Design Document

For

Dragon Course Scheduler

Prepared by

Nathan Gelfant,
Kevin Huang,
Stan Kolakowski,
Mark Scheid

At
Drexel University

Design Document Revisions History

Date	Description	Version	Editor(s)
2/25/2013	Document created.	1.0	Nathan Gelfant, Kevin Huang, Stan Kolakowski, Mark Scheid

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to detail the implementation of the Dragon Course Scheduler as outlined in the Software Requirements Specification document. The Dragon Course Scheduler itself is intended as a tool to simplify and improve the experience of planning a student schedule using the information provided by the Drexel Term Master Schedule.

1.2 Intended Audience

The intended audience for this document are designers, developers, testers and reviewers of the Dragon Course Scheduler program.

1.3 Scope

The scope of this design document is to describe the design and architectural details of the Dragon Course Scheduler beta release, intended audience of this document is for the developer and tester and reviewer of this software.

1.4 Definitions, Acronyms, and Abbreviations

1.4.1 Definitions

Browser: A piece of client side software, such as Firefox or Internet Explorer, capable of displaying web pages written in HTML, CSS and JavaScript code, as well as running java applets.

Class Selection Interface: A web page through which the user views the list of available classes for the selected term based upon their previously entered courses.

Concentration: Also referred to as "tracks", will be major specific information required or defined by Drexel.

Database: A database is a structured collection of data that is organized for later retrieval.

Dependent classes: Classes which require the course in question as a prerequisite.

Prerequisites table: A table in the database that is populated based on Drexel Course Catalog, detailing what courses are necessary before one can register for a given class.

Program: The java backend that manages data flow between the interfaces and the database.

Selected Schedule Interface: A web page through which the user views their current weekly plan of classes for the selected term.

Server: A content hosting system that will deliver content to end users and host aggregated information.

TMS Synchronizer: The server-side tool that automatically updates the database with available course information and major concentration definitions.

User-Background Interface: A web page through which the user can input their major, concentrations, and the list of classes they have already taken.

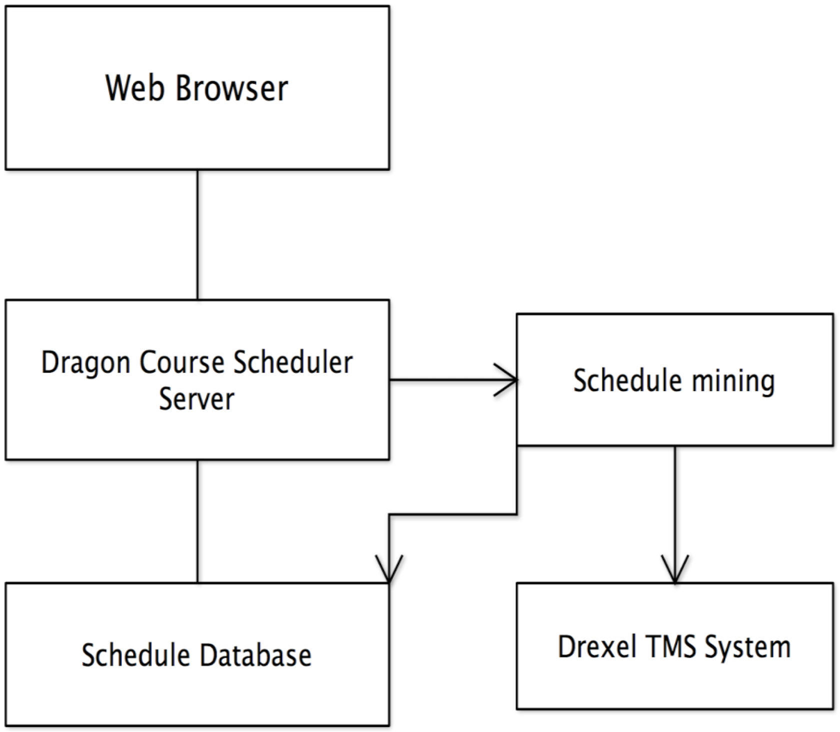
User-History Object (UHO) The user history object contains a data structure capable of holding information containing all previous classes completed by the user, as well as the currently selected list of classes for the scheduled terms.

1.4.2 Abbreviations

DCS	Dragon Course Scheduler
CRN	Course Reference Number
JRE	Java Runtime Environment
JVM	Java Virtual Machine
TBD	To be dated
TBA	To be announced
TMS	Term Master Schedule

1.5 Context Diagram

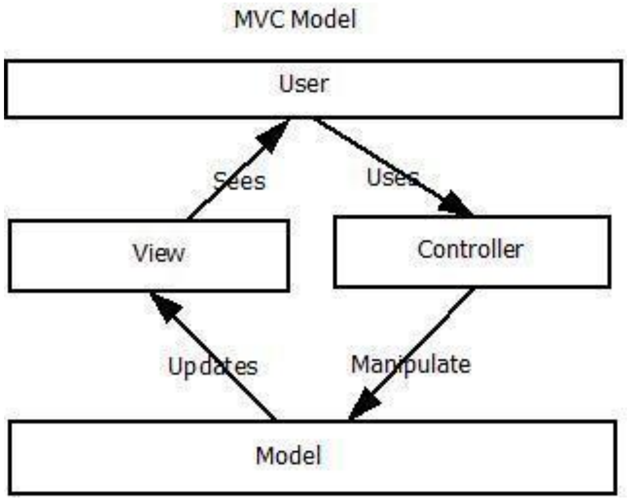
The program will run on a web server, so the natural context is the web. And since it also interacts with Drexel TMS and a remote database, the following graph showed as the context diagram.



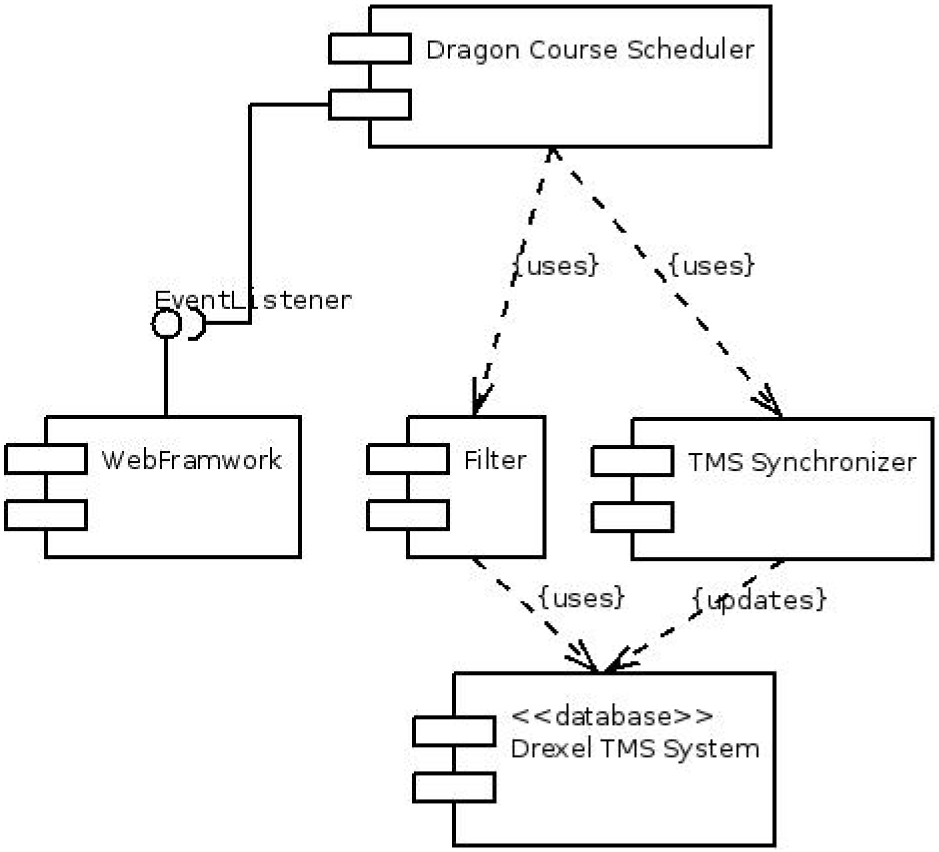
2. Architecture

2.1 Overview

The general architecture of the Dragon Course Scheduler is MVC (Model, View, and Controller) structure. The following architect graph is shown to illustrate the overall design. Model is the TMS Database, and the controller is composed of Filter and Dragon Course Scheduler, and the View is the WebFramwork model.

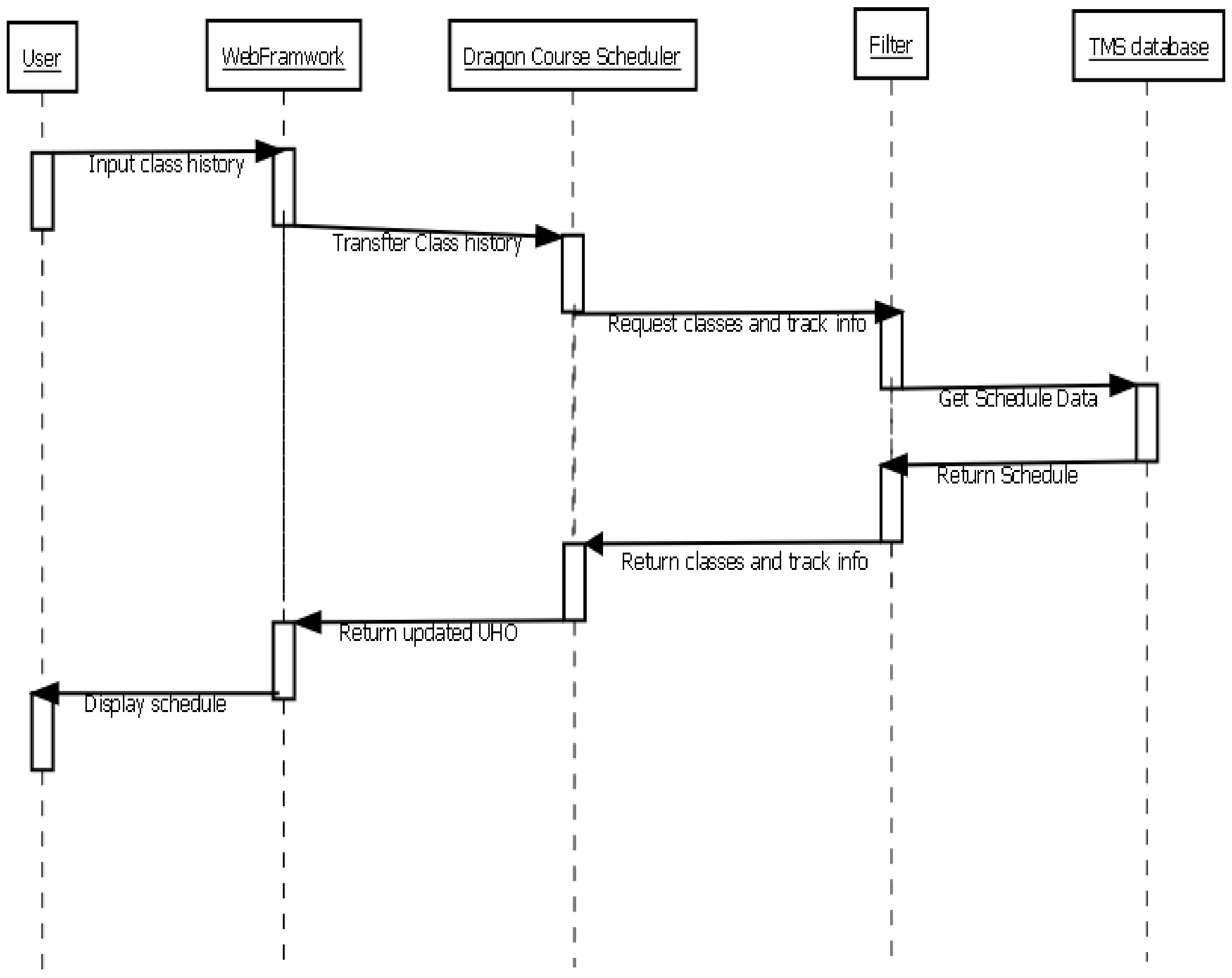


A component diagram is show as follows:



The program is composed of five components, and The Drexel TMS System is the remote database component which holds schedule information. The important interface EventListenter is provided by the WebFramework so that it will listen to user action and transmit to DCS, which will then control other components to accomplish related tasks.

In order to understand how a most common use case of the program, a sequence diagram is included as follows to show the time to time interaction between modules and components.



2.2 Survey of Technologies Used

The Dragon Course Scheduler makes use of a MySQL database and the Java virtual machine to generate the functionality for the end user.

2.3 Presentation Layer Components

2.3.1 Swing WebFramework

The dragon course scheduler is a web application, and thus the presentation layer is a web display. The Swing framework is used for the implementation of this presentation layer.

2.3.2 Cascading Style Sheet (CSS)

The program utilizes CSS to add certain style elements to pages it displays.

2.4 Data Layer Components

2.4.1 Database Engine

The system used MySQL database with InnoDB storage engine. MySQL is portable and boasts fast read performance.

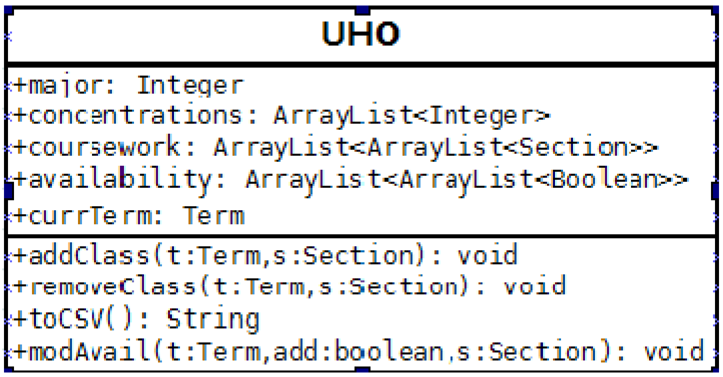
2.5 External Components

2.5.1 TMS Synchronizer

Since the program utilized the Drexel TMS data, this module is needed to fetch information from Registrar website to record scheduling updates. This TMS Synchronizer is an external model of the system, invoked from server side on a daily basis. The TMS Synchronizer is designed and implemented in python.

3. Design Features

3.1 UHO(User History Object)



```
UHO
+major: Integer
+concentrations: ArrayList<Integer>
+coursework: ArrayList<ArrayList<Section>>
+availability: ArrayList<ArrayList<Boolean>>
+currTerm: Term
+addClass(t:Term,s:Section): void
+removeClass(t:Term,s:Section): void
+toCSV(): String
+modAvail(t:Term,add:boolean,s:Section): void
```

The UserHistoryObject(UHO) is an important class that encapsulates crucial user data. It is responsible for storing all of the user’s academic data. This includes major, concentration(s), courses that have been taken in the past, as well as courses that the user has selected to take in the upcoming school year.
.
Upon instantiating the DragonCourseScheduler class, it creates a blank UHO. The DCS then makes a call to the framework, populating the UHO with user data. The UHO stores the user’s classes in an coursework ArrayList of ArrayLists, in which the zeroth indexed ArrayList represents the courses taken in the past, and the first, second, third, and fourth indexed ArrayLists account for the Fall, Winter, Spring, and Summer term, respectively.

3.1.1 addClass(Term t, Section s) / removeClass()

A UHO must have the ability to add and remove new classes to UHO in order to keep updated with any changes the user may make. This ability is satisfied with the addClass() and removeClass() methods. The methods will remove or add the selected Section from the appropriate inner ArrayList of *coursework* provided a constant of the Term enum (History, Fall, Winter, Spring, and Summer maps to the 0th, 1st, 2nd, 3rd, 4th indexed ArrayList, respectfully).

3.1.2 toCsv()

The UHO must also be able to export the data in a csv string. The string is generated using the toCsv() method. This is used for the export feature of the program

3.2 Dragon Course Scheduler class

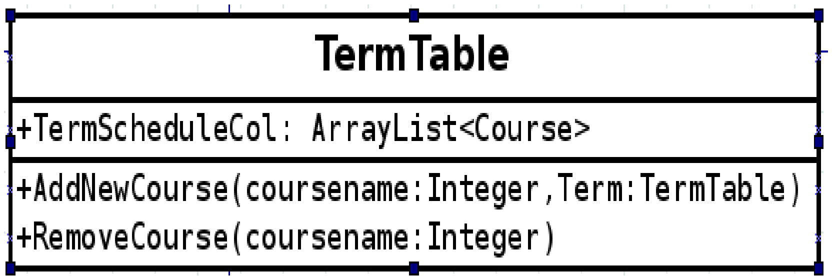
The DCS class is the central class that orchestrates the operation of the different parts of the program. Taking user requests from the web framework, it generates and maintains the UHO with schedule information, and gives that to the filter so that it can provide an accurate, updated list of possible class options back to the framework for user review. It handles conversion of data formats to facilitate other functions of the program.

3.3 Filter Model

The Filter Model holds the key data of the program, links with the database and is responsible for feeding data for the View. Controlled by the Dragon Course Scheduler, the Filter model consists of several classes, detailed below:

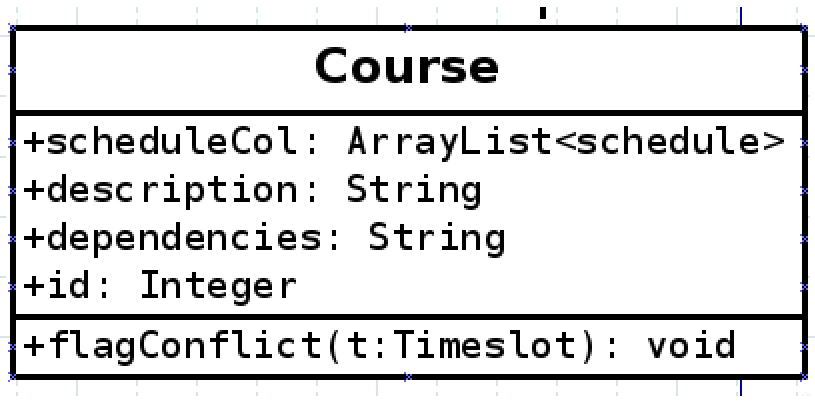
3.3.1 Term Table Class

The term table class holds the collection of courses being offered after being filtered against the UHO data. Any of these courses can be added to the user's planned schedule successfully, as they are verified by the checkFree class which explained in the algorithm section(3.3.4)
The Term Table class diagram is shown as follows:


TermTable
+TermScheduleCol: ArrayList<Course>
+AddNewCourse(coursename:Integer,Term:TermTable)
+RemoveCourse(coursename:Integer)

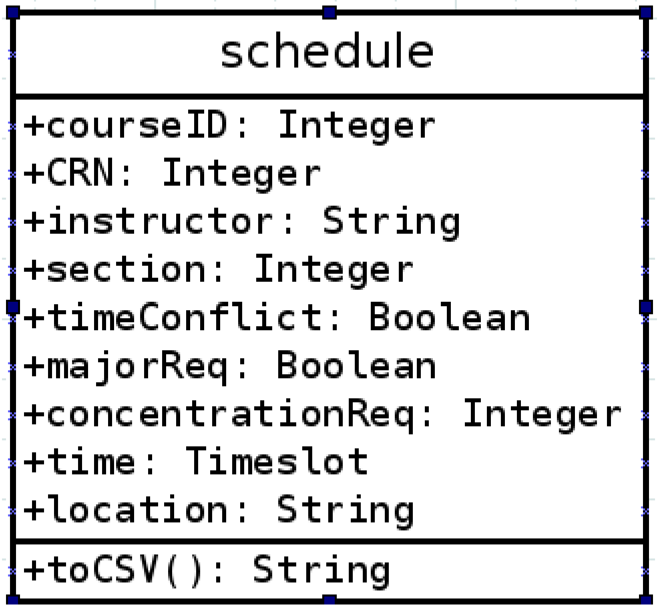
3.3.2 Course Class

The course class holds the detailed schedule of a class and the prerequisites and as well as description of a course and dependencies with future classes. The class diagram is as follows:

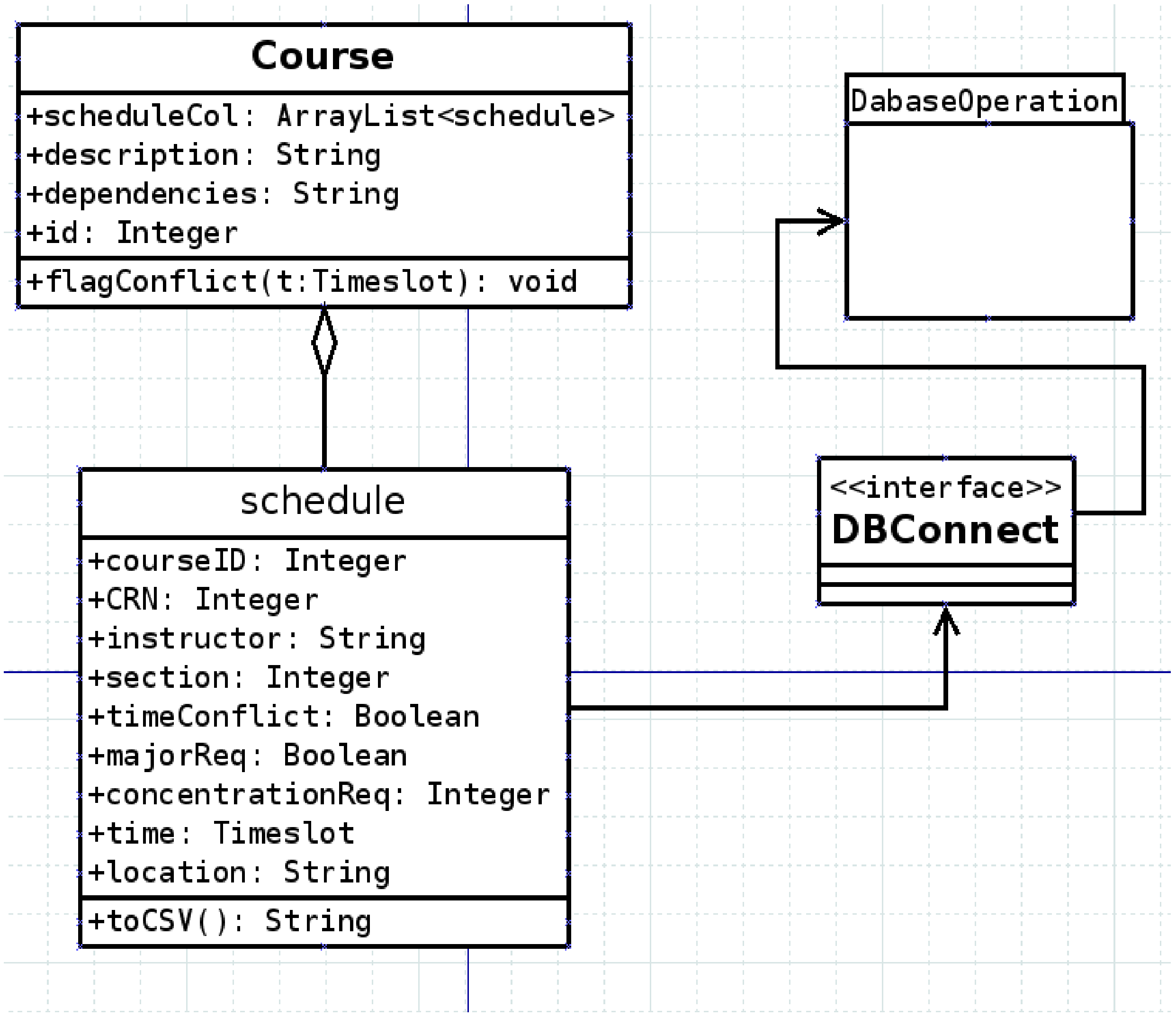

Course
+scheduleCol: ArrayList<schedule>
+description: String
+dependencies: String
+id: Integer
+flagConflict(t:Timeslot): void

3.3.3 Schedule Class

The Schedule class is directly interacting with the database and therefore, holds the detailed schedule information for individual courses.



The schedule class uses a DBConnect Interface to interact with the database as shown in following diagram



3.3.4 Algorithm

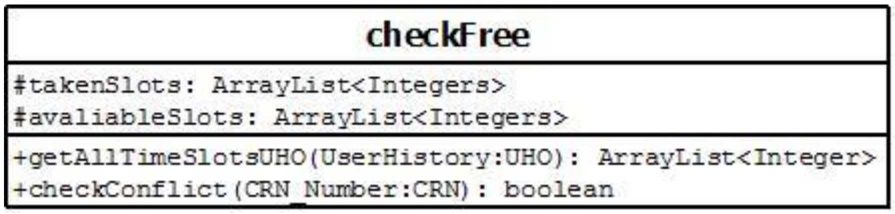
3.3.4.1 Description

The checkFree class handles the filtration process. Having received the updated list of selected classes from the UHO, and the full list of classes from the database for a given term, it weeds out all courses that conflict with the user data.

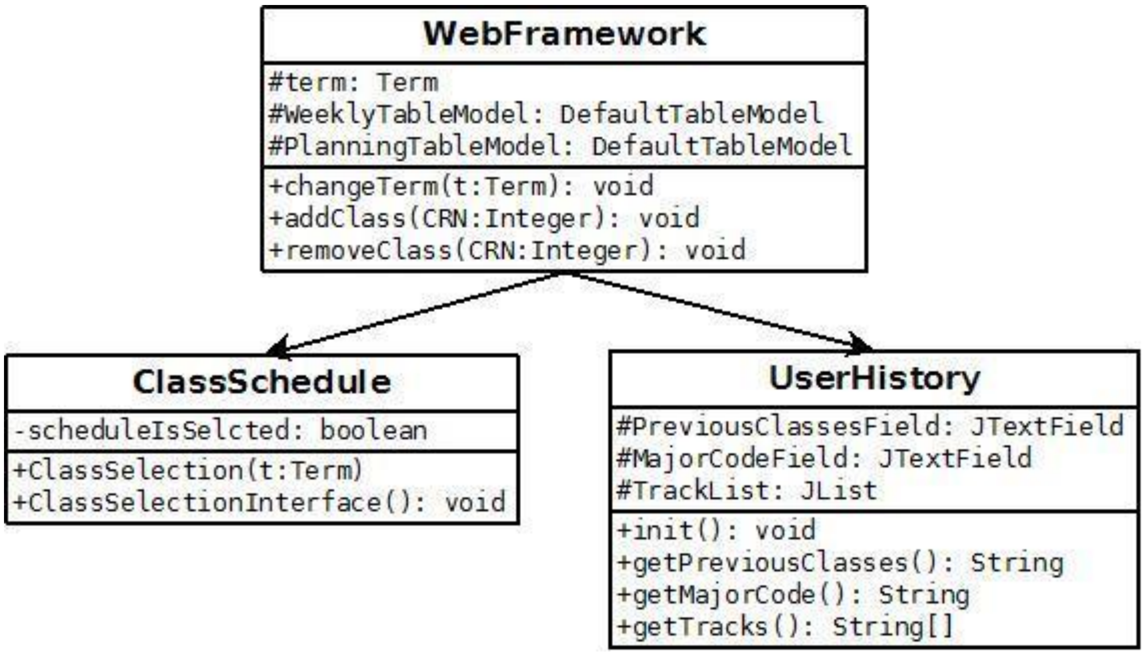
3.3.4.2 Detailed process

Given a term t with n class offerings, and m selected classes, the course checks for any conflicts. For each class x in the database, it must compare the timeslot against the UHO availability table. If any portion of the classes overlap, the course is removed. Next, the item x is compared to the list of classes m for all terms prior to t, and if the course id is found, it is removed. FInally, if the course fits the user's schedule and history, the list of pre-requisites for x is compared against the selected classes for terms prior to t. if any prerequisites are missing, then the class is removed. Should the class x have passed these 3 filters, then checkFree returns it as a valid class to be passed back to the caller.

The following graph displays UML details of the class..

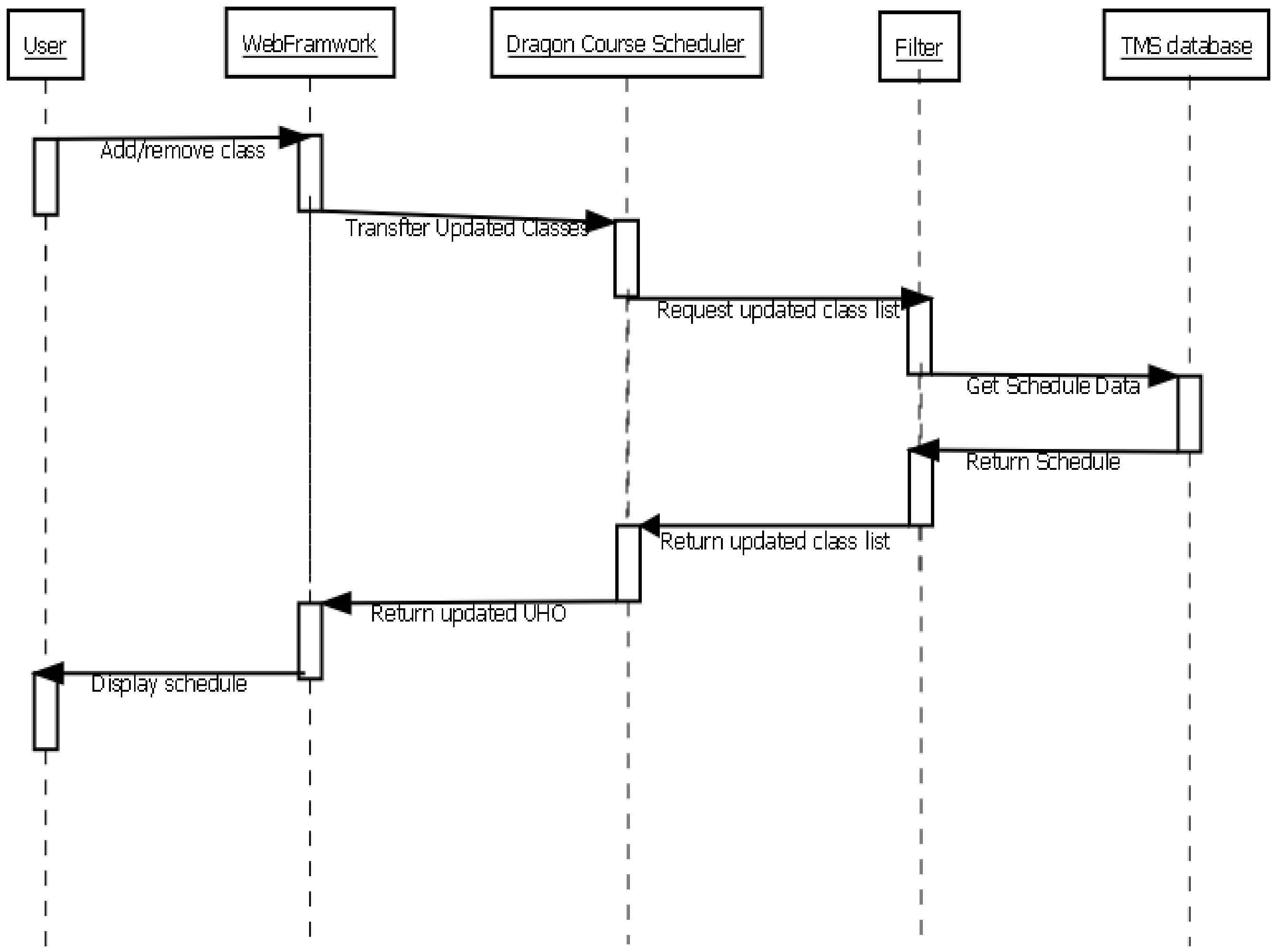

checkFree
#takenSlots: ArrayList<Integers>
#avaliableSlots: ArrayList<Integers>
+getAllTimeSlotsUHO(UserHistory:UHO): ArrayList<Integer>
+checkConflict(CRN_Number:CRN): boolean

3.4 WebFramework Model


WebFramework
#term: Term
#WeeklyTableModel: DefaultTableModel
#PlanningTableModel: DefaultTableModel
+changeTerm(t:Term): void
+addClass(CRN:Integer): void
+removeClass(CRN:Integer): void
ClassSchedule
-scheduleIsSelcted: boolean
+ClassSelection(t:Term)
+ClassSelectionInterface(): void
UserHistory
#PreviousClassesField: JTextField
#MajorCodeField: JTextField
#TrackList: JList
+init(): void
+getPreviousClasses(): String
+getMajorCode(): String
+getTracks(): String[]

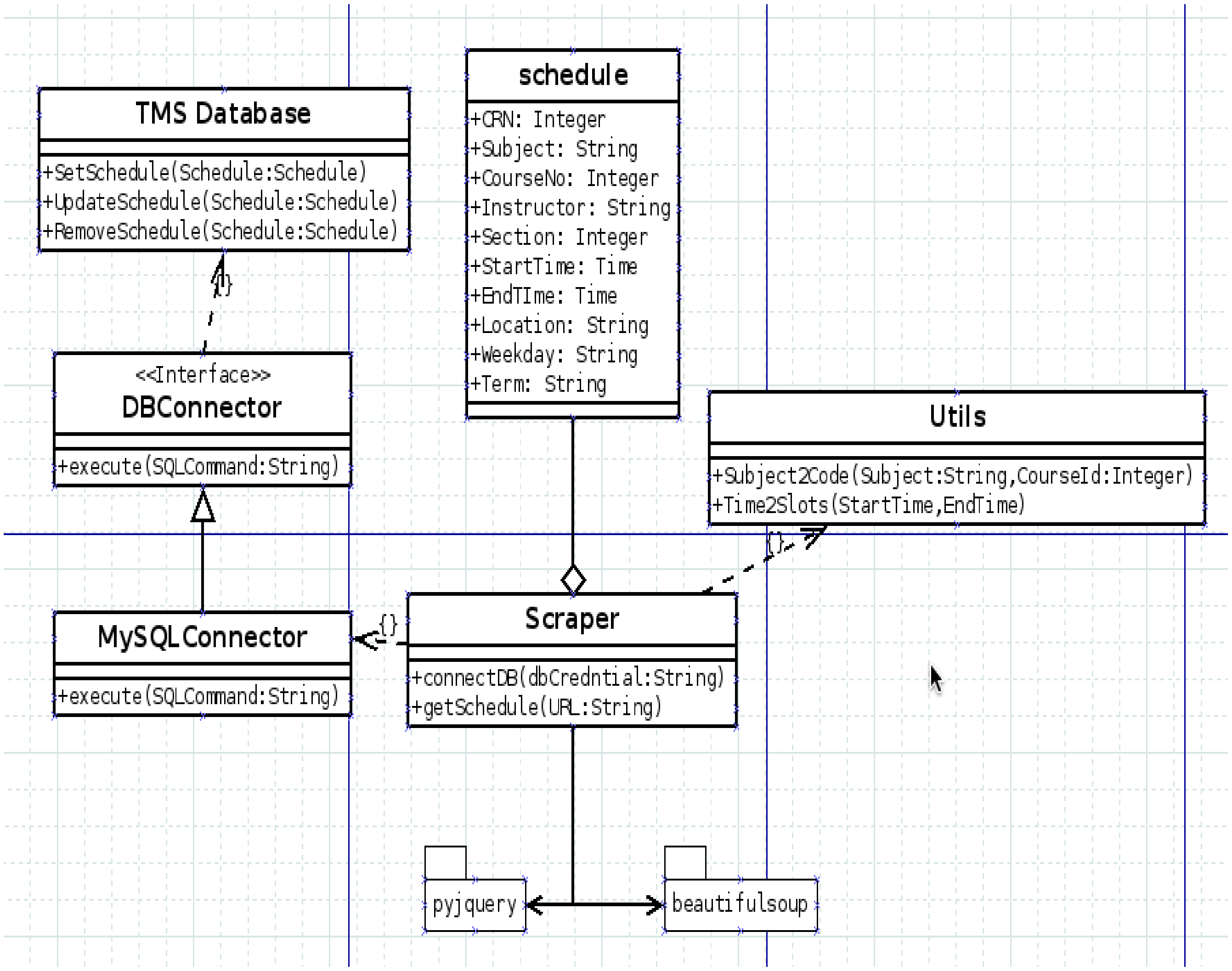
The above WebFramework model extends JApplet in order to create a Java Applet to display on the hosting webpage. It mainly talks with the DragonCourseScheduler class in order to either send it UserHistory class information or to populate the ClassSchedule class. These classes also contain the entire front end interface and graphics. Error checking for user input also occurs at this stage. The UserHistory class generally gathers the user information to send back to the DragonCourseScheduler class. The ClassSchedule class is actually a combination of two interfaces. The first state allows the user to select a schedule. The second state (when the private variable is true) allows the user to view their selected schedule based on the term being viewed.

To explain how the WebFramworks handles updating user information, the following sequence diagram is used to illustrate a sequence of module interactions.



3.5 TMSSynchronizer Utility

The TMSSynchronizer is design outside of the main program, and evoked only on the server side as a scheduled task. This utility is written in Python and is consisted of a few python class and two libraries as demonstrated in following class diagram.



4. Database Design

4.1 Overview

4.1.1 Tables

The database is consisted of 4 tables

schedule	Schedule for Specific Terms
require	Consists all prerequisites and post requisites of a specific class
courseRef	Consists a unique identifier for a course, and its human-readable name and description

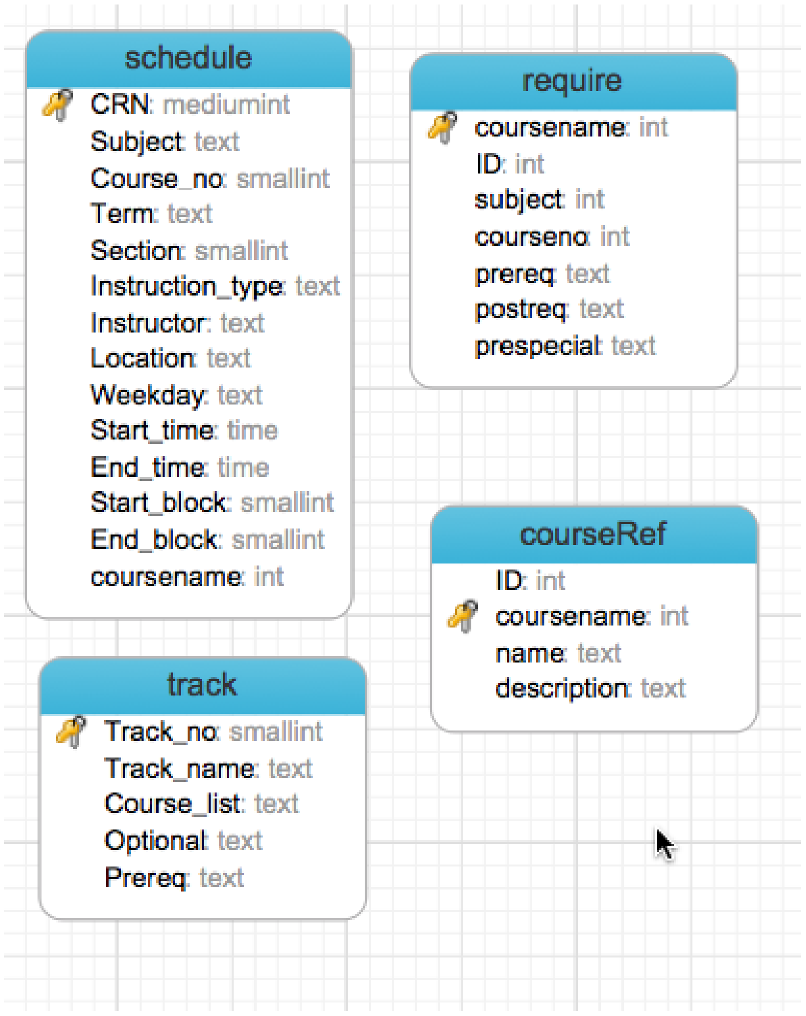
track	Consists all tracks information for any given concentration or track

4.1.2 Data Generation

The database is updated, and maintained by the TMSSynchronizer tool, which is run at the server side on a daily basis.

4.2 Database Schema

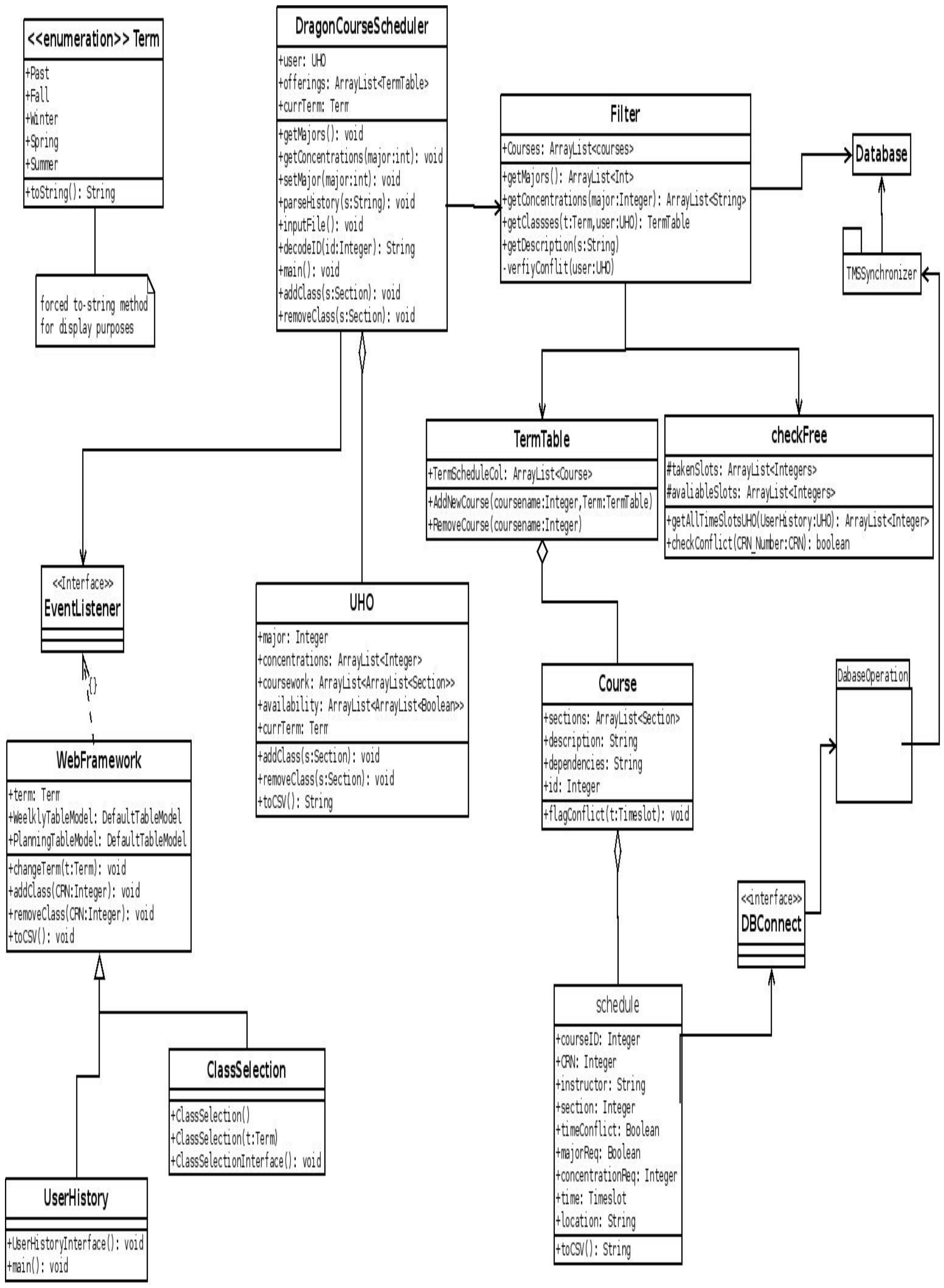
The coursename filed is both the primary and foreign key(of the courseRef table) in the require table.



5. Summary

5.1 Visual System overview

The following page is the essential classes and methods of the Java Class UML diagram of the program.



<<enumeration>> Term
+Past
+Fall
+Winter
+Spring
+Summer
+toString(): String
forced to-string method for display purposes
DragonCourseScheduler
+user: UHO
+offerings: ArrayList<TermTable>
+currTerm: Term
+getMajors(): void
+getConcentrations(major:int): void
+setMajor(major:int): void
+parseHistory(s:String): void
+inputFile(): void
+decodeID(id:Integer): String
+main(): void
+addClass(s:Section): void
+removeClass(s:Section): void
Filter
+Courses: ArrayList<courses>
+getMajors(): ArrayList<Int>
+getConcentrations(major:Integer): ArrayList<String>
+getClassses(t:Term,user:UHO): TermTable
+getDescription(s:String)
-verfiyConflit(user:UHO)
Database
TMSSynchronizer
TermTable
+TermScheduleCol: ArrayList<Course>
+AddNewCourse(coursename:Integer,Term:TermTable)
+RemoveCourse(coursename:Integer)
checkFree
#takenSlots: ArrayList<Integers>
#avaliableSlots: ArrayList<Integers>
+getAllTimeSlotsUHO(UserHistory:UHO): ArrayList<Integer>
+checkConflict(CRN_Number:CRN): boolean
<<Interface>>
EventListener
{}
UHO
+major: Integer
+concentrations: ArrayList<Integer>
+coursework: ArrayList<ArrayList<Section>>
+availability: ArrayList<ArrayList<Boolean>>
+currTerm: Term
+addClass(s:Section): void
+removeClass(s:Section): void
+toCSV(): String
Course
+sections: ArrayList<Section>
+description: String
+dependencies: String
+id: Integer
+flagConflict(t:Timeslot): void
DabaseOperation
WebFramework
+term: Term
+WeelklyTableModel: DefaultTableModel
+PlanningTableModel: DefaultTableModel
+changeTerm(t:Term): void
+addClass(CRN:Integer): void
+removeClass(CRN:Integer): void
+toCSV(): void
<<interface>>
DBConnect
ClassSelection
+ClassSelection()
+ClassSelection(t:Term)
+ClassSelectionInterface(): void
schedule
+courseID: Integer
+CRN: Integer
+instructor: String
+section: Integer
+timeConflict: Boolean
+majorReq: Boolean
+concentrationReq: Integer
+time: Timeslot
+location: String
+toCSV(): String
UserHistory
+UserHistoryInterface(): void
+main(): void

5.2 Advantages

The advantage of current design is simple and powerful. By leveraging a MVC model, we have a highly modularized software to maintain and small changes in the term master schedule can be adjusted in the Filter model without affecting other models.

5.3 Limitation

The current design does not take a few non-functional limitation into consideration. As the data grows bigger, and the program expanding to accept all majors, it will take significantly longer time to generate suitable schedule and overhead of database connection might lead to redesign partial model of the software.

5.4 Future changes

The projected future changes will be allowing more majors and more complicated scheduling system. Currently the program will only offer major based information, and does not consider course completion requirements, and therefore, does not suit for a comprehensive course scheduling.

Appendix Traceability Matrix

3.2.1.1, 3.2.1.2, 3.2.1.3, 3.2.1.7, 3.2.1.8,3.2.2,	3.1.1,3.4
3.2.1.9	3.1,3.4,3.3.1
3.2.1.5, 3.2.1.10,3.2.7	3.2,3.4
3.2.1.6, 3.2.1.11	3.3.3,3.3.4
3.2.3	3.1.2
3.2.4,3.2.5,3.2.6	3.4
3.2.8	3.5