**Question 1**

**A)**

# Use Case Name

Student managing their schedule

## Actors/Roles

Student

Fee system

## Pre-Conditions –

1. Student must be enrolled in the college

## Basic Flow

Include labels (extension points) in curly braces as required.

|  |  |
| --- | --- |
|  | 1. The use case starts when a student has logged into the system |
| {Add course} | 1. The user selects to add a course |
| {drop course} | 1. The user selects to drop a course 2. The use case ends when a student exits the registration system |

## Alternative Flows

{User is enrolled in too many classes}

1. At {Add course}, the system checks how many courses the student is enrolled in.
2. If they’re enrolled in the maximum number of courses, they can’t enroll in a new one.
3. If they have any unpaid fees, they can’t enroll in any courses.

{User can’t drop any more courses}

1. If at {Drop course} the user is only enrolled in 1 course, then they can’t drop a course.

## Sub-Flows

{User adds a course}

1. At {Add course}, the system checks if they’re enrolled in too many courses.
2. The system checks if they have any unpaid fees.
3. The user is permitted to add a course.

{User drops a course}

1. At {drop course} the user can view all courses they’re enrolled in.
2. The user selects a course to drop.

{Student views their courses}

1. At any point between {Add course} and {Drop course}, the student can view all courses they’re enrolled in and all available courses to enroll in.

**B)**

# Use Case Name

Manage courses

## Actors/Roles

Student

Department staff

## Pre-Conditions – optional

## Basic Flow

|  |  |
| --- | --- |
|  | 1. The use case starts when a department staff logs into the system |
| {Add course} | 1. The staff member adds a course to the system |
| {Update course} | 1. The staff member updates an existing course in the system |
| {remove course} | 1. The staff member removes a course from the system. |
|  | 1. The use case ends when the department staff exits the system |

## Alternative Flows

{The course already exists}

1. At {Add course} if a course with the same course number exists, it shouldn’t get added

{The course is currently being accessed}

1. At {update course} if someone is currently modifying the course, the staff member should be locked out

## Sub-Flows

{Department staff adds a course}

1. At {Add course} The staff member views all courses in the department
2. The staff member can select to add a course
3. The staff member enters all the course info and adds it.

{Department staff updates a course}

1. At {update course} The staff member views all courses in the department
2. The staff member selects a course to modify
3. The staff member enters all the course info and saves their changes

{Department staff removes a course}

1. At {remove course} The staff member views all courses in the department
2. The staff member selects a course to delete

# Question 2:

**A)**

# Use Case Name

Catalogue Video

## Actors/Roles

Video store employee

## Pre-Conditions – optional

The employee must have access to the database

## Basic Flow

|  |  |
| --- | --- |
|  | 1. The use case starts when a video store employee accesses the database |
| {Add video} | 1. A video store employee adds a video to the database |
|  | 1. The use case ends when a video store employee exits the database |

## Alternative Flows

{A video already exists}

1. At {Add video} if the video already exists, it won’t get added.

## Sub-Flows

{The adding video process}

1. At {Add video}, the customer must select to add a new video to the database
2. The customer enters all the video information
3. The video is added to the database

**B)**

# Use Case Name

Obtain account

## Actors/Roles

Customer

## Pre-Conditions – optional

The customer doesn’t already have an account

## Basic Flow

|  |  |
| --- | --- |
|  | 1. The use case starts when a customer accesses the customer account system. |
| {Add account} | 1. The a user adds a new account |
| {Update account} | 1. A user logs into the account and edits it. |
|  | 1. The use case ends when a customer exits the account system |

## Alternative Flows

{Username for the given account already exists}

1. In {add account}, An account for that username already exists

## Sub-Flows

{User adds a new account}

1. At {add account}, the user selects to add a new account
2. The user enters all required information for their account.
3. User adds the account to the system

{user modifies an account}

1. At {update account}, the user logs into their account.
2. The user selects to modify their information
3. The user saves and changes that they made.