**OBJECT ORIENTED ANALYSIS AND DESIGN COURSE**

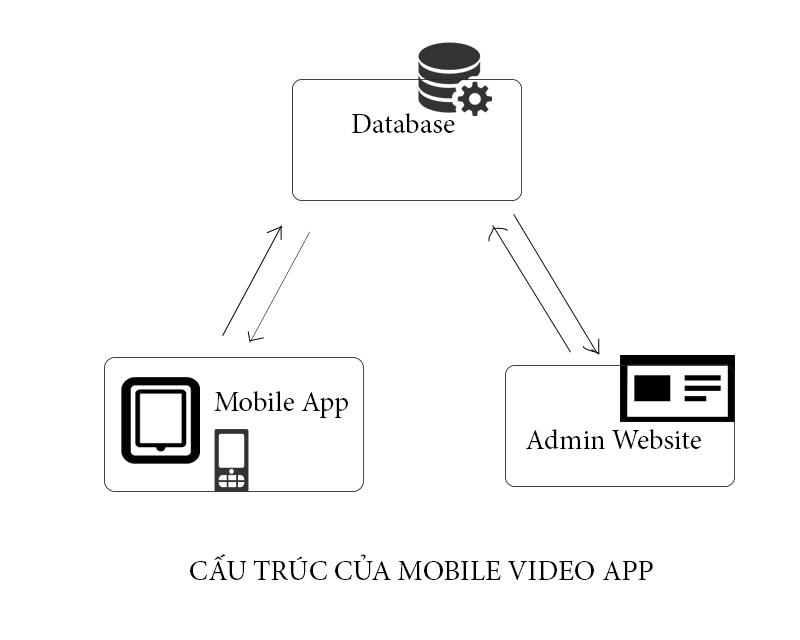
**YOUTUBE VIDEO** **PLAYER**

**Project Team:**

* Phan Dang Thanh
  + Msv: 11020429
  + Leader
  + **Web admin**
* Tran Duc Muoi
  + Msv: 11020204
  + Member
  + Mobile App
* Nguyen Thanh Toan
  + Msv: 11020342
  + Member
  + Thiết kế nội dung
* Chu Van Tao
  + Msv: 11020271
  + Member
  + Web Api

1. **YOUTUBE VIDEO PLAYER REQUIREMENT**
   1. **Problem** **Statement**

* Now, there are many tools to play videos on Youtube. However, videos do not categorized well. Therefore, we create this application , YOUTUBE VIDEO PLAYER, to help customers easily watch categorized videos.
  + This application allows administrators to add, delete, update information of videos and categories as well.
  + Customers can watch and mark the quality of videos.
* Our application has 2 part, web application on personal computer and mobile application on hand-held systems .



* 1. **Glossary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Type** | **Length** | **Note** |
| **Video** | | | |
| Id | Number | 10 | Video Identity |
| Title | String | 100 | Video title |
| Url | String | 100 | Url of video |
| duringTime | String | 10 | Video duration |
| imageUrl | String | 100 | Image url |
| Description | String | 100 | Description of video |
| Rating | Number | 3 | Evaluating Point of video |
| **Category: category of video** | | | |
| Id | Number | 10 | Category identity |
| Cat\_name | String | 100 | Name of category |
| **Video\_Cat: table that represent relationships of video and category** | | | |
| Id | Number | 10 | Relation identity |
| videoId | Number | 10 | Video Identity |
| catId | Number | 10 | Category Identity |
| **Admin:** The person who manage info, and content of video Management System. | | | |
| **Customer:** Person who use app to view video | | | |

* 1. **Suplementary Specification**
* Objectives
  + The purpose of this document is to define requirements of the Youtube Video player System. This Supplementary Specification lists the requirements that are not readily captured in the use case of the use-case model. The Supplementary Specification and the use-case model together capture a complete set of requirements on the system.
* Scope
  + This Supplementary Specification applies to the Youtube Video Player, which will be developed by OOAD students.
  + This specification defines non-functional requirements of the system; such as reliability, usability, performance and supportability, as well as functional requirements that are common across a number of use cases.
* Usability
  + The software must be easy to use so that a new user can learn how to use the system within 30 minutes. This is very important requirement.
  + The user interface has to be nice and clear.
* Security
  + The system must prevent people are not manager to modify bill after store to the database.
  + Almost changes of the system databases can only be done by the manager.
  + Require confirm password before submit the changes.
* Design Constraints
  + This system will provide only for Mobile Application Interface.
  1. **Use** **case** **model**

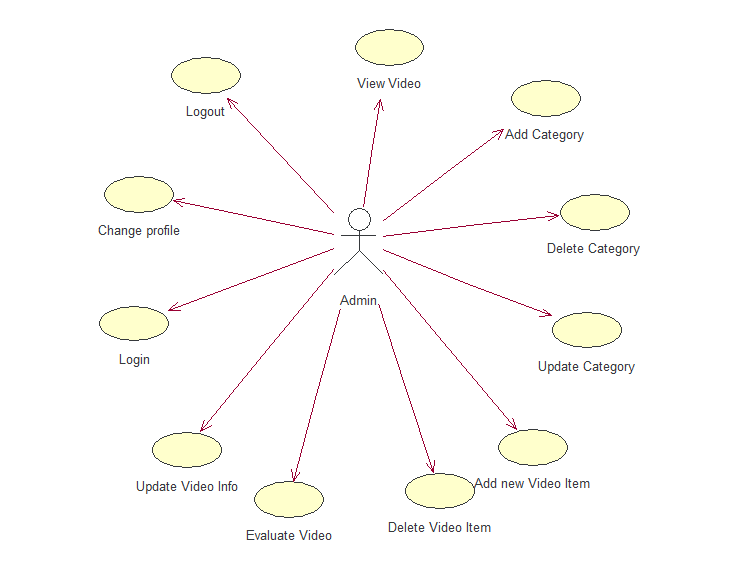
****

Figure1: Admin Usecase Diagram

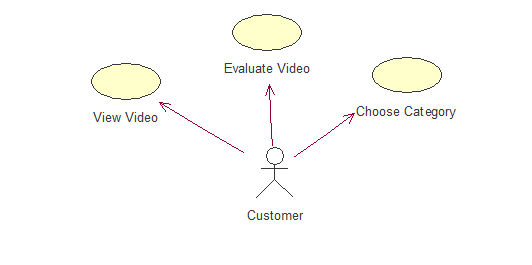


Figure2: Customer Usecase Diagram

* The System has 2 actors: Admin, and Customer
  + Admin can insert video items, delete items, update items, login, change profile…
  + Customer can view video, as well as evaluate video.
    1. **Login**
  + **Brief Description**
    - This use case describe how users login to Youtube Video Player Admin Controlpanel.
  + **Flow of events**
    - **Basic Flow**
      * This use case starts when the actor wishes to Login to the admin webpage**.**
      * Actor enter his username and password
      * System validates his info and logs the actor into system
    - **Alternative Flows**
      * Actor enter invalid username/password
      * System alert actor and still return login page
  + **Special Requirements**
    - None
  + **Pre-Conditions**
    - The system has the login page displayed.
  + **Post-Conditions**
    - If user enter valid info then , the user can login in to admin page. If not, the state isnot changed.
  + **Extension Points**
    - None
    1. **Logout**
  + **Brief Description**
    - This use case describe how user logout from admin page.
  + **Flow of events**
    - User click on the logout button, then system return logout page.
  + **Special Requirements**
    - None
  + **Pre-Conditions**
    - User is logged.
  + **Post-Conditions**
    - none
  + **Extension Points**
    - none
    1. **Change profile**
  + **Brief Description**
    - This use case describe how user change user info, and password.
  + **Flow of events**
    - **Basic Flow**
      * User enter updating info
      * System validates info, then change his info
    - **Alternative Flows**
      * User enter invalid info, then system alert user + his info is not changed.
  + **Special Requirements**
    - User is logged.
    - User enter valid required info such as old password.
  + **Pre-Conditions**
    - User is logged.
  + **Post-Conditions**
    - Enter valid old password
    - Enter valid update info
  + **Extension Points**

None.

* + 1. **Add category**
  + **Brief Description**
    - This use case describe how user can add new category of videos.
  + **Flow of events**
    - **Basic Flow**
      * Admin enter name of new category. Press Add button
      * System get new category name, then add to database
    - **Alternative Flows**
      * Name that admin enter is invalid format, then system alert user. Database is not changed.
  + **Special Requirements**
    - User is logged
  + **Pre-Conditions**
    - User is logged.
    - User press add new category button
  + **Post-Conditions**
    - Valid Info, then database is updated
    - Invalid info then database is unchanged.
  + **Extension Points**
    - **none**
    1. **Delete Category**
  + **Brief Description**
    - This use case describe how admin can delete a category of videos
  + **Flow of events**
    - **Basic Flow**
      * Admin press delete button, then system delete this category. Database is updated.
  + **Special Requirements**
    - User is logged.
  + **Pre-Conditions**
    - User is logged.
    - User enter into category manager page.
    - User press delete button
  + **Post-Conditions**
    - System check database, then delete it.
  + **Extension Points**
    - **None.**
    1. **Update Category**
  + **Brief Description**
    - This use case describe how admin can update name of a category
  + **Flow of events**
    - **Basic Flow**
      * Admin enter new name of category
      * System validates this name, then update database information.
    - **Alternative Flows**
      * If new name is empty as well as invalid, then system alert user, database is unchanged.
  + **Special Requirements**
    - User is logged.
  + **Pre-Conditions**
    - User enter to Category management page.
    - User enter new name, then press update button.
  + **Post-Conditions**
    - New info is valid
  + **Extension Points** 
    - **None**
    1. **Add new Video**
  + **Brief Description**
    - This use case describe how admin ca add new video item.
  + **Flow of events**
    - **Basic Flow**
      * Admin enter info of new video such as url, title, description…
      * System validates info. If the info is valid, then database is updated.
    - **Alternative Flows**
      * If entered information is invalid, then system alert admin. Database is unchanged.
  + **Special Requirements**
    - User is logged.
  + **Pre-Conditions**
    - Admin enter insert video page, enter valid info.
  + **Post-Conditions**
    - if information are valid, the new video item is added into database.
  + **Extension Points**
    - **None.**
    1. **Delete Video**
  + **Brief Description**
    - This use case describe how admin can delete a video.
  + **Flow of events**
    - **Basic Flow**
      * Admin press delete button, then system delete this video.
  + **Special Requirements**
    - User is logged.
  + **Pre-Conditions**
    - User is logged.
    - User press delete button.
  + **Post-Conditions**
    - System check in database, then this video information are deleted.
  + **Extension Points**
    - None.
    1. **Update Video Info**
  + **Brief Description**
    - This use case describe how admin can update video information.
  + **Flow of events**
    - **Basic Flow**
      * Admin update new information of a video.
      * System check these information. If valid, then database is updated.
    - **Alternative Flows**
      * If new information is invalid, then the system alert user, database is unchanged.
  + **Special Requirements**
    - User is logged.
  + **Pre-Conditions**
    - User is logged.
  + **Post-Conditions**
    - Information that admin entered are valid. Then database is updated.
  + **Extension Points**
    - **None.**
    1. **View Video (Customer)**
  + **Brief Description**
    - This use case describe how customer can view videos.
  + **Flow of events**
    - **Basic Flow**
      * Customer enter mobile app.
      * System return list of videos.
      * Customer press on any video.
      * System return this video on mobile screen.
    - **Alternative Flows**
      * Video is not found. Then system alert info to customer.
  + **Special Requirements**
    - Mobile has a working internet.
  + **Pre-Conditions**
    - Mobile app is working well
    - Mobile has a working internet.
    - Customer enter this app
    - Customer press play this video.
  + **Post-Conditions**
    - Server works well.
    - System return video well.
  + **Extension Points**
    - **None.**
    1. **Evaluate Video (Customer)**
  + **Brief Description**
    - This use case describe how customer can evaluate any video.
  + **Flow of events**
    - **Basic Flow**
      * Customers rate video follow IMDB rate.
      * Mobile app get info, then send info to database.
      * Database is updated.
  + **Special Requirements**
    - Mobile app is working well
    - Mobile has a working network.
  + **Pre-Conditions**
    - Server is working well.
    - Mobile has a working network.
    - Customer rates this video.
  + **Post-Conditions**
    - Server is working well.
  + **Extension Points**
    - **None.**
    1. **Choose Category (Customer)**
  + **Brief Description**
    - This use case describe how customer can choose list videos belong to a category.
  + **Flow of events**
    - **Basic Flow**
      * Customer open mobile app.
      * Customer press on category thumbnail.
      * System return list of videos belong to this category.
  + **Special Requirements**
    - Mobile has a working network.
  + **Pre-Conditions**
    - Customer press on a category thumbnail.
    - Mobile has a working network.
  + **Post-Conditions**
    - Server working well, then it return list of videos that belongs to this category.
  + **Extension Points**

**None.**

**II. Analysis:**

1. **Architectural Analysis:**
   1. **High-Level Component**

****

**Define:**

Application View:

Includes all things that can display for user and interacts with user.

Application Method:  
 Include databases and methods to use the databases, to manage

things displayed for users.

Manager:

Manage databases, methods… and receive requests from the user and call the accordance methods to handle them.

* 1. **Analysis Mechanisms for each class:**

LogIn: Error detection / handling / reporting

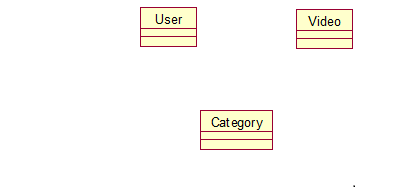
ChangeProfile: Error detection / handling / reporting

Video: Distribution

Category: Distribution

User: Security

* 1. **Key Abstractions:**

****

**Definition:**

**User**: is an account of the website that manage database video, maybe a manager or admin

Analysis Mechanism: Persistency, Security

Video: contain informations of video, method to change, add, delete… video

Analysis Mechanism: Distribution, Persistency

Category: classify a video to have lists of videos with same type, maybe add, delete, change the category.

Analysis Mechanism: Distribution

* 1. **Use-Case Realization:**
  2. LogIn:



* 1. **Change Profile:**

****

* 1. **Add new category**

****

* 1. **Update a category(include deleting)**

****

* 1. **Add video to a category:**

****

* 1. **Update a video**

****

* 1. **View video**

****