**Section 4 - Class Diagram Detailed Description**

The following section describes a detailed description of the concerning the class diagram of the system. The scheduler contains a total of 8 classes. The classes *Student, Admin* and *User* represent essentially the user. *Student* and *Admin* are subclasses of the *User* class. This corresponds to the possibility of a student and an admin to log in to the system. The *User* class interacts with the *UI* class where the this class simply manages the information to be displayed. The classes *Scheduler - Admin* and *Scheduler-Student*. Depending on the type of user, the UI will redirect that user the the appropriate *Scheduler* system. These classes are the core of the system since they provide functionality with the help of the other classes *Preferences, Classes & Sections.*

**Section 4.1.1 - Student Generate Schedule**

The *Student Scheduler* has a method called *generateSchedule(Student)* which generates a schedule based on the courses, preferences and sections. A method called *autogenerateTakenClasses(int)* shall automatically display the courses that are added even before finalization. The Student Scheduler contains Preferences, Course, Section and Scheduler Student. Each of these classes communicate between each other to generate schedule.

Student generated schedule is provided as part of the schedule generation component, the method listed below are elaborated:

|  |  |
| --- | --- |
| **Classes Involved** | **Preferences, Course, Section, Scheduler Student** |
| **Method(s) Implemented** | generateSchedule(Student): void  **Implemented in Class**: Scheduler Student  **Description**: Method that generates schedule for Student  **Input Parameter(s)**: An object Student  **Return Type**: void  getTakenCourses(User): Course []  **Implemented in Class**: Scheduler Student  **Description**: Method that retrieves taken courses from database  **Input Parameter(s)**: User  **Return Type**: void  getNeededCourses(User): Course []  **Implemented in Class**: Scheduler Student  **Description**: Method that returns needed courses  **Input Parameter(s)**: User  **Return Type**: Course []  autogenerateTakenClasses(int): String []  **Implemented in Class**: Scheduler Student  **Description**: Method that autogenerates taken classes  **Input Parameter(s)**: int  **Return Type**: String []  getPreferences(User):Preferences  **Implemented in Class**: Scheduler Student  **Description**: Method that returns preferences of a user  **Input Parameter(s)**: User  **Return Type**: Preferences |

**4.1.2 - Student Manage Courses**

This subsystem portrays the path for a student to manage his course load. The scheduler possesses methods such as *addNeededCourse(), dropTakenCourse()*, which allows for the user to simply add courses to his/her schedule or simply drop them. The methods *dropTakenCourse()* and *addTakenCourse()* are functions where the user himself creates his own transcript. The project and specifications defines the courses taken in the previous semesters as inputs to be put into the database.

Student Manage Courses is part of the component Course Management, the methods listed below are elaborated.

|  |  |
| --- | --- |
| **Classes Involved** | **Scheduler Student** |
| **Method(s) Implemented** | addTakenCourse(JSON): boolean  **Implemented in Class**: Scheduler Student  **Description**: Method that adds a taken course in the scheduler  **Input Parameter(s)**: takes JSON  **Return Type**: boolean  addNeededCourse(JSON): boolean  **Implemented in Class**: Scheduler Student  **Description**: Method that adds a needed course that has not been taken yet in the scheduler  **Input Parameter(s)**: takes JSON  **Return Type**: boolean  dropTakenCourse(JSON):boolean  **Implemented in Class**: Scheduler Student  **Description**: Method that remove taken course requested by the student and returns a boolean  **Input Parameter(s)**: JSON  **Return Type**: boolean  dropNeededCourse(JSON):boolean  **Implemented in Class**: Scheduler Student  **Description**: Method that remove a course that is needed by the student and returns a boolean  **Input Parameter(s)**: JSON  **Return Type**: boolean |

**4.1.3 - Student manages preferences**

For this subsystem, the scheduler contains one method called *setPreferences()* where the user who is a student can set his preferences. The *Preference* class contains attributes such as *timeOfDay, offDay*.

Student manages preferences is part of the component preferences manager and the following methods are elaborated

|  |  |
| --- | --- |
| **Classes Involved** | **Scheduler Student, Preferences** |
| **Method(s) Implemented** | setPreferences(User,Preferences):boolean  **Implemented in Class**: Scheduler Student  **Description**: Method that returns a boolean for preferences modification  **Input Parameter(s)**: User and Preferences  **Return Type**: boolean  getPreferences(User):Preferences  **Implemented in Class**: Scheduler Student  **Description**: Method that returns preferences of a user  **Input Parameter(s)**: User  **Return Type**: Preferences  setLoad(int):void  **Implemented in Class**: Preferences  **Description**: Method that sets the load for preferences  **Input Parameter(s)**: int  **Return Type**: void  setOffDay(String):void  **Implemented in Class**: Preferences  **Description**: Method that sets the days that the students wants to be off  **Input Parameter(s)**: String  **Return Type**: void  setPrefDay(String):void  **Implemented in Class**: Preferences  **Description**: Method that sets the preferred day that the student wants  **Input Parameter(s)**: String  **Return Type**: void  setTimeOfDay(String):void  **Implemented in Class**: Preferences  **Description**: Method that sets the preferred time that the student wants the course to start  **Input Parameter(s)**: String  **Return Type**: void  Preferences(): void  **Implemented in Class**: Preferences  **Description**: Default constructor that initializes preferences  **Input Parameter(s)**: Empty  **Return Type**: void |

**4.1.4 - Admin Manage Course Database**

This subsystem portrays the relationship between the priviliges of the admin and the database. The admin will set all the courses and sections that students can take through the *Scheduler Admin* class. This system contains methods that manipulate the database such as *addSectionInDB()*, *modifyCourseInDB()*.

|  |  |
| --- | --- |
| **Classes Involved** | **Scheduler Administrator** |
| **Method(s) Implemented** | addSectionInDB(JSON): boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that adds a section in the database  **Input Parameter(s)**: takes JSON  **Return Type**: boolean  dropSectionInDB(JSON): boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that removes a section from the database  **Input Parameter(s)**: takes JSON  **Return Type**: boolean  modifySectionInDB(JSON):boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that modifies section for a course inside database  **Input Parameter(s)**: JSON  **Return Type**: boolean  addCourseInDB(JSON):boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that adds course inside database  **Input Parameter(s)**: JSON  **Return Type**: boolean  dropCourseInDB(JSON):boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that removes course from database  **Input Parameter(s)**: JSON  **Return Type**: boolean  modifyCourseInDB(JSON):boolean  **Implemented in Class**: Scheduler Admin  **Description**: Method that modifies the course inside database  **Input Parameter(s)**: JSON  **Return Type**: boolean |

**4.1.5 - Admin and Student Verification**

This subsystem has one functionality which is to verify which type of user is accessing the system(student/admin). The *UI* class represents the bridge between the systems and will direct the user to their corresponding Scheduler system depending on their type(student/admin).

|  |  |
| --- | --- |
| **Classes Involved** | **Student, User, UI** |
| **Method(s) Implemented** | login(Student): void  **Implemented in Class**: User  **Description**: Method that allows user to login. For security purposes, hashing will be used in order to avoid potential breaches and unwanted access.  **Input Parameter(s)**: takes User  **Return Type**: void  signUp(JSON): void  **Implemented in Class**: Student  **Description**: Method that signs up the student  **Input Parameter(s)**: takes JSON  **Return Type**: void  verificationUserType(User): void  **Implemented in Class**: UI  **Description**: Method that checks the user type and redirects to their respective scheduler according to their type  **Input Parameter(s)**: takes User  **Return Type**: void |

**4.1.6 - Admin and Student Managing Account Information**

In this subsystem, both Scheduler systems(admin/student) contain methods that allow for the user to change its account information: *changeAccountInformation()*, *resetUserPassword()*. This subsystem simply permits the user to change his/her information and characteristics.

Admin and Student Managing Account Information is part of the component Account Management.

|  |  |
| --- | --- |
| **Classes Involved** | **Student, User, Admin, Scheduler Student, Scheduler Admin** |
| **Method(s) Implemented** | setPassword(String, String): void  **Implemented in Class**: User  **Description**: Method that allows user to change their password  **Input Parameter(s)**: takes two Strings, first string is the old password and second string the new password  **Return Type**: void  setEmail(String): void  **Implemented in Class**: User  **Description**: Method that allows to change email address for the user  **Input Parameter(s)**: takes String  **Return Type**: void  sendEmailForgotPass(**User**): void  **Implemented in Class**: Student  **Description**: Method that sends the password to the Student  **Input Parameter(s)**: takes User  **Return Type**: void  changeAccountInformation(User): void  **Implemented in Class**: Scheduler Admin  **Description**: Method that changes the account information of a user  **Input Parameter(s)**: takes User  **Return Type**: void  changeAccountInformation(User): void  **Implemented in Class**: Scheduler Student  **Description**: Method that changes the account information of a user  **Input Parameter(s)**: takes User  **Return Type**: void |

**4.2 Unit Descriptions**

|  |  |
| --- | --- |
| **Class Name** | **Admin** |
| **Description** | A type of user inherited from the User class representing the Admin. |
| **Attribute(s)** | N/A |
| **Operation(s)** | N/A |

|  |  |
| --- | --- |
| **Class Name** | **Student** |
| **Description** | A type of user inherited from the User class representing the Student. |
| **Attribute(s)** | * neededCourses : Course[] * preferences : Preferences * takenCourses : Courses |
| **Operation(s)** | Ø signUp(JSON\_File : JSON): Boolean  o Registers the student into the system for the first time.  Ø Student(): void  o Creates a preset Student object.  Ø sendEmailForgotPasswordt(UserObj : User): void  o Sends an email to the student explaining how to log back on if they forgot their password. |
|  |  |

|  |  |
| --- | --- |
| **Class Name** | **User** |
| **Description** | The generic model of the user of the system from which Student and Admin inherent from. |
| **Attribute(s)** | * username : String * password: String * email : String |
| **Operation(s)** | Ø User(username : String, password : String, email : String): void  o Generates a new account for a new user.  Ø getName(): String  o Returns the user’s username.  Ø getPassword(): String  o Returns the user’s password.  Ø getEmail(): String  o Returns the user’s email.  Ø setPassword(oldpassword : String, newpassword : String): void  o Allows the user to the change their password  Ø setEmail(newEmail : String): void  o Changes the user’s email.  Ø login(StudentObj: Student): void  o Provides the student user access to the system.  Ø ConfirmEmail(): Boolean  o Sends an automated confirmation email to the user confirming that a new account has been successfully. |

|  |  |
| --- | --- |
| **Class Name** | **UI** |
| **Description** | Provides the front end user interface (GUI) to the user. |
| **Attribute(s)** | N/A |
| **Operation(s)** | Ø VerificationUserType(UserObj : User): void  o Determines whether a user is a Student or Admin and only gives them access to parts of the system that they are allowed to interact with. |

|  |  |
| --- | --- |
| **Class Name** | **Preferences** |
| **Description** | Manages and stores the preferences the Student User selects if any. Used by Scheduler Student. |
| **Attribute(s)** | * load : int * offDay : String * prefDay : String * timeofDay : String |
| **Operation(s)** | Ø getLoad(): int  o Returns the number of courses a student wishes to take per semester.  Ø getOffDay(): String  o Returns the selection of days the Student selected to try and have off in a semester.  Ø getPrefDay(): String  o Returns the preferred day(s) that the Student wants to have classes on.  Ø getTimeOfDay(): String  o Displays the time of the day the Student wishes to not have classes at.  Ø setLoad(NumOfCourses : int): void  o Student sets the number of courses they wish to take per semester.  Ø setOffDay(DayOff : String): void  o Student sets the days they wish to have no classes..  Ø setPrefDay(PrefDay : String): void  o Student sets the days they wish to have classes on.  Ø setTimeOfDay(TimeOfDay : String): void  o Sets the time of day Student would like to have classes at.  Ø Preferences(): void  o Generates the preferences of the Student User. |

|  |  |
| --- | --- |
| **Class Name** | **Course** |
| **Description** | The object model of an academic course inside the system from which Section is derived. Used by Scheduler Student to construct a Schedule for the Student User. |
| **Attribute(s)** | * name : String * id : String * description : String * lectures : Section [ ] * credits : double * tutorial : Section [ ] * labs : Section [ ] * prereqs : Course [ ] |
| **Operation(s)** | Ø getName(): String  o Retrieves and displays the name of Course for Admin.  Ø getDescription(): String  o Retrieves and displays the description of Course for the Admin User.  Ø getLectures(): Section [ ]  o Retrieves and displays the Section(s) of Course that contain lectures.  Ø getCredits(): double  o Retrieves the number of credits for the Course.  Ø getTutorial(): Section [ ]  o Retrieves and displays the Section(s) of Course that contain lectures.  Ø getLab(): Section [ ]  o Retrieves and displays the Section(s) of Course that contain labs.  Ø getPrereqs(): Course [ ]  o Retrieves and displays the name of Course for Admin. |

|  |  |
| --- | --- |
| **Class Name** | **Section** |
| **Description** | Contains the information of a particular section for a given Course which can be accessed and manipulated by an Admin User. |
| **Attribute(s)** | * time : String * id : String * classroom : String * semester : String * type : String |
| **Operation(s)** | Ø getTime(): String  o Admin retrieves and displays the time the course is offered during a week.  Ø getId(): String  o Returns the ID of the Course to the Admin.  Ø getClassroom(): String  o Retrieves and Displays the Classroom number to the Admin.  Ø getSemester(): String  o Returns and displays the semester the Section is being offered to the Admin.  Ø getTypye(): String  o Returns and displays the type of the Section (Lab, Tutorial or Lecture).  Ø setTime(time : String): void  o Admin can set the time of a section during the week.  Ø setID(ID : String): void  o Admin sets the ID of the Section.  Ø setClassroom(classroom : String): void  o Admin sets the classroom number of the Section.  Ø setSemester(semester : String): void  o Admin sets the semester the Section is offered in.  Ø setType(type : String): void  o Admin sets the type of Section (lab, tutorial, lecture) for a particular Course.  Ø Section(time : String, id: String, classroom : String, semester : String, type : String): void  o Generates a new Section for a Course for the Admin. |