**SRS FOR “Course Registration System - CRS”**

**Team members:**

*Murad Hajiyev (610691), Uuganbayar Oyun (610751), Mohamed Ibrahim (610695), Odbayar Nanjidjamts (610526)*

**Github repository:**

<https://github.com/muradhajiyev/cs425-CRS>

**1. Introduction**

Several years ago in the Computer Professional MS in CS program, there were three

entries per year and student entry numbers were 20-40 per entry. Often there was just one

elective class being offered per block and all students in an entry took the same classes in

the same sequence. Scheduling of classes and faculty was done with a relatively simple

Excel spreadsheet, and students were assigned to classes via a manual process.

As Compro has grown, we now offer 4 entries per year and there are often 100 – 130

students per entry. In some blocks, we may offer 8 or 9 elective classes, plus there are

often 3 FPP classes and 5 MPP classes offered per entry. There are several areas of

specialization for classes such as:

- Web Applications

- Data Science

- SW Design

- Networking

- Operating Systems

- Compilers

- Parallel Programming, etc.

Most faculties have one or two areas of specialization and a set of classes that they

would like to teach. In addition, they have preferences for what blocks they can teach.

Faculty needs to be able to enter their profile and be able to view their scheduled classes.

Compro students should be able to view the schedule and register for classes.

A few 500 level courses have 400 level **course prerequisites**, so the 400 level courses

should be offered for each entry in their first blocks on campus.

The 500 level classes should be provided for their later blocks on campus.

Most students take 4 elective blocks on campus.

Some U.S. resident students take 9 elective blocks on campus.

Some OPT students take 5 courses on campus.

CRS is a new software tool that will build a Compro schedule of classes with

faculty assigned to each class and will also offer a simple tool for students to register for

those classes.\*

(\*Note – the student registration part will be kept simple for our project. It is added for

the purposes of having a separate student register subsystem – to be explained in later.)

**2. Positioning**

**2.1 Problem Statement**

|  |  |
| --- | --- |
| The problem of | *managing the Compro schedule and allowing students to*  *register for classes* |
| Affects | *administrators, faculty, and students* |
| the impact of which is | *scheduling is complex, must be manually maintained, and*  *changed frequently* |
| a successful solution would be | *one tool which builds a Compro schedule that integrates the*  *business rules for faculty availability and courses needed by*  *students per entry. This tool will provide a Database and a*  *user interface that is easy to use for faculty, staff, and*  *students.* |

**2.2 Product Position Statement**

|  |  |
| --- | --- |
| For | *administrators, faculty, and students* |
| Who | *manages and registers course schedule.* |
| The (CRS) | *Is Course registration system* |
| That | *Allows students to register to courses, and give chance to administrators to manage courses and assign them to faculties, etc.* |
| Unlike | *Current system – excel spreadsheet* |
| Our product | *Allows students to register to courses, and give chance to administrators to manage courses and assign them to faculties through web platform.* |

**3. Stakeholder Descriptions**

**3.1 Stakeholder Summary**

****

**3.2 User Environment**

It is intended for university administrators, faculty members and all students. Minimum target user count should be 100. It is changeable. The task cycle takes 1 week. Member may spend 1 day for each activity. It may change. Excel is being used right now. CRS will be used in future.

**4. Product Overview**

**4.1 Product Perspective**

It is a web based system implementing client-server model. The CRS provides simple mechanism for students to register to the courses.

The following are the main features that are included in CRS

User account: The system allows the user to create their accounts in the system and provide features of updating and viewing profiles.

Number of users being supported by the system: Though the number is precisely not mentioned but the system is able to support a large number of online users at a time.

Search: search is simply local search engine based on key words.

FAQs section: Frequently asked section contain answer of problem which CRS user frequently faced.

**4.2 Assumptions and Dependencies**

A student will take a course each block. Professor can give maximum a course per block on campus. He may teach DE course at the same time.

**4.3 Needs and Features**























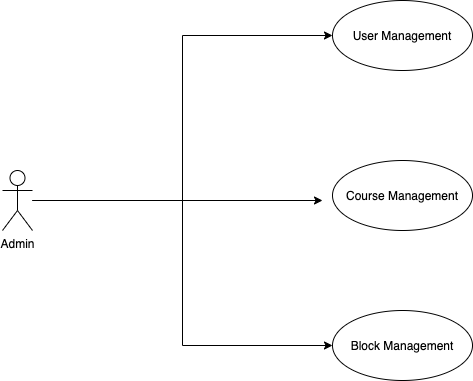
**4.4 Alternatives and Competition**

The other groups in classroom are our alternatives and competitors.

**5. Other Product Requirements**

System should be implemented using Java Spring. Relation database might be selected as Mysql or Postgresql.

**CRS use case diagram**

****

**CRS: Use Case Description**

|  |  |  |
| --- | --- | --- |
| **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 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** 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 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: 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 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 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 | | |
|  | | |