|  | **CAN THO UNIVERSITY SOFTWARE CENTER** |  |
| --- | --- | --- |
| **MEKONG DELTA - APTECH** |

**Aptech Certified Computer Professional (ACCP)**

**SEMESTER IV**

**SHOP MUSIC ONLINE**

| **Instructor:** | **Class: CP2095H02** |
| --- | --- |
| MSc. Lê Thị Minh Loan | **Team:Group 5** |
| **Students:**  1. A18138 - Hứa Hoàng Duy  2. A18059 – Lê Nguyễn Hạt  3. A18122 – Huỳnh Minh Tuấn |

Can Tho, 07/2021

**PROBLEM DEFINITION OF PROJECT**

**1. Problems in the Existing Scenario:**

There is no doubt that music comes in different forms, styles and combinations. In fact

it would not be wrong to state that it has become a part of our daily life. It goes a long

way in soothing nerves and also being a balm for tension and stress. In today’s world

where there is much uncertainty around and so much unhappiness, there is hardly any

doubt that going through such recreational modes help a lot in both physical and

mental health. It can certainly be a big balm for helping elevate moods and even

treating various types of depression.

**2. Proposed Solution:**

Music, Game, Movie is an important part of our life. Nowadays, listening to music

online is hobby of most modern people. So we need to build website where users will

get relevant information about the latest music, game, movie releases in the market.

This website has web pages that give information about the new releases in the field

of music, game, and movie in the National market.

The site will allow users to view all kinds of music, game, movie - includes National

and International, listen to some music, see trailers game, movie, and download them

for free some but not full, vote for an album, game, movie and send feedback about

the site, etc. In addition, this website also provides a forum that users could discuss.

The website has search and browse functions too.

When users need to know information about recent releases of music, game, movie

they can open this website to see or search some information, and naturally, they can

buy some music if they want.

If users want to know information regarding music, game, movie or they wish to buy a

new music, they need not visit a shop at all. Instead of this, they can just open the

website to refer to information about music displayed on the web pages and place an

order.

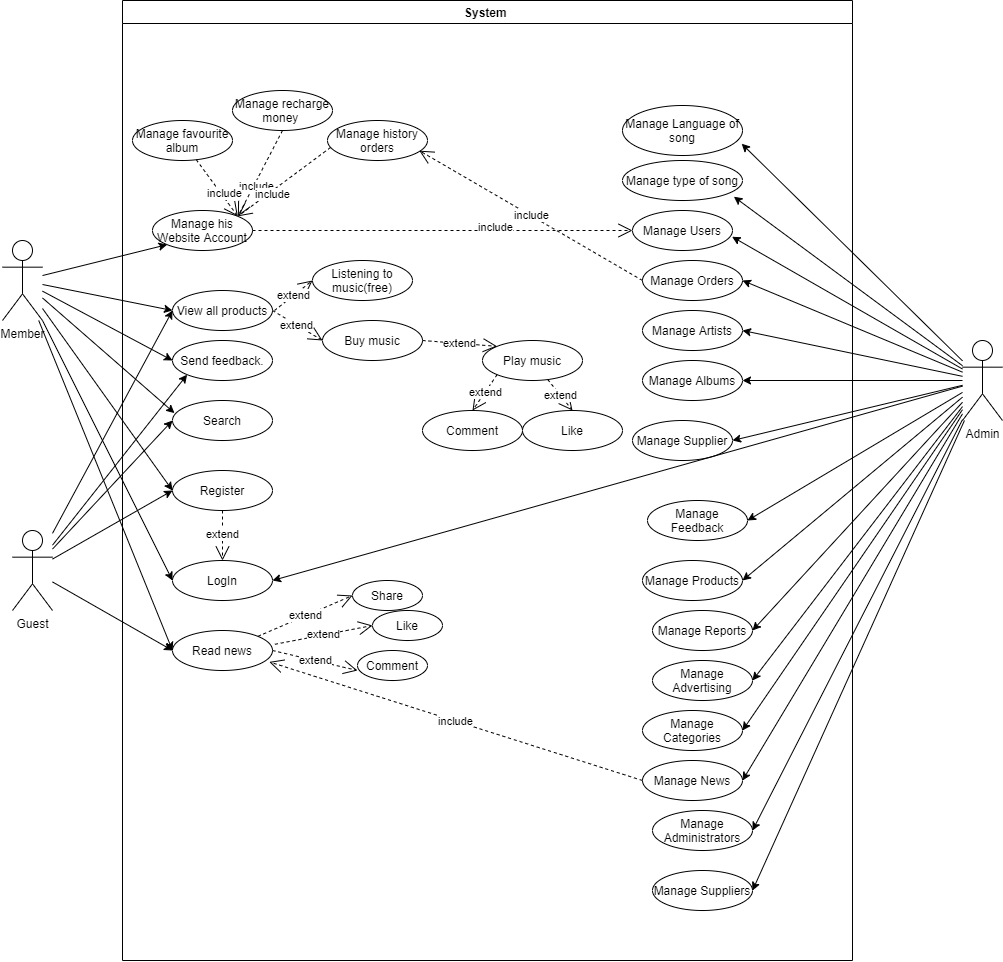
Finally, what’s thing that we want customers: come to see, and buy new releases

music.

**CUSTOMER REQUIREMENTS SPECIFICATION (CRS)**

**Name of Project**: SHOP MUSIC ONLINE

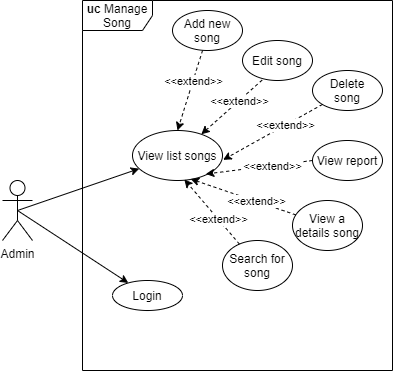
1. **Modeling requirements of customer (Use case Diagram)**



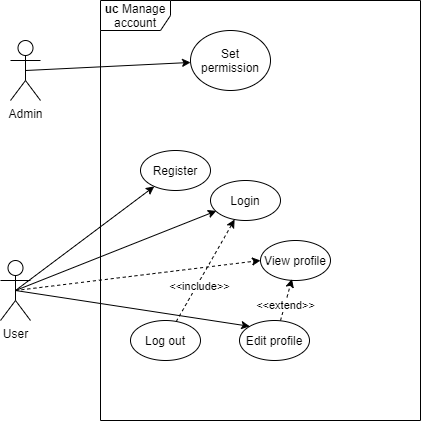
1. **Actor Description**

| **No.** | **Actor** | **Use Case** | **Description** |
| --- | --- | --- | --- |
|  | Guest | Read news |  |
| Search news | Browse or search information about an artist, an album, a song or a lyric, or see trailer game, movie or information content about them. |
| Login | Listen to free trailer movie, and some music online (but not full DVD) |
| Send feedback | Allow guest to send feedback about services on the website |
| View all song | Allow guest to view all song available on the website |
|  |  | Register | Allow guest who doesn't an account could sign up a new an account |
|  | Member | Buy music on website | Allow |
| Make Purchase |  |
| Login |  |
| Write the review | Write a review about an album, movie, game or rating. |
| Change profile | (view profile, edit profile, change password, cancel account) |
|  |  | Manage account member | (Such as Check Balance, View Transactions Log). |
|  |  | Manage history orders | (Such as view history orders …). |
|  | Admin | Manage Categories |  |
|  |  | Manage News |  |
|  |  | Manage Artists |  |
|  |  | Manage Albums |  |
|  |  | Manage Songs |  |
|  |  | Manage Feedback | This function contains view, deleting, reply and search feedback of users. |
|  |  | Manage Suppliers | Supplier is a company which supplies music’s discs for websites. This function contains view suppliers, adding a new supplier, deleting, editing, search and recovery supplier. |
|  |  | Manage Orders | This function contains view orders, deleting, editing, and searching. |
|  |  | Manage Reports | This function helps the admin can view reports. |
|  |  | Manage Advertising |  |

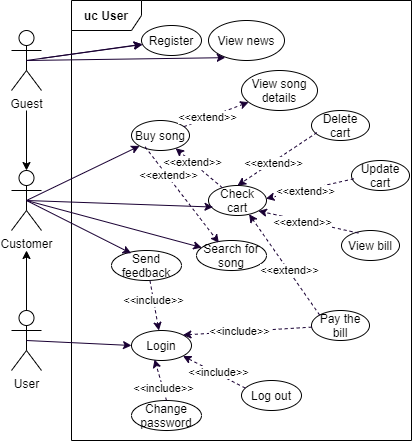
1. **Use case Description**
   1. **Manage Song:** This module allow Admin to manage information of product consist of **add new song**, **update information of song** and **delete song**.



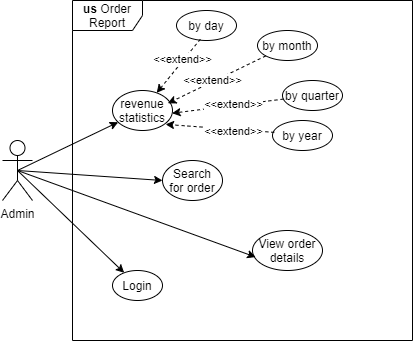
* + 1. **Add New Song**
* Purpose: this function allows the Admin to add new information about the song.
* Inputs:
  + Name of song
  + Description of song
  + Date of release
  + Price of song
  + Type of song
  + Language of song
  + Thumbnail of song
  + Audio of song
* Outputs: New information about the song is added to the table of the database.
  + 1. **Update Song**
* Purpose: this function allows the Admin to update information of the song
* Inputs:
  + The same data when adding a new song.
* Outputs: New information is updated to table of the database
  + 1. **Delete Song**
* Purpose: this function allows Admin to delete any song if the song has no data related.
* Inputs:
  + ID of the song
* Outputs: A record is deleted from table of the database
  1. **Login:** This module allows guests to login to the system and use some services on the website**.**
  + Purpose: this function allow guest can login to system
  + Inputs:
    - Username
    - Password
  + Outputs: Display message and redirect to home if information is correct
  1. **Manage Account:** This module allows Admin to manage information about Account consist of add, update and delete.

****

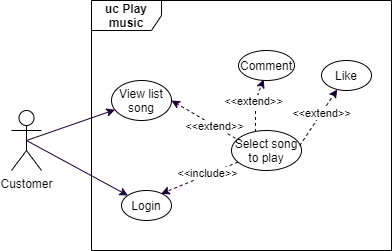
* 1. **Buy Song:** This module allows customers to buy some songs available on the website, add any song to cart, view order details and pay the bill.

****

* 1. **Manage Report:** This module allows admin to view reports, filter data or search.

****

* 1. **Play music:** This module allows Customers to perform actions about playing music and comment about the song or reaction.

****

1. **Hardware and Software required for implementing the project**

|  | **Server** | **Client** |
| --- | --- | --- |
| **Hardware** | - CPU: Intel Core i5-5200U, 2.5GHz or higher  - RAM: 4GB or higher  - HDD: 500GB Hard Disk space or higher  - Modem/ADSL Connecting to internet | - CPU: Intel Celeron or higher  - RAM: 2GB or higher  - HDD: 100GB Hard Disk space or higher |
| **Software** | - Windows 8 or later  - JDK 1.8.0  - Glassfish 4.1.2  - SQL Server 2012 or higher | Chrome, Internet Explorer/Edge 9.0 or higher, Firefox 3.5 or higher, Safari, Opera 10.5 or higher. |
| **Additional**  **Technology** |  | |

**CUSTOMER ACCEPTANCE CRITERIA**

| **No.** | **Customer’s Acceptance Criteria** |
| --- | --- |
|  | Show all products |
|  | Allow Users register and manage their account |
|  | Allow Users read news |
|  | Allow Users browse by category such as Artists, Albums,… |
|  | Allow Users place an order and view their history orders |
|  | Allow Users rate products and write their comments |
|  | Allow Users add products may buy |
|  | Allow Users play music, trailer movie or trailer game |

**PROJECT PLAN**

1. **Project Details**
2. Name of the Client: Shop Music Online
3. Date of the Project Plan: 08/06/2021
4. Project Vision/Objectives: (Define the project vision/objectives as stated by the client): You need to build a website where users will get relevant information about the latest music, game, movie releases in the market. This website has web pages that give information about the new releases in the field of music, game, and movie in the National market.
5. Scope: (Mention the scope of the project giving the location that will be covered, processed, range of services, and so on)

* **Manage categories**.(This function contains view categories, adding a new category, deleting, editing, and searching.)
* **Manage Artists**.(This function contains view artists, adding a new artists, deleting, editing, search)
* **Manage Albums**.( This function contains view albums, adding a new album, deleting, editing, search)
* **Manage Songs, games, movies**.( This function contains views , adding a new , deleting, editing, search.)
* **Manage Users**.( User can be a guest or a member or a customer)
* **Manage Feedback**.( This function contains view, deleting, reply and search feedback of users.)
* **Manage Suppliers**.( Supplier is a company which supplies music’s discs for websites. This function contains view suppliers, adding a new supplier, deleting, editing, search and recovery supplier.)
* **Manage Producers**:( Producer is a company which publishes. This function contains view producers, adding a new producer, deleting, editing, search and recovery producer.)
* **Manage Products**:(A product is an album which is sold on the website.
* This function contains view products, adding a new product, deleting, editing, search.)
* **Manage Orders**:(This function contains view orders, deleting, editing, search.)
* **Manage Reports**:(This function helps the admin to view reports.)
* **Manage Advertising**:(This function contains changing banner of website, and changing all ads of website. The Ads can be placed in any position.)
* **Manage News**:(This function contains view, adding, deleting, editing, search and recovery news of music, game, movie. Include National and International news.)
* **Manage Album Reviews**:(Admin manages all reviews about albums of users. This function contains adding, deleting, editing and searching album reviews.)
  1. **Guest: (Guest is a user who doesn’t register at a website)**.
* Read news about music, games, movies on websites.
* Browse or search information about an artist, an album, a song or a lyric, or see trailer game, movie or information content about them.
* Listen to free trailer movies, and some music online.
* Send feedback
* Register
  1. **Member** Member is a user who registered at a website. Signing in the website, a Member can use these below functions:
* All functions of Guest.
* Download music, trailer movies, or movie games.
* Buy music on website:
* Write a review about an album, movie, game or rating.
* Change profile (view profile, edit profile, change password, cancel account)
* Send feedback.
* Members can manage his Website Account (such as Check Balance, View Transactions Log).
* Members can manage history orders (such as view history orders …)

1. Our understanding of the client organization: (Give the range of services, functions, overview of processes, and so on)

Admin: Manage categories, manage artists, manage types of songs,manage albums, manage songs,games, movies, manage users, manage feedback, manage suppliers, manage products, manage orders, manage reports, manage advertising, manage news, manage album reviews, manage guests, manage members.

Member: Login/logout, change password, edit personal information, view log, send feedback.

Guest: Register, view log, view news, view details song.

1. Project Organization with Responsibilities and Authorities: (Give the name of Project team members their roles and responsibilities)

- Duy Hua Hoang (Leader):

Manage product: insert, update, delete, search,statistics.

Manage artist, producer, supplier: insert , update, delete, search.

Manage type of songs

Listen to music, Manage account: register account, personal manage information, change password, statistics, block account.

- Hat Le Nguyen (Member):

Manage News: insert, update, delete, search, statistics.

Manage Categories: insert, update, delete, search, statistics.

Manage feedback.

- Tuan Huynh Minh (Member):

Product buy manager:display list products, display details products, cart, recharge money.

Manage orders: display orders, statistics(day, week, month, year).

1. **Project Initiation/Requirement Documents**

(Information required from the client as inputs regarding his/her system; could be the information about the services, processes, and so on)

● Show all products.

● Allow Users register and manage their account.

● Allow Users to read news.

● Allow Users browse by category such as Artists, Albums,…

● Allow Users place an order and view their history orders.

● Allow Users rate products and write their comments.

● Allow Users to add products they may buy.

● Allow Users to play music, trailer movies or trailer games.

● Allow Users download songs as they want with condition movies or games or music for free.

● Allow Users to send feedback in order to improve the website better.

● Allow Users create your collection album to save a list of some music videos for free.

1. **Deliverables**

(The documents to be handed over to the client - such as CRS, Design Document, Installation Manual, User Manual, Maintenance Manual, and Code of documents)

● Source code.

● Database.

● Documentation.

- Table of Contents.

- Problem Definition.

- Customer Requirement Specification.

- Architecture & GUI standard of application.

- Interface design document.

- Coding standards document.

- Testing document.

- Activity.

- Database Design.

- Task Sheet.

- Project review.

- Final Checklist.

● Help Documentation.

- User Guide.

- Installation Guide.

- Developer’s Guide.

1. **Project Dependencies**

(Any event or task outside the scope of the project which will affect the success of the project)

● **Cost**: the financial constraints or budget of a project.

● **Scope**: all the tasks required to complete the project in full.

● **Time**: The proposed timeline or schedule for the project to be completed.

1. **Major Project Milestones**

(Generating CRS, Building a Prototype, Gartt - describe the main milestones of the project)

* Take customer requests.
* Design analysis.(database, interfaces)
* Coding.
* Deploy and maintain.

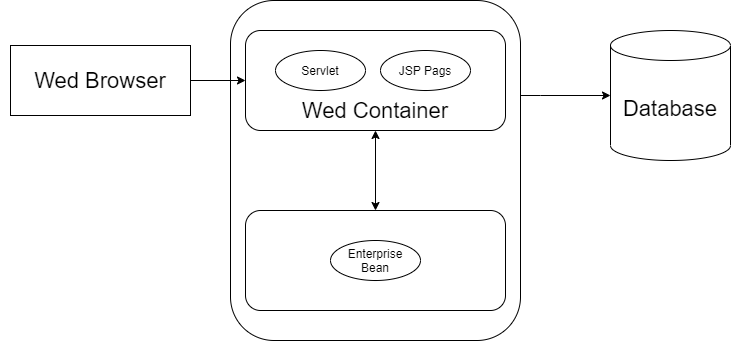
1. **Quality Plan**
2. Review Activities (Review meeting participants, frequency, and so on)
   * + - Once a week.
3. Testing Activities (Final Test)
   * + - Unit testing: see component testing.
       - Component testing: the testing of individual software components.
       - Unit testing task.
       - Unit test plan: prepare, review, rework, baseline.
       - Unit Test cases/Script: prepare, review, rework, baseline.
       - Unit test: perform.

1. Backup and Recovery Strategies (In case of disk crash, network failures, and so on):

* Github, Google Drive.

**ARCHITECTURE & GUI STANDARD OF APPLICATION**

1. **Architecture of Application**



1. **GUI Standard Application**

| **Property** | **Value** |
| --- | --- |
| Document Theme and Color Scheme | Custom |
| Form - Background Color | Custom |
| Title - Font Size | Custom |
| Title - Font Color | Custom |
| Title - Font Style | Custom |
| Title - Alignment | Custom |
| Background color of Controls on the form | Custom |
| Foreground color of Controls on the form | Custom |
| Control Caption - Font Size | Custom |
| Control Caption - Font Color | Custom |
| Control Caption - Font Style | Custom |
| Control caption and controls - Alignment | Custom |
| Command button - Alignment |  |

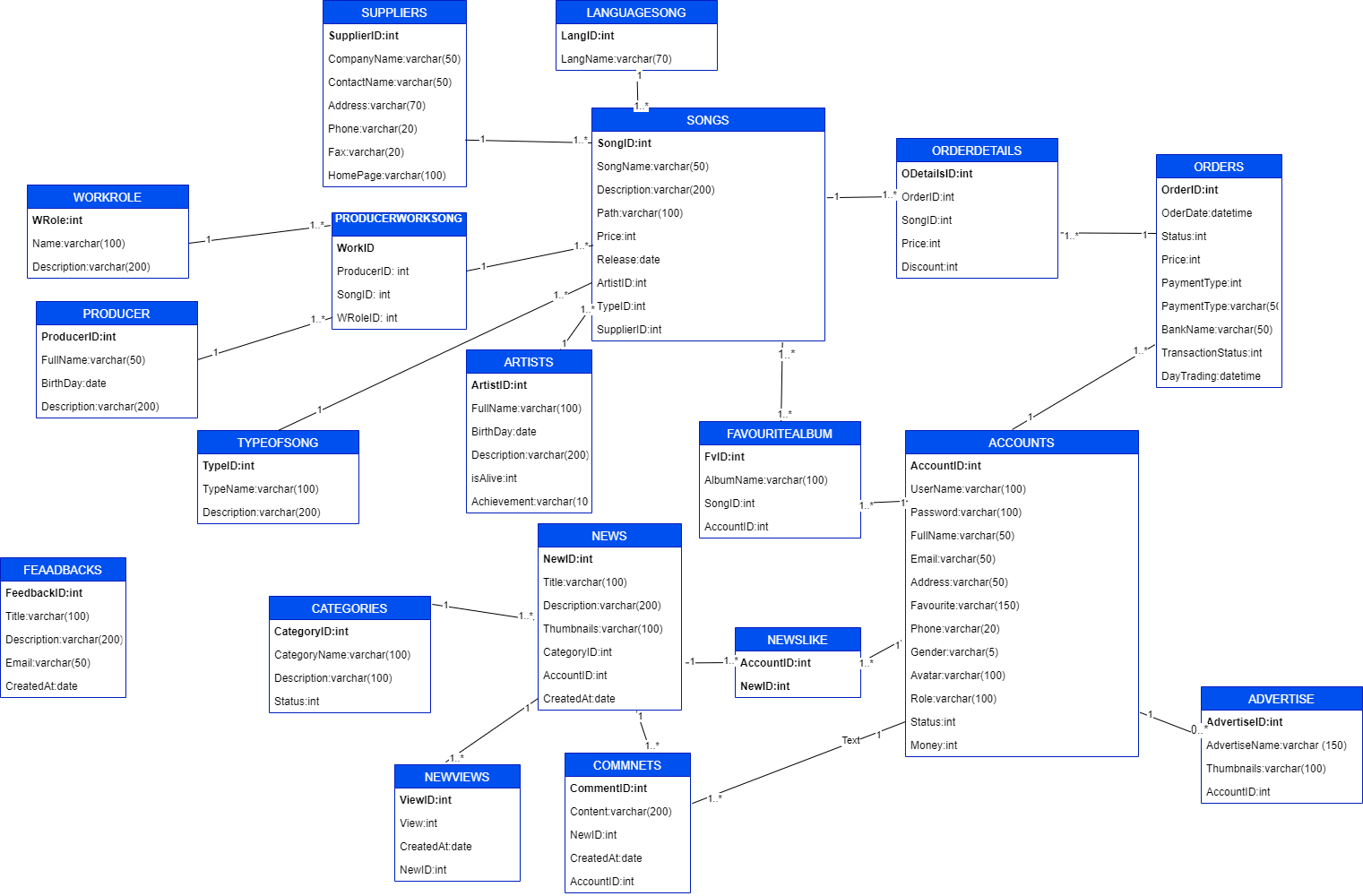
**INTERFACE DESIGN DOCUMENT**

List of forms to be created

| **Document Name** | **Description** | **Controls on the Document** |
| --- | --- | --- |
|  |  |  |

**DATABASE DESIGN DOCUMENT**

**I. ER/Class Diagram**

****

**II. Table Description**

| **(NEWS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| NewID | int | Primary Key | New version's primary key |
| Title | varchar(100) |  | Type of news |
| Description | varchar(200) |  | Description for news |
| Thumbnails | varchar(100) |  | Cover photo of the article |
| CategoryID | int | Foreign Key | Foreign Key table of Category |
| AccountlID | int | Foreign Key | Foreign Key table of Category |
| CreatedAt | int |  | Post creation date |

| **(NEWVIEWS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| ViewID | int | Primary Key | newview version's primary key |
| View | int |  | Number of views |
| CreatedAt | varchar(100) |  | view date |
| NewID | int | Foreign Key | Foreign Key table of New |

| **(COMMENTS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| CommentID | int | Primary Key | Comments version's primary key |
| Content | varchar(200) |  | Comment's content |
| NewID | int |  | Foreign Key table of New |
| CreatedAt | date |  | Post creation date |
| AccountID | int | Foreign Key | Foreign Key table of Account |

| **(FEEDBACKS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| FeedbackID | int | Primary Key | Feedbacks version's primary key |
| Title | varchar(100) |  | Type of Feedback |
| Description | varchar(100) |  | Description for Feedback |
| Email | varchar(50) |  | Email Account |
| CreatedAt | date |  | Post creation date |

| **(CATEGORIES)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| CategoryID | int | Primary Key | Category version's primary key |
| CategoryName | varchar(100) |  | Category's name |
| Description | varchar(100) |  | Pictures of Category |
| Status | int |  | Status |

| **(NEWSLIKE)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| NewID | int | Foreign Key | NewID version's primary key |
| AccountID | int | Foreign Key | Foreign Key table of Account |

| **(ADVERTISE)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| AdvertiseID | int | Primary Key | NewID version's primary key |
| AdvertiseName | varchar(150) |  | Name Of Advertise |
| Thumbnails | varchar(100) |  | Img of Advertise |
| AccountID | int | Foreign Key | Foreign Key table of Account |

| **(Accounts)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| AccountID | int | Primary Key | Account version's primary key |
| UserName | varchar(100) |  | Login account enter user |
| PassWord | varchar(100) |  | Login password |
| FullName | varchar(50) |  | User name |
| Email | varchar(50) |  | Email user |
| Address | varchar(50) |  | Address user |
| Favourite | varchar(150) |  | Favourite |
| Phone | varchar(20) |  | User's phone number |
| Gender | varchar(5) |  | Gender |
| Avatar | varchar(100) |  | User Image |
| Role | varchar(100) |  | User Role |
| Status | int |  | Status |
| Money | int |  | Money |

| **(TYPE OF SONG)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| TypeID | int | Primary Key | Type version's primary key |
| TypeName | varchar(100) |  | Name Type |
| Description | varchar(200) |  | Image Type |

| **(ARTISTS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| ArtistiID | int | Primary Key | Artisti version's primary key |
| FullName | varchar(100) |  | Name of Artisti |
| BirthDay | date |  | BirthDay of Artisti |
| Description | varchar(200) |  | Image of Artisti |
| isAlive | int |  | Still alive or not? |
| Achievement | varchar(10) |  | Achievement |

| **(FAVOURITE ALBUM)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| FvID | int | Primary Key | FAVOURITE ALBUM version's primary key |
| AlbumName | varchar(100) |  | Name of Album |
| SngID | int | Foreign Key | Foreign Key table of Songs |
| AccountID | int | Foreign Key | Foreign Key table of Account |

| **(PRODUCER)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| ProducerID | int | Primary Key | Producer version's primary key |
| FullName | varchar(50) |  | Name of Producer |
| BirthDay | date |  | date |
| Description | varchar(200) |  | Image Of Producer |

| **(PRODUCERWORKSONG)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| WorkID | int | Primary Key | PRODUCERWORKSONG version's primary key |
| ProducerID | int | Foreign Key | Foreign Key table of Producer |
| SongID | int | Foreign Key | Foreign Key table of Songs |
| WoeID | int | Foreign Key | Foreign Key table of Woe |

| **(WORKROLE)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| WRole | int | Primary Key | WORKROLE version's primary key |
| Name | varchar(100) |  | Name |
| Description | varchar(200) |  | Image |

| **(SONGS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| SongID | int | Primary Key | Songs version's primary key |
| SongName | varchar(50) |  | Name of songs |
| Description | varchar(200) |  | Image of songs |
| Path | varchar(100) |  | path |
| Price | int |  | Price of song |
| Release | date |  | Date of manufacture |
| ArtistiID | int |  | Foreign Key table of Artists |
| TypeID | int |  | Foreign Key table of Types |
| SupplierID | int |  | Foreign Key table of Suppliers.. |

| **(ORDER DETAILS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| ODetailsID | int | Primary Key | OrderDetails version's primary key |
| OrderID | int | Foreign Key | Foreign Key table of Order.. |
| SongID | int | Foreign Key | Foreign Key table of  Songs. |
| Paice | int |  | Price at purchase |
| Discount | int |  | Discount |

| **(ORDERS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| OrderID | int | Foreign Key | Order version's primary key |
| OrderDate | datetime |  | Order Date |
| Status | int |  | Status |
| Price | int |  | Price |
| PaymantType | int |  | Payment Type |
| BankName | varchar(50) |  | Payment unit name |
| TransactionStatus | int |  | Transaction Status |
| DayTrading | datetime |  | Day Trading |

| **(LANGUAGE SONG)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| LangID | int | Primary Key | LANGUAGE SONG version's primary key |
| LangName | varchar(70) |  | vietnamese, english, france |

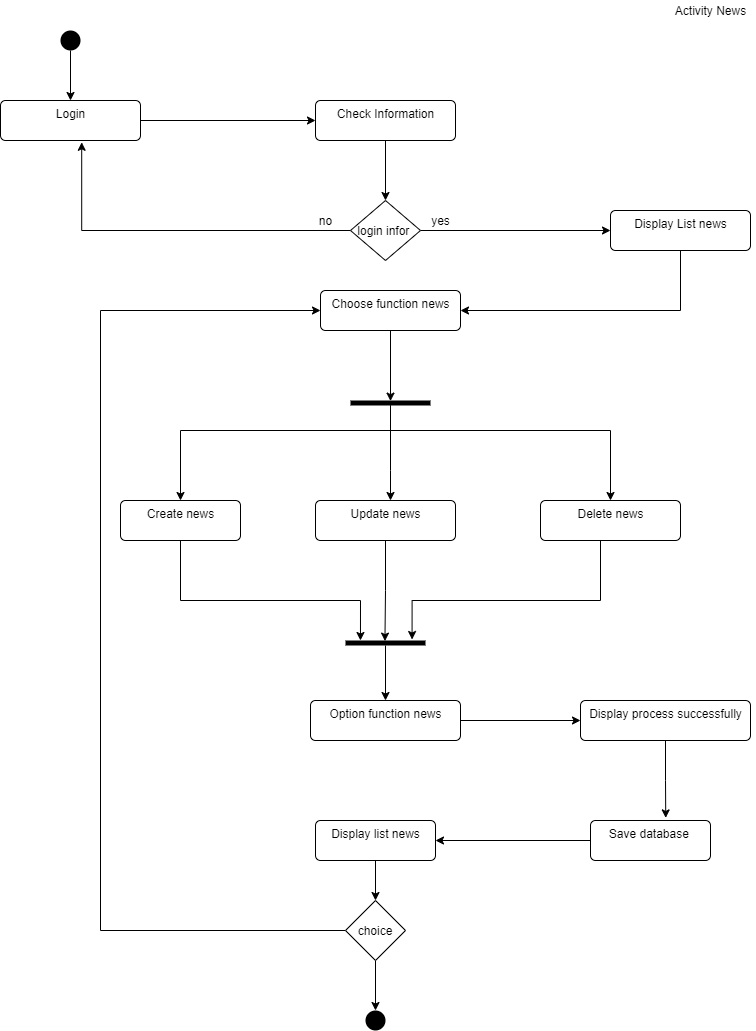
| **(SUPPLIERS)** | | | |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint  (Primary Key, Foreign Key, Identity, Not Null,...)** | **Description** |
| SupplierID | int | Primary Key | Supplier version's primary key |
| CompanyName | varchar(70) |  | Company Name |
| ContacName | varchar(50) |  | Contact Name |
| Address | varchar(70) |  | Address of Supplier |
| Phone | varchar(20) |  | Phone of Supplier |
| Fax | varchar(20) |  | Fax of Supplier |
| HomePage | varchar(50) |  | Wed of Supplier |

**PROCESS DESIGN DOCUMENT**

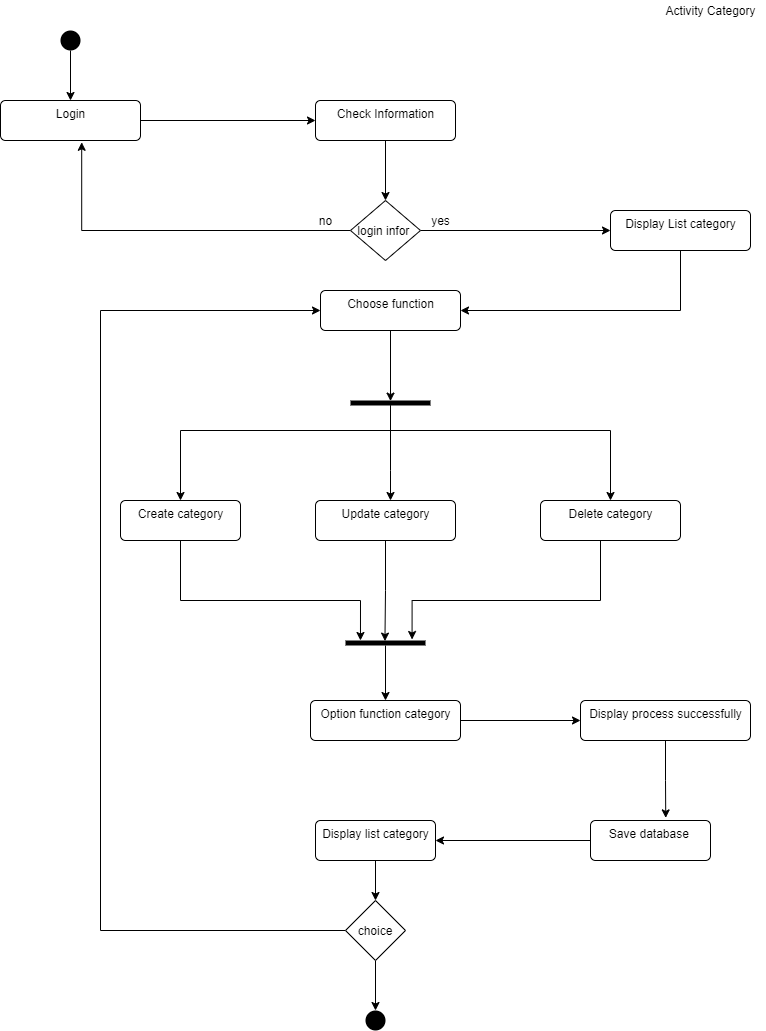
(Activity diagram, sequence diagram)

1.Activity

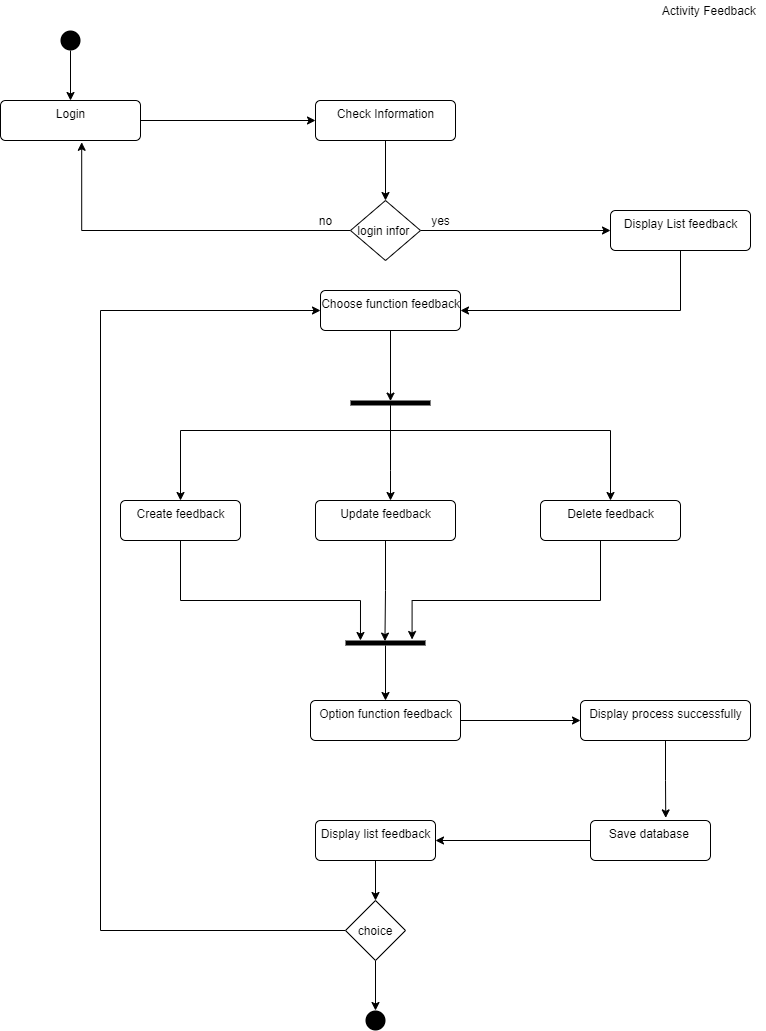
1. 1.1 Process Name: Manage News.
2. 1.2 Process Description and Details



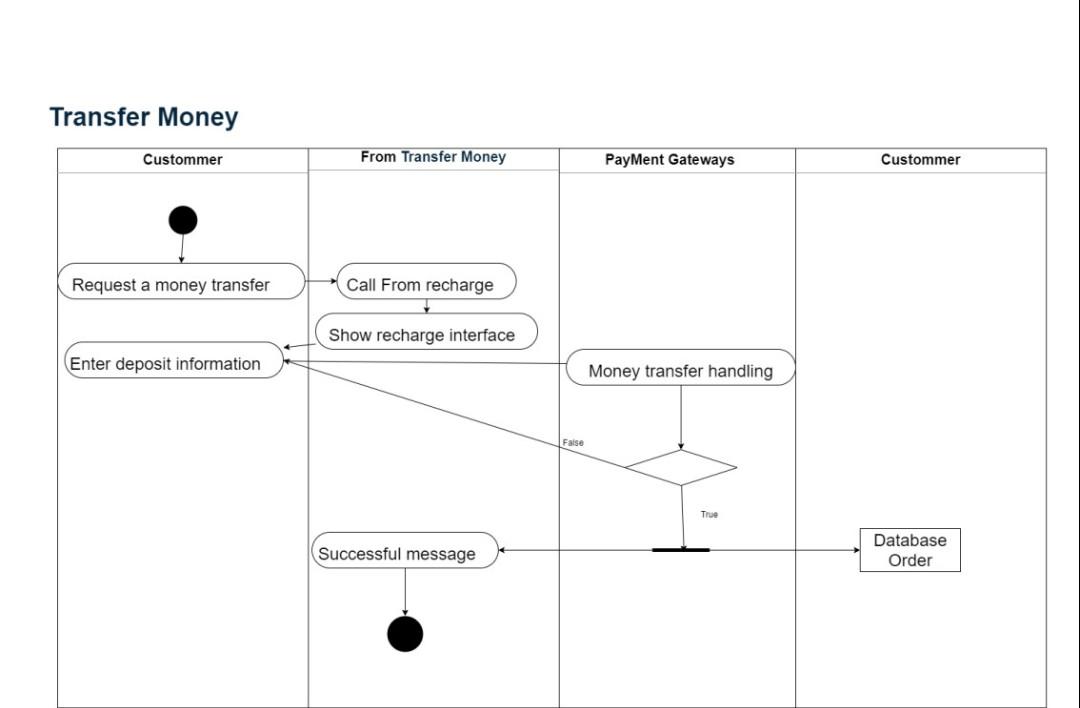
1. 2.1 Process Name: Manage Category.
2. 2.2 Process Description and Details

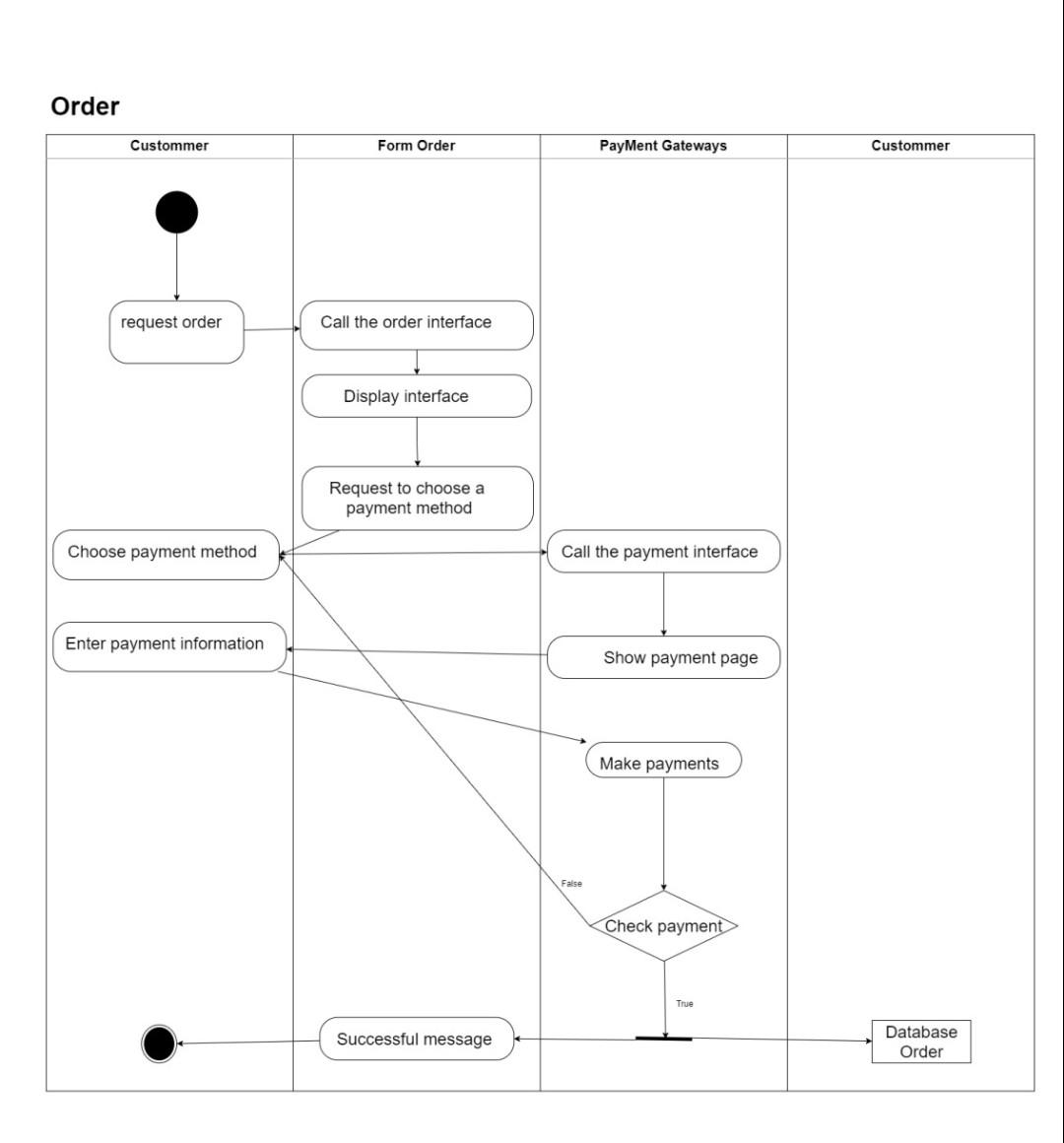


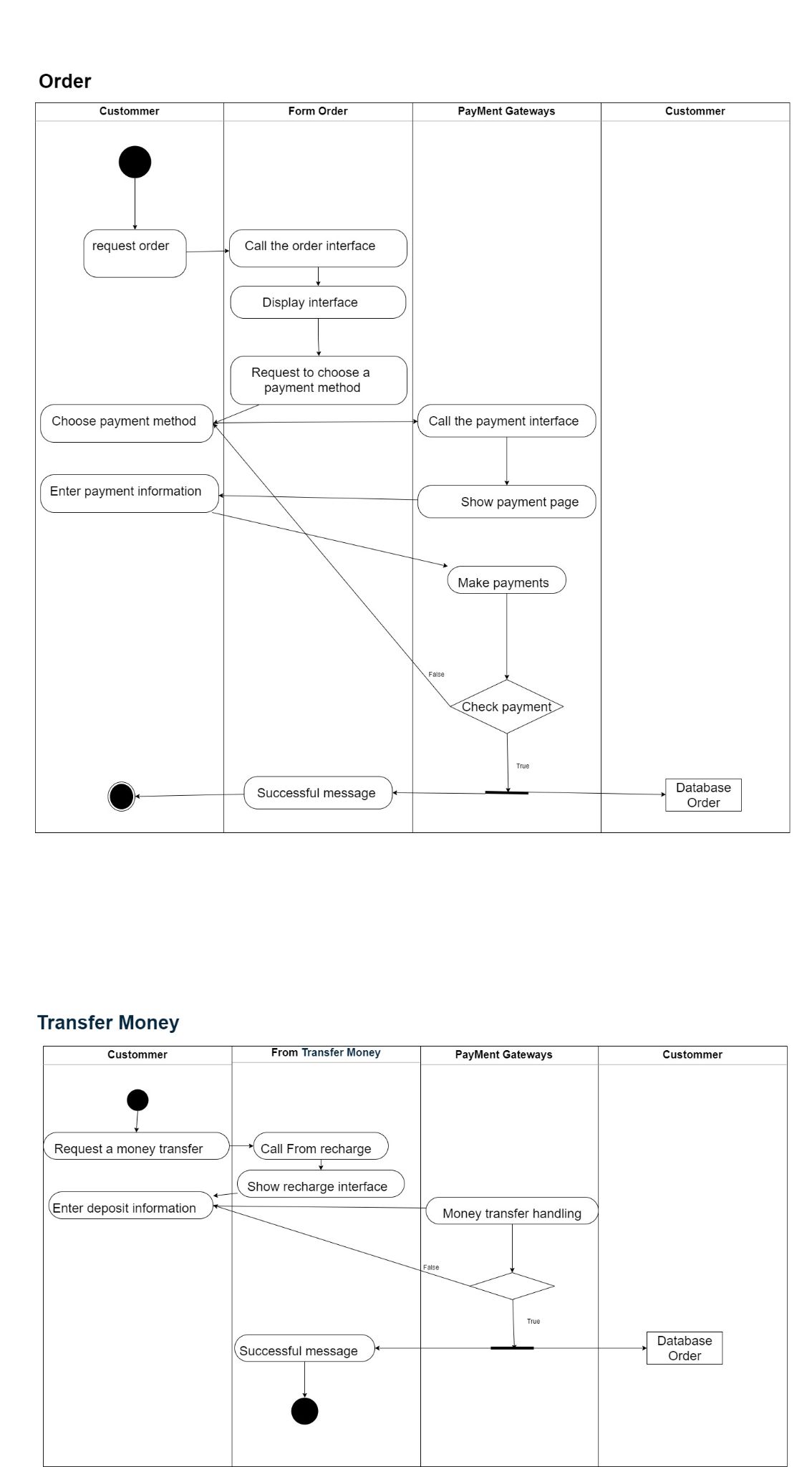
1. 3.1 Process Name: Manage feedback.
2. 3.2 Process Description and Details



1. 4.1 Process Name: Manage Order .
2. 4.2 Process Description and Details



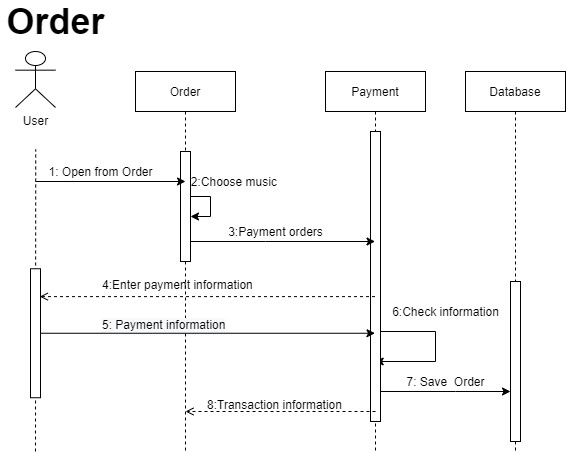




2.Sequence

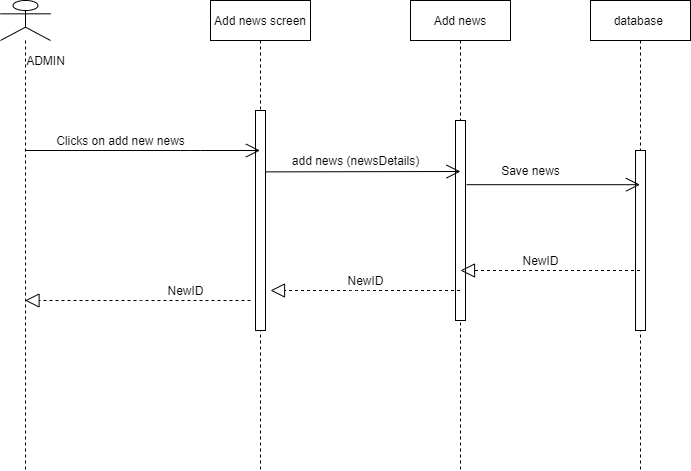
2.1 Process name: Manage Order.

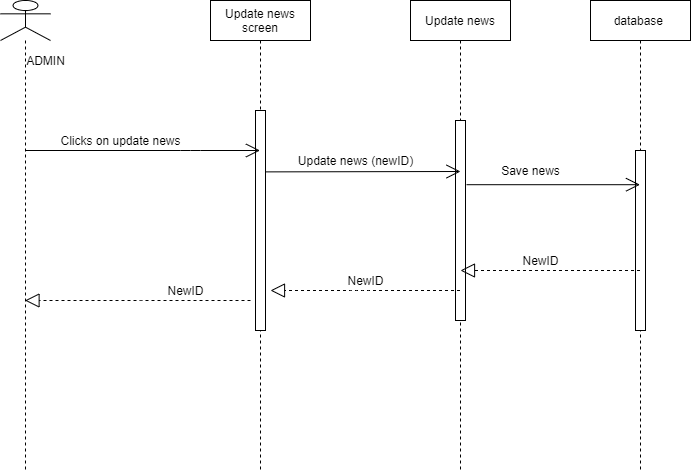
2.2 Process Description and Details

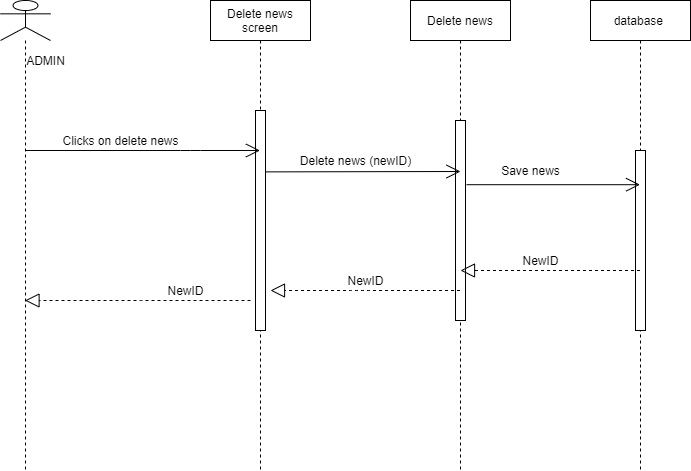


3.1 Process name: Manage News.

3.2 Process Description and Details







**CODING STANDARDS DOCUMENT**

1. Programming Standards

### 1 @Override: always used

A method is marked with the @Override annotation whenever it is legal. This includes a class method overriding a superclass method, a class method implementing an interface method, and an interface method respecifying a superinterface method.

**Exception:** @Override may be omitted when the parent method is @Deprecated.

### 2 Caught exceptions: not ignored

Except as noted below, it is very rarely correct to do nothing in response to a caught exception. (Typical responses are to log it, or if it is considered "impossible", rethrow it as an AssertionError.)

When it truly is appropriate to take no action whatsoever in a catch block, the reason this is justified is explained in a comment.

try {

int i = Integer.parseInt(response);

return handleNumericResponse(i);

} catch (NumberFormatException ok) {

// it's not numeric; that's fine, just continue

}

return handleTextResponse(response);

**Exception:** In tests, a caught exception may be ignored without comment *if* its name is or begins with expected. The following is a very common idiom for ensuring that the code under test *does* throw an exception of the expected type, so a comment is unnecessary here.

try {

emptyStack.pop();

fail();

} catch (NoSuchElementException expected) {

}

### 3 Static members: qualified using class

When a reference to a static class member must be qualified, it is qualified with that class's name, not with a reference or expression of that class's type.

Foo aFoo = ...;

Foo.aStaticMethod(); // good

aFoo.aStaticMethod(); // bad

somethingThatYieldsAFoo().aStaticMethod(); // very bad

### 4 Finalizers: not used

It is **extremely rare** to override Object.finalize.

1. Standards for code writing style

#### 1 General form

The *basic* formatting of Javadoc blocks is as seen in this example:

/\*\*

\* Multiple lines of Javadoc text are written here,

\* wrapped normally...

\*/

public int method(String p1) { ... }

... or in this single-line example:

/\*\* An especially short bit of Javadoc. \*/

The basic form is always acceptable. The single-line form may be substituted when the entirety of the Javadoc block (including comment markers) can fit on a single line. Note that this only applies when there are no block tags such as @return.

#### 2 Paragraphs

One blank line—that is, a line containing only the aligned leading asterisk (\*)—appears between paragraphs, and before the group of block tags if present. Each paragraph but the first has <p> immediately before the first word, with no space after.

#### 3 Block tags

Any of the standard "block tags" that are used appear in the order @param, @return, @throws, @deprecated, and these four types never appear with an empty description. When a block tag doesn't fit on a single line, continuation lines are indented four (or more) spaces from the position of the @.

4 The summary fragment

Each Javadoc block begins with a brief **summary fragment**. This fragment is very important: it is the only part of the text that appears in certain contexts such as class and method indexes.

This is a fragment—a noun phrase or verb phrase, not a complete sentence. It does **not** begin with A {@code Foo} is a..., or This method returns..., nor does it form a complete imperative sentence like Save the record.. However, the fragment is capitalized and punctuated as if it were a complete sentence.

5 Where Javadoc is used

At the *minimum*, Javadoc is present for every public class, and every public or protected member of such a class, with a few exceptions noted below.

6 Exception: self-explanatory methods

Javadoc is optional for "simple, obvious" methods like getFoo, in cases where there *really and truly* is nothing else worthwhile to say but "Returns the foo".

7 Exception: overrides

Javadoc is not always present on a method that overrides a supertype method.

8 Non-required Javadoc

Other classes and members have Javadoc *as needed or desired*.

Whenever an implementation comment would be used to define the overall purpose or behavior of a class or member, that comment is written as Javadoc instead (using /\*\*).

Non-required Javadoc is not strictly required to follow the formatting rules of Sections 7.1.2, 7.1.3, and 7.2, though it is of course recommended.

1. Standards for declaring Variables

##### 1 One variable per declaration

Every variable declaration (field or local) declares only one variable: declarations such as int a, b; are not used.

**Exception:** Multiple variable declarations are acceptable in the header of a for loop.

##### 2 Declared when needed

Local variables are **not** habitually declared at the start of their containing block or block-like construct. Instead, local variables are declared close to the point they are first used (within reason), to minimize their scope. Local variable declarations typically have initializers, or are initialized immediately after declaration

1 Local variable names

Local variable names are written in [lowerCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case).

Even when final and immutable, local variables are not considered to be constants, and should not be styled as constants.

2 Type variable names

Each type variable is named in one of two styles:

A single capital letter, optionally followed by a single numeral (such as E, T, X, T2)

A name in the form used for classes (see Section 5.2.2, [Class names](https://google.github.io/styleguide/javaguide.html#s5.2.2-class-names)), followed by the capital letter T (examples: RequestT, FooBarT).

#### 3 Non-constant field names

Non-constant field names (static or otherwise) are written in [lowerCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case).

These names are typically nouns or noun phrases. For example, computedValues or index.

4 Parameter names

Parameter names are written in [lowerCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case).

One-character parameter names in public methods should be avoided.

1. Standards for Function Declarations

The function’s name is function feature.

The first characters are not uppercase and the first characters the next word is uppercase.

Example:

public void getId()

{

//block

}

1. Other Standards

#### 1 Package names

Package names are all lowercase, with consecutive words simply concatenated together (no underscores). For example, com.example.deepspace, not com.example.deepSpace or com.example.deep\_space.

#### 2 Class names

Class names are written in [UpperCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case).

Class names are typically nouns or noun phrases. For example, Character or ImmutableList. Interface names may also be nouns or noun phrases (for example, List), but may sometimes be adjectives or adjective phrases instead (for example, Readable).

There are no specific rules or even well-established conventions for naming annotation types.

*Test* classes are named starting with the name of the class they are testing, and ending with Test. For example, HashTest or HashIntegrationTest.

#### 3 Method names

Method names are written in [lowerCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case).

Method names are typically verbs or verb phrases. For example, sendMessage or stop.

Underscores may appear in JUnit *test* method names to separate logical components of the name, with *each* component written in [lowerCamelCase](https://google.github.io/styleguide/javaguide.html#s5.3-camel-case). One typical pattern is *<methodUnderTest>*\_*<state>*, for example pop\_emptyStack. There is no One Correct Way to name test methods.

#### 4 Constant names

Constant names use CONSTANT\_CASE: all uppercase letters, with each word separated from the next by a single underscore. But what *is* a constant, exactly?

Constants are static final fields whose contents are deeply immutable and whose methods have no detectable side effects. This includes primitives, Strings, immutable types, and immutable collections of immutable types. If any of the instance's observable state can change, it is not a constant. Merely *intending* to never mutate the object is not enough

**TASK SHEET**

1. Project Title: Shop online for DVDs.

| **No.** | **Task** | **Actual Start Date** | **Actual Days** | **Teammate Name** | **Status** |
| --- | --- | --- | --- | --- | --- |
|  | Manage Product | 28/06/2021 | 3 | Hua Hoang Duy | Completed |
|  | Type of song | 25/06/2021 | 1 | Hua Hoang Duy | Completed |
|  | Manage artist | 02/07/2021 | 2 | Hua Hoang Duy | Completed |
|  | Manage producer | 05/07/2021 | 1 | Hua Hoang Duy | Completed |
|  | Manage supplier | 05/07/2021 | 1 | Hua Hoang Duy | Completed |
|  | Product buy manager | 26/06/2021 | 5 | Huynh Minh Tuan | Completed |
|  | Manage orders | 02/07/2021 | 5 | Huynh Minh Tuan | Completed |
|  | Manage news | 26/06/2021 | 7 | Le Nguyen Hat | Completed |
|  | Manage category | 05/07/2021 | 3 | Le Nguyen Hat | Completed |
|  | Manage feedback | 08/07/2021 | 1 | Le Nguyen Hat | Completed |
|  | Manage account | 06/07/2021 | 2 | Hua Hoang Duy | Completed |
|  | Manage statistic | 08/06/2021 | 1 | Hua Hoang Duy | Completed |
|  | form 1,2a | 15/06/2021 | 1 | Hua hoang Duy | Completed |
| 14 | form 2b | 16/06/2021 | 1 | Hua Hoang Duy | Completed |
| 15 | form 3 | 15/06/2021 | 1 | Le Nguyen Hat | Completed |
| 16 | form 4 | 15/06/2021 | 1 | Huynh Minh Tuan | Completed |
| 17 | form 5 | 16/062021 | 1 | Huynh Minh Tuan | Completed |
| 18 | form 6 | 17/06/2021 | 1 | Huynh Minh Tuan | Completed |
| 19 | form 7 | 19/06/2021 | 1 | Le Nguyen Hat | Completed |
| 20 | form 8 | 10/07/2021 | 1 | Le Nguyen Hat | Completed |
| 21 | form 9 | 12/07/2021 | 1 | Le Nguyen Hat | Completed |
| 22 | form 11,12 | 14/07/2021 | 2 | Le Nguyen Hat | Completed |
| 23 | form 10 |  |  | Hua Hoang Duy |  |
|  |  |  |  |  |  |

**TESTING DOCUMENT**

| **ID** | **Test Case Name** | **Task** | **Result** | |
| --- | --- | --- | --- | --- |
| **Expected** | **Actual Result**  Pass/Fail-Comments |
|  | Add account details | Enter account details:  AccountID: 123456 Username: Steve  FullName: MOSS | Record is added to the database | Pass |
|  | Add account | - First scenario:  Username: abc  Password: admin@123  FullName:Spiderman Email: abc@gmail.com  Address : Road 10, American  Favourite: 2010  Money: 1$  Phone:0123456789  Gender: 0/1  Avatar: image  Role:  Status: 0/1 | Record is added to the database | Pass |
| - Second scenario:  Same data except for the following part: ISBN-10: 000123 | System gives a warning that the ISBN-10 format is not correct. | Fail - System crash - See screen shot at this address |
|  |  |  |  |  |

**PROJECT REVIEW**

| **Date** | **Project Plan Activity/Milestone** | **Work Specification** | **Status of the Activity** | **Remarks** | **Responsibility** |
| --- | --- | --- | --- | --- | --- |
| 08/06/2021 | draw use case | Draw use cases | pass | pretty good | All |
| 14/6/2021 | class diagram | describe the components in the system | Pass | pretty good | All |
| 20/6/2021 | database | view database in project | Pass | pretty good | All |
| 25/6/2021 | news | Manage news: insert, update,delete, search | pass | pretty good | Le Nguyen Hat |
| 25/6/2021 | account | Manage account: login/logout, profile, change password,statistics. | pass | pretty good | Hua Hoang Duy |
| 02/07/2021 | product | Manage songs: play music, comment, like, statistics. | pass | pretty good | Hua Hoang Duy |
| 25/06/2021 | product | Purchase manager: order, pay, statistics. | pass | pretty good | Huynh Minh Tuan |
| 05/07/2021 | feedback | customer’s feedback | pass | pretty good | Le Nguyen Hat |
| 06/07/2021 | TypeOfSong,artist, supplier | Manage song type | pass | pretty good | Hua Hoang Duy |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**FINAL CHECKLIST**

| **S.No.** | **Aspect tested** | **Suggestion/Remarks** |
| --- | --- | --- |
|  | Have all the modules been property integrated and are they completely functional? | Yes |
|  | Are all the validations happening as specified in Process Design? | Yes |
|  | Have all Design and Coding standards been followed and implemented? | Yes |
|  | Have all the modules been property integrated and are they completely functional? | Yes |
|  | Are all the validations happening as specified in Process Design? | Yes |
|  | Have all Design and Coding standards been followed and implemented? | Yes |
|  | Have all the modules been property integrated and are they completely functional? | Yes |
|  | Are all the validations happening as specified in Process Design? | Yes |