4. Detailed Design

4.1 Detailed Design Diagram

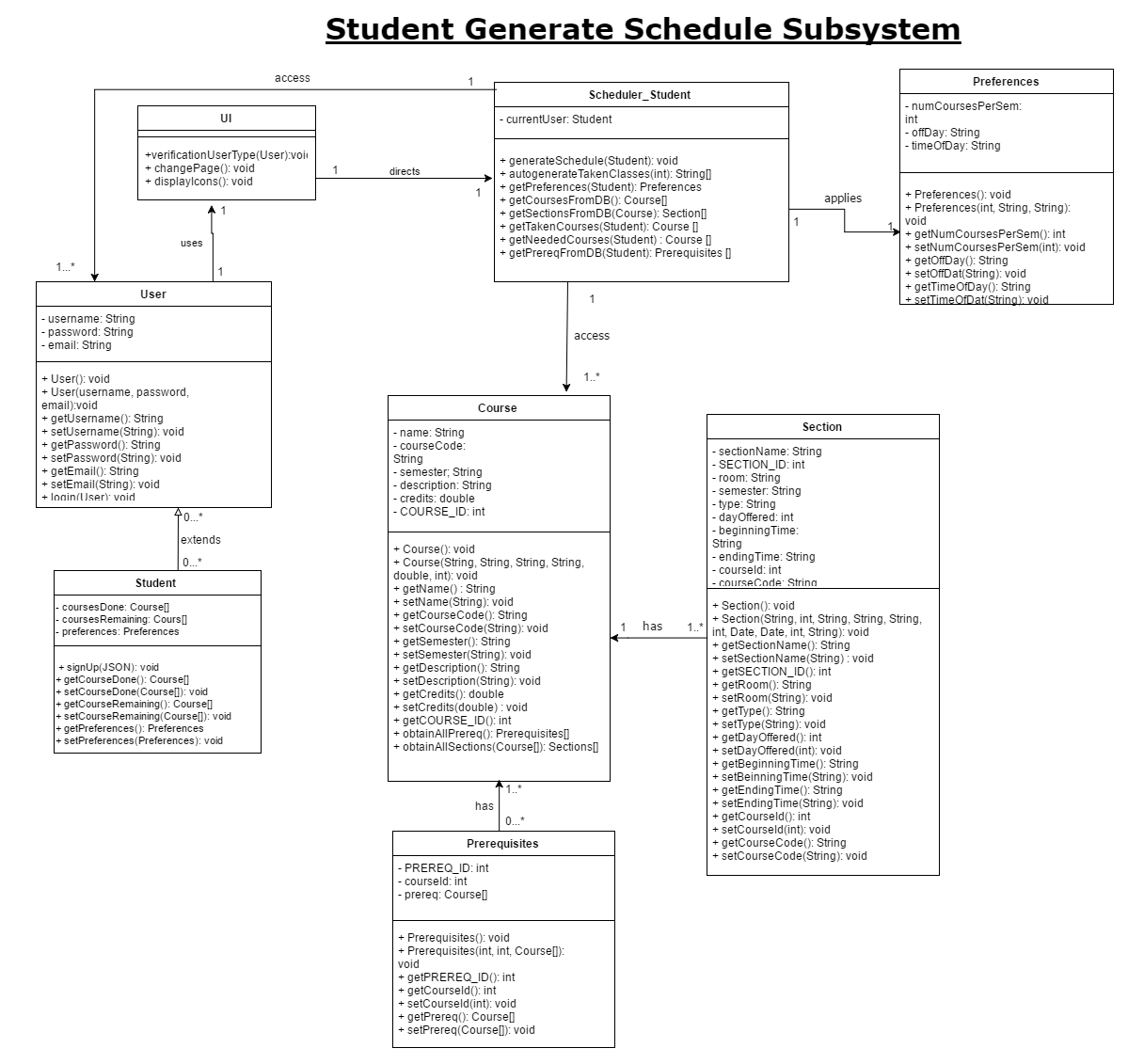
The following section describes a detailed description of the concerning the class diagram of the system. The scheduler contains a total of 10 classes. The classes *Student, Admin* and *User* represent essentially the user. *Student* and *Admin* are subclasses of the *User* class. This corresponds to the possibility of a student and an admin to log in to the system. The *User* class interacts with the *UI* class where this class simply manages the information to be displayed. For the classes *Scheduler\_Admin* and *Scheduler\_Student*, depending on the type of user, the *UI* will redirect that user 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 & Prerequisites.*

**4.1.1 Student Generate Schedule Subsystem**

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 subsystem contains *Preferences, Course, Section, Prerequisites* and *Scheduler\_Student* as classes. Each of these classes communicate between each other to generate schedule. *Preferences, Courses, Section* and *Prerequisites* classes are only composed of getters and setters for their attributes. For more details, refer section 4.2.

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

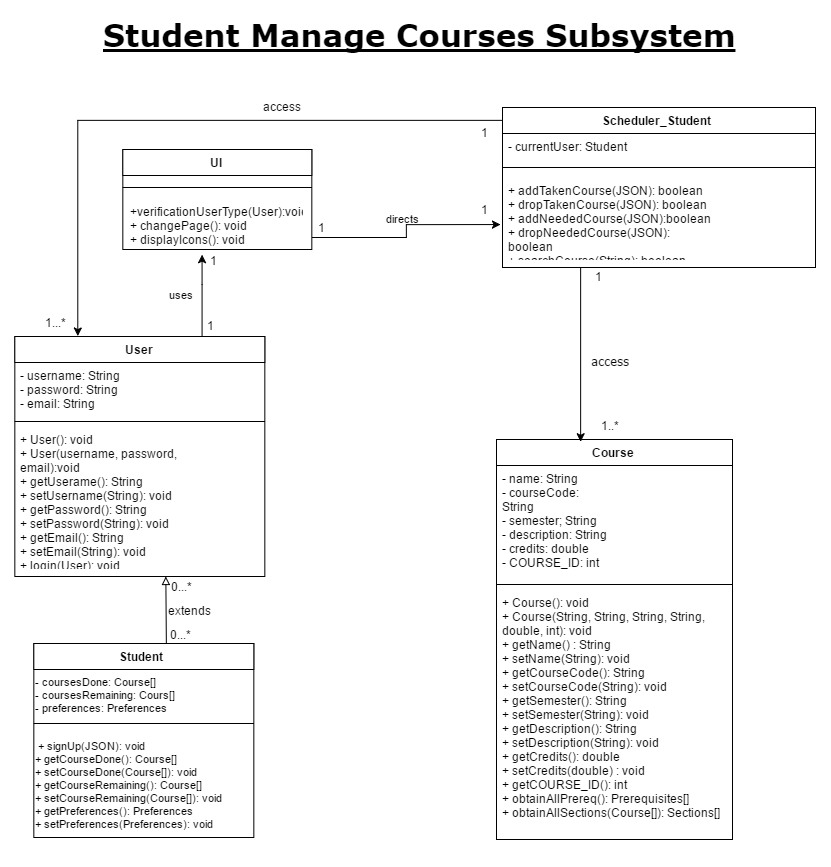
|  |  |
| --- | --- |
| **Classes Involved** | **Preferences, Course, Section, Prerequisites, Scheduler\_Student** |
| **Method(s) Implemented** | generateSchedule(Student): void  **Implemented in Class**: Scheduler\_Student  **Description**: Method that generates schedule for Student  **Input Parameter(s)**: Student  **Return Type**: void  getTakenCourses(Student): Course []  **Implemented in Class**: Scheduler\_Student  **Description**: Method that retrieves taken courses from database  **Input Parameter(s)**: Student  **Return Type**: void  getNeededCourses(Student): Course []  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns needed courses  **Input Parameter(s)**: Student  **Return Type**: Course []  getPreferences(Student): Preferences  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns preferences of a user  **Input Parameter(s)**: Student  **Return Type**: Preferences  getPrereqFromDB(Student): Prerequisites[]  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns the prerequisites of all classes  **Input Parameter(s)**: Student  **Return Type**: Prerequisites[]  autogenerateTakenClasses(int): String []  **Implemented in Class**: Scheduler\_Student  **Description**: Method that autogenerates taken classes  **Input Parameter(s)**: int  **Return Type**: String []  getCoursesFromDB(): Course[]  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns all courses from the sequence of courses from the database  **Input Parameter(s)**: -  **Return Type**: Course[]  getSectionsFromDB(Course): Section[]  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns all sections of a specific course  **Input Parameter(s)**: Course  **Return Type**: Section [] |



**4.1.2 Student Manage Courses Subsystem**

This subsystem portrays the path for a student to manage his course load. First, the course is searched with *searchCourse(String)* method. The scheduler possesses methods such as *addNeededCourse(JSON), dropTakenCourse(JSON)*, which allows for the user to simply add courses to his/her schedule or simply drop them. The methods *dropTakenCourse(JSON)* and *addTakenCourse(JSON)* 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.

|  |  |
| --- | --- |
| **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)**: 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)**: 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  searchCourse(String):boolean  **Implemented in Class**: Scheduler\_Student  **Description**: Searches for a course through the database and returns true if course exists, false otherwise.  **Input Parameter(s)**: String  **Return Type**: boolean |

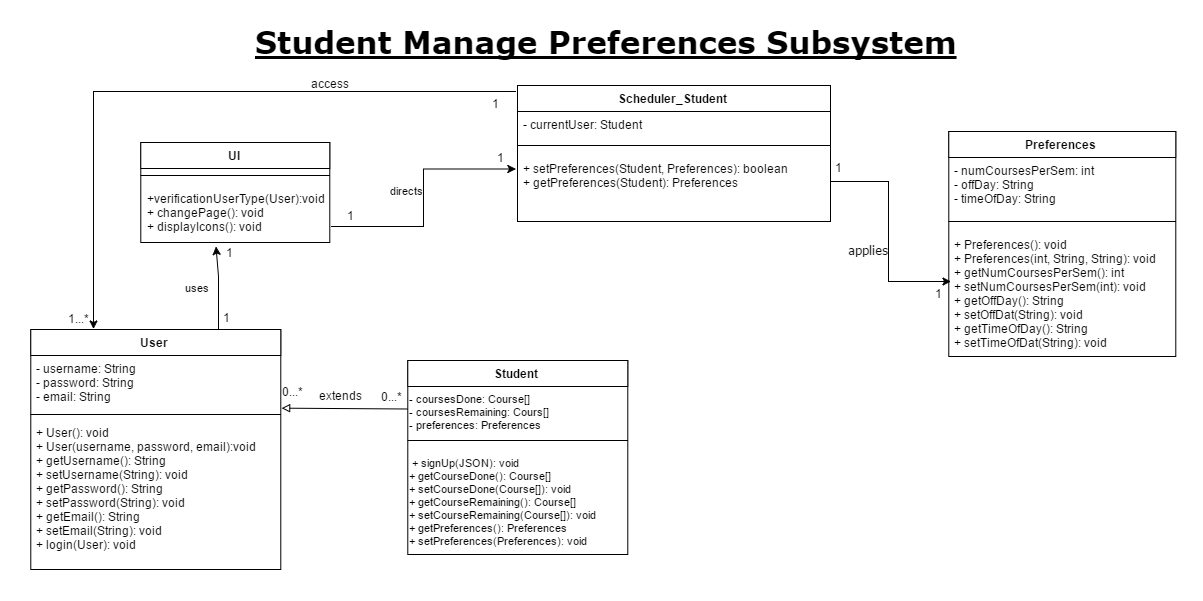


**4.1.3 - Student Manage Preferences Subsystem**

For this subsystem, the scheduler contains two methods called *getPreferences(Student)* and *setPreferences(Student, Preferences)* inside class *Scheduler\_Student* where the user who is a student can set and get the preferences. The *Preference* class contains attributes such as *timeOfDay, offDay* and *numCoursesPerSem*.

Class *Preferences* is mainly composed of getters and setters for its 3 attributes. For more details, please refer section 4.2.

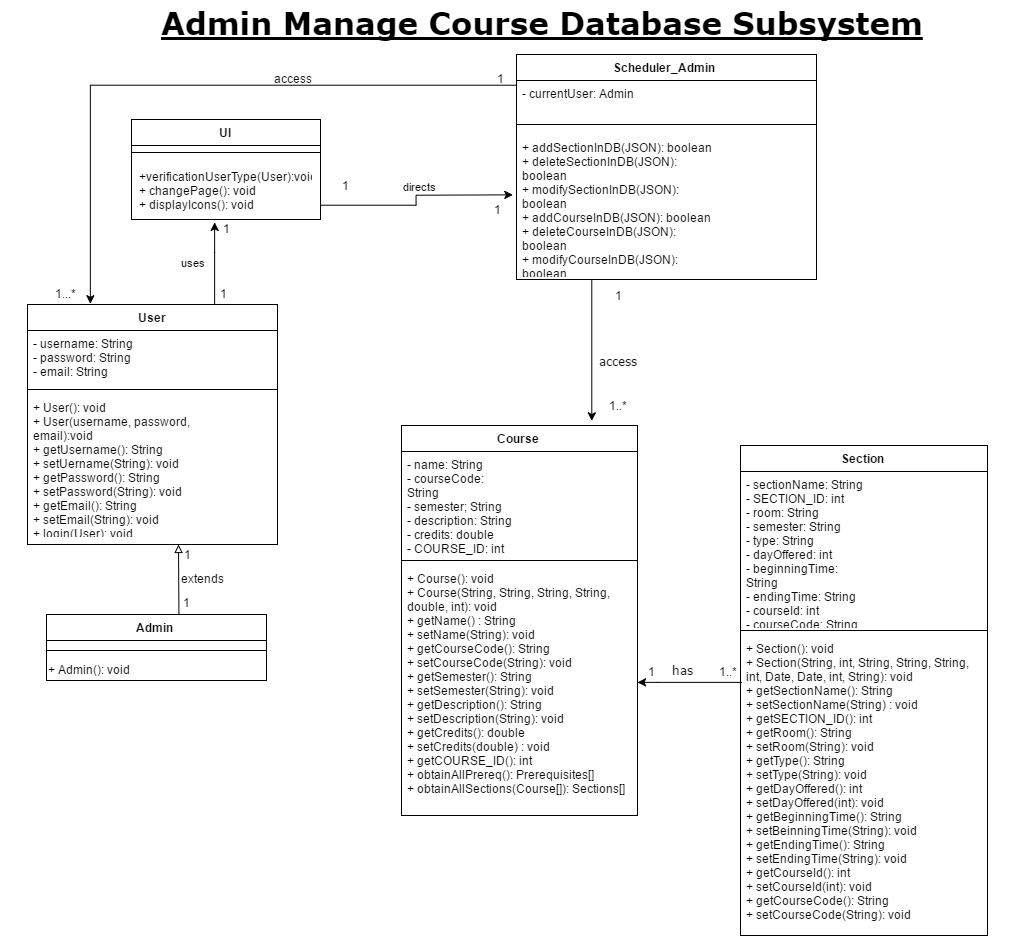
|  |  |
| --- | --- |
| **Classes Involved** | **Scheduler\_Student, Preferences** |
| **Method(s) Implemented** | setPreferences(Student,Preferences):boolean  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns a boolean for preferences modification  **Input Parameter(s)**: Student and Preferences  **Return Type**: boolean  getPreferences(Student):Preferences  **Implemented in Class**: Scheduler\_Student  **Description**: Method that returns preferences of a user  **Input Parameter(s)**: Student  **Return Type**: Preferences |



**4.1.4 - Admin Manage Course Database Subsystem**

This subsystem portrays the relationship between the privileges 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()*. *Course* and *Section* classes are only composed of getters and setters. For more details, please refer to section 4.2.

|  |  |
| --- | --- |
| **Classes Involved** | **Scheduler\_Admin, Course, Section,** |
| **Method(s) Implemented** | addSectionInDB(JSON): boolean  **Implemented in Class**: Scheduler\_Admin  **Description**: Method that adds a section in the database  **Input Parameter(s)**: JSON  **Return Type**: boolean  dropSectionInDB(JSON): boolean  **Implemented in Class**: Scheduler\_Admin  **Description**: Method that removes a section from the database  **Input Parameter(s)**: 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 |

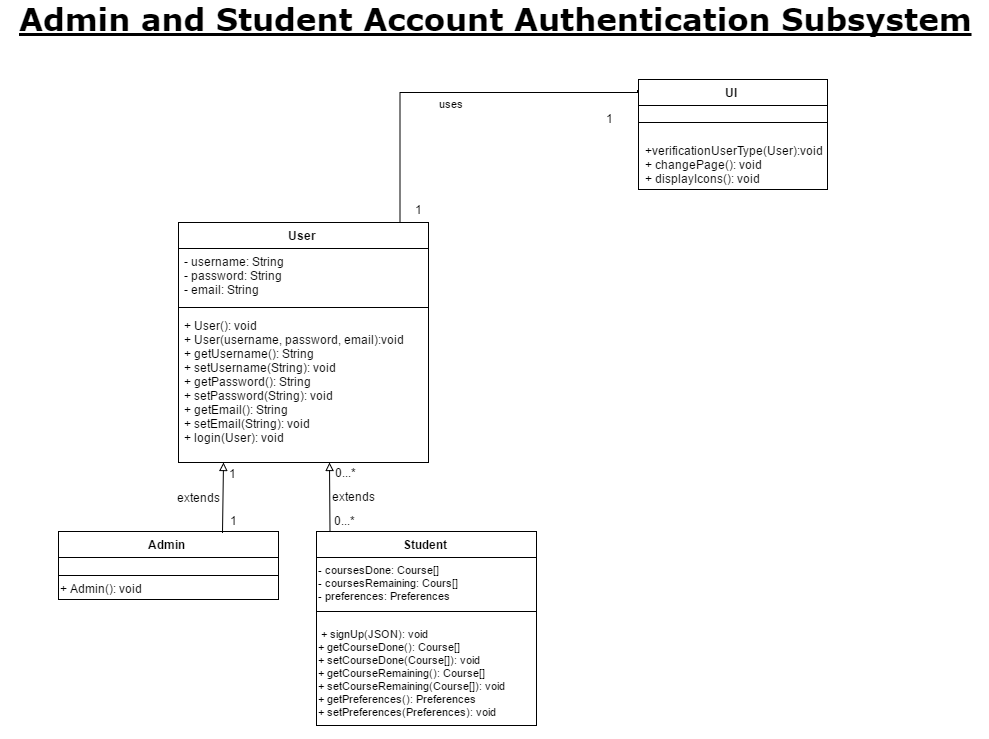


**4.1.5 - Admin and Student Account Authentication Subsystem**

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). It is important to note both *admin* and *student* can login, but only a *student* can sign up, as no one can register as an *admin* and have access to the admin page.

*User* and *Student* are both composed of attributes, and each class has getters and setters for those attributes. For more details, please refer section 4.2.

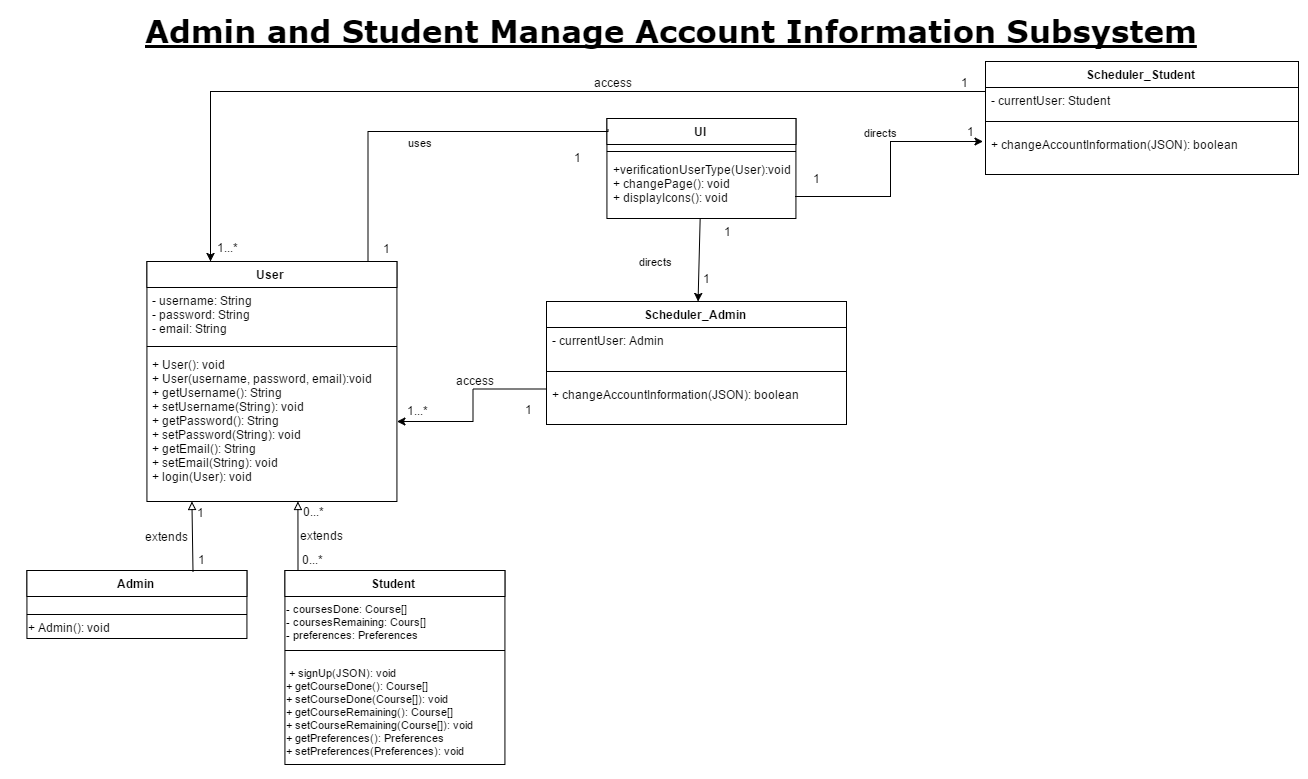
|  |  |
| --- | --- |
| **Classes Involved** | **Student, Admin, User, UI** |
| **Method(s) Implemented** | login(User): void  **Implemented in Class**: User  **Description**: Method that allows user to login. For security purposes, hashing will be used in order to avoid potential breaches and unwanted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



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

In this subsystem, both Scheduler systems(admin/student) contain a method that allows for the user to change its account information: *changeAccountInformation()*. This subsystem simply permits the user to change his/her information and characteristics inside the *User* and *Student* classes, such as *username, password, email* for both *Student* and *Admin*, but also *courseDone, coursesRemaining* and *preferences* inside of *Student*. For more details regarding the setters and getters of this attributes, please refer to section 4.2.

|  |  |
| --- | --- |
| **Classes Involved** | **Student, User, Admin, Scheduler\_Student, Scheduler\_Admin** |
| **Method(s) Implemented** | changeAccountInformation(JSON): boolean  **Implemented in Class**: Scheduler\_Admin  **Description**: Method that changes the account information of the admin. It returns true if the changes are successful, false otherwise.  **Input Parameter(s)**: JSON  **Return Type**: boolean  changeAccountInformation(JSON): boolean  **Implemented in Class**: Scheduler\_Student  **Description**: Method that changes the account information of a student. It returns true if the changes are successful, false otherwise.  **Input Parameter(s)**: JSON  **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)** | Ø Admin(): void  o Creates a preset Admin object. |

|  |  |
| --- | --- |
| **Class Name** | **Student** |
| **Description** | A type of user inherited from the User class representing the Student. |
| **Attribute(s)** | * coursesDone : Course[] * coursesRemaining : Course[] * preferences : Preferences |
| **Operation(s)** | Ø Student(): void  o Creates a preset Student object.  Ø signUp(JSON\_File : JSON): Boolean  o Registers the student into the system for the first time.  Ø getCourseDone(): Course[]  o Return an array of already taken/done courses for the student  Ø setCourseDone(coursesDone: Course[]): void  o Changes the array of already taken/done courses for the student  Ø getCourseRemaining(): Course[]  o Return an array of remaining/needed courses for the student  Ø setCourseRemaining(coursesRemaining: Course[]): void  o Changes the array of remaining/needed courses for the student  Ø getPreferences(): Preferences  o Returns student's preferences  Ø setPreferences(preferences: Preferences): void  o Changes the preferences of the student |

|  |  |
| --- | --- |
| **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 User (): void  o Creates a preset User object.  Ø getName(): String  o Returns the user’s username.  Ø setName(newUsername): String  o Allows the users to change their username  Ø getPassword(): String  o Returns the user’s password.  Ø setPassword(oldpassword : String, newpassword : String): void  o Allows the users to the change their password  Ø getEmail(): String  o Returns the user’s email.  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. |

|  |  |
| --- | --- |
| **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.  Ø changePage(): void  o   Changes the page displayed to User based on User’s interaction with the system.  Ø displayIcons(): void  o   Displays different icons depending on page being displayed to User such as buttons and checkboxes. |

|  |  |
| --- | --- |
| **Class Name** | **Preferences** |
| **Description** | Manages and stores the preferences the Student User selects if any. Used by Scheduler\_Student. |
| **Attribute(s)** | * numCoursesPerSem : int * offDay : String * timeofDay : String |
| **Operation(s)** | Ø Preferences(): void  o Initializes preferences for a Student.  Ø Preferences(): void  o Generates the preferences of the Student User.  Ø getNumCoursesPerSem(): int  o Returns the number of courses a student wishes to take per semester.  Ø setNumCoursesPerSem(numCourses : int): void  o Student sets the number of courses they wish 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 setOffDay(dayOff : String): void  o Student sets the days they wish to have no classes.  Ø getTimeOfDay(): String  o Displays the time of the day the Student wishes to not have classes at.  Ø setTimeOfDay(timeOfDay : String): void  o Sets the time of day Student would like to have classes at. |

|  |  |
| --- | --- |
| **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 * courseCode: String * semester : String * Description: String * credits: double * COURSE\_ID: int |
| **Operation(s)** | Ø Course(): void  o Creates an object of type Course and initializes all attributes.  Ø Course(name: String, code: String, semester: String, desc: String, credits: int, id: int): void  o Creates and object of type Course and sets all attributes.  Ø getName(): String  o Returns name of the course  Ø setName(newName: String): void  o Changes the name of the course  Ø getCourseCode(): String  o Returns name of the course code  Ø setCourseCode(code: String): void  o Changes the name of the course code  Ø getDescription(): String  o Returns the description of a course.  Ø setDescription(desc: String): void  o Changes the description of the course  Ø getCredits(): double  o Returns the number of credits for the Course.  Ø setCredits(credit: double): void  o Changes the number of credits for the Course.  Ø getCOURSE\_ID (): int  o Returns a final and unique id for a course, a number between 1 and ~140). This attribute cannot be modified.  Ø obtainAllPrereq (): Prerequisites[]  o Returns an array of all prerequisites and corequisites from the DB.  Ø obtainAllSections (neededCourses: Course[]): Section[]  o Returns an array of all sections from the needed classes. |

|  |  |
| --- | --- |
| **Class Name** | **Section** |
| **Description** | Contains the information of a particular section for a given Course which can be accessed and manipulated by an Admin, and also utilized to generate the schedule. |
| **Attribute(s)** | * sectionName: String * SECTION\_ID: int * room: String * semester: String * type: int * dayOffered: int * beginningTime: String * endingTime: String * courseId: int * courseCode: String |
| **Operation(s)** | Ø Section(): void  o Creates an object of type Section and initializes all attributes.  Ø Section(name: String, id: int, room: String, semester: String, type: int, days: int, beginning: String, end: String, courseID: int, courseCode: String): void  o Creates an object of type Section and sets all attributes  Ø getSectionName(): String  o Returns the section name .  Ø setSectionName(name: String): void  o Changes the name of a section.  Ø getSECTION\_ID(): String  o Returns a final and unique id for a section, a number between 1 and ~400). This attribute cannot be modified.  Ø getRoom(): String  o Returns the classroom number.  Ø setRoom(room: String): void  o Changes the classroom/location of a section.  Ø getSemester(): String  o Returns and displays the semester the Section is being offered. The possibilities are Fall, Winter, Summer.  Ø setSemester(sem:String): void  o Changes the semester that a section is offered at. The possibilities are Fall, Winter, Summer.  Ø getType(): String  o Returns and displays the type of the Section (Lab, Tutorial or Lecture).  Ø setType(newType: String): void  o Changes the type of the Section (Lab, Tutorial or Lecture).  Ø getDayOffered(): int  o Returns the days at which the section is offered. 1 represents Monday, 2 represents Tuesday, 24 represents Tuesday-Thursday, and so on.  Ø setDayOffered(day: int): void  o Changes the day(s) at which the section is offered.  Ø getBeginningTime(): String  o Returns the time at which a section begins, in the hh:mm:ss format.  Ø setBeginningTime(time: String): void  o Changes the beginning time of a section.  Ø getEndingTime(): String  o Returns the time at which a section ends, in the hh:mm:ss format.  Ø setEndingTime(time: String): void  o Changes the ending time of a section.  Ø getCourseId(): int  o Returns the ID of the course.  Ø setCourseId(id: int): void  o Changes the id of a course.  Ø getCourseCode(): String  o Returns the coruse code of a course.  Ø setCourseCode(code: String): void  o Changes the course code of a section. |

|  |  |
| --- | --- |
| **Class Name** | **Prerequisites** |
| **Description** | Contains all prerequisites and corequisites of a specific course. This is only used in generating a schedule. |
| **Attribute(s)** | * PREREQ\_ID: int * courseId: int * prereq: Course[] |
| **Operation(s)** | Ø Prerequisites(): void  o Creates an object of type prerequisites and initializes all attributes.  Ø Prerequisites(prereqId: int, courseId: int, prereq: Course[]): void  o Creates an object of type Prerequisites and sets all attributes  Ø getPREREQ\_ID(): String  o Returns a final and unique id for one set of prerequisites for one course, a number between 1 and ~140). This attribute cannot be modified.  Ø getCourseId(): int  o Returns the id number of the course.  Ø setCourseId(id: int): void  o Changes the id number of the course.  Ø getPrereq(): Course[]  o Returns and displays the semester the Section is being offered. The possibilities are Fall, Winter, Summer.  Ø setPrereq(prereqs: Course[]): void  o Changes the set of courses that a prerequesites/corequisites for a specific course. |

|  |  |
| --- | --- |
| **Class Name** | **Scheduler\_Admin** |
| **Description** | The object model that allows only an Admin User to modify all aspects of the database. |
| **Attribute(s)** | * currentUser: Admin |
| **Operation(s)** | Ø  changeAccountInformation(UserObj : User): boolean  o Allows an Admin User to change their account information.  Ø  addSectionInDB(JSON\_File : JSON): boolean  o Adds a new Section object to the database.  Ø  dropSectionInDB(JSON\_File : JSON): boolean  o Removes a Section object from the database.  Ø  modifySectionInDB(JSON\_File : JSON): boolean  o Admin modifies the information of a Section object inside the database.  Ø  addCourseInDB(JSON\_File : JSON): boolean  o Adds a new Course object to the database.  Ø  dropCourseInDB(JSON\_File : JSON): boolean  o Removes a Course object from the database.  Ø  modifyCourseInDB(JSON\_File : JSON): boolean  o Admin modifies the information of a Course object inside the database.  Ø SchedulerAdmin(): void  o Generates a new Scheduler\_Admin object which allows an Admin User to modify the database. |

|  |  |
| --- | --- |
| **Class Name** | **Scheduler\_Student** |
| **Description** | The object model that creates a schedule for the Student based on information taken from the Preferences, Course, Section and Student classes. |
| **Attribute(s)** | * currentUser: Student |
| **Operation(s)** | Ø  addTakenCourse(JSON\_File : JSON): boolean  o Updates database JSON file with record of Courses taken by Student.  Ø addNeededCourse(JSON\_File :JSON): boolean  o Updates database JSON file with record of Courses that Student User needs to take.  Ø dropTakenCourse(JSON\_File : JSON): boolean  o Removes a Course that was taken by Student from database JSON file.  Ø  dropNeededCourse(JSON\_File : JSON): boolean  o Removes a Course that Student needs to take from JSON file.  Ø  getTakenCourses(UserObj : Student): Course [ ]  o Returns an array of Course that the Student has taken.  Ø  getNeededCourses(UserObj : Student): Course [ ]  o Returns and displays an array of Course that Student needs to take.  Ø changeAccountInformation(JSON\_File : JSON): boolean  o Student User may change certain information on their account.  Ø generateSchedule(UserObj : Student): void  o Generates and displays a new schedule to the Student based on information from Preferences, database, Course and Section.  Ø autoGenerateTakenClasses(numj : int): String [ ]  o Generates and displays to Student classes that have already been taken.  Ø getCoursesFromDB(): Course [ ]  o Retrieves all Courses from the database.  Ø getSectionsFromDB(CourseObj : Course): Section [ ]  o Retrieves all Course Section(s) from the database.  Ø getPreferences(UserObj : Student): Preferences  o Retrieves all of Student’s selected preferences.  Ø setPreferences(UserObj : Student): boolean  o Student can set their preferences and are told whether action was successful or not.  Ø getPrereqFromDB(UserObj : Student): Prerequisites []  o Returns all prerequesites of all courses  Ø SchedulerStudent(): void  o Default constructor.  Ø searchCourse(courseCode: String): boolean  o Returns true if course exist in database, false otherwise. |