

**Confidential**

Robosoft Technologies Pvt. Ltd.

Registered & Corporate Office

217, N.H.66, Santhekatte

New Udupi – 576 105

Karnataka, India

Tel: +91 820 2593930

Fax: +91 820 2593900

**Document Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Authors** | **Reviewed By** | **Approved By** | **Brief Description** |
| 13-Feb-21 | 0.01 | Dona Thomas | <Reviewer>  (One person/ Group Review - PQAL) | - | Initial draft |
| <dd-Mmm-yy> | 0.02 | <Author> | Verified by | - | List the changes incorporated as per review |
| <dd-Mmm-yy> | 0.03 | <Author> | <Customer> | <Customer> | List the changes incorporated as per Customer feedback and sent to Customer for approval |
| <dd-Mmm-yy> | 1.00 |  |  |  | Approved by Customer and baselined |

**Contents**

[**1.**](#_heading=h.2jxsxqh) **High-Level Design 3**

[*1.1*](#_heading=h.z337ya) *Introduction 3*

[1.1.1](#_heading=h.3j2qqm3) Purpose 3

[1.1.2](#_heading=h.1y810tw) Scope 3

[1.1.3](#_heading=h.4i7ojhp) Audience 3

[1.1.4](#_heading=h.2xcytpi) References 3

[1.1.5](#_heading=h.1ci93xb) Terms and Abbreviations 3

[*1.2*](#_heading=h.3whwml4) *Software Product Overview 3*

[*1.3*](#_heading=h.2bn6wsx) *Design Considerations 3*

[1.3.1](#_heading=h.2s8eyo1) Assumptions and dependencies 3

[1.3.2](#_heading=h.qsh70q) Development methods 3

[1.3.3](#_heading=h.3rdcrjn) General constraints 3

[*1.4*](#_heading=h.3as4poj) *Architectural Strategies 3*

[*1.5*](#_heading=h.1pxezwc) *Software Product Architecture 3*

[*1.6*](#_heading=h.49x2ik5) *Component Architecture 3*

[*1.7*](#_heading=h.2p2csry) *Bibliography 3*

[**2.**](#_heading=h.147n2zr) **Reference 3**

**Figures**

# **High-Level Design**

## **Introduction**

### **Purpose**

### **Scope**

### **Audience**

### **References**

### **Terms and Abbreviations**

## **Software Product Overview**

## **Design Considerations**

### **Assumptions and dependencies**

### **Development methods**

### **General constraints**

## **Architectural Strategies**

## **Software Product Architecture**

## **Component Architecture**

## **Bibliography**

# **Reference**

## 

## Software Product Overview

**Client Side Implementation**

* Create a project workspace for the app.
* Create folders that hold the files which differentiate the respective functionalities.
* Create app local DB if needed.
* UI designing for all the app screens(storyboard or xib or both).
* Rest service calls for each functionality and handling the success and the error scenarios.
* Modelling of the data object and converting the REST response to the data models.
* Render the data objects in the screen.
* Add the third party library for sign in using the ‘LinkedIn’, editing the video and other UI components.
* Create an app and provide all the app related information in the appstore/playstore.

**Server-Side Implementation**

Create a database and tables that stores all the app related data.

Setup and deploy the project on the application server.

## 1.3 Design Considerations

**1.3.1 Assumption, dependencies and additional requirements**

* **OS support**: Set the app’s minimum OS support.
* **Supported Devices**: iPhone/iPad or both
* **Design Pattern**: MVVM
* **Folder Structure**: App can be structured based on the functionalities, UI components, Common Utils, Manager classes, data model classes.
* **Data storage**: App sensitive data storage can be done in keychain/files and also in local DB like SQlite using core data or realm.
* **Dependency Manager**: Cocoapods or carthage
* **Third party libraries**: Can be used for interactive UI components, linkedin login, code formatting(like swiftlint), rest service calls(AFNetworking).
* **App Build/Release Support:**
* Create the App and add the app related information in the app store. Also the necessary certificates and provisional profiles need to be created for the development team.
* Automated release tools like fastlane can be used.
* **Additional components** like user activity tracking, video streaming quality tracking, casting the video to other OTT platforms can be added.
* **Reusable components:**
* Video editing component that will accept any video and provides the edited video as the output.
* Login component
* Statistical representation of data.

**API List**

● API for new user SignUp.

● API for LogIn credentials validation.

● API for forgot password functionality.

● API for getting the Hamburger menu

● API for getting the trending videos

● Get a Video Details

● Get Comment details

● Post video comment

● Get playlist

● Create/Update video playlist

● Get video queue

● Create/Update video queue

● Get watched video history

● Get user profile info

● Get user profile statistics info

● Get user top videos info

● Get user top views

● Get user profile analytics statistics info

● Get user profile analytics top videos info

● Get user profile analytics top views

● Get user published video list

● Get user unpublished video list

● Publish video.

API Name :SaveUserSignUpInfo

Type: Post

Description: To save the sign up information like username, password, first name , last name , dob and mobile number.

Request Payload:

{

"username":"",

"password":"",

"firstName":"",

"lastName":"",

"dob":"",

"mobileNumber":""

}

Response

{

"code":200,

"message":"User signed up successfully"

"data":""

}

===================================================================

API Name :validateUserLogin

Type: GET

Description: To check the user credentials and validate whether it's an authenticated user.

Request Payload:

{

"username":"",

"password":"",

}

Response

{

"code":200,

"message":"User login successfully"

"data":""

}

===================================================================

API Name :forgotPassword

Type: POST

Description: To capture the user email id and send a reset link to their registered email or as SMS to their mobile number

Request Payload:

{

"emailId":"",

}

Response

{

"code":200,

"message":"Password reset link sent successfully"

"data":""

}

===================================================================

API Name :resetPassword

Type: POST

Description: To update the password in the database with the new password captured in the Reset password page

Request Payload:

{

"emailId":"",

"newPassword":""

}

Response

{

"code":200,

"message":"Password reset successfully"

"data":""

}

===================================================================

API Name :getHamburgerMenu

Type: GET

Description: To get the menu data after a successful login. Same Api can be used to supply menu for registered and guest users .

Request Payload:

{

"emailId":""

}

Response

{

"code":200,

"message":"Menu data retrieved successfully"

"data":"

[

{name:"","iconImage":"","navigationHandler":""},

{name:"","iconImage":"","navigationHandler":""},

{name:"","iconImage":"","navigationHandler":""}

]"

}

===================================================================

API Name :getTrendingVideoList

Type: GET

Description: To retrieve the trending videos based on a limit

Request Payload:

{

"offset":,

"limit":

}

Response

{

"code":200,

"message":"Trending video data retrieved successfully"

"data":"

[

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

]"

}

===================================================================

API Name :getVideoCategories

Type: GET

Description: To retrieve the video categories

Request Payload:

{

}

Response

{

"code":200,

"message":"Category data retrieved successfully"

"data":"

[

{categoryId:"","description":"","categoryName":""},

{categoryId:"","description":"","categoryName":""},

{categoryId:"","description":"","categoryName":""},

]"

}

===================================================================

API Name :likeVideo

Type: POST

Description: To save the like for the video

Request Payload:

{

"id":123,

"like":1,

"email":""

}

Response

{

"code":200,

"message":"Like saved successfully for the video"

"data":""

}

===================================================================

API Name :commentVideo

Type: POST

Description: To save the comment for the video

Request Payload:

{

"id":123,

"comment":"nice video",

"email":""

}

Response

{

"code":200,

"message":"Comment saved successfully for the video"

"data":""

}

===================================================================

API Name :shareVideo

Type: POST

Description: To save the video sharing details and updating the shared count

Request Payload:

{

"id":123,

"sharedThrough":"",

"sharedTo":"",

"email":""

}

Response

{

"code":200,

"message":"Video sharing details saved successfullyo"

"data":""

}

===================================================================

API Name :getUserVideoQueue

Type: GET

Description: To retrieve the video queue

Request Payload:

{

"email":""

}

Response

{

"code":200,

"message":"Video queue retrieved successfully"

"data":"

[

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

]"

}

===================================================================

API Name :updateVideoQueue

Type: POST

Description: To add video to the users playing queue.

Request Payload:

{

"email":"",

"videoData":"[123,422,344]"

}

Response

{

"code":200,

"message":"Queue updated successfully"

"data":""

===================================================================

API Name :getPlayListForTheUser

Type: GET

Description: To retrieve the video playlist for the user

Request Payload:

{

"email":""

}

Response

{

"code":200,

"message":"Video playlist retrieved successfully"

"data":"

[

{"playlistName":"","noOfVideos":,"videoData":[

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

]},

{"playlistName":"","noOfVideos":,"videoData":[

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

{"id":"",title:"","description":"","creatorName":"","creatorImage":"","noOfViews":,"noOfComments":,"noOfLikes":,"noOfShares":},

]},

"

}

===================================================================

API Name :createOrUpdateUserPlaylist

Type: POST

Description: To create a new playlist for the user.

Request Payload:

{

"email":"",

"playListId":454,

"playListName":"",

"videoData":"[123,422,344]"

}

Response

{

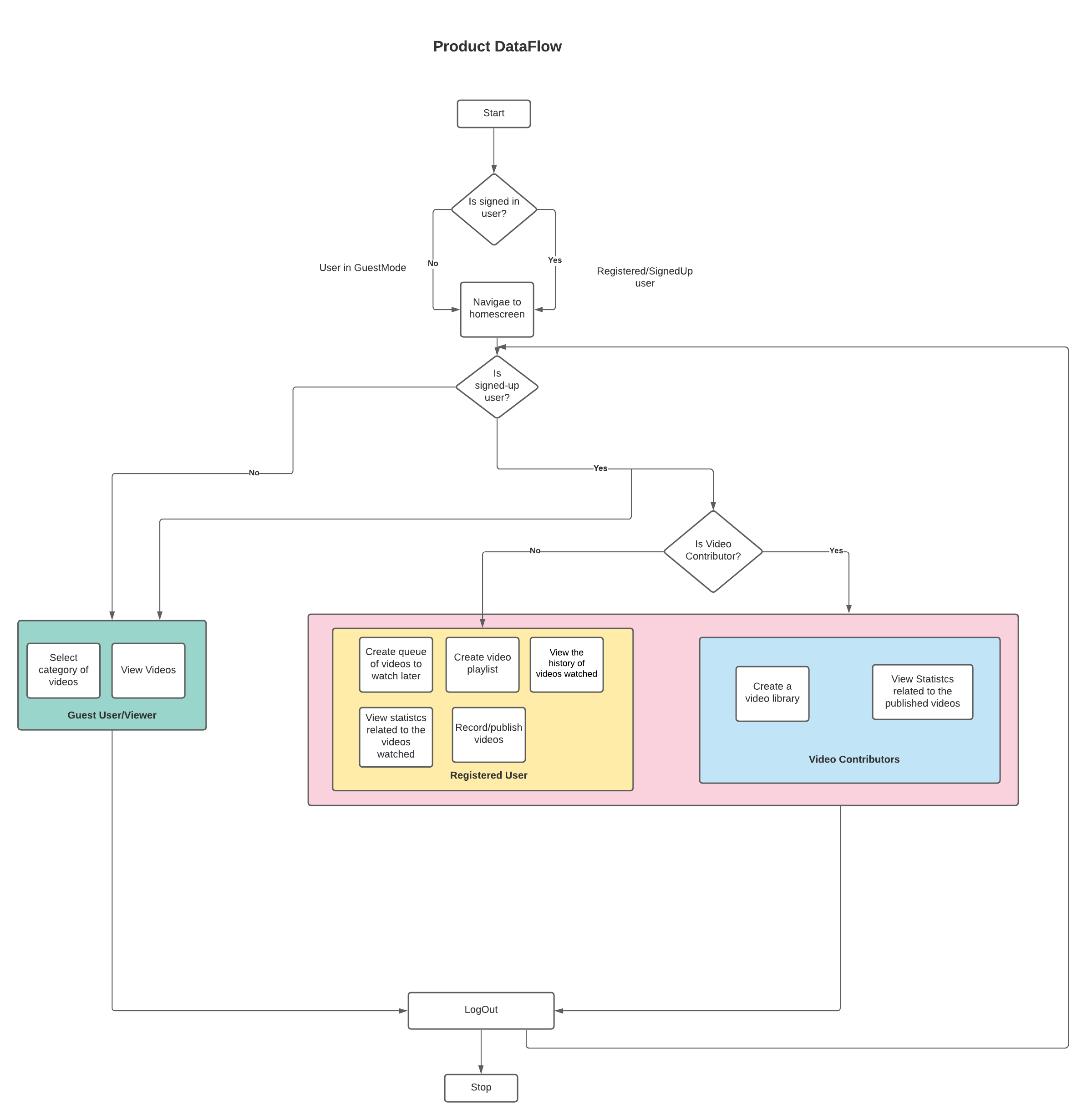
"code":200,

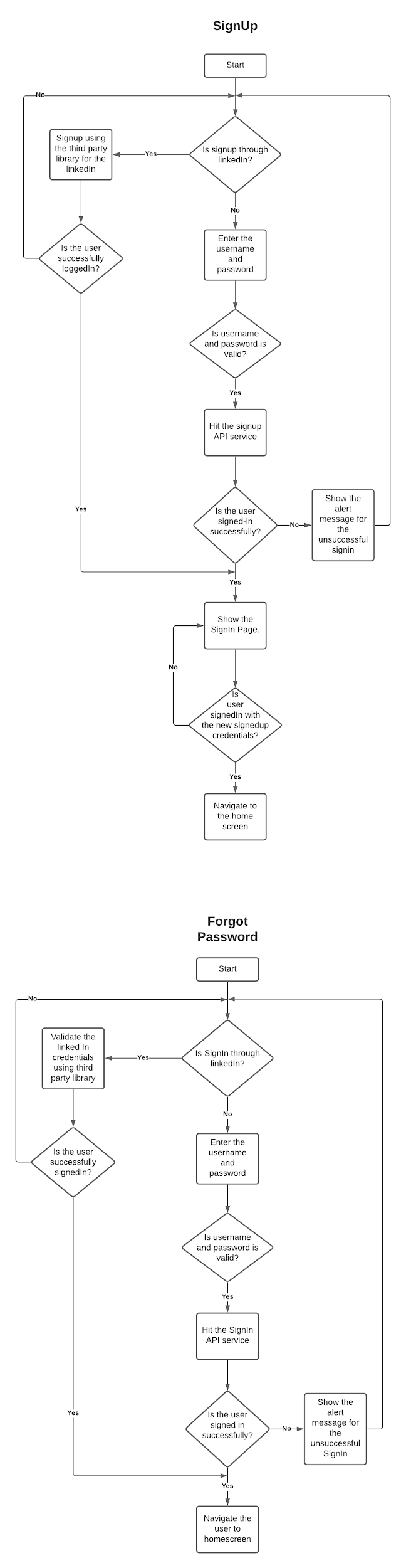
"message":"New playlist successfully created/updated with added videos"

"data":""

===================================================================

**Data Flow Diagrams**





**Class diagrams**

