Exam System for VIA University College – Analysis Document

Summary

VIA University College Software Engineering needs a system that can arrange all exam dates, time and rooms needed. The exam type needs to be specific (Oral or Written) in order to work with our system.

The system is focusing only on scheduling exams for Software Engineering students between 1st and 4th semester, but it can be adjusted for other study programs in the university. The exam system needs to be easily accessible in order to arrange the proper time, date and room needed for any given exam. The program should be able to handle names and store information about rooms (when they are free and when they are used for classes or something else.)

System will be used mostly from the secretaries, so we put them as our main target user, design wise should be simplistic. The program should be easy to use and understand.

Functional requirements

**Critical priority:**

1. As a secretary, I want to be able to add an exam at a specific date and room, in order to prevent overlapping.
2. As a secretary, I want to be able to make lists of exams, for students to know where and when the exam is held.
3. As a secretary, I want to have a list of examiners and which exams they are attending, in order to add them in the system.
4. As a secretary, I want to be able to add, edit and delete scheduled individual exams, in order to make, change and remove data and exams if necessary.
5. As a secretary, I want to be able to change and update the schedule at any given time, in order to prevent overlapping on dates and rooms.
6. As a secretary, I want to be able to check and edit if necessary, whether a class has more than one written exam in one day, in order to not exceed the limit of one written exam per day per student.

**High priority:**

1. As a secretary, I want to have access to student information, in order to be able to arrange proper exams.
2. As a secretary, I want to have a list with equipment for each class, in order to make sure oral exams can take place there, e. g.: has HDMI.
3. As a secretary, I want to be able to add teacher names for the exams and change when

necessary, in order to keep track of number of examiners attending.

1. As a secretary, I want to be able to edit the lists of examiners, courses and rooms, in order to keep the information updated.

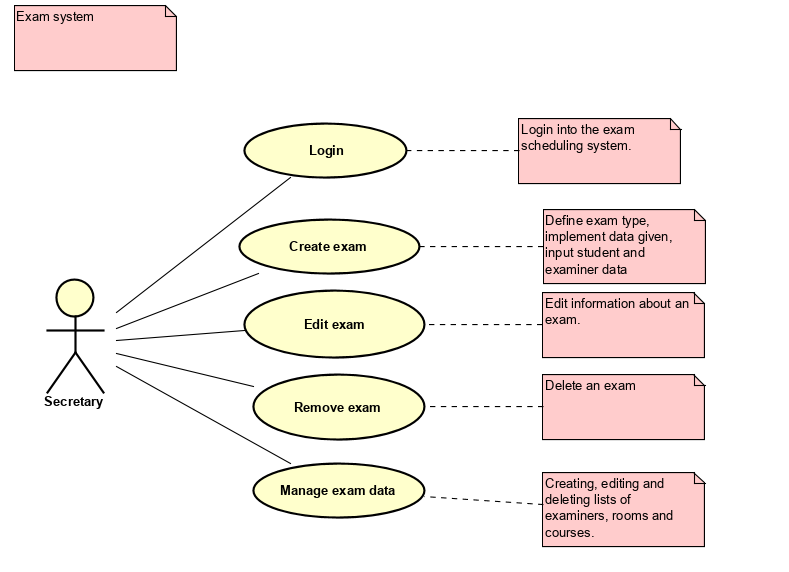
**Low priority:**

1. As a secretary, I want to be able to arrange a 3-day SEP exam, as this is the standard procedure.
2. As a secretary, I want to be able to notify students for upcoming exams, room numbers and dates and send them a reminder message 3 days before the exam, in order to make sure they are aware of the schedule.
3. As a secretary, I want to be able to have a login, for security reasons.

Non-functional requirements

1. User interface should be practical and easy to use.
2. The main programming language of the system must be JAVA.
3. The system must be a single user software.

User Case Diagram



Use case descriptions

|  |  |
| --- | --- |
| **Use case** | **Login** |
| **Summary** | The secretary has to know the username and the password in order to enter the system. |
| **Actor** | Secretary |
| **Precondition** | The user must know the login details. |
| **Postcondition** | The user is logged into the system. |
| **Base sequence** | 1. The system displays the login window. 2. The secretary types the username. 3. The secretary types the password. 4. The system displays the main window. |
| **Exception Sequence** | If the username or password are wrong, the user has to type them again. |
| **Note** |  |

|  |  |
| --- | --- |
| **Use case** | **Create exam** |
| **Summary** | Add an exam and add information for the exam – semester, date, room, participants and what type of exam it is. |
| **Actor** | Secretary |
| **Precondition** | Information should be available, or created (date, room, type of exam, participants, semester) |
| **Postcondition** | An exam has been created. The information about the exam has been added, which includes date, room, examiners, course. |
| **Base sequence** | 1. Add the course which includes the semester and the class 2. Add the date for the exam 3. Add the examiners for the exam 4. Add the room for exam |
| **Exception Sequence** | If the user tries to create the same exam for a subject that has already been created once:  Step 1-6 as base sequence.  The process should be terminated, to prevent overlapping of data. |
| **Note** | The process can be canceled at any time.  If any information written and scheduled is wrong, the information will be easy to change and put in properly in the next use case. |

|  |  |
| --- | --- |
| **Use case** | **Edit exam** |
| **Summary** | Edit exam data |
| **Actor** | Secretary |
| **Precondition** | The exam should be created. |
| **Postcondition** | The exam data is now visible to users. Wrong data can be changed. |
| **Base sequence** | In case of wrong input, change any information needed in the next steps:   1. Choose an exam you have previously created. 2. Change type of exam, if not needed choose the same one. 3. Change date, if not needed choose the same one. 4. Change examiners, if not needed choose the same ones. 5. Change the room, if not needed choose the same one. |
| **Exception Sequence** | If the user adds an exam and a student/examiner already has/is assigned to an exam or the room is occupied. The process should be terminated in order to prevent overlapping. |
| **Note** | For example, if the user chooses the wrong examiner and realizes the mistake after saving the information, the user will easily access the same data about the examiner and change to the right one.  If a room is reserved for another exam already, the program will give the user a notice and make the user change the room to an available one from the list. |

|  |  |
| --- | --- |
| **Use case** | **Remove exam** |
| **Summary** | Delete wrong exams. |
| **Actor** | Secretary |
| **Precondition** | The exam should be created. |
| **Postcondition** | The exam chosen is now deleted from the system. |
| **Base sequence** | 1. Select an exam. 2. Remove the exam. |
| **Exception sequence** |  |
| **Note** |  |

|  |  |
| --- | --- |
| **Use case** | **Manage exam data** |
| **Summary** | List creator for examiners, courses and rooms. Selected examiners, courses and rooms can be edited and/or deleted. |
| **Actor** | Secretary |
| **Precondition** | - |
| **Postcondition** | The lists are created or updated. |
| **Base sequence** | 1. Create course 2. Create room 3. Create examiner 4. If needed, change information about course. 5. If needed, change information about room. 6. If needed, change information about examiner. 7. If needed, delete selected course/room/examiner. |
| **Exception sequence** |  |
| **Note** |  |

Link between requirements and use cases

|  |  |
| --- | --- |
| **Use Case** | **Covered requirements** |
| Login | 13 |
| Create exam | 1, 4, 9, 11 |
| Edit exam | 4, 5, 6, 9 |
| Remove exam | 4 |
| Manage exam data | 2, 3, 4, 5, 7, 8, 9, 10, 12 |

|  |  |
| --- | --- |
| **Requirements** | **Related Use Case** |
| 1 | Create exam |
| 2 | Manage exam data |
| 3 | Manage exam data |
| 4 | Create exam, Edit exam, Remove exam, Manage exam data |
| 5 | Edit exam, Manage exam data |
| 6 | Edit exam |
| 7 | Manage exam data |
| 8 | Manage exam data |
| 9 | Create exam, Edit exam, Manage exam data |
| 10 | Manage exam data |
| 11 | Create exam |
| 12 | Manage exam data |
| 13 | Login |

Activity diagram

