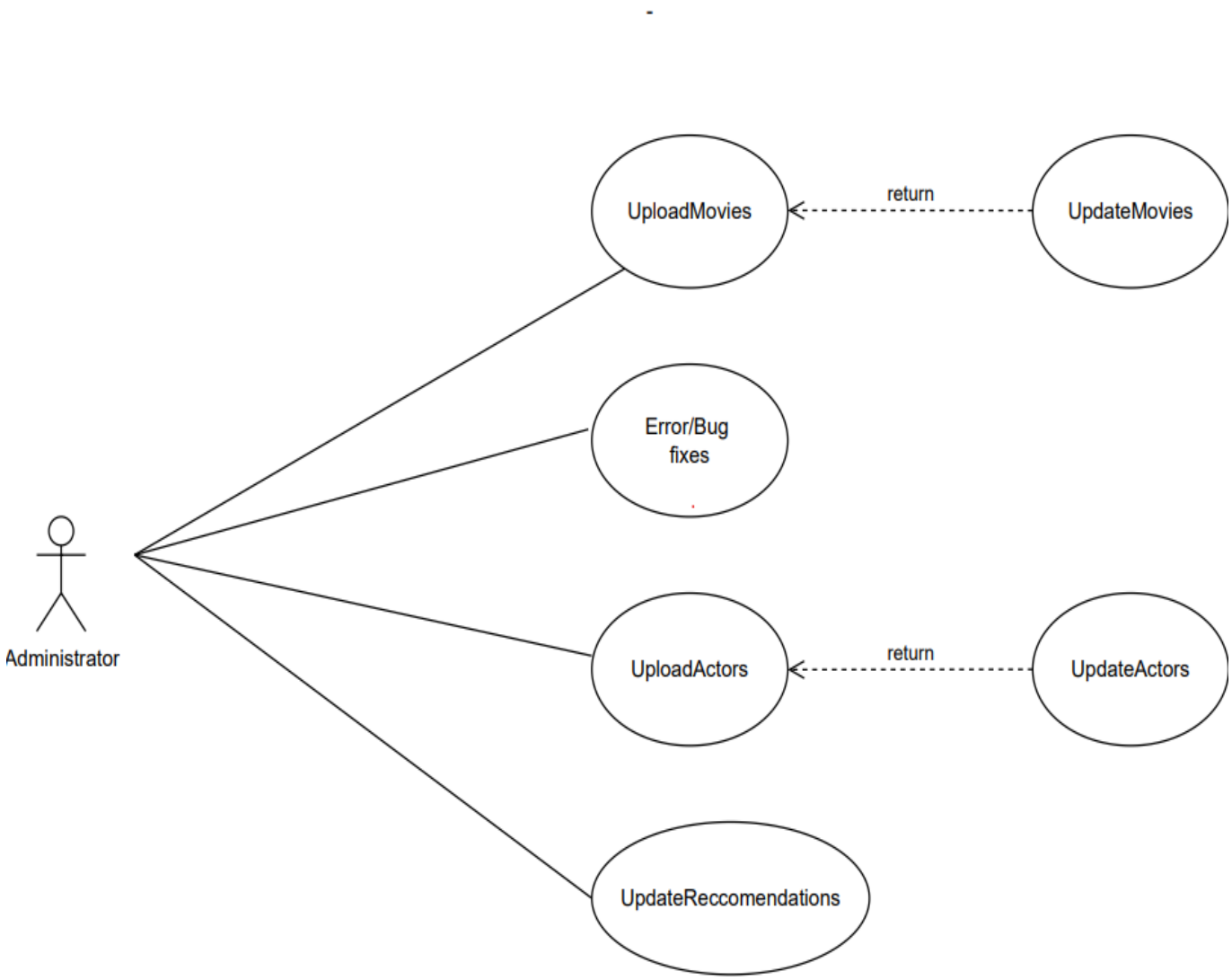

UITableViewCell
inherits
Network
△ page
△ totalPage
△ isPaginating
▲ fetchData(urlString:pagination:expecting:completion:)
MError
UrlManager
△ urlComponents
△ queryItems
▲ setup(tempQueryItems:path:)
▲ getUrl()
ReviewsTableViewCell
△ avatarImageView : UIImageView!
△ usernameLabel : UILabel!
△ dateLabel : UILabel!
△ commentLabel : UILabel!
△ ratingLabel : UILabel!
▲ awakeFromNib()
▲ setSelected(_:animated:)
▲ configureReviewsTableViewCell(review:)
▲ trimDate(date:)


UIViewController
UICollectionViewDelegate
UICollectionViewDataSource
UISearchResultsUpdating
UISearchBarDelegate
UICollectionViewDelegateFlowLayout
UITableViewDataSource
inherits
MovieDetailViewController
movieNameLabel : UILabel!
backgroundPosterImageView : UIImageView!
posterImagesContainerView : UIView!
releaseDateConstantLabel : UILabel!
genreConstantLabel : UILabel!
budgetConstantLabel : UILabel!
releaseDateLabel : UILabel!
revenueConstantLabel : UILabel!
durationConstantLabel : UILabel!
productionCompaniesConstantLabel : UILabel!
originalTitleConstantLabel : UILabel!
homePageLinkConstantLabel : UILabel!
originalTitleLabel : UILabel!
homepageLinkTextView : UITextView!
overviewConstantLabel : UILabel!
castConstantLabel : UILabel!
recommendationsConstantLabel : UILabel!
genreLabel : UILabel!
budgetLabel : UILabel!
revenueLabel : UILabel!
durationLabel : UILabel!
imdbLabel : UILabel!
productionCompaniesLabel : UILabel!
movieOverviewLabel : UILabel!
castCollectionView : UICollectionView!
recommendationsCollectionView : UICollectionView!
foregroundPosterImageView : UIImageView!
movieDetailsContainerView : UIView!
overviewContainerView : UIView!
seeReviewsButton : UIButton!
favouriteButton : UIButton!
numberOfItems
movie : Movie?
dummyMovie
cornerRadius
movieElement : MovieDetails?
favoriteManager
network
genres : [Genre]?
productionCompanies : [ProductionCompany]?
urlManager
peopleOfCast : [Person]?
recommendedMovies : [Movie]?
viewDidLoad()
seeReviewsButtonTapped(_:)
favouriteButtonTapped(_:)
configureCollectionViews()
setupMoviePosterImage()
customizeViewsAppearance()
blurImage(imageView:)
configureView(response:)
setProductionCompaniesLabel(productionCompanies:)
setGenresLabel(genres:)
setPlaceholders(releaseDate:budget:revenue:duration:imdb:overview:originalTitle:homepageLink:)
getMovieDetails(movie:)
getMovieCredits()
getRecommendedMovies()
CastDetailsViewController
castImageView : UIImageView!
castInfosView : UIView!
castNameLabel : UILabel!
birthdateConstantLabel : UILabel!
birthdateLabel : UILabel!
deathdayConstantLabel : UILabel!
deathdateLabel : UILabel!
placeOfBirthConstantLabel : UILabel!
placeOfBirthLabel : UILabel!
castMoviesCollectionView : UICollectionView!
castMovies : [Movie]?
bioLabel : UILabel!
person : Person?
personDetails : PersonDetails?
network
urlManager
viewDidLoad()
configureCollectionView()
configureCastDetailView(personDetails:)
getCastDetails(personID:)
getCastMovies(personID:)
collectionView(_:numberOfItemsInSection:)
collectionView(_:cellForItemAt:)
FavoritesViewController
favoriteMoviesCollectionView : UICollectionView!
defaultNumberOfItems
favouriteManager
favoriteMovies : [FavoriteMovie]?
viewDidLoad()
configureCollectionView()
reloadMyTable(notification:)
viewWillAppear(_:)
ViewController
popularMoviesCollectionView : UICollectionView!
network
favoriteManager
searchController
searchBar
scopeButton
urlManager
defaultNumberOfItems
widthConstant
heightConstant
queryItems
searchMovieQueryItems
searchPersonQueryItems
people : [PersonInSearch]?
movies : [Movie]?
viewDidLoad()
initSearchController()
configureCollectionView()
loadMovies()
updateSearchResults(for:)
searchBarCancelButtonClicked(_:)
viewWillAppear(_:)
ReviewsPageViewController
reviewsTableView : UITableView!
movie : Movie?
network
urlManager
reviews : [Review]?
viewDidLoad()
getReviews()
tableView(_:numberOfRowsInSection:)
tableView(_:cellForRowAt:)
ext
«extension»
FavoritesViewController
collectionView(_:numberOfItemsInSection:)
collectionView(_:cellForItemAt:)
movieFromFavoriteMovie(favoriteMovie:)
collectionView(_:didSelectItemAt:)
«extension»
MovieDetailViewController
collectionView(_:numberOfItemsInSection:)
collectionView(_:cellForItemAt:)
collectionView(_:didSelectItemAt:)
scrollViewDidEndDragging(_:willDecelerate:)
setupLocalization()
«extension»
ViewController
collectionView(_:numberOfItemsInSection:)
collectionView(_:cellForItemAt:)
scrollViewDidEndDragging(_:willDecelerate:)
«extension»
ViewController
collectionView(_:layout:sizeForItemAt:)
«extension»
ViewController
collectionView(_:didSelectItemAt:)

Use Case Diagram:



Use Case Diagram 2:



4. RESTRICTIONS, LIMITATIONS, AND CONSTRAINTS

4.1 Restriction

- The largest constraint in software development is often the limited availability of time, and when there is a shortage of developers, this can exacerbate the issue. In the current project, there is a defined timeframe within which the development of the Cine Map app must be completed.

4.2 Limitation

- In this case, the huge problem is the budget and technical shortage. Since Swift was used to develop this application, we cannot use it on Android platforms. Moreover, some of the team members don't have a Mac OS, so they had to use a virtual machine, which can't run on iOS devices.

4.3 Constraints

- As constraints, computers are limited by factors such as memory, battery life, and security. Additionally, these devices are unable to adapt to varying screen sizes and orientations.

5. WORK DISTRIBUTION TABLE

- We have regular meetings with our team members. As a result of the meetings, we shared the workload of the project between us. Details are available in the table.

WORK SHARING CHART			
No	**Name Of the Team Member**	**Department**	**Detail**
1)	Asaf Talha Gültekin	Uml Diagrams	Use-Case and UML
2)	Ertan Karaoğlu	Design Consideration	Assumption&Dep, Constraints
3)	Emir Said Haliloğlu	Uml Diagrams	Use-Case and UML
4)	Fatih Akgündüz	Design Consideration	System Env, Dev. Methods
5)	Hasan Fatih Başar	Introduction	Major const., Definitons
6)	Oruç Berat Turan	Uml Diagrams	Use-Case and UML
7)	Sefa Görkem Keçeci	Introduction	Acronyms&Abbreviations
8)	Yasin Çörekci	Conclusion	Use-Case and UML, Restrictions&Limitations&Const.

6. CONCLUSION

Taking everything into consideration, this document covered various design considerations for the project's development cycle. It provided a detailed overview of the system's functionality, its decomposition into components, their design architecture, and how they connect with each other. Additionally, the document discussed the data design and flows, along with user interactions through the design of user interfaces. Furthermore, it outlined the libraries and tools that will be utilized during the system's development and operation. Overall, this document serves as a comprehensive guide for the project's design and development, ensuring the smooth functioning of the system.