**Group 7:**

**Oussama Jablaoui /Adham Mohamed /Abanoub Kaiser / Mark Ghatas**

**MUMSched: Use Case Description**

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 1** | | |
| **Name** Manage user profile | | |
| **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 to save the details | The system verifies that there’s no other profile in 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 faculty from the list of profiles | The system returns the profile of the faculty as a 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 want to update | The system displays an editable faculty profile form pre-populated with the faculty profile details |
| 3 | The admin updates the fields they want to update and requests system to save the new details | The system updates the record and returns the 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 list of faculty profiles | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the profile | The system confirms the faculty is not assigned to 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 to save the details | The system checks if no other student profile has 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 want to update | The system displays back an editable form with the student profile details pre-filled. |
| 3 | The admin updates the parts they want to update and requests system to save the new details | The system updates the record and returns a 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 want | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the profile | The system checks to confirm the student is not 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** Manage entries | | |
| **Brief description** This use case provides a functionality for managing entries | | |
| **Actors** Admin | | |
|  | | |
| **Preconditions** | | |
| The admin must logged 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 identifier number, and a name and requests the system to save the details | The system checks if there’s no other entry with same name, start and end dates and saves the block and 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** | | |
| An entry must be unique. A unique entry 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 entries | The system returns a list of all entries. |
| 2 | The admin selects to view an 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 entries | The system returns a list of all entries. |
| 2 | The admin selects the entry they want to update | The system displays back an editable form with the entry details pre-populated |

|  |  |  |
| --- | --- | --- |
| 3 | The admin changes the parts they want to update and requests the system to save the new details | The system updates the entry record and returns a success message or a fail 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 entries | The system returns a list of all blocks. |
| 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 window to confirm deleting the block | The system checks that the entry is not linked to any block 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 blocks. |
| **Postconditions** | | |
| The entry will be deleted | | |
| **Business Rule** | | |
| The entry must not be linked to some blocks | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 3** | | |
| **Name** Manage blocks | | |
| **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 identifier number, and a name and requests the system to save the details | The system checks if there’s no other block with same name, start and end dates and saves the block and 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 |
| 3 | The admin changes the parts they want to update and requests the system to save the new details | The system updates the block record and returns a 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 window to confirm deleting the block | The system checks that the block is not linked to any schedule 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 schedules. |
| **Postconditions** | | |
| The block will be deleted | | |
| **Business Rule** | | |
| The block must not be linked to some schedules | | |
|  | | |

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

|  |  |  |
| --- | --- | --- |
| 3 | The admin changes the parts they want to update and requests the system to save the new details | The system updates the section record and returns a success message or a fail message on exception. |
| **Postconditions** | | |
| The section details will be updated | | |
| **Business Rule** | | |
| The section details should be unwritable. | | |
|  | | |
| **1.1.3 Delete Section** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The admin requests for a list of sections | The system returns a list of all sections. |
| 2 | The admin selects to delete the section they want | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the section | The system checks that the section is not linked to 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 section is associated to some sections. |
| **Postconditions** | | |
| The section will be deleted | | |
| **Business Rule** | | |
| The section must not be linked to some blocks | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 5** | | |
| **Name** Manage courses | | |
| **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 system to save the details | The system verifies there’s no other course with 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 list. | The system returns the course object as a string 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 and requests system to save the new details | The system updates the course and returns a 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 courses | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the course | The system confirms the course is not assigned to 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: 6** | | |
| **Name** Manage schedules | | |
| **Brief description** This use case provides a functionality for managing schedules | | |
| **Actors** Admin | | |
|  | | |
| **Preconditions** | | |
| The admin must logged 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 schedule form with sequence code with list of available blocks |
| 2 | The Admin fills out the form and link the blocks to the schedule and requests the system to save the details | The system verifies there’s no other schedule details returns a success message or a fail message in case of an exception. It should return a specific message when a duplicate schedule exists. |
| **Postconditions** | | |
| The schedule is persisted in the system | | |
| **Business Rule** | | |
| No two schedule can have the same code and same list of blocks. | | |
|  | | |
| **1.1 Read/view Schedule** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The admin requests for a list of schedules | The system returns a list of all schedules per entry. |
| 2 | The admin selects to view a schedule they want from the list. | The system returns the schedule object as a string with code and list of blocks associated with. |
|  | | |
| **1.2 Update Schedule** | | |
| 1 | The admin requests a list of schedules | The system returns a list of schedules. |
| 2 | The admin selects the schedule they want to update | The system displays an editable form with the schedule details |

|  |  |  |
| --- | --- | --- |
| 3 | The admin changes the parts they want to update and requests system to save the new details | The system updates the schedule and returns a success message or a fail message on exception. |
| **Postconditions** | | |
| The schedule details will be updated | | |
| **Business Rule** | | |
| The schedule code should be unwritable. | | |
|  | | |
| **1.1.3 Delete Schedule** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The admin requests for a list of schedules | The system returns a list of all schedules. |
| 2 | The admin selects to delete a schedule from a list of schedules | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the schedule | The system confirms the schedule returns a success message or a fail message on exception. |
| **Postconditions** | | |
| The schedule will be deleted | | |
|  | | |
| **Business Rule** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 7** | | |
| **Name** Login | | |
| **Brief description** This use case provides a functionality for user logging | | |
| **Actors** Admin  Faculty  Student | | |
|  | | |
| **Preconditions** | | |
| No precondition required | | |
| **Flows of Events:** | | |
| **1. Basic Flows** | | |
| **1.0 Login** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The user request the login form | The system displays the login form with email address and password |
| 2 | The user fills out the form and requests the system to log in through submit button | The system verifies if the login details matches the data persisted and returns a success message or a fail message in case of an exception. It should return a specific message when a the user doesn’t exist or password doesn’t match. |
| **Postconditions** | | |
| The user login into the system | | |
| **Business Rule** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 8** | | |
| **Name** view schedules | | |
| **Brief description** This use case provides a functionality for viewing schedules | | |
| **Actors** Faculty  Student | | |
|  | | |
| **Preconditions** | | |
| The used must be logged in to the system | | |
| **Flows of Events:** | | |
| **1. Basic Flows** | | |
| **1.0 view Schedules** | | |
| **Step** | **User Actions** | **System Actions** |
| 1 | The user request the list of schedules | The system displays the list of schedules as a response to the request |
| **Postconditions** | | |
| None | | |
| **Business Rule** | | |
| Users except admin don’t have the right to manage schedules. | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 9** | | |
| **Name** Manage profile | | |
| **Brief description** This use case allows the faculty and student to manage their profiles | | |
| **Actors** Faculty  Student | | |
| **Preconditions** | | |
| The user must be logged in to the system | | |
| **Flows of Events:** | | |
| **1. Basic Flows** | | |
| **1.1.1 Read/view Profile** | | |
| **Step** | **User Actions** | **System Actions** |

|  |  |  |
| --- | --- | --- |
| 1 | The user selects to view his profile | The system returns user profile with all its details. |
|  | | |
| **1.1.2 Update Profile** | | |
| **Step** | **User Actions** | **System Action** |
| 1 | The user selects to view his profile | The system returns user profile with all its details. |
| 2 | The user updates the fields they want to update and requests system to save the new details | The system updates the record and returns the success message or a fail message on exception. |
| **Postconditions** | | |
| The user profile will be updated | | |
| **Business Rule** | | |
| The email field should be unwritable. | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 10** | | |
| **Name** Sign up | | |
| **Brief description** This use case allows the student to sign up | | |
| **Actors** Student | | |
| **Preconditions** | | |
| No precondition required | | |
| **Flows of Events:** | | |
| **1. Basic Flows** | | |
| **1.1.1 Sign up** | | |
| **Step** | **User Actions** | **System Action** |
| 1 | The student request the sign up form | The system returns sign up form with all details to be filled. |
| 2 | The student fills the form and request the system to save it . | The system updates the record and returns the success message or a fail message on exception if it finds a duplicate student with same information as name , surname and email address. |
| **Postconditions** | | |
| The student will be created but not activated yet by the faculty advisor | | |
| **Business Rule** | | |
| No duplicate email addresses for student | | |

|  |  |  |
| --- | --- | --- |
| **Use Case Number: 11** | | |
| **Name** Register for a section | | |
| **Brief description** This use case allows the student to register for a section | | |
| **Actors** Student | | |
| **Preconditions** | | |
| The student must be logged in to the system | | |
| **Flows of Events:** | | |
| **1. Basic Flows** | | |
| **1.1.1 Register for a section** | | |
| **Step** | **User Actions** | **System Action** |
| 1 | The student request the list of sections | The system returns the list of sections persisted and available . |
| 2 | The student select his/her preferable section and request the system to enroll . | The system check the student meets the requirements for enrolling to that specific section and returns message or a fail message on exception the student profile doesn’t matches with wanted requirements. |
| **Postconditions** | | |
| The section will be added to student schedule | | |
| **Business Rule** | | |
| No duplicate sections can be selected by the student . | | |