****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **MUMSched: Use Case Description For Admin** | |
|  |  | | |  |
| **Use Case Number: 1** | | | |  |
| **Name** | |  | User Profile CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case allows the admin to create profiles for faculty and student | |
|  |  | |  |  |
| **Actors** | | | Admin |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Preconditions** | | |  |  |
| The admin must logged in to the system | | | |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | | |  |
| **1.1.0 Create Faculty Profile** | | | |  |
|  |  |  |  |  |
| **Step** |  |  | **User Actions** | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin calls the create faculty profile command | | The system displays the faculty profile form with |
|  |  |  |  | the fields for firstname, lastname, email, password, |
|  |  |  |  | specializations, courses and blocks. |
|  |  |  | |  |
| 2 |  | The Admin fills out the form and requests the system | | The system verifies that there’s no other profile in |
|  |  | to save the details | | the database with the same email address and |
|  |  |  |  | saves the faculty and returns the success message |
|  |  |  |  | on success or a fail message in case of failure. In |
|  |  |  |  | case another profile exists with the email address, |
|  |  |  |  | the system returns the message indicating a |
|  |  |  |  | duplicate entry exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The staff profile is persisted in the system | | | |  |



**Business Rules**

No duplicate faculty profiles. A unique profile is identified by email address.

**1.1.1 Read/view Faculty Profile**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 1 | The admin selects to view a list of faculty profiles | The system returns a list of all faculty profiles. |
|  |  |  |
| 2 | The admin selects to view a profile of one of the | The system returns the profile of the faculty as a |
|  | faculty from the list of profiles | string with the names (firstname and lastname), |
|  |  | email, specializations, courses and the blocks of |
|  |  | preference. |
|  |  |  |
|  |  |  |
| **1.1.2 Update Faculty Profile** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Action** |
|  |  |  |
| 1 | The admin selects to view a list of faculty profiles | The system returns a list of all faculty profiles. |
|  |  |  |
| 2 | The admin selects the profile of the faculty they | The system displays an editable faculty profile |
|  | want to update | form pre-populated with the faculty profile details |
|  |  |  |
| 3 | The admin updates the fields they want to update | The system updates the record and returns the |
|  | and requests system to save the new details | success message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The staff profile will be updated | |  |
|  |  |  |
| **Business Rule** | |  |
| The email field should be unwritable. | |  |
|  |  |  |
|  |  |  |
| **1.1.3 Delete Faculty Profile** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin selects to view a list of faculty profiles | The system returns a list of all faculty profiles. |
|  |  |  |
| 2 | The admin selects to delete a faculty profile from a | The system displays a confirmation dialogue |
|  | list of faculty profiles | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system confirms the faculty is not assigned to |
|  | window to confirm deleting the profile | any sections or blocks and deletes the profile. The |
|  |  | system returns message a success message on |
|  |  | success or a failure message. The system should |
|  |  | return a message indicating the profile could not |
|  |  | be deleted because it is already assigned to some |
|  |  | blocks and sections |
|  |  |  |
| **Postconditions** | |  |
| The staff profile will be deleted | |  |
|  |  |  |
| **Business Rule** | |  |
| Only a faculty profile that is not assigned to any blocks or sections. | | |
|  |  |  |
| **1.2.0 Create Student Profile** | |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Step** | **User Action** | **System Action** |
|  |  |  |
| 1 | The admin calls the create student profile command | The system displays the student profile form with |
|  |  | firstname, lastname, registration number, email |
|  |  | and initial password |
|  |  |  |
| 2 | The Admin fills out the form and requests the system | The system checks if no other student profile has |
|  | to save the details | the same registration number and email and saves |
|  |  | the student, and displays a success message. The |
|  |  | system returns a fail message on exception or a |
|  |  | duplicate entry message in case of a duplicate. |
|  |  |  |
| **Postconditions** | |  |
| The student profile is persisted in the system | |  |
|  |  |  |
|  |  |  |
| **Business Rules** | |  |
| No duplicate student profiles. A unique profile is identified by email address and registration number. | | |
|  |  |  |
|  |  |  |
| **1.2.1 Read/view Student Profile** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin selects to view a list of student profiles | The system returns a list of all student profiles. |
|  |  |  |
| 2 | The admin selects the profile they want to view. | The system returns the profile of the student as a |
|  |  | string with names (firstname, lastname), |
|  |  | registration number and email. |
|  |  |  |
| **1.2.2 Update Student Profile** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Action** |
|  |  |  |
| 1 | The admin selects to view a list of student profiles | The system returns a list of all student profiles. |
|  |  |  |
| 2 | The admin selects the profile of the student they | The system displays back an editable form with |
|  | want to update | the student profile details pre-filled. |
|  |  |  |
| 3 | The admin updates the parts they want to update and | The system updates the record and returns a |
|  | requests system to save the new details | success message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The student profile will be updated | |  |
|  |  |  |
| **Business Rule** | |  |
| The registration number and email fields should be unwritable | | |
|  |  |  |
|  |  |  |
| **1.2.3 Delete Student Profile** | |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin selects to view a list of student profiles | The system returns a list of all student profiles. |
|  |  |  |
| 2 | The admin selects to delete a student profile they | The system displays a confirmation dialogue |
|  | want | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system checks to confirm the student is not |
|  | window to confirm deleting the profile | assigned to any sections and deletes the profile |
|  |  | and returns a success message or a fail message on |
|  |  | failure. It must also return a specific message |
|  |  | when the student is assigned to some sections. |
|  |  |  |
| **Postconditions** | |  |
| The student profile will be deleted | |  |
|  |  |  |
| **Business Rule** | |  |
|  |  |  |
| The student must have not been assigned to any of the sections. | | |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Number: 2** | | |  |
| **Name** | | Course CRUD |  |
|  |  | |  |
| **Brief description** This use case provides a functionality for managing courses | | | |
|  |  |  |  |
| **Actors** | | Admin |  |
|  |  | |  |
|  |  |  |  |
| **Preconditions** | | |  |
| The admin must logged in to the system | | |  |
|  |  | |  |
| **Flows of Events:** | | |  |
| **1. Basic Flows** | | |  |
|  |  | |  |
| **1.0 Create Course** | | |  |
|  |  |  |  |
| **Step** |  | **User Actions** | **System Actions** |
|  |  |  |  |
| 1 |  | The admin calls the create course command | The system displays the course form with name, |
|  |  |  | code and pre-requisites |
|  |  |  |  |
| 2 |  | The Admin fills out the form and requests the | The system verifies there’s no other course with |
|  |  | system to save the details | the same code and saves the course and returns a |
|  |  |  | success message or a fail message in case of an |
|  |  |  | exception. It should return a specific message |
|  |  |  | when a duplicate course exists. |
|  |  | |  |
| **Postconditions** | | |  |
| The staff profile is persisted in the system | | |  |
|  |  | |  |
| **Business Rule** | | |  |
| No two courses can have the same code. | | |  |
|  |  | |  |
|  |  | |  |
| **1.1 Read/view Course** | | |  |
|  |  |  |  |
| **Step** |  | **User Actions** | **System Actions** |
|  |  |  |  |
| 1 |  | The admin requests for a list of courses | The system returns a list of all courses. |
|  |  |  |  |
| 2 |  | The admin selects to view a course they want from | The system returns the course object as a string |
|  |  | the list. | with course code, and course name |
|  |  | |  |
|  |  | |  |
| **1.2 Update Course** | | |  |
|  |  |  |  |
| 1 |  | The admin requests a list of courses | The system returns a list of courses. |
|  |  |  |  |
| 2 |  | The admin selects the course they want to update | The system displays an editable form with the |
|  |  |  | course details |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| 3 | The admin changes the parts they want to update | The system updates the course and returns a |
|  | and requests system to save the new details | success message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The course details will be updated | |  |
|  |  |  |
| **Business Rule** | |  |
|  |  |  |
| The course code should be unwritable. | |  |
|  |  |  |
|  |  |  |
| **1.1.3 Delete Course** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin requests for a list of courses | The system returns a list of all courses. |
|  |  |  |
| 2 | The admin selects to delete a course from a list of | The system displays a confirmation dialogue |
|  | courses | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system confirms the course is not assigned to |
|  | window to confirm deleting the course | any sections and deletes and returns a success |
|  |  | message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The course will be deleted | |  |
|  |  |  |
|  |  |  |
| **Business Rule** | |  |
| The course must not have been assigned to some section | | |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Number: 3** | | | |  |
| **Name** | |  | Block CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case provides a functionality for managing blocks | |
|  |  | |  |  |
| **Actors** | | | Admin |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Preconditions** | | |  |  |
| The admin must logged in to the system | | | |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | |  |  |
| **1.0 Create Block** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin calls the create block command | | The system displays the create block form with name, |
|  |  |  |  | sequence number, start date and end date. |
|  |  |  | |  |
| 2 |  | The Admin enters the block start and end dates, the block | | The system checks if there’s no other block with same |
|  |  | identifier number, and a name and requests the system to | | name, start and end dates and saves the block and |
|  |  | save the details |  | returns a success message or a fail message on failure. |
|  |  |  |  | It should return a specific message when a duplicate |
|  |  |  |  | block exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The block is persisted in the system | | | |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Business Rules** | | |  |  |
| A block must be unique. A unique block is identified by the name and the start and end dates | | | | |
|  |  | |  |  |
|  |  | |  |  |
| **1.1 Read/view Block** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin requests for a list of blokcs | | The system returns a list of all blocks. |
|  |  |  | |  |
| 2 |  | The admin selects to view a block they want | | The system returns the block object as a string |
|  |  |  |  | with block name, start and end dates |
|  |  | |  |  |
|  |  | |  |  |
| **1.2 Update Block** | | |  |  |
|  |  |  | |  |
| 1 |  | The admin requests for a list of blocks | | The system returns a list of all blocks. |
|  |  |  | |  |
| 2 |  | The admin selects the block they want to update | | The system displays back an editable form with |
|  |  |  |  |  |

the block details pre-populated

* The admin changes the parts they want to update and The system updates the block record and returns a

|  |  |
| --- | --- |
| requests the system to save the new details | success message or a fail message on exception. |

**Postconditions**

The block details will be updated

**Business Rule**

The name, start and end dates should be unwritable.

**1.1.3 Delete Block**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin requests for a list of blocks | The system returns a list of all blocks. |
|  |  |  |
| 2 | The admin selects to delete the block they want | The system displays a confirmation dialogue |
|  |  | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system checks that the block is not linked to |
|  | window to confirm deleting the block | any section and deletes the it and returns a success |
|  |  | message or a fail message on an exception. |
|  |  | Otherwise, it will return a message indicating the |
|  |  | block is associated to some sections. |
|  |  |  |

**Postconditions**

The block will be deleted

**Business Rule**

The block must not be linked to some sections

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Number: 4** | | | |  |
| **Name** | |  | Entry CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case provides a functionality for managing Entry’s | |
|  |  | |  |  |
| **Actors** | | | Admin |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Preconditions** | | |  |  |
| The admin must log in to the system | | | |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | |  |  |
| **1.0 Create Entry** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin calls the create Entry command | | The system displays the create Entry form with name, |
|  |  |  |  | sequence number, start date and end date. |
|  |  |  | |  |
| 2 |  | The Admin enters the Entry start and end dates, the Entry | | The system checks if there’s no other Entry with same |
|  |  | identifier number, and a name and requests the system to | | name, start and end dates and saves the Entry and |
|  |  | save the details |  | returns a success message or a fail message on failure. |
|  |  |  |  | It should return a specific message when a duplicate |
|  |  |  |  | Entry exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The Entry is persisted in the system | | | |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Business Rules** | | |  |  |
| A Entry must be unique. A unique block is identified by the name and the start and end dates | | | | |
|  |  | |  |  |
|  |  | |  |  |
| **1.1 Read/view Entry** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin requests for a list of Entry | | The system returns a list of all Entry’s. |
|  |  |  | |  |
| 2 |  | The admin selects to view a Entry they want | | The system returns the Entry object as a string |
|  |  |  |  | with Entry name, start and end dates |
|  |  | |  |  |
|  |  | |  |  |
| **1.2 Update Entry** | | |  |  |
|  |  |  | |  |
| 1 |  | The admin requests for a list of Entry | | The system returns a list of all Entry’s. |
|  |  |  | |  |
| 2 |  | The admin selects the Entry they want to update | | The system displays back an editable form with |
|  |  |  |  |  |

the Entry details.

* The admin changes the parts they want to update, and the system updates the block record and returns a

|  |  |
| --- | --- |
| requests the system to save the new details | success message or a failure message on exception. |

**Postconditions**

The Entry details will be updated.

**Business Rule**

The name, start and end dates should be unwritable.

**1.1.3 Delete Entry**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin requests for a list of Entry | The system returns a list of all Entry’s. |
|  |  |  |
| 2 | The admin selects to delete the Entry they want | The system displays a confirmation dialogue |
|  |  | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system checks that the Entry is not linked to |
|  | window to confirm deleting the Entry | any section and deletes the it and returns a success |
|  |  | message or a fail message on an exception. |
|  |  | Otherwise, it will return a message indicating the |
|  |  | Entry is associated to some sections. |
|  |  |  |

**Postconditions**

The Entry will be deleted.

**Business Rule**

The Entry must not be linked to block.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Number: 5** | | | |  |
| **Name** | |  | Schedule CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case provides a functionality for managing Schedule | |
|  |  | |  |  |
| **Actors** | | | Admin |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Preconditions** | | |  |  |
| The admin must log in to the system | | | |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | |  |  |
| **1.0 Create Schedule** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin calls the create Schedule command | | The system displays the create Entry form with name, |
|  |  |  |  | sequence number, start date and end date. |
|  |  |  | |  |
| 2 |  | The Admin enters the Entry start and end dates, the Entry | | The system checks if there’s no other Schedule with same |
|  |  | identifier number, and a name and requests the system to | | name, start and end dates and saves the Schedule and |
|  |  | save the details |  | returns a success message or a fail message on failure. |
|  |  |  |  | It should return a specific message when a duplicate |
|  |  |  |  | Schedule exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The Schedule is persisted in the system | | | |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Business Rules** | | |  |  |
| N/A | | | | |
|  |  | |  |  |
|  |  | |  |  |
| **1.1 Read/view Schedule** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin requests for a list of Schedule | | The system returns a list of all Schedule we have . |
|  |  |  | |  |
| 2 |  | The admin selects to view a Schedule they want | | The system returns the Entry object as a string |
|  |  |  |  | with Schedule name, start and end dates |
|  |  | |  |  |
|  |  | |  |  |
| **1.2 Update Entry** | | |  |  |
|  |  |  | |  |
| 1 |  | The admin requests for a list of Schedule | | The system returns a list of all Schedules. |
|  |  |  | |  |
| 2 |  | The admin selects the Schedule they want to update | | The system displays back an editable form with |
|  |  |  |  |  |

the Schedule details .

* The admin changes the parts they want to update and The system updates the block record and returns a

|  |  |
| --- | --- |
| requests the system to save the new details | success message or a fail message on exception. |

**Postconditions**

The Schedule details will be updated

**Business Rule**

The name, start and end dates should be unwritable.

**1.1.3 Delete Schedule**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin requests for a list of Schedule | The system returns a list of all Schedules. |
|  |  |  |
| 2 | The admin selects to delete the Schedule they want | The system displays a confirmation dialogue |
|  |  | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system checks that the Schedule is not linked to |
|  | window to confirm deleting the Schedule | any section and deletes the it and returns a success |
|  |  | message or a fail message on an exception. |
|  |  | Otherwise, it will return a message indicating the |
|  |  | Schedule is associated to some sections. |
|  |  |  |

**Postconditions**

The Schedule will be deleted.

**Business Rule**

N/A.