**Detailed Design**

This detailed design will include the logical back-end organization of “The Force” thus far. These diagrams differ from the class diagrams in that they represent the system as it is organized in the database. Non labeled arrows show a “Has-a” relationship.

Subsystems

User management subsystem

. 1. **Detailed design diagram**

**Enrollment**

Student\_idStudent: INT

Sections\_Section: VARCHAR

Sections\_course\_Master\_list\_id : VARCHAR

**Transcripts**

Grade: VARCHAR

Completed: BOOLEAN

Semester: VARCHAR

1 1

1..\*

1

**Student**

idStudent: INT

PermanentCode:VARCHAR

Is a

**Addresses**

idAddresses: INT

address: VARCHAR

line2: VARCHAR

city: VARCHAR

province: VARCHAR

country: VARCHAR

zip\_code: VARCHAR

mailing: BOOLEAN

home: BOOLEAN

Is a

**Admin**

userName : VARCHAR

Views

**User**

FirstName : VARCHAR

LastName : VARCHAR

Email : VARCHAR

Password : VARCHAR

1..\* 1

This subsystem includes all student and admin information information. It allows for the creation of an account on “The Force”. The student may first sign up, and then subsequently sign in with their credentials. Any account will be considered a “User”, but depending on restrictions the account will either be of type “student” or “admin”. Furthermore, the subsystem stores relevant information regarding transcripts and student records. This allows the subsystem to communicate clearly with the course registration subsystem. The Student class will be referenced by “Enrollment” in the database, which in turn will be referenced by the “transcripts” class. An instance of the “transcript” class refers to a course the student has taken. Any superfluous information about the student that is not directly needed to enroll in a course is stored in a separate class called “Addresses” which references the “Student” class. The address table is included in the design for two reasons. The first reason is so the user can maintain his/her current address through this portal (stretch goal of the system). The second was to account for this system’s storage to draw information for other systems the school could use. This separation allows for easier manipulation of a user. The “enrollment” class is what will allow the user management subsystem to communicate with the course registration subsystem. Finally, if an admin wishes to view the transcript of a student in a specific course, they may because the enrollment and transcript classes are linked.

2. **Unit description**

**Class User**

* FirstName (VARCHAR, String): stores the user’s first name
* LastName (VARCHAR, String): stores the user’s last name
* Email (VARCHAR, String) : stores the user’s email.
* Password (VARCHAR, String) : stores the user’s password for authentification purposes.

**Class Admin**

* userName (VARCHAR, String) : stores the admin’s User Name to display.

**Class Student**

* idStudent (INT): stores the student id number
* PermanentCode (VARCHAR, String) : stores the student’s permanent code

All above information is what is vital to the student’s ability to registering for courses and singing in.

**Class Addresses**

* idAddresses (INT): Because a user may possess many addresses, idAddresses stores an id number which may help refer to the same user without confusing the address instances.
* address (VARCHAR, String) : stores the user’s address
* line 2 (VARCHAR, String) : stores the user’s alternate address
* city (VARCHAR, String) : stores the user’s city
* province (VARCHAR, String) : stores the user’s province
* country (VARCHAR, String) : stores the user’s country
* zip\_code (VARCHAR, String) : stores the user’s zip code (postal code)
* mailing (BOOLEAN) : determines whether or not the given address is the mailing address to send any important documentation.
* home (BOOLEAN) : stores whether or not the address is a home address or not

**Class Enrollment**

* Student\_idStudent (INT) : stores a student’s id number. This allows the instance of “enrollment” to associate with a specific student.
* Sections\_section (VARCHAR, String) : stores the section the student is enrolled in.
* Sections\_course\_Master\_list\_id (VARCHAR, String) : stores the id number of the course the student is enrolled in from the database. This allows for the program to determine which courses have been completed or not.

**Class Transcripts**

* Grade (VARCHAR, String): stores the student’s grade in the given course
* Completed (BOOLEAN) : stores whether or not the student successfully completed the course or not.
* Semester (VARCAHR, String): stores the semester in which the student took the course.

Course subsystem

1. **Detailed Design Diagram**
2. 1..\*

**Section**

Section : VARCHAR

Semester : VARCHAR

Course\_Master\_List\_id : VARCHAR

**Course**

(course\_Master\_List)

Id : INT

Course\_code : VARCHAR

Number : INT

Description VARCHAR

Credits : INTEGER

Suggested\_Semester : INTEGER

**Prerequisites**

MainCourseID : VARCHAR

PrereqCoruseID : VARCHAR

1

1

1

**Enrollment**

Student\_idStudent: INT

Sections\_Section: VARCHAR

Sections\_course\_Master\_list\_id : VARCHAR

\*

\*

1..\*

**Timeslot**

Id : INT

Sections\_Section : VARCHAR

Sections\_Course\_Master\_List\_id : VARCHAR

Start : time

End : time

DOW : CHAR

The course subsystem contains all information regarding course numbers, IDs, and times. Having an entire subsystem where this information can be stored allows for a clean separation of tasks between the two subsystems. Although a student will enroll to a course from their point of view, the system sees it as an enrollment in a section. This is because different sections have different timeslot instances. This allows the system to simultaneously enroll the student while checking for time conflicts. It is important to note that any information regarding the course name and number starts at the course class. From there, you can even find the suggested semester. As you go further away from the course class, there is a “breadcrumb trail” that can always lead you back to it easily. Class Section contains an ID that matches to a class in the master list. Enrollment and timeslot classes have the same course name and number, although we append “Section\_” at the front, to force the path through the section class, without accidentally accessing every single section in the course. On top of this information, they contain the section ID in order to not pass through and get enrolled into the wrong section.

1. **Unit Descriptions**

**Class Enrollment**

* See “User management subsystem”
* This class serves as a “UI” between the two subsystems
* Sections\_Course\_Master\_List\_id (VARCHAR, String) : Refers to the id of a course. This id is first matched to a section along with Sections\_Section (VARCHAR, String) to match to a section the student is enrolled in.

**Class Section**

* Section (VARCHAR, String) : Stores the section number of a course the student is enrolled in.
* Semester (VARCHAR, String) : Stores the semester of the course the student is enrolled in.
* Course\_Master\_list\_id (VARCHAR, String) : Stores the ID of the course the student is enrolled in. This is matched to a specific ID in the database (Course class).

**Class Course**

* Takes the form of Course\_Master\_list in the database. Contains all information regarding courses.
* Id (INT) : Stores the ID number of a course.
* Course\_Code (VARCHAR, String) : Stores the course code in a string (4 letters)
* Number (INT) : Stores the course number (3 digits).
* Description (VARCHAR, String) : Stores the description of a course so that the student may know what they are chosing.
* Credits (INTEGER) : Stores the number of credits the course provides.
* Suggested\_Semester (INTEGER) : Stores the suggested semester of the year for the student (between 1 and 3). This allows to provide the student with a recommended course sequence.

**Class Prerequisites**

* MainCourseID (VARCHAR, String) : Stores the course name and number of the course the student is trying to enroll in.
* PrereqCourseID (VARCHAR, String): Stores the course name(s) and number(s) of the course(s) the student should have previously completed in order to enroll to “MainCourse”.

**Class Timeslot**

* ID (INT) : contains the student number of the student taking a course in that given timeslot.
* Sections\_Section (VARCHAR, String) : Stores the section number of the course of the timeslot the student is enrolled in.
* Sections\_Course\_Master\_List\_id (VARCHAR, String) : Stores the course name and number of the section of the timeslot the student is enrolled in.
* Start (Time) : the starting time of the timeslot.
* End (Time) : The ending time of the timeslot.
* DOW (CHAR): Day of the Week represented by a single character.

Scheduler Subsystem

**1.Detailed Design Diagram**

1 Accesses 1..\* 1 Uses 1 1

**Schedule**

Month: VARCHAR

Time: VARCHAR

Year: VARCHAR

Day: VARCHAR

Semester: VARCHAR

SelectedCourses: List<Course>

**Student**

idStudent: INT

PermanentCode:VARCHAR

FirstName : VARCHAR

LastName : VARCHAR

Email : VARCHAR

Password : VARCHAR

**Scheduler**

CurrentSchedule : Schedule

**Transcript**

**Course**

**(Prerequisites)**

1 Creates 1

Uses

1..\*

The Scheduler Subsystem includes all elements of the database and app that allow a student user to create a comprehensive schedule that they can use. The Student class has been shown with all the information of a user included to show that that information is crucial to creating the correct schedule and assigning it to the correct student. The part of the diagram that details that the student accesses the transcript is simplified. From the database’s point of view, the user accesses their transcript of each course separately. The app, however, collects these into one transcript to make it easier for the student to view. This is essential to know what courses are necessary to enroll in. The Scheduler itself is contained within the app, and it uses the student’s selections of courses to generate a schedule. Once again, this aprt has been slightly simplified. The prerequisites and courses have been grouped in order to have a cleaner diagram. Finally, the scheduler creates a schedule. This schedule has all the information from timeslot classes in it, as well as a list of courses. This subsystem has a purpose of bridging the gap between the user and the database. Effectively, it provides all information to the student clearly, as well as manipulates it. This is probably the most essential of subsystems, since without it, there is no app. Finally, it serves as a bridge between the other subsystems as well, functionally speaking (organizationally, they are already linked, but not functionally without this subsystem).

1. **Unit Description**

**Class Student**

* See “user management subsystem” to see the separation between “user” and “Student”. In this diagram, those have been shown as one.

**Class Transcript**

* See “user management subsystem” to see the separation between “enrollment” and “transcript” in the database. In the app, however, a transcript contains all relevant information of all courses the student may have taken.

**Class Scheduler**

* CurrentSchedule: Stores the schedule which the scheduler generates.

**Class Schedule**

* Month (VARCHAR, String) : Stores the month the schedule is displaying.
* Time (VARCHAR, String) : Stores the time that the schedule is displaying.
* Year (VARCHAR, String) : Stores the year that the schedule is displaying.
* Day (VARCHAR, String) : Stores the day that the schedule is displaying.
* Semester (VARCHAR, String) : Stores the semester that the schedule is displaying.
* SelectedCourses (List<Courses) : List of courses the student has chosen to include in the schedule.

**Class Course (Prerequisites)**

* See “Course subsystem” to see the separation between the “Course” and “Prerequisites” classes in the database. In the app, these are joined together.