Faculty of Computers and Artificial intelligence-
Cairo University (credit hours system)

Cairo University, Faculty of
Computers and Artificial intelligence

Final Year Project 2020

Youtube.com

Subject: Software Modeling

Subject code: SCS252

Project ID: PM-441

Under the supervision of: Dr. Soha Makady
TA. Desoky Abdelqawy

ID	Name	الأسم	Group
20186031	Nada Mohamed	ندي محمد	3
20186007	Ayat Hany	أيات هاني	
20186008	Sarah Khaled	سارة خالد	
20186043	Mark Rofaeel	مارك روفائيل	

Contents

1. Requirements: 3
2. Use case diagram: 4
3. Use case tables: 5
4. Class Diagram: 9
5. Sequence diagrams: 10
6. Updated class diagram (involving 2 design principles): 14
7. Evaluation of class diagram (against all the solid principles): 15

1. Requirements:

1.1 The system shall allow any guest user(not signed-in user) to **sign-into** the system using personal email and password. Once the user signs in, user can easily create a channel, write a description, and add posts to community.

1.2 The system shall allow any user(guest or signed-in user) to click one preferred video at a time to **load** and start watching it all or a part form it.

1.3 The system shall allow only signed-in users to **upload** one video at a time with recommended formatting specifications to their channel.

1.4 The system shall allow only signed-in users to **download** specific videos to their “downloads” list in the system when they load a video.

1.5 The system shall allow only signed-in users to **like** one loaded video at a time but cannot dislike at the same time.

1.6 The system shall allow only signed-in users to **dislike** one loaded video at a time but cannot like at the same time.

1.7 The system shall allow only signed-in users to **write**, edit, delete **comment** on one loaded video at a time.

1.8 The system shall allow only signed-in users to **subscribe** to other channels to see their newly uploaded videos or they can unsubscribe.

1.9 The system shall allow any user (guest or signed-in user) to **share** one video at a time by copying the video URL or via different applications such as Facebook, Twitter, Blogger, etc..

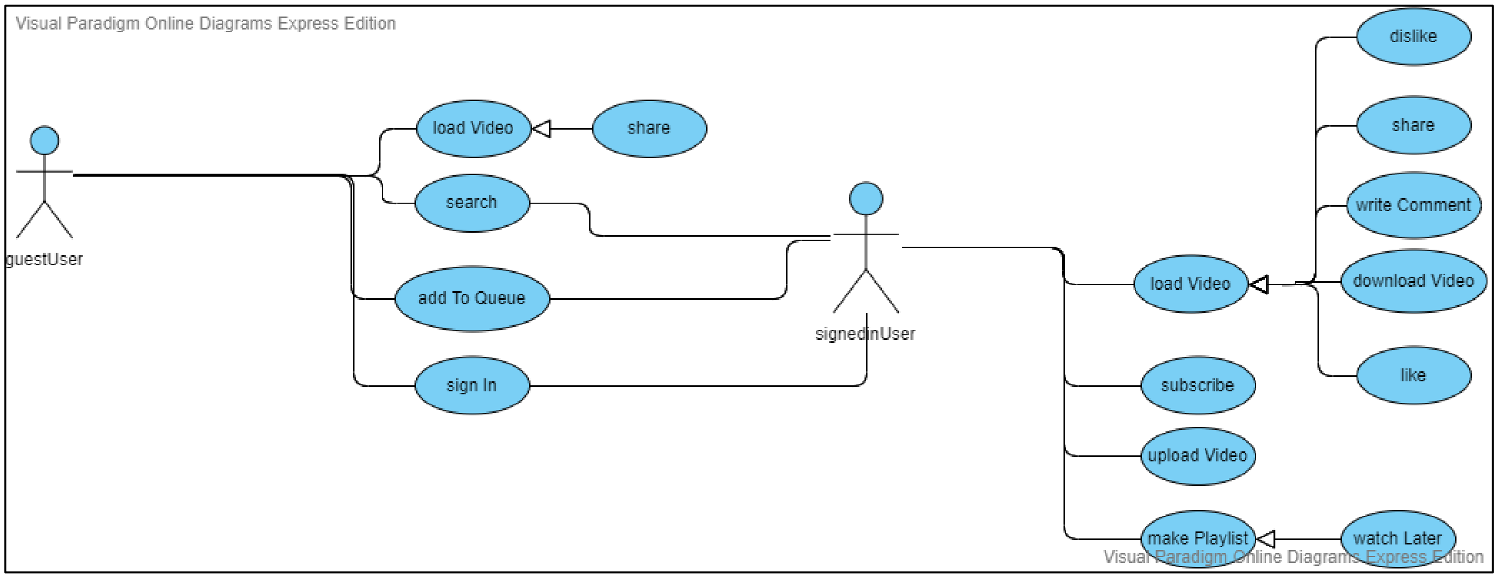
1.10 The system shall allow only signed-in users to **make** their own **playlists**, add one video at a time, or remove one video at a time from that playlist.

1.11 The system shall allow any user (guest or signed-in user) to **search** about specific topic at a time.

1.12 The system shall allow only signed-in users to **add** one video at a time **to watch later** playlist or remove one video at a time.

1.13 The system shall allow any user (guest or signed-in user) to **add** one video at a time **to queue** or clear one video at a time from that queue.

2. Use case diagram:



As shown on the above use case diagram, there are some functionalities that the **guest user** can do such as **search, load video, share, add to queue, sign-in**.

The **signed-in user** can do those functionalities as well (except sign-in, as user has already signed in). Also, the signed-in user can do other functionalities such as **subscribe, upload video, load video** that can allow him to **like, dislike, write comment, download video, share.** Moreover, signed-in user can **make playlists** that can allow him to **"add to watch later"**, which is one type of playlists.

3. Use case tables:

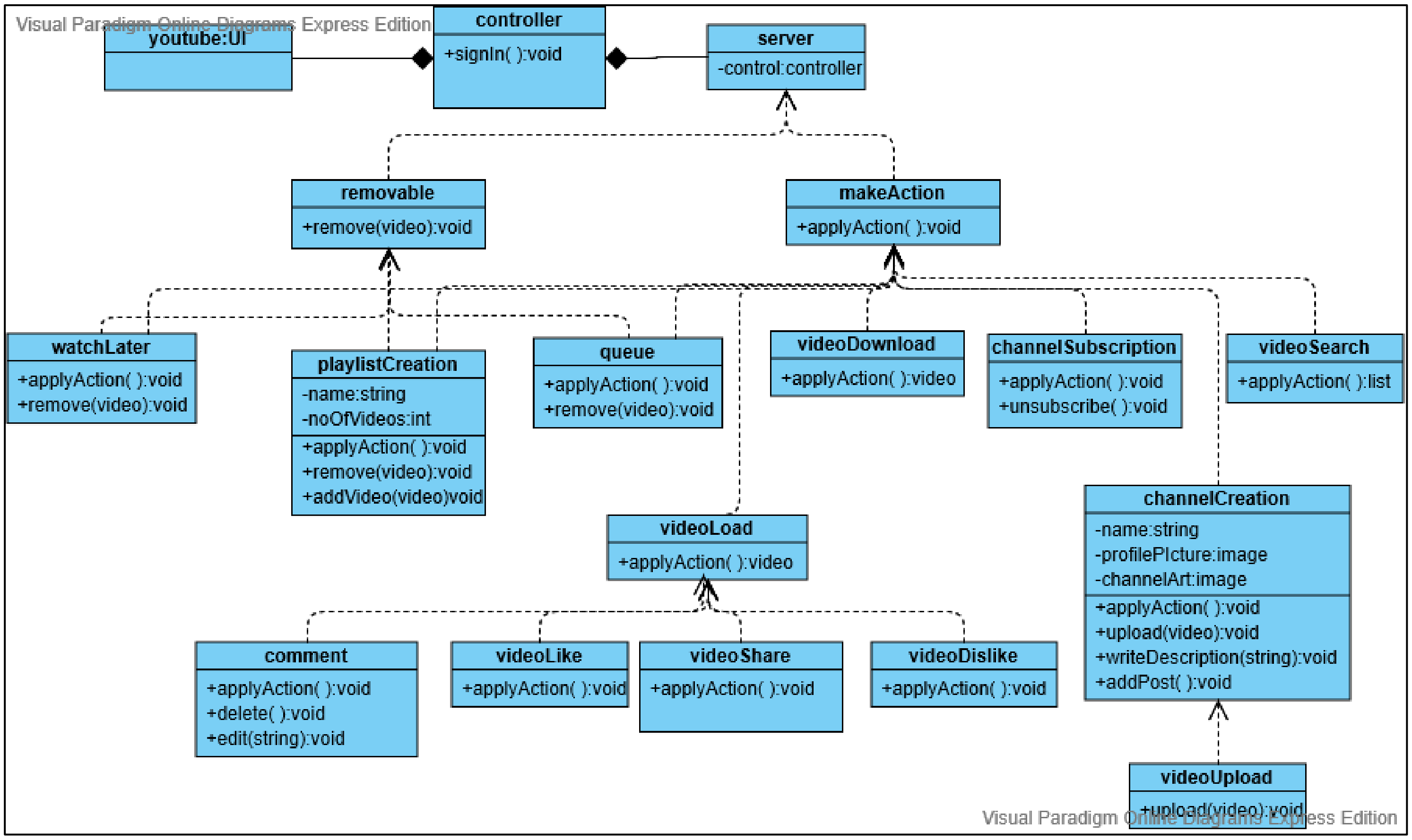
Use Case ID:	1.1	
Use Case Name:	Sign in	
Actors:	Guest user	
Pre-conditions:	Guest user needs to have an active internet connection.	
Post-conditions:	After a successful login, guest user becomes a signed-in user and has a channel on the system.	
Flow of events:	**User Action**	**System Action**
	1- User requests youtube.com system.	
		2- System recognizes user as guest user.
	3- Guest user selects “sign in”.	
		4- System lets guest user fill in his e-mail and password and then verify this data.
	5- Guest user becomes signed-in user.	
		6- System proposes different actions as “Write description” or “Add post”
Exceptions:	**User Action**	**System Action**
	1- Guest user selects “sign in”.	
		2- System lets guest user fill in his e-mail and password and then find that this data is invalid, so it rejects the guest user’s request.
Notes and Issues:	E-mail address should be valid, and password should contain at least 8 characters.	

Use Case ID:	1.13	
Use Case Name:	Add to queue	
Actors:	Guest user and signed-in user	
Pre-conditions:	Both users need to have an active internet connection.	
Post-conditions:	Both users can now add one video at a time to queue or clear one video at a time from that queue.	
Flow of events:	**User Action**	**System Action**
	1- User requests youtube.com system.	
		2- System recognizes user as guest or signed-in user.
	3- User can select “Add to queue” of one of the videos.	
		4- System adds the selected video to the user’s queue.
	5- User can select “Clear from queue” of one of the videos.	
		6- System removes the selected video from the user’s queue.
Exceptions:	**User Action**	**System Action**
	1- User adds several videos to the queue and then closes the system.	
		2- System will not save the queue as it is temporary and will not be retrieved again.
Notes and Issues:	If there was a queue made by the user and the user closed the system, that queue cannot be retrieved again.	

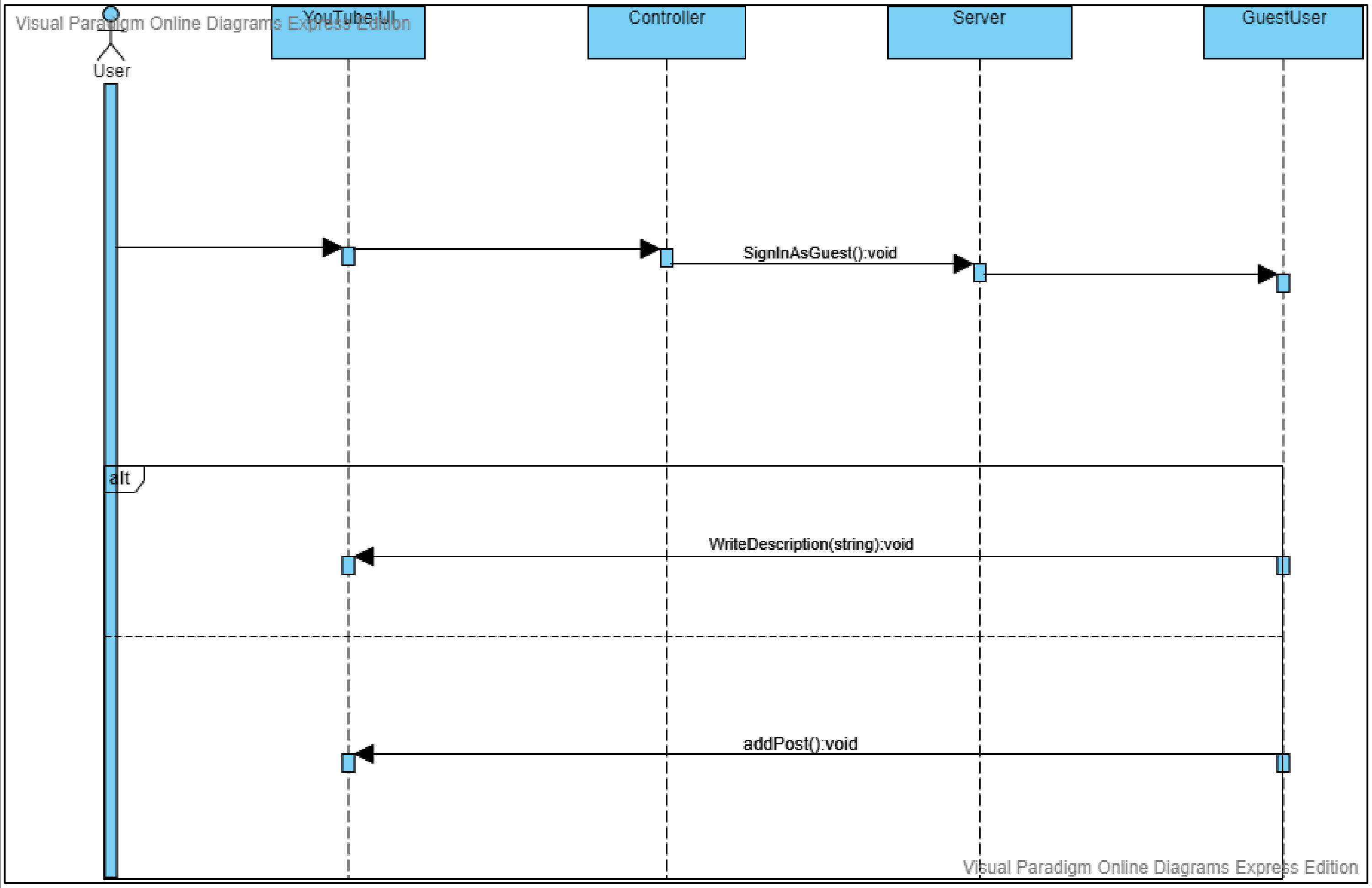
Use Case ID:	1.10	
Use Case Name:	Make playlists	
Actors:	Signed-in users	
Pre-conditions:	Signed-in user needs to have an active internet connection.	
Post-conditions:	The signed-in user can now make their own playlists, adds one video at a time, or removes one video at a time from that playlist.	
Flow of events:	**User Action**	**System Action**
	1- User requests youtube.com system.	
		2- System recognizes user as signed-in user.
	3- Signed-in user selects “Make playlists”	
		4- System creates the playlist.
	5- Signed-in user can select “Add to x playlist” of one of the videos to add it to his specific playlist.	
		6- System adds the selected video to the user’s x playlist.
	7- Signed-in user can select “Remove from x playlist” of one of the videos to add it to his specific playlist.	
		8- System removes the selected video from the user’s x playlist.
Exceptions:	**User Action**	**System Action**
	1- User tries to make the 201st playlist.	
		2- System refuses to make it as number of playlists allowed is 200 only.
Includes:	Watch later	
Notes and Issues:	Signed-in user can only make 200 playlists.	

Use Case ID:	1.7	
Use Case Name:	Write comment	
Actors:	Signed-in user	
Pre-conditions:	Signed-in user needs to have an active internet connection.	
Post-conditions:	The signed-in users can now write comment on one loaded video at a time.	
Flow of events:	**User Action**	**System Action**
	1- User requests youtube.com system.	
		2- System recognizes user as signed-in user.
	3- Signed-in user selects one video to load.	
		4- System opens that video.
	5- Signed-in user can click to write a comment below one of the videos.	
		6- System allows user to add the comment.
	7- Signed-in user can delete the comment he wrote below one of the videos.	
		8- System allows user to delete the comment.
	9- Signed-in user can edit the comment he wrote below one of the videos.	
		10- System allows user to edit comment.
Notes and Issues:	Signed-in user can write, edit, delete comments as long as he like.	

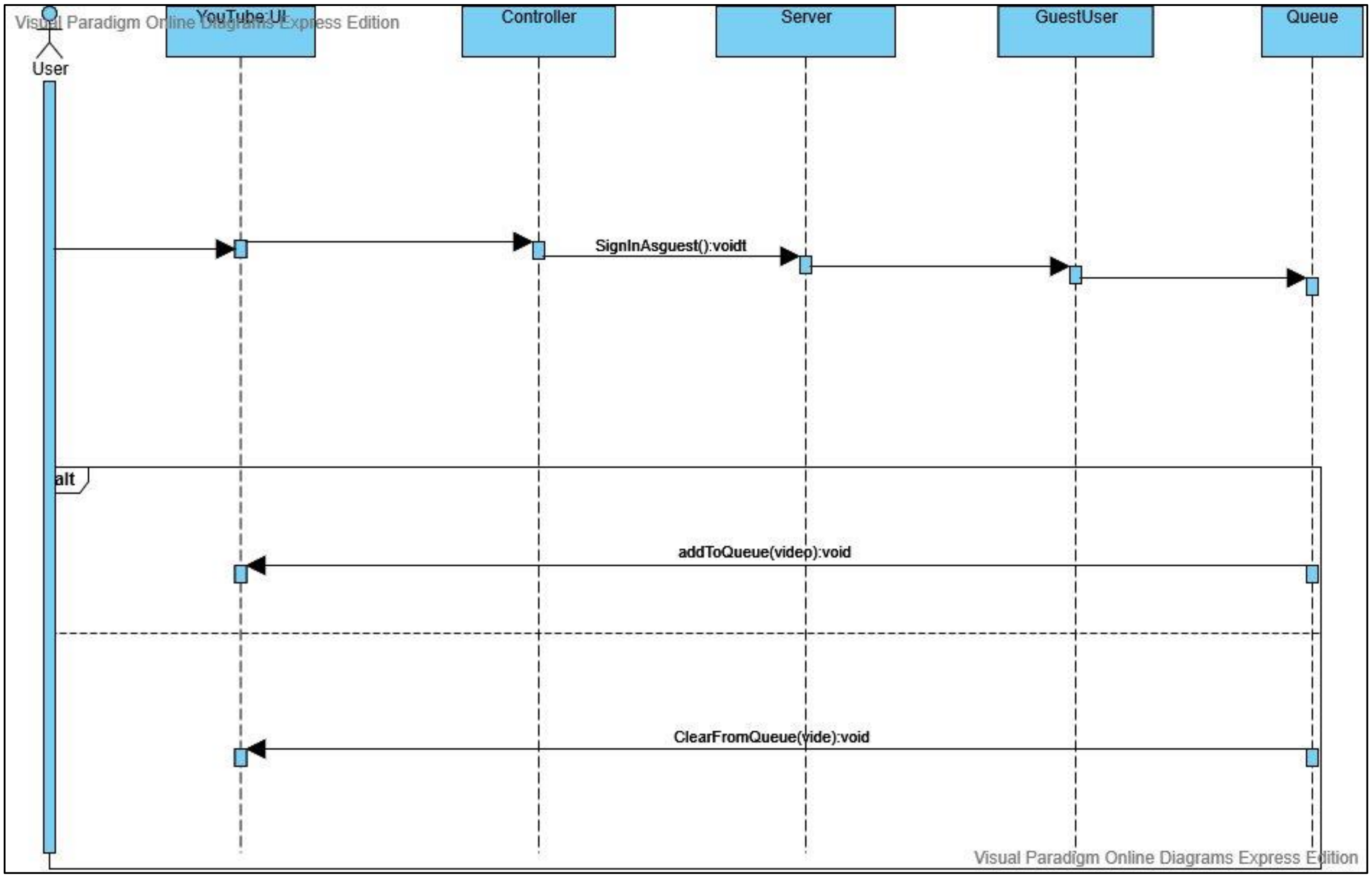
4. Class Diagram:

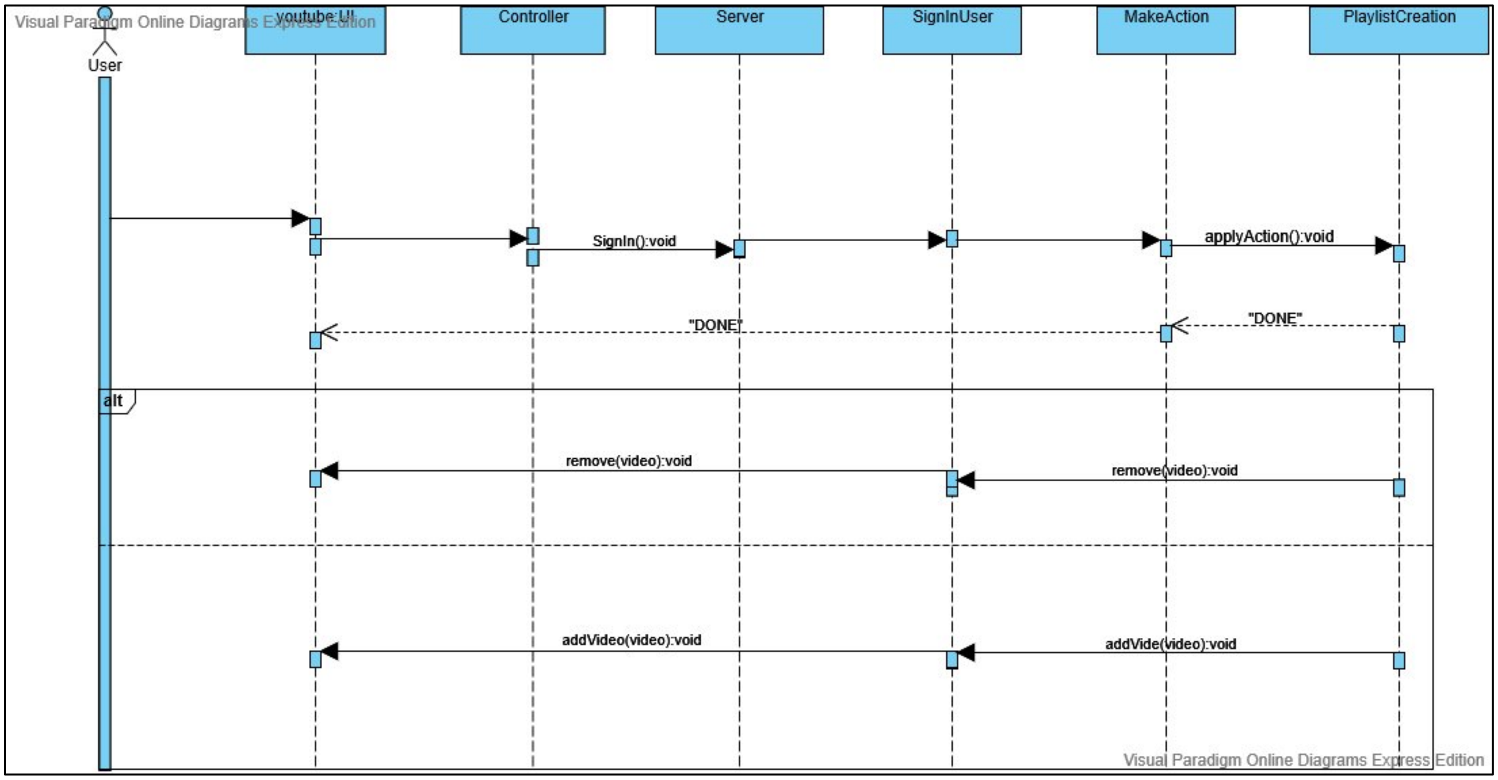


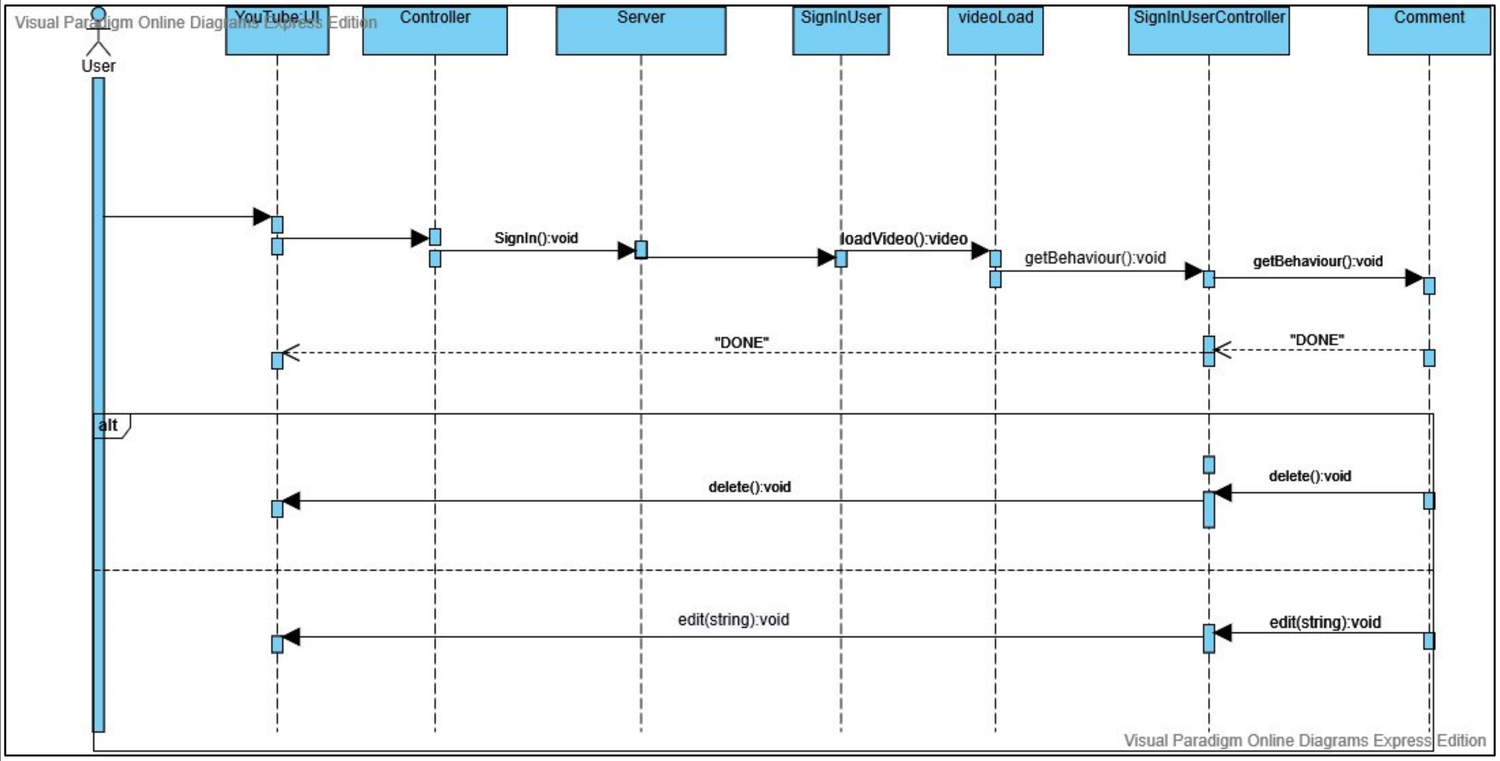
5. Sequence diagrams:



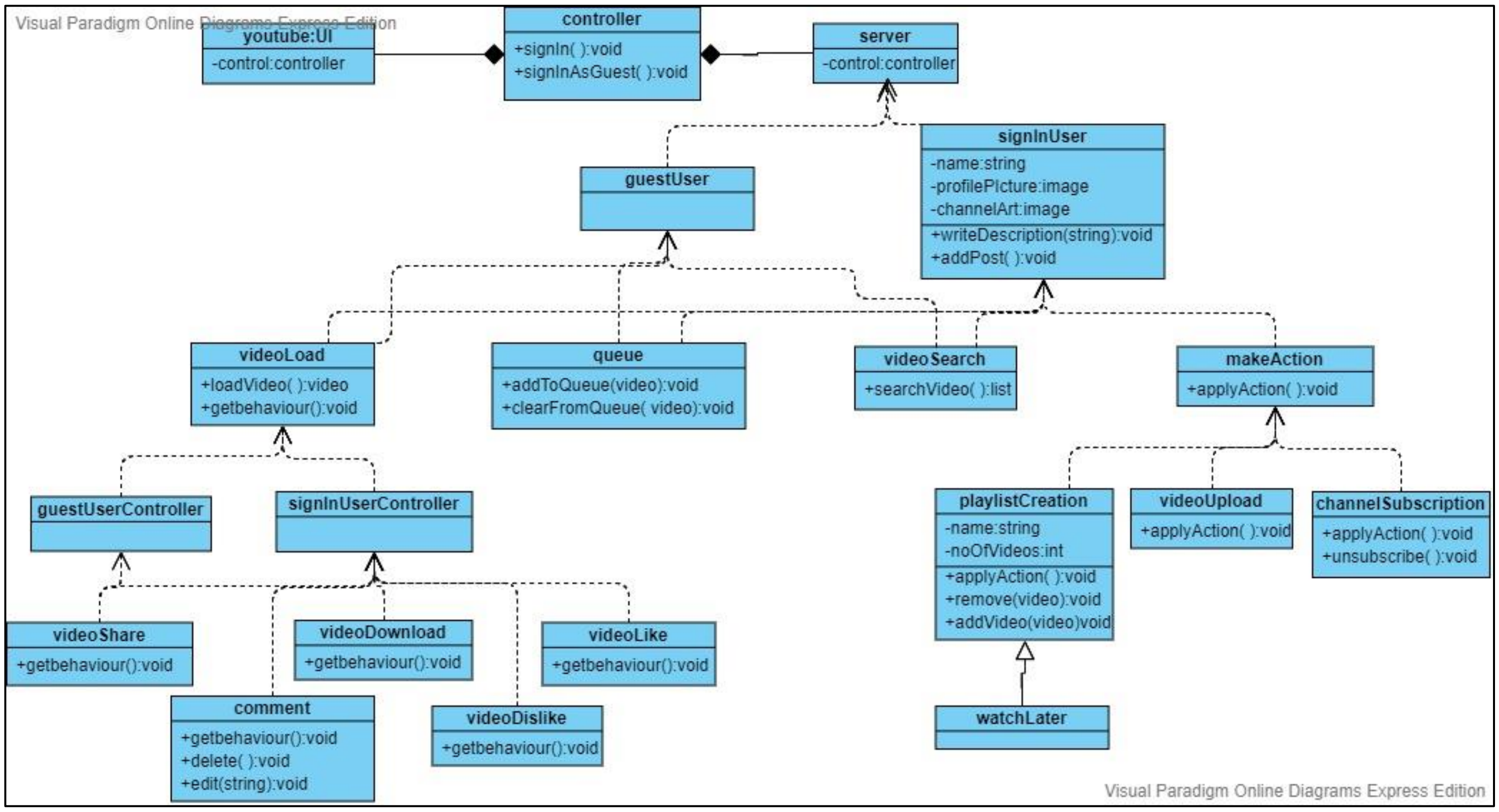
Visual Paradigm Online Diagrams Express Edition
User
YouTube:UI
Controller
Server
GuestUser
SignInAsGuest():void
alt
WriteDescription(string):void
addPost():void
Visual Paradigm Online Diagrams Express Edition


Visual Paradigm Online Diagrams Express Edition
User
YouTube:UI
Controller
Server
GuestUser
Queue
SignInAsguest():voidt
alt
addToQueue(video):void
ClearFromQueue(vide):void
Visual Paradigm Online Diagrams Express Edition


Visual Paradigm Online Diagrams Express Edition
User
youtube:UI
Controller
Server
SignInUser
MakeAction
PlaylistCreation
SignIn():void
applyAction():void
"DONE"
"DONE"
alt
remove(video):void
remove(video):void
addVideo(video):void
addVide(video):void
Visual Paradigm Online Diagrams Express Edition


Visual Paradigm Online Diagrams Express Edition
User
YouTube:UI
Controller
Server
SignInUser
videoLoad
SignInUserController
Comment
SignIn():void
loadVideo():video
getBehaviour():void
getBehaviour():void
"DONE"
"DONE"
alt
delete():void
delete():void
edit(string):void
edit(string):void
Visual Paradigm Online Diagrams Express Edition

6. Updated class diagram (involving at least two design principles):



- The observer pattern: there is one-to-many dependency where **YouTube** is the concrete subject, **guest users** and **signed-in users** are the observers. They are notified and automatically updated, when a new video is uploaded by any of their subscriptions.
  We are coding here to implementation not interface.
  We can add new observers(users) at any time. The only thing the YouTube knows about a user is that it implements a certain interface. Changes to either the YouTube or a user will not affect each other.
  Publisher + subscribers = observer pattern; where the publisher is the **YouTube** and the subscribers are the **guest users** and the **signed-in users**.
- The factory pattern: it encapsulates object (user) creation by letting subclasses decide what objects to create. Here we separated users into guest users and signed-in users. So, on the long run, we can easily create different users with different functionalities.
- The strategy pattern: it is used to perform an operation (or set of operations) in a particular manner. There is different algorithm implementation under the videoLoad which depends on the user's choice (as videoLike, videoDislike, etc). So, on the long run, we can easily create, for example (videoLove or videoAngry) classes with the same functionality without any change in design. Also, we can add (LiveVideo) class that has the same functionality without any change in design that any user can use directly (as queue, videoLoad, etc).

7. Evaluation of class diagram (against all the solid principles):

- **Single-responsibility Principle(SRP):** Every object in our system is having a single responsibility and all the object's services are focused on carrying out that single responsibility, so we applied that principle.

- **Open-closed Principle(OCP):** Classes are open for extension and closed for modification. Because we applied the factory design, we applied that solid principle.

- **Liskov substitution principle(LSP):** Objects in our design are replaceable with instances of their subtypes without changing the correctness of the system. We applied inheritance in (watchLater), and we applied delegation such as in (signInUser/guestuser/makeAction/videoLoad). So, we applied that principle.

- **Interface segregation principle(ISP):** Many users-specific interfaces are better than one general-purpose interface. No user should be forced to depend on methods it does not use, such as the guest user cannot use the methods of the signed-in user. Impact of changes to one interface are not as big. So, we applied that principle.

- **Dependency Inversion principle(DIP):** High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions. We applied that principle because if you notice the queue class or videoLoad class, it has the same implementation for both the guest user and the signed-in user. So, we applied that principle.