Section 2.2 Development view

The development view describes our scheduling system from our programmer's perspective and the management of our software. It is illustrated by the component diagram which consists of components and links that show the dependencies between them.

<insert component diagram>

<https://drive.google.com/file/d/0B40wpYofiC-ZT2otdlk4MFA1M28/view?usp=sharing>

The main components of our system consists of the model, view and controller. Since these 3 components are not taken by the same framework, we have specified on the component diagram that the view is handled using React and that data manipulation and database queries are handled by Laravel. Most importantly, in our model, we have defined our subsystems to be account management, courses management, preferences management, schedule generation and database management. The different subsystems interact with each other through interfaces, with the open end of the link being a required interface and the lollipop being the provided interface. Also, the model is connected to a MySQL database.

In our component diagram the two frameworks, Laravel for server side model and controller, and react for the client side view, are described. The components of the model view controller system interact as following: The model requires connection to the SQL database through Database Connection port and control instructions provided by the controller. The view requires object oriented and control information in order to present the user with the system on a browser. The controller requires input from the user through the view.

Within the model, the database management subsystem can provide the user credentials in order to manage the account subsystem and allow a confirmed admin to modify the database. The account subsystem provides preferences and course management subsystems for students. Each of these subsystems (course management and preferences) each provide their own criteria to a schedule generator that will use the inputs in order to provide a schedule.

Section 3.2 Subsystem Interfaces Specifications

The subsystem interfaces consist of: ManageCourses, PreferenceSettings, UserVerification, CourseAvailabilty, PreferenceImplementation, ScheduleGeneration. Each of the interface connect two subsystems together. In this section, each of these interfaces will be described and the function calls exchanged between its subsystems will be given along with the specific description of the parameters passed.

3.2.1

ManageCourses

The manage courses interface joins the provided interface of account management with the required interface of the subsystem courses management.

|  |  |
| --- | --- |
| **Classes involved** | SchedulerStudent, Course |
| **List of Methods** | 1. addTakenCourses(JSON): boolean   **Implemented in Class**: Scheduler Student  **Description**: Method to add courses to the list of taken courses.  **Input Parameter(s)**: serialized Course -course to be added  **Return Type**: boolean   1. getTakenCourses(User): Course[]   **Implemented in Class**: Scheduler Student  **Description**: Method to get the list of courses taken.  **Input Parameter(s)**: User of type Student -user making inquiry  **Return Type**: an array of type Course   1. addNeededCourses(JSON): boolean   **Implemented in Class**: Scheduler Student  **Description**: Method to add courses to the list of needed courses.  **Input Parameter(s)**: serialized Course -courses to be aded  **Return Type**: boolean   1. getNeededCourses(User): Course[]   **Implemented in Class**: Scheduler Student  **Description**: Method to get the list of courses needed.  **Input Parameter(s)**: User type Student -user making inquiry  **Return Type**:  an array of type Course   1. autogenerateTakenClasses(int): String[]   **Implemented in Class**: Scheduler Student  **Description**: Method to generate a list of taken courses.  **Input Parameter(s)**: int -number of classes t  **Return Type**:  an array of Strings -names of courses   1. getCoursesFromDB(int): void   **Implemented in Class**: Scheduler Student  **Description**: Method to access courses in database.  **Input Parameter(s)**: none  **Return Type**:  N/A |

3.2.2

PreferencesSettings

The preference settings interface joins the provided interface of account management with the required interface of the preferences management.

|  |  |
| --- | --- |
| **Classes involved** | User Student, Preferences, Scheduler Student |
| **List of Methods** | 1. setPreferences(User, Preferences): boolean   **Implemented in Class**: Scheduler Student  **Description**: Method to set the preferences for a user of type Student  **Input Parameter(s)**: object of type Preferences and User of type Student -current user and their new preferences  **Return Type**: boolean   1. getPreferences(User): Preferences   **Implemented in Class**: Scheduler Student  **Description**: Method to get the preferences  **Input Parameter(s)**: User of type Student -current user and their preferences  **Return Type**: An object of type Preferences |

3.2.3

UserVerification

The UserVerification interface is provided by the user account in order to manage the database. The credentials entered in the UI are compared to the User’s in order to establish a valid login type.

|  |  |
| --- | --- |
| **Classes involved** | User |
| **List of Methods** | 1. getName() : String   **Implemented in Class**: User  **Description**: Method to access the name of a user.  **Input Parameter(s)**: N/A  **Return Type**: String   1. getPassword() : String   **Implemented in Class**: User  **Description**: Method to access the password of a user.  **Input Parameter(s)**: N/A  **Return Type**: String   1. getEmail() : String   **Implemented in Class**: User  **Description**: Method to access the email of a user.  **Input Parameter(s)**: N/A  **Return Type**: String   1. login(Student) : void   **Implemented in Class**: User  **Description**: Method to access the student object associated with a user account.  **Input Parameter(s)**: object of type Student  **Return Type**: void |

Verification of user login type is used to ensure proper access to the database.

|  |  |
| --- | --- |
| **Classes involved** | UI |
| **List of Methods** | 1. verificationUserType(User) : void   **Implemented in Class**: UI  **Description**: Method to access the user’s dynamic type Student/Admin.  **Input Parameter(s)**: object of type User -user to be verified  **Return Type**: void |

Enables an Admin to modify any User’s information, allow Student to modify its own information.

|  |  |
| --- | --- |
| **Classes involved** | Scheduler Student, Scheduler Student |
| **List of Methods** | 1. changeAccountInformation(User) : boolean   **Implemented in Class**: Scheduler Admin/Student  **Description**: Method to modify a user’s account information.  **Input Parameter(s)**: object of type User -user who is verified as either Admin or Student.  **Return Type**: boolean |

3.2.4

AdminDataBaseModifications

This interface allows an administrator to edit the courses and sections list in the database. This is not accessible for a student because of the UserValidation interface.

|  |  |
| --- | --- |
| **Classes involved** | Scheduler Admin, User Admin |
| **List of Methods** | 1. getCoursesFromDB(): void   **Implemented in Class**: Scheduler Admin  **Description**: Method to access all Courses in database.  **Input Parameter(s)**: N/A  **Return Type**: void   1. addCourseInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to add a Course in database.  **Input Parameter(s)**: serialized Course -to be added  **Return Type**: boolean   1. dropCourseInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to remove a Course from database.  **Input Parameter(s)**: serialized Course -to be removed  **Return Type**: boolean   1. modifyCourseInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to modify an existing Course in database.  **Input Parameter(s)**: serialized Course  -to be modified  **Return Type**: boolean   1. modifySectionInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to modify an existing section of a course in database.  **Input Parameter(s)**: serialized Section -to be modified  **Return Type**: boolean   1. addSectionInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to add a Section for a Course in database.  **Input Parameter(s)**: serialized Section -to be added  **Return Type**: boolean   1. dropSectionInDB(JSON) : boolean   **Implemented in Class**: Scheduler Admin  **Description**: Method to remove a section for a course.  **Input Parameter(s)**: serialized Section -to be removed  **Return Type**: boolean |

3.2.5

CourseAvailability

The Course availability Interface generates, from a list of Courses that a student may take, a list of course Sections that could match together in a schedule. This interface provides viewing of courses list, not modification.

The Scheduler Student provides a list of courses needed and their sections. This list is solely based on which course could be taken for the schedule; semester and prerequisites are the only constraints.

|  |  |
| --- | --- |
| **Classes involved** | Scheduler Student, User Student, Course |
| **List of Methods** | 1. getTakenCourses(User) : Course[]   **Implemented in Class**: Scheduler Student  **Description**: Method to access the list of courses taken.  **Input Parameter(s)**: User of type Student -who has taken the courses  **Return Type**: array of objects of type Course   1. getNeededCourses(User) : Course[]   **Implemented in Class**: Scheduler Student  **Description**: Method to access the list of needed courses.  **Input Parameter(s)**: User of type Student -who needs the courses  **Return Type**: array of objects of type Course   1. getCoursesFromBD() : void   **Implemented in Class**: Scheduler Student  **Description**: Method to access the list of all courses.  **Input Parameter(s)**: none  **Return Type**: N/A   1. getSectionsForCourse(String) : Section[]   **Implemented in Class**: Scheduler Student  **Description**: Method to access the list of sections for a course.  **Input Parameter(s)**: String -name of course  **Return Type**: array of objects type Section |

The Course attributes are accessed  in order to create the schedule, more importantly they provide a list of Sections.

|  |  |
| --- | --- |
| **Classes involved** | Course, Section |
| **List of Methods** | 1. getName(): String   **Implemented in Class**: Course  **Description**: Method to access the name of a course.  **Input Parameter(s)**: N/A  **Return Type**: string   1. getNumber(): int   **Implemented in Class**: Course  **Description**: Method to access the number of a course.  **Input Parameter(s)**: N/A  **Return Type**: int   1. getPrereqs(): Courses[]   **Implemented in Class**: Course  **Description**: Method to access the list prerequisite courses of a course.  **Input Parameter(s)**: N/A  **Return Type**: array of objects of type Course -sections not initialized   1. getLectures(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list of lectures for a course.  **Input Parameter(s)**: N/A  **Return Type**: array of objects of type Course Section   1. getTutorial(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list of tutorials for a course.  **Input Parameter(s)**: N/A  **Return Type**: array of Course objects of type  Section   1. getLabs(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list labs of a course.  **Input Parameter(s)**: N/A  **Return Type**: array of Course objects of type  Section |

Sections are contained within Courses, stored in the arrays lecture, tutorial and Lab -from which they get their section type. Sections provide availability and more precise scheduling information.

|  |  |
| --- | --- |
| **Classes involved** | Section |
| **List of Methods** | 1. getSemester(): String   **Implemented in Class**: Section  **Description**: Method to access the semester in which the section is taught.  **Input Parameter(s)**: N/A  **Return Type**: string |

3.2.6

PreferenceImplementation

The preference implementation interface provides the student preferences in order to generate a schedule with the current student preferences.

|  |  |
| --- | --- |
| **Classes involved** | Scheduler Student, User |
| **List of Methods** | 1. getPreferences(User) : Preferences   **Implemented in Class**: Scheduler  **Description**: Method to access the preferences of a User of type Student.  **Input Parameter(s)**: User type Student -who will access their preferences  **Return Type**: object of type Preferences |

3.2.7

ScheduleGeneration

The schedule generation interface is a combination of the course availability and the preferences of a student. A schedule is generated from a list of sections for the courses to be taken that are offered during the semester and have times corresponding with preferences. The Schedulegeneration then chooses sections from the lists that fulfil the requirements to be added to a schedule until it is filled. This schedule can then be displayed with section information -times and classrooms. In order to modify the schedule, preferences or courses to be taken can be modified to generate a new schedule.

|  |  |
| --- | --- |
| **Classes involved** | Scheduler Student, Student |
| **List of Methods** | 1. generateSchedule(Student) : void   **Implemented in Class**: Scheduler  **Description**: Method to produce a schedule for a semester according to a list of courses and preferences.  **Input Parameter(s)**: object of type Student  **Return Type**: void. |

More Information is required in order to draw a schedule: the course names, number and section.

|  |  |
| --- | --- |
| **Classes involved** | Course, Section |
| **List of Methods** | 1. getName(): String   **Implemented in Class**: Course  **Description**: Method to access the name of a course.  **Input Parameter(s)**: N/A  **Return Type**: string   1. getNumber(): int   **Implemented in Class**: Course  **Description**: Method to access the number of a course.  **Input Parameter(s)**: N/A  **Return Type**: int   1. getCredits(): double   **Implemented in Class**: Course  **Description**: Method to access the number of credits of a course.  **Input Parameter(s)**: N/A  **Return Type**: double   1. getPrereqs(): Courses[]   **Implemented in Class**: Course  **Description**: Method to access the list prerequisite courses of a course.  **Input Parameter(s)**: N/A  **Return Type**: array of objects of type Course -sections not initialized   1. getLectures(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list of lectures for a course.  **Input Parameter(s)**: N/A  **Return Type**: array of objects of type Course Section   1. getTutorial(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list of tutorials for a course.  **Input Parameter(s)**: N/A  **Return Type**: array of Course objects of type  Section   1. getLabs(): Section[]   **Implemented in Class**: Course  **Description**: Method to access the list labs of a course.  **Input Parameter(s)**: N/A  **Return Type**: array of Course objects of type  Section |

Still more Information is required in order to draw a schedule:the section IDs, Times, Types, Classrooms.

|  |  |
| --- | --- |
| **Classes involved** | Section |
| **List of Methods** | 1. getID(): String   **Implemented in Class**: Section  **Description**: Method to access the id of a section.  **Input Parameter(s)**: N/A  **Return Type**: string   1. getTime(): String   **Implemented in Class**: Section  **Description**: Method to access the time of a section.  **Input Parameter(s)**: N/A  **Return Type**: string   1. getClassroom(): String   **Implemented in Class**: Section  **Description**: Method to access the location of a section.  **Input Parameter(s)**: N/A  **Return Type**: stringA  **Return Type**: string |