# **5.** **Dynamic Design Scenarios**

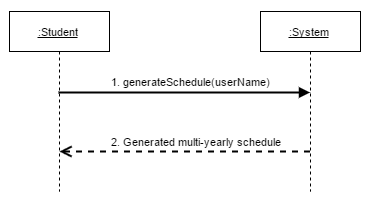
**5.1 Generate Schedule**

*Generate schedule*, as the name says, consists of creating a schedule for a student actor by using his/her preferences, list of taken classes and list of needed classes.

5.1.1 Full Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Generate Schedule | Author | Ying-Chen Chu |
| Identifier: | UC13 | Version: | 2.0 |
| Date Created: | 2015-02-03 | Last Modified: | 2015-03-22 |
| Importance: | 5/5 | | |
| Actor(s): | Student | | |
| Goal: | Generate a schedule | | |
| Summary: | Based on the course selection list and or without the list of preferences, the system generates schedule(s) for the selected courses. | | |
| Related use-cases: | UC "**View Schedule**" | | |
| Preconditions | -User has been authenticated | | |
| Trigger: | User activates the "Generate Schedule" process | | |
| Basic Flow: | 1. System generates multi-yearly schedules by fetching and using the preferences, needed courses list, taken courses list.  2. Schedules are generated are displayed to the user. | | |
| Post-Conditions: | Schedule for the course selected (and or without the preferences) is generated. | | |
| Minimum Guarantee: | Previous state of the system remains the same. | | |
| Risk Assessment: | High | | |
| Notes | A schedule could always be generated even though the preferences and needed/taken courses where not set by the student. A schedule based on the regular course sequence would display starting from the current semester, assuming the student is a first year. | | |

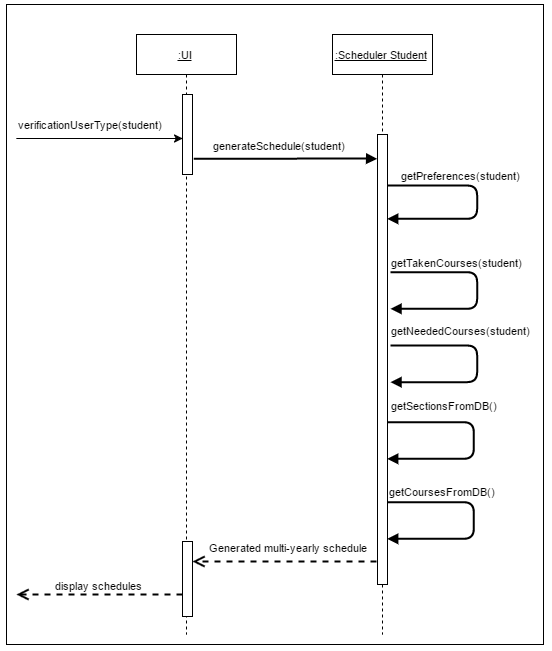
5.1.2 System Sequence Diagram



5.1.2.1 *generateSchedule(userName)* Operating Contracts & Respective Sequence Diagram

|  |  |
| --- | --- |
| **Name** | Contract 1.1 Generate Schedule |
| **Operation** | generateSchedule(Student:astudent) |
| **Cross Reference** | UC13 Generate Schedule |
| **Pre conditions** | -User is logged into the system |
| **Post conditions** | -If there is any restriction (preferences or course lists), the schedule is displayed according to those restrictions.  -If there are no restrictions, the schedule is displayed according to the general course sequence, starting from the current semester, and assuming this is the students first semester. |

Sequence Diagram for contract 1.1



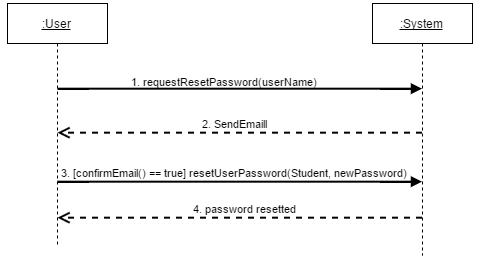
**5.2 Reset Password**

*Reset password* consists of changing a forgotten password to a new one, meaning the user cannot login into the system. An email is sent to the user. Once confirmed, the user can proceed and change the password.

5.2.1 Full Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Reset Password | Author | Salma Aly |
| Identifier: | UC3 | Version: | 2.0 |
| Date Created: | Jan 30, 2016 | Last Modified: | 2015-03-22 |
| Importance: | 5/5 | | |
| Actor(s): | Student, Admin | | |
| Goal: | To reset password | | |
| Summary: | The user requests password reset and the system sends them an email that should be confirmed . The user then enters a new password. | | |
| Related use-cases: | UC "**Confirm Email**" | | |
| Preconditions | 1-The user has already signed up to the system  2-The user has provided a valid email address | | |
| Trigger: | The user initiates a password reset | | |
| Basic Flow: | 1-System receives a request to reset the password to a specific user  2-System sends an email to user  3-If email confirmed, the user enters a new password  4-System displays a password reset successfully message | | |
| Post-Conditions: | -Previous password has been replaced by the new password  -User can now login with the new password. | | |
| Minimum Guarantee: | The user can view the login page with reset password option | | |
| Risk Assessment: | Low | | |

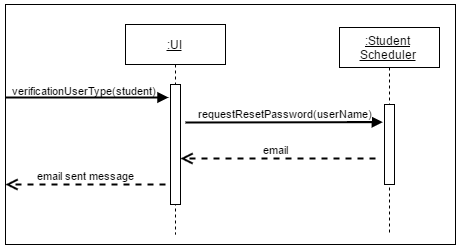
5.2.2 System Sequence Diagram



5.2.2.1 *requestResetPassword(userName)* Operating Contracts

|  |  |
| --- | --- |
| **Name** | Contract 2.1 Request Password Reset |
| **Operation** | requestNewPassword(String: userName) |
| **Cross Reference** | UC3 Reset password |
| **Pre conditions** | -User has an existing account  -User has a valid email linked to their account |
| **Post conditions** | -Email is sent to the user to be confirmed |

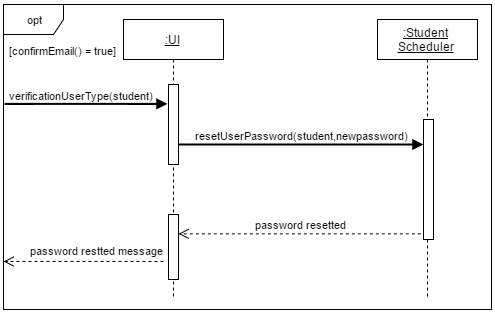
Sequence diagram for contract 2.1



5.2.2.2 *[confirmEmail() == true] resetUserPassword(Student, newPassword*) Operating Contracts

|  |  |
| --- | --- |
| **Name** | Contract 2.2 Reset Password |
| **Operation** | resetUserPassword(User:auser,String:newPassword) |
| **Cross Reference** | UC3 Reset password |
| **Pre conditions** | -User has an existing account  -User has a valid email linked to their account  -User has confirmed the email sent |
| **Post conditions** | -Old password has been replaced by the new password  -User can login into the system using the new password |

Sequence diagram for contract 2.2



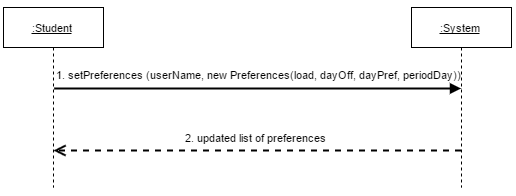
**5.3 Set Preferences**

*Set preferences* consists of saving the students preferences into the database, so they can be used later while generating the schedules.

5.3.1 Full Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Set Preferences | Author | Ying-Chen Chu |
| Identifier: | UC11 | Version: | 2.0 |
| Date Created: | 2015-02-03 | Last Modified: | 2015-03-22 |
| Importance: | 5/5 | | |
| Actor(s): | Student | | |
| Goal: | Input the schedule preferences to the system | | |
| Summary: | Set the schedule preferences and save them to the system. | | |
| Related use-cases: | - | | |
| Preconditions | User has been authenticated | | |
| Trigger: | Student sets his preferences and initiates the "set preferences" process | | |
| Basic Flow: | 1. System replaces the old preferences by the new ones.  2. System displays the list of the new preferences. | | |
| Post-Conditions: | -The user’s old preferences are replaced by the new preferences.  -Preferences are associated with the current student. | | |
| Minimum Guarantee: | Previous state of the system remains unchanged. | | |
| Risk Assessment: | Low | | |
| Notes | Preferences are not mandatory for the schedule. Therefore, the "old preferences" could also be the default preferences, which is basically no preferences (all elements set to null). | | |

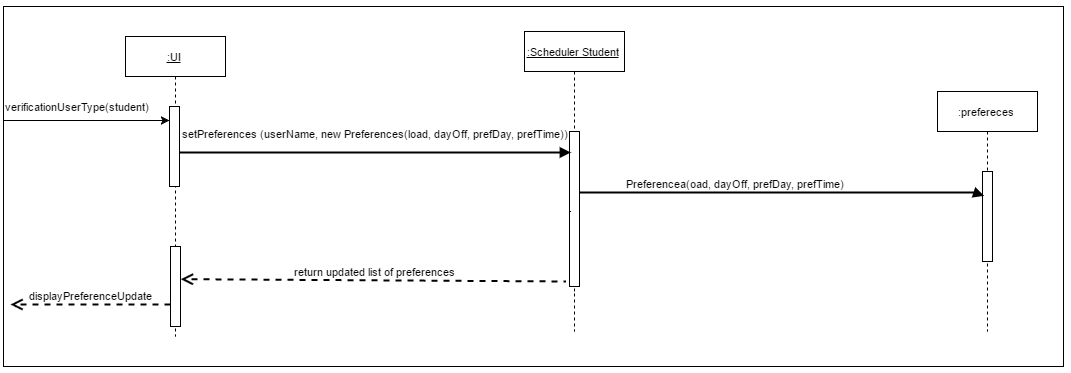
5.3.2 System Sequence Diagram



5.3.2.1 *setPreferences (userName, new Preferences(load, dayOff, dayPref, periodDay))* Operating Contracts

|  |  |
| --- | --- |
| **Name** | Contract 3.1 Set Prefrences |
| **Operation** | setPreferences(Student: student, Preferences: pref) |
| **Cross Reference** | UC11 Set Preferences |
| **Pre conditions** | Student logged into the System |
| **Post Conditions** | -Student’s preferences are saved into the database and associated to this specific user.  -Old preferences are replaced by the new preferences. |

Sequence Diagram for Contract 3.1



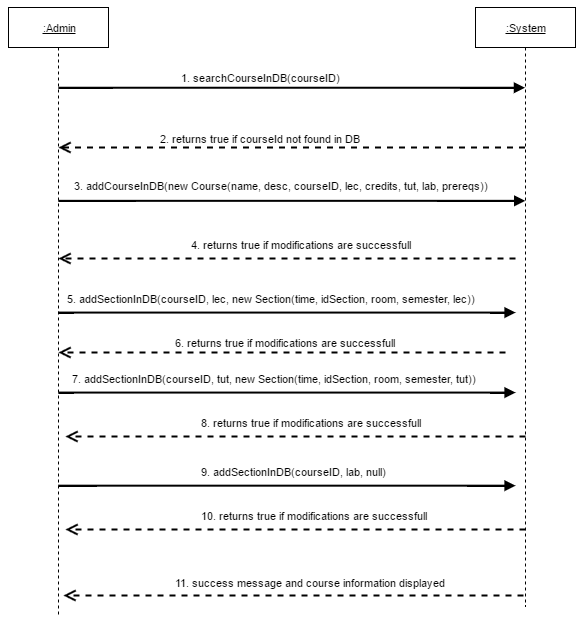
**5.4 Add Course to Program**

*Add course to program* consists of adding a course to the entire database of courses by the admin. By adding a course, the admin has to add the corresponding sections for lab, tutorial and lecture to this course.

5.4.1 Full Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Add Course to Program | Author | Adil Hssaini |
| Identifier: | UC23 | Version: | 2.0 |
| Date Created: | Feb 2, 2016 | Last Modified: | 2015-03-22 |
| Importance: | 5/5 | | |
| Actor(s): | Admin | | |
| Goal: | To add a new course to a specific program. | | |
| Summary: | The Administrator updates the list of required courses for a specific program by adding a new course. | | |
| Related use-cases: | - | | |
| Preconditions | 1. Actor is logged on as administrator.  2. The course is not part of the program sequence stored by the system. | | |
| Trigger: | Administrator activates the “Add Course to Program” process. | | |
| Basic Flow: | 1.System searches for a specific course id into the database.  2. If the course is not found, system returns a success message.  3. System takes the inputted specifications of the course and adds it to the database.  4.System returns a success message and the results of adding the course  5. System takes in the inputted data for the lecture section and adds to the current course database.  6.System displays success message and result of adding new course.  7. System takes in the inputted data for the tutorial section and adds to the current course database.  8.System displays success message and result of adding new course.  9. System takes in the inputted data for the laboratory section and adds to the current course database.  10.System displays success message and result of adding new course.  11. Success message and the entire course information and its sections are displayed. | | |
| Post-Conditions: | Course now exists in the database. | | |
| Minimum Guarantee: | List of courses in the program stored by the system will not be affected. | | |
| Risk Assessment: | Low | | |
| Note: | If a course does not have a lab for example, the admin will be able to specify this while filling in the information. The non-existant type of section, in this case, lab, will simply be saved as null. | | |

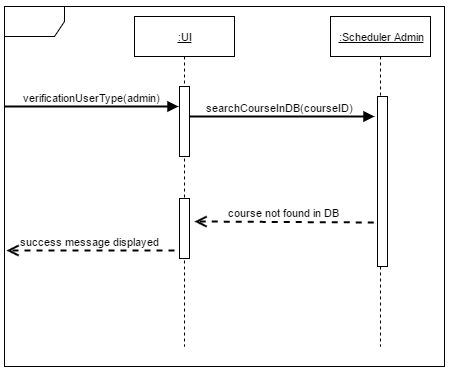
5.4.2 System Sequence Diagram



5.4.2.1 *searchCourseInDB()* Operation Contracts

|  |  |
| --- | --- |
| **Name** | Contract 4.1 Search course in database |
| **Operation** | searchCourseInDB() |
| **Cross Reference** | UC23 Add Course to Program |
| **Preconditions** | -User Logged in as administrator |
| **Postconditions** | -Course is successfully not found in database. |

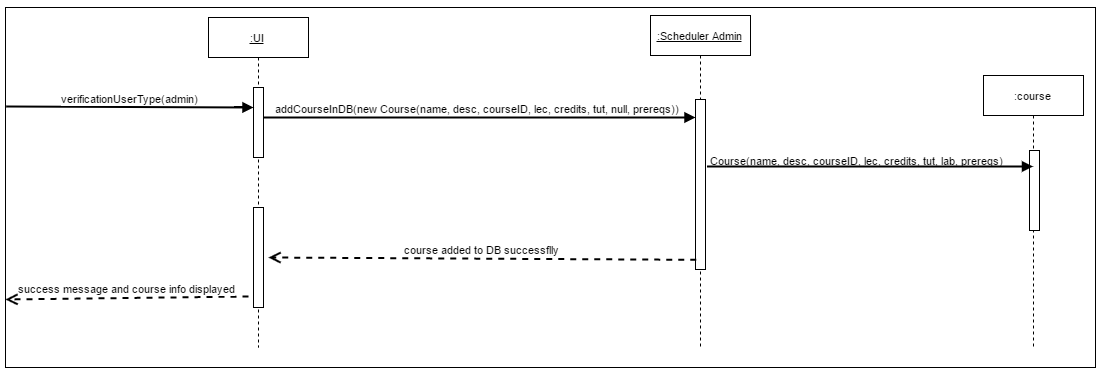
Sequence Diagram for Contract 4.1



5.4.2.2 *addCourseInDB(new Course(name, desc, courseID, lec, credits, tut, lab, prereqs))* Operation Contracts

|  |  |
| --- | --- |
| **Name** | Contract 4.2 Add course in database |
| **Operation** | modifyCourseInDB(JSON) |
| **Cross Reference** | UC23 Add Course to Program |
| **PreConditions** | -User Logged in as administrator  -Course does not already exist |
| **Post Conditions** | -Course can searched and found in database. |

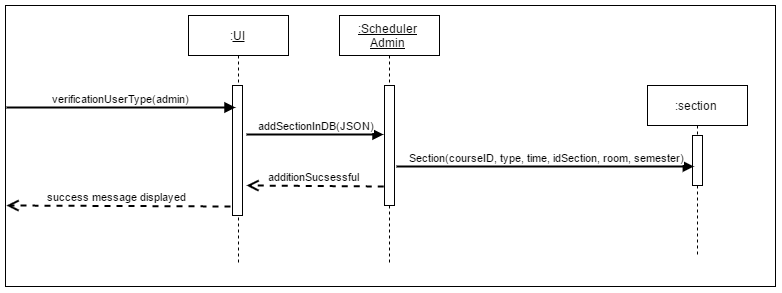
Sequence Diagram for Contract 4.2



5.4.2.3 *addSectionInDB(JSON)* Operation Contracts

|  |  |
| --- | --- |
| **Name** | Contract 4.3 Add section in database |
| **Operation** | addSectionInDB(JSON) |
| **Cross Reference** | UC23 Add Course to Program |
| **PreConditions** | -User Logged in as administrator  -There is an existing course to which the section will be added. |
| **Post Conditions** | -New section can be found inside of the current course database. |

Sequence Diagram for Contract 4.3



**Note:** The last three arrows in the system sequence diagram for this use case , are calling the same method, but adds different types of sections (Lecture, Tutorial, Lab) . Therefore contract 4.3 and its sequence diagram represent these 3 last arrows.