PROJECT PROPOSAL

Temple Course Helper

(Application for helping students find courses, using C# and SQL)

Table of Contents

Project Abstract

Temple Course Helper is an application that will take the users' course numbers and find the course information from the catalog on Coursicle’s website as well as teacher ratings from it. The application will take all of this information and store it in a database along with the student ID entered. The database will allow returning users to open up their previous search for ease of use. On the screen, all of the information (course time, professor, rating, description) will be displayed, and the user will be asked if they would like to be sent this information via email as well. This will help Temple students better plan their semesters and hopefully will save them time.

High-Level Requirement

The application will first prompt the user if they have an already entered student ID to display their previous data. If it is a first-time user the user will enter their student ID as well as the courses they want to find. This script will use web driver from the selenium package. This is a package that allows the script to interact with a web browser, in the case of this project the browser will be Chrome. Once the script gets online it will find the necessary xpaths that are found by inspecting the page. The script will pull the information from these xpaths, and store the info on a HashMap. The information will be stored in a different class where all general information about the course will be stored (besides the course ID). This class will be used as the value in the HashMap while the course ID will be used as the key. Following this, the script will fill in a record in a database so the user can have their information saved in case they want to revisit it. The information will be displayed on the screen, and if the user wants, they can be sent the information through a bot.

Conceptual Design

Around 7 classes should be present (main/GUI interaction class, database communication class, worker class (operating the bot), email bot class, course conflicts checker class, course builder class). Done either in C# (easier to work with GUI) or Java (more familiarity) depending on the level of comfort with the team. Knowledge of SQL commands and Access will also be useful and can be improved on with this project. How to properly use the inspect tool on browsers as well as being able to break down what information is being given in HTML will also be a skill that will be necessary and can be improved on. We will be scraping info from the site by using xpaths. Xpaths are the codes that designate the different parts, or controls, of the UI that users see on browsers such as buttons, text boxes to type etc.

Proof of Concept

https://github.com/cis3296s22/ArthurKozhevnikProject

Background

This project will be built from scratch with no resources taken from any open sources. However, the functionality will be similar to a search tool. If for example, you enter a course number on the Temple’s website, you will be brought to the description of the course entered. Temple’s website does not have any functionality that displays multiple courses at once or sends you the information via phone, which is something that is planned for this project.

CUNYFirst is also quite similar to what we are trying to build. From my understanding, it will search globally to find a course that suits the users. The search is much more in-depth requiring a lot more information to conduct the search, but its range is quite large. Temple

Course Helper on the other hand will be focused on Temple Universities Catalog and will also be displaying ratings of professors teaching those classes based on reviews from multiple websites.

Coursicle is very similar to what we are trying to build and in fact, we scrape the information from Coursicle itself. Where the functionality will differ is that our program will be like an app you open instead of a site, and the app will remember the users' previous search and show them so they can re-enter the information and get their results from the last search. Our team also hopes to provide a conflicts checker to make sure the searched courses do not conflict with each other.

Sources:
https://home.cunyfirst.cuny.edu/psp/cnyepprd/GUEST/CAMP/c/COMMUNITY_ACCESS.CLASS_SEARCH.GBL?FolderPath=PORTAL_ROOT_OBJECT.HC_CLASS_SEARCH_GBL&IsFolder=false&IgnoreParamTempl=FolderPath%2cIsFolder

https://www.coursicle.com/

Required Resources

Operating System: Windows OS is required because the .NET framework runs primarily on it. Windows Forms, the way we will be building the GUI only works on Visual Studio and is not included in Visual Studio Code. Therefore, without a Windows OS, the only other option would be Virtual Machine

Software:

1. Visual Studio is the best IDE for this project because of its built-in Windows Forms. 2. Chrome will be needed for the script to interact. It will be the easiest browser to work with. Along with the browser, an add-on must be installed for the interaction to be facilitated: https://www.tutorialspoint.com/how-to-launch-chrome-browser-via-selenium 3. Selenium Package. This can be installed onto Visual Studio and will provide the library needed for the script to interact with Chrome.

4. Microsoft Access will be used to store user data for future use.

5. Twilio Package. This can be installed onto Visual Studio and will provide the library needed for the script to send a text out to the user.

Vision Statement

For Temple's students, their parents, and professors who need an easier and more informative course-info software the Temple Course Helper is an application that will make finding information about courses and professors easier than ever. Unlike Temple's website or other websites, our product is open for everyone to use, provides all the information in one application, stores this information, and sends them to the user.

Feature List

- As a freshman student I want to have easy access to my recent inquiry so that I can access them easier later.
- As a parent, I want to be able to see all of Temple's courses information so I can help my kid with registration.

- As a professor, I want to see my rating so I can get more feedback.
- As a senior year student, I want to get an email with my inquiries so I can access the information everywhere.

Personas

Bobby, a freshman from Delaware. (Roberto Nano)

Bobby, age 18, just graduated from Brandywine High School, a school in Wilmington, Delaware. Bobby is eager to live the college-student life and cannot wait for the Summer to be over. Even though he was excited to leave his parents' house, he did not want to go very far and so he searched for colleges close to him. Temple University was the ideal choice for him.

Unfortunately, Bobby does not know anyone that goes to Temple and so he is having a hard time finding the right classes with the best professors available for him. Bobby is eager to use Temple's Course Helper because, he thinks, is the best available option for him that will help him get all the information for all the classes he has to take along with the ratings for the different professors.

Sarah, a High School teacher. (Sofia Drachuk)

Sarah, age 39, is a high-school teacher from Wilmington, Delaware. Sarah, born and raised in Wilmington has been a high school teacher for over 10 years and she loves to teach kids and prepare them for college. Sarah is Bobby's mom, and she is excited that her son is going to college after the Summer.

However, Sarah, like any other parent, is worried that her son, Bobby, will leave her house and move to Philly. Sarah wants her son to succeed, and she wants to be there for him whenever Bobby needs her help to find the perfect classes and the best professors for him. Since Sarah has experience on that field, she is excited to use Temple Course Helper to find the best classes and the best professors for her son whenever he needs her opinion and help.

Alex, a College Professor. (Arthur Kozhevnik)

Alex, age 55, is a college professor at Temple University. Arthur, born and raised in South Philly, has been a professor for over 25 years. He is a Political Science major professor in the College of Liberal Arts, and he loves to teach the theory of government and politics at the state and national levels. Alex loves and cares about his kids and he wants them to enjoy his class and learn from it.

However, one time Alex heard one of his students mention a rating about a professor. Since Alex cares a lot about his performance and wants them to enjoy his class, he is going to use Temple Course Helper to monitor his rating and make sure he gets great feedback.

Rebecca, a Senior year student. (Rachel Rubin)

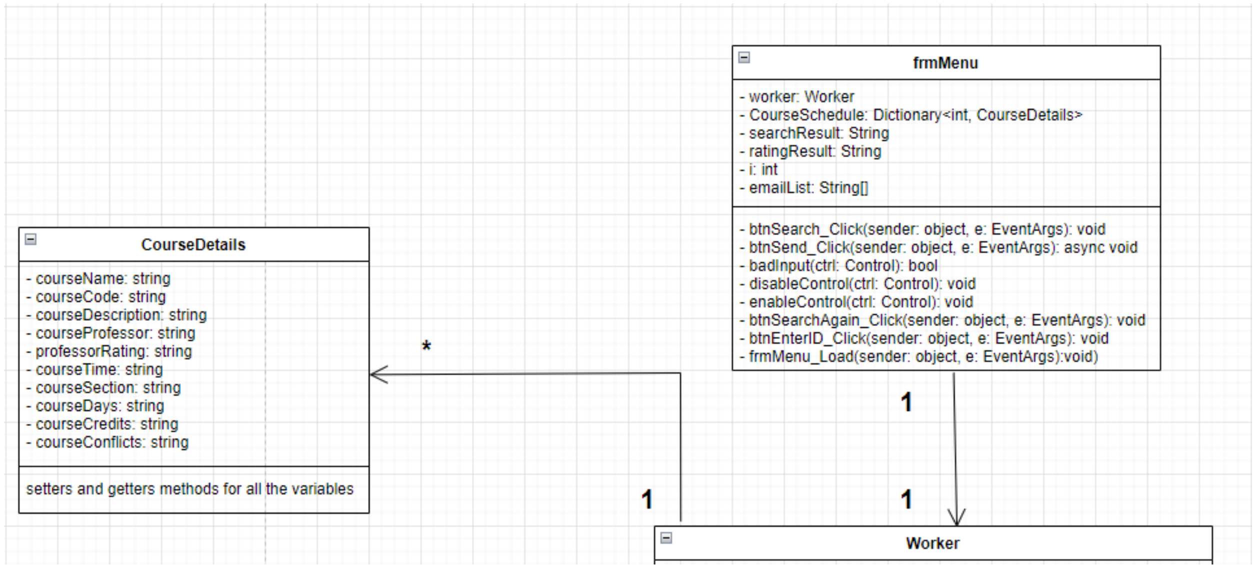
Rebecca, age 23, is a senior year Computer Science student at Temple University. Rebecca was born in New Jersey but moved to Philadelphia around 10 years ago. Ever since she was a kid, she loved computers and games. Her years at Temple have been a little challenging but she has managed to almost finish her degree with a perfect GPA.

Since Rebecca cares about her grades, she wants her final year at Temple to be as easy as possible, especially while she must get the more difficult classes in her major. Rebecca is excited to use Temple Course Helper to find some interesting and not challenging free elective courses taught by professors with great ratings. Finally, since Rebecca will be very busy during her last year, she wants to have these records saved and receive an email with all the information she searched.

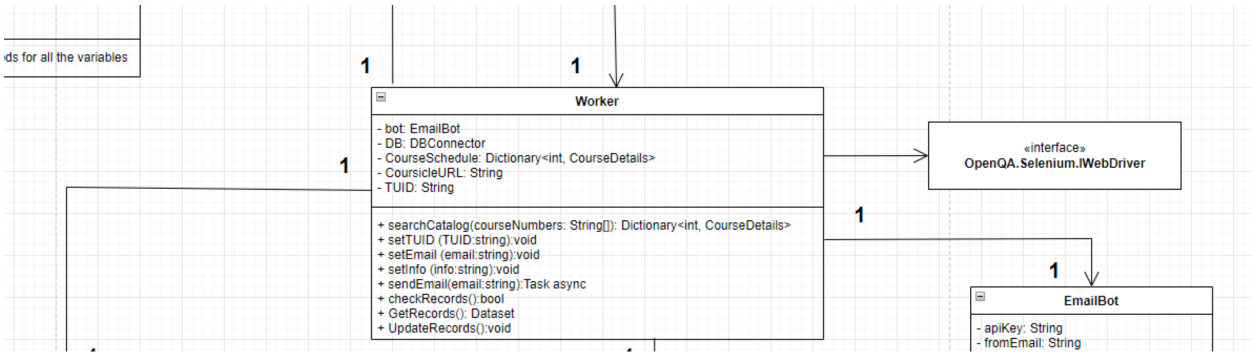
Class Diagram

https://drive.google.com/file/d/1DiDHXDGLA-WezxDrCi93MevsyH-pvpt2/view?usp=sharing

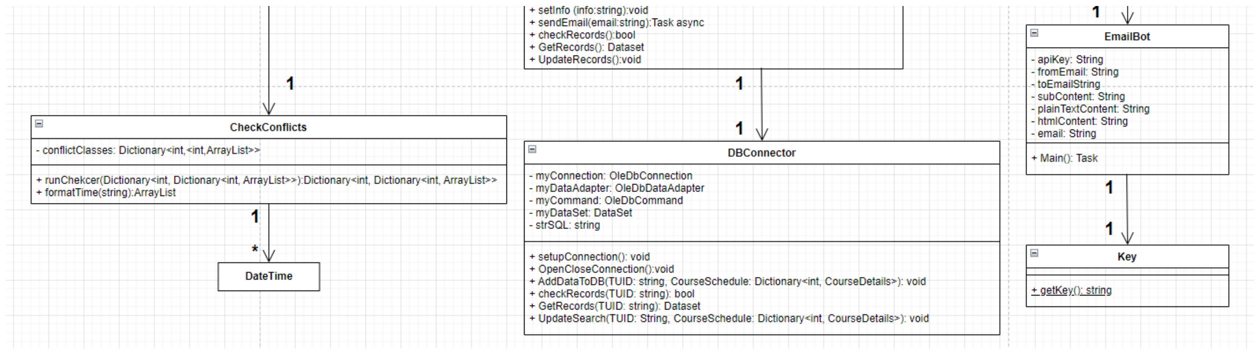
Part-1


frmMenu
- worker: Worker
- CourseSchedule: Dictionary<int, CourseDetails>
- searchResult: String
- ratingResult: String
- i: int
- emailList: String[]
- btnSearch_Click(sender: object, e: EventArgs): void
- btnSend_Click(sender: object, e: EventArgs): async void
- badInput(ctrl: Control): bool
- disableControl(ctrl: Control): void
- enableControl(ctrl: Control): void
- btnSearchAgain_Click(sender: object, e: EventArgs): void
- btnEnterID_Click(sender: object, e: EventArgs): void
- frmMenu_Load(sender: object, e: EventArgs):void)
CourseDetails
- courseName: string
- courseCode: string
- courseDescription: string
- courseProfessor: string
- professorRating: string
- courseTime: string
- courseSection: string
- courseDays: string
- courseCredits: string
- courseConflicts: string
setters and getters methods for all the variables
*
1
1
1
Worker

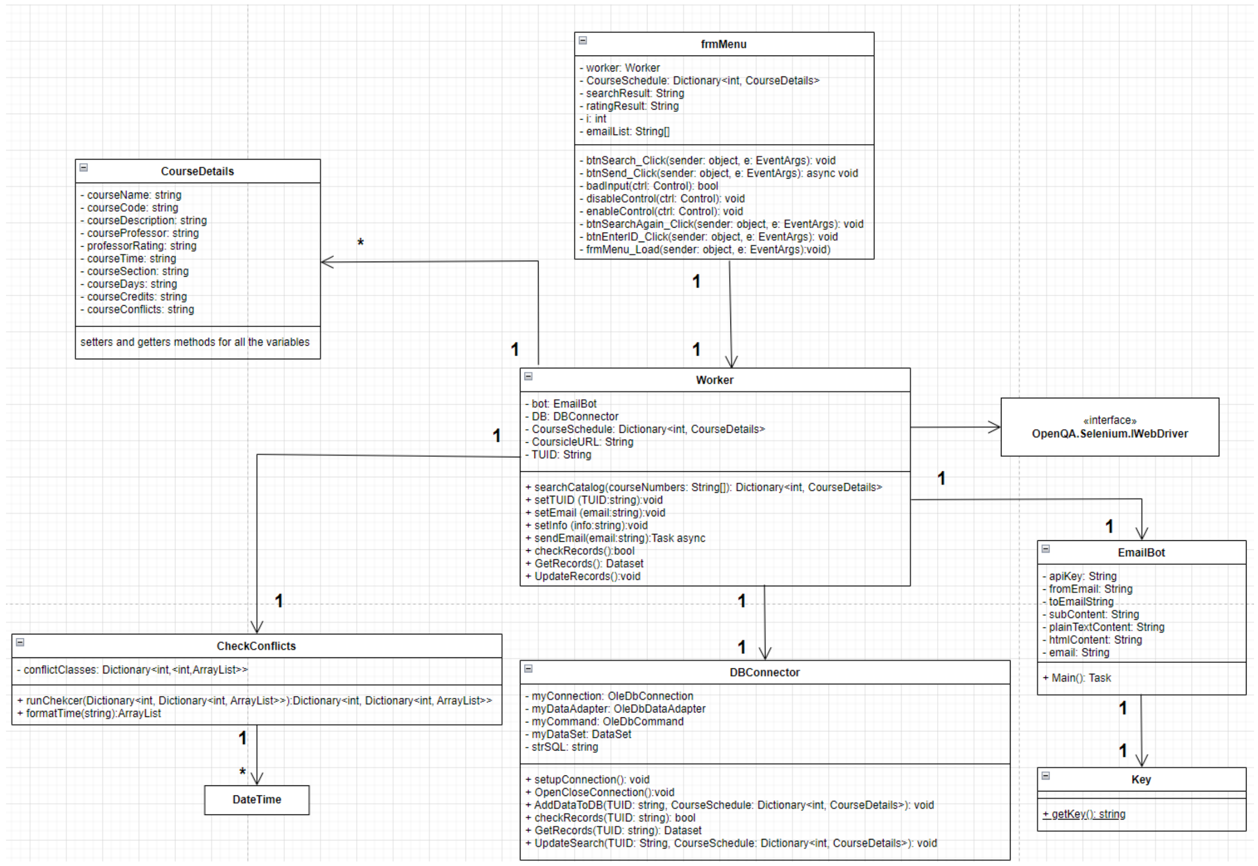
Part-2


ds for all the variables
1
1
Worker
- bot: EmailBot
- DB: DBConnector
- CourseSchedule: Dictionary<int, CourseDetails>
- CoursicleURL: String
- TUID: String
+ searchCatalog(courseNumbers: String[]): Dictionary<int, CourseDetails>
+ setTUID (TUID:string):void
+ setEmail (email:string):void
+ setInfo (info:string):void
+ sendEmail(email:string):Task async
+ checkRecords():bool
+ GetRecords(): Dataset
+ UpdateRecords():void
1
«interface»
OpenQA.Selenium.IWebDriver
1
1
EmailBot
- apiKey: String
- fromEmail: String

Part-3


+ setInfo (info:string):void
+ sendEmail(email:string):Task async
+ checkRecords():bool
+ GetRecords(): Dataset
+ UpdateRecords():void
CheckConflicts
- conflictClasses: Dictionary<int,<int,ArrayList>>
+ runChekcer(Dictionary<int, Dictionary<int, ArrayList>>):Dictionary<int, Dictionary<int, ArrayList>>
+ formatTime(string):ArrayList
DateTime
DBConnector
- myConnection: OleDbConnection
- myDataAdapter: OleDbDataAdapter
- myCommand: OleDbCommand
- myDataSet: DataSet
- strSQL: string
+ setupConnection(): void
+ OpenCloseConnection():void
+ AddDataToDB(TUID: string, CourseSchedule: Dictionary<int, CourseDetails>): void
+ checkRecords(TUID: string): bool
+ GetRecords(TUID: string): Dataset
+ UpdateSearch(TUID: String, CourseSchedule: Dictionary<int, CourseDetails>): void
EmailBot
- apiKey: String
- fromEmail: String
- toEmailString
- subContent: String
- plainTextContent: String
- htmlContent: String
- email: String
+ Main(): Task
Key
+ getKey(): string
1
*

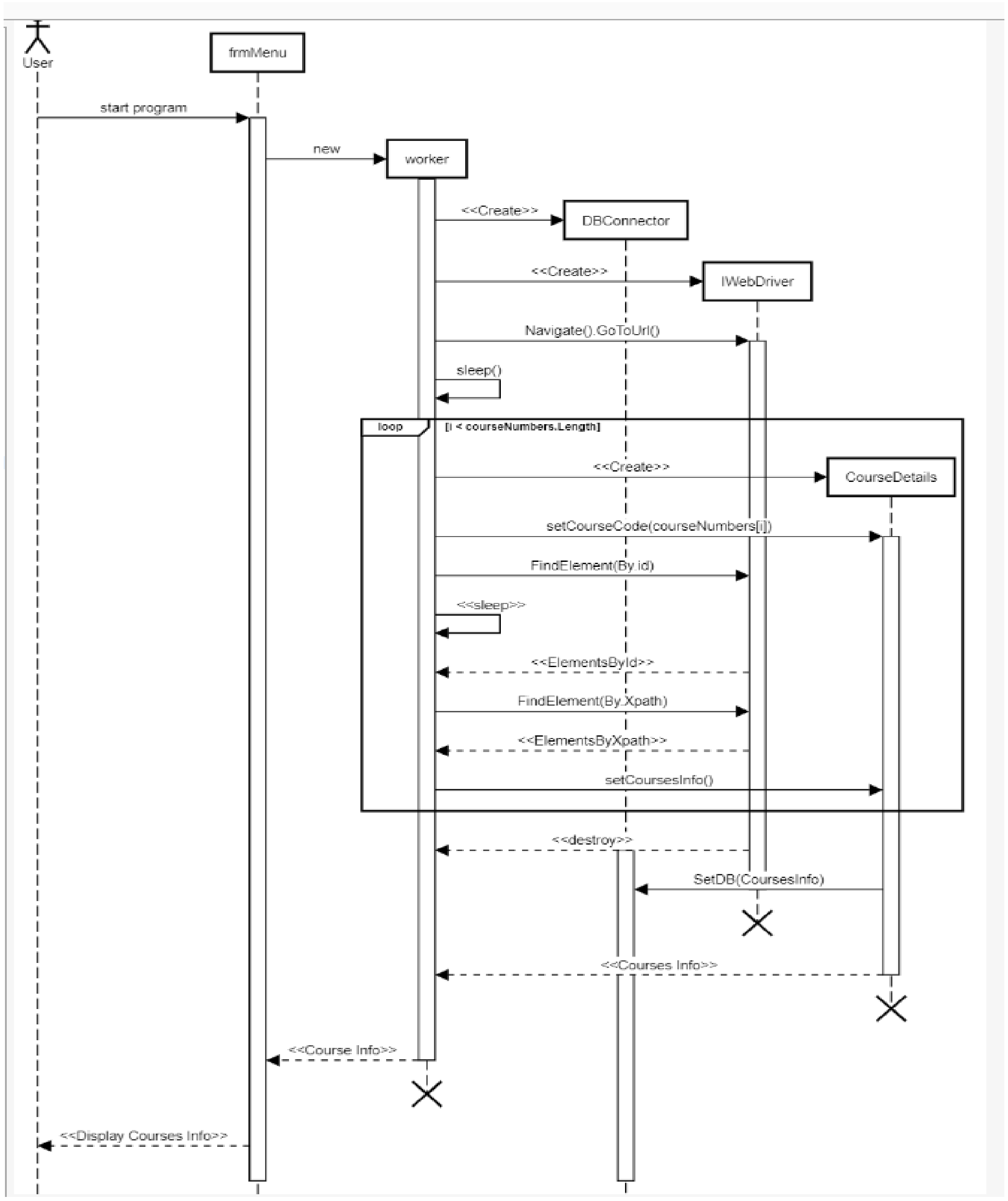
Full Image


frmMenu
- worker: Worker
- CourseSchedule: Dictionary<int, CourseDetails>
- searchResult: String
- ratingResult: String
- i: int
- emailList: String[]
- btnSearch_Click(sender: object, e: EventArgs): void
- btnSend_Click(sender: object, e: EventArgs): async void
- badInput(ctrl: Control): bool
- disableControl(ctrl: Control): void
- enableControl(ctrl: Control): void
- btnSearchAgain_Click(sender: object, e: EventArgs): void
- btnEnterID_Click(sender: object, e: EventArgs): void
- frmMenu_Load(sender: object, e: EventArgs):void)
CourseDetails
- courseName: string
- courseCode: string
- courseDescription: string
- courseProfessor: string
- professorRating: string
- courseTime: string
- courseSection: string
- courseDays: string
- courseCredits: string
- courseConflicts: string
setters and getters methods for all the variables
Worker
- bot: EmailBot
- DB: DBConnector
- CourseSchedule: Dictionary<int, CourseDetails>
- CoursicleURL: String
- TUID: String
+ searchCatalog(courseNumbers: String[]): Dictionary<int, CourseDetails>
+ setTUID (TUID:string):void
+ setEmail (email:string):void
+ setInfo (info:string):void
+ sendEmail(email:string):Task async
+ checkRecords():bool
+ GetRecords(): Dataset
+ UpdateRecords():void
«interface»
OpenQA.Selenium.IWebDriver
EmailBot
- apiKey: String
- fromEmail: String
- toEmailString
- subContent: String
- plainTextContent: String
- htmlContent: String
- email: String
+ Main(): Task
Key
+ getKey(): string
CheckConflicts
- conflictClasses: Dictionary<int,<int,ArrayList>>
+ runChekcer(Dictionary<int, Dictionary<int, ArrayList>>):Dictionary<int, Dictionary<int, ArrayList>>
+ formatTime(string):ArrayList
DateTime
DBConnector
- myConnection: OleDbConnection
- myDataAdapter: OleDbDataAdapter
- myCommand: OleDbCommand
- myDataSet: DataSet
- strSQL: string
+ setupConnection(): void
+ OpenCloseConnection():void
+ AddDataToDB(TUID: string, CourseSchedule: Dictionary<int, CourseDetails>): void
+ checkRecords(TUID: string): bool
+ GetRecords(TUID: string): Dataset
+ UpdateSearch(TUID: String, CourseSchedule: Dictionary<int, CourseDetails>): void
1
*

The above UML Class Diagram is representing how our program runs and the relationship between our different classes. At first, the frmMenu class (Which is essentially the GUI class) is going to call the Worker class. The worker class will handle the web scrapping(By using the webdriver which is the instance of the Selenium package) and will set all the fields in the CourseDetails class. The Worker class will then give all the course details to the DBConnector class in order to store the information in the database. Finally, the worker class will check for any Conflicts between the classes searched by the user, and it will also (if necessary) call the emailConnector class and give all the necessary information to it, where the email connector will handle sending an email to the user.

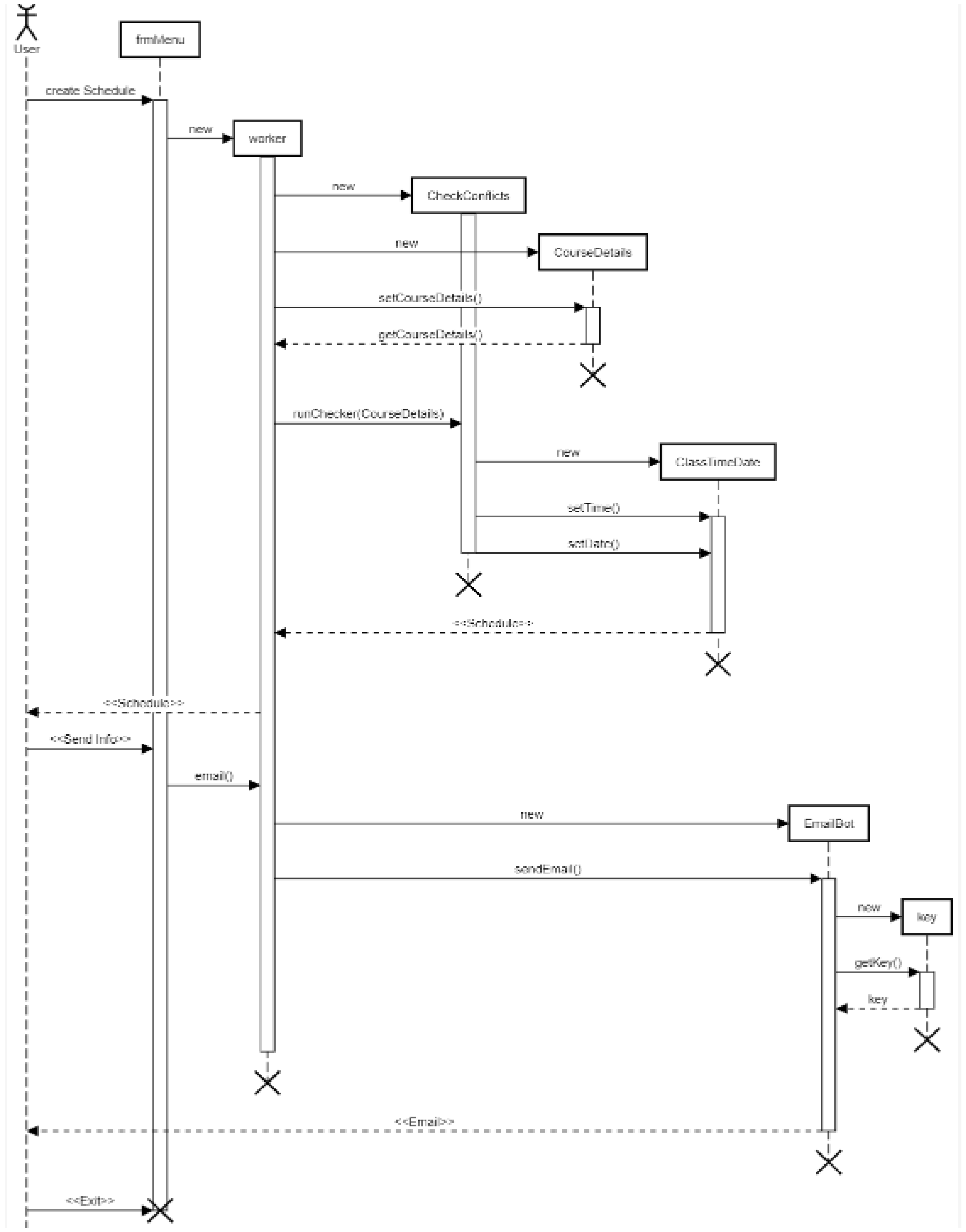
Sequence Diagrams

Sequence Diagram 1: Search Classes



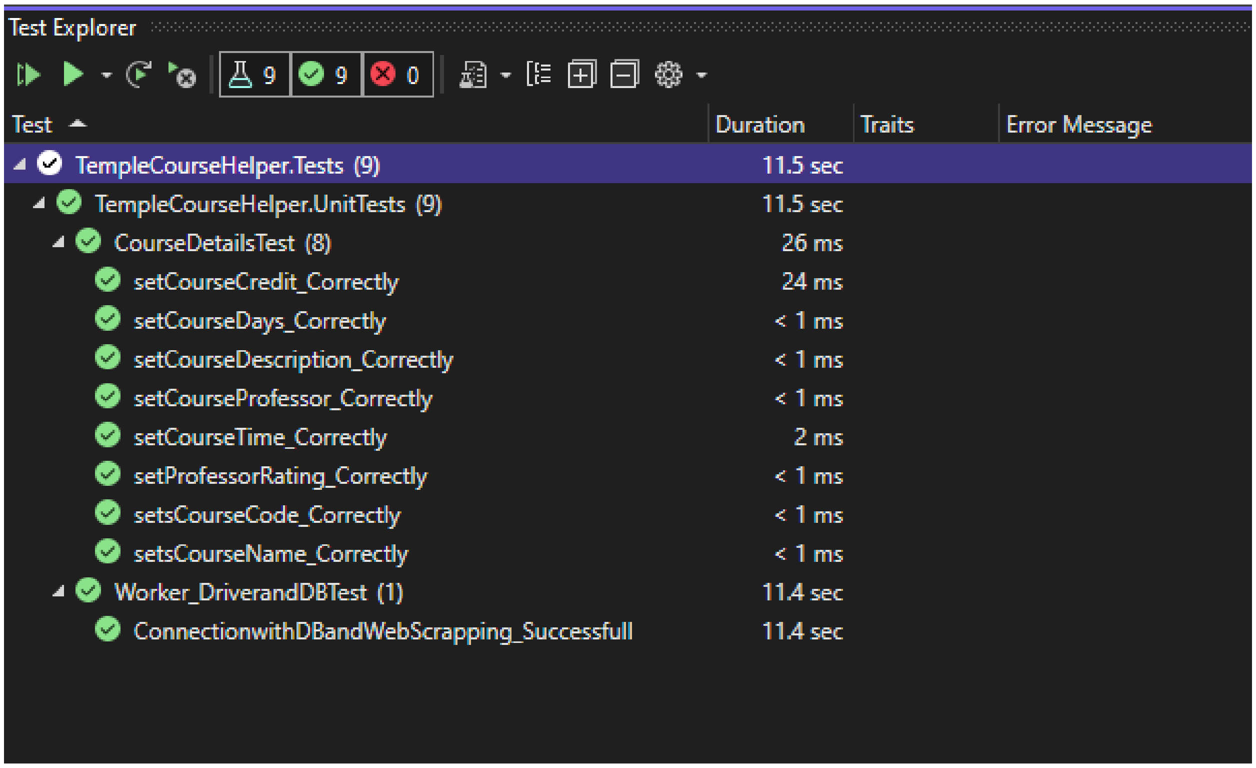
As soon as the User starts the program, the frmMenu class (which is the GUI class) is created. Then, a worker object of the class is created which will then create objects of the DBConnector class and the IWebDriver. As the IWebDriver is handling the jobs received by the worker, the main program will have to sleep for some time until the whole progress is done. Once each course (and its specific info) is taken, it will be set in the CourseDetails class. When the loop is over and all courses are set, the IWebDriver is not necessary anymore and it will be destroyed. The CourseDetails object will then store all the info for all the courses inside the database for the specific user, and it can be accessed easily next time. Finally, the information for the different courses is sent back to the user by going through the worker and the GUI. After this, the CourseDeatils and Worker objects are destroyed.

Sequence Diagram 2: Create Schedule and Send Email



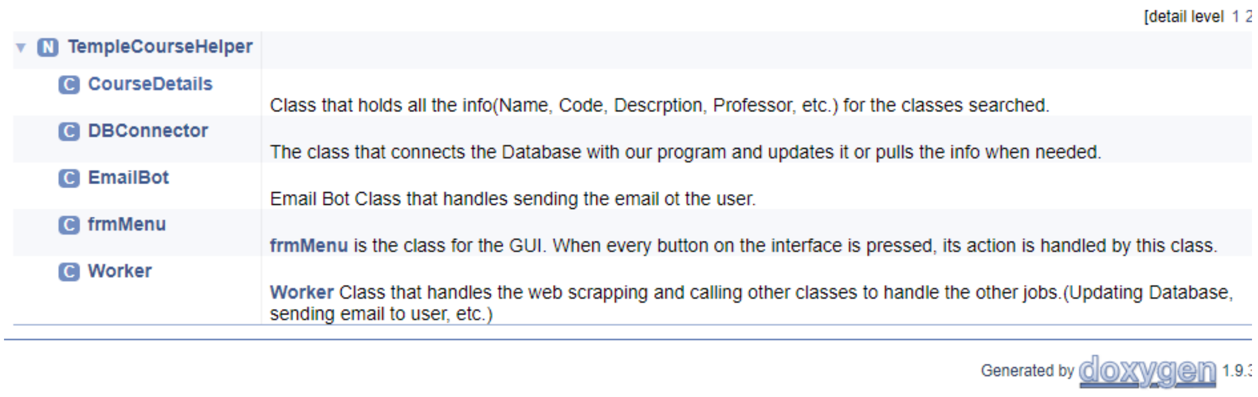
Our second sequence diagram is showing how our CourseConflicts and EmailBot classes work while the user is trying to create a certain plan with various courses and after that, send that information to his/her email. At the beginning (and after all the steps from Diagram 1 are complete), the user decides to create his/her schedule and so the frmMenu is creating the worker object again which then creates the CheckConflicts and CourseDetails objects. The CourseDetails object is sending all the information to the worker and the worker is passing that info inside the runChecker method from the CheckConflicts class. That method will create another object of the ClassTimeDate class which will handle all the conflicts between the different classes. When all the conflicts are handled, the results are returned to the user, passing by the worker object. Finally, if the user requests to receive an email, the worker will initialize the EmailBot object, which will handle sending the email to the provided email address after getting the key from the Key class. The program exits when the user decides to exit.

Automated Testing Report



Test	Duration	Traits	Error Message
✓ TempleCourseHelper.Tests (9)	11.5 sec		
✓ TempleCourseHelper.UnitTests (9)	11.5 sec		
✓ CourseDetailsTest (8)	26 ms		
✓ setCourseCredit_Correctly	24 ms		
✓ setCourseDays_Correctly	< 1 ms		
✓ setCourseDescription_Correctly	< 1 ms		
✓ setCourseProfessor_Correctly	< 1 ms		
✓ setCourseTime_Correctly	2 ms		
✓ setProfessorRating_Correctly	< 1 ms		
✓ setsCourseCode_Correctly	< 1 ms		
✓ setsCourseName_Correctly	< 1 ms		
✓ Worker_DriverandDBTest (1)	11.4 sec		
✓ ConnectionwithDBandWebScrapping_Successfull	11.4 sec		

Detailed Design API

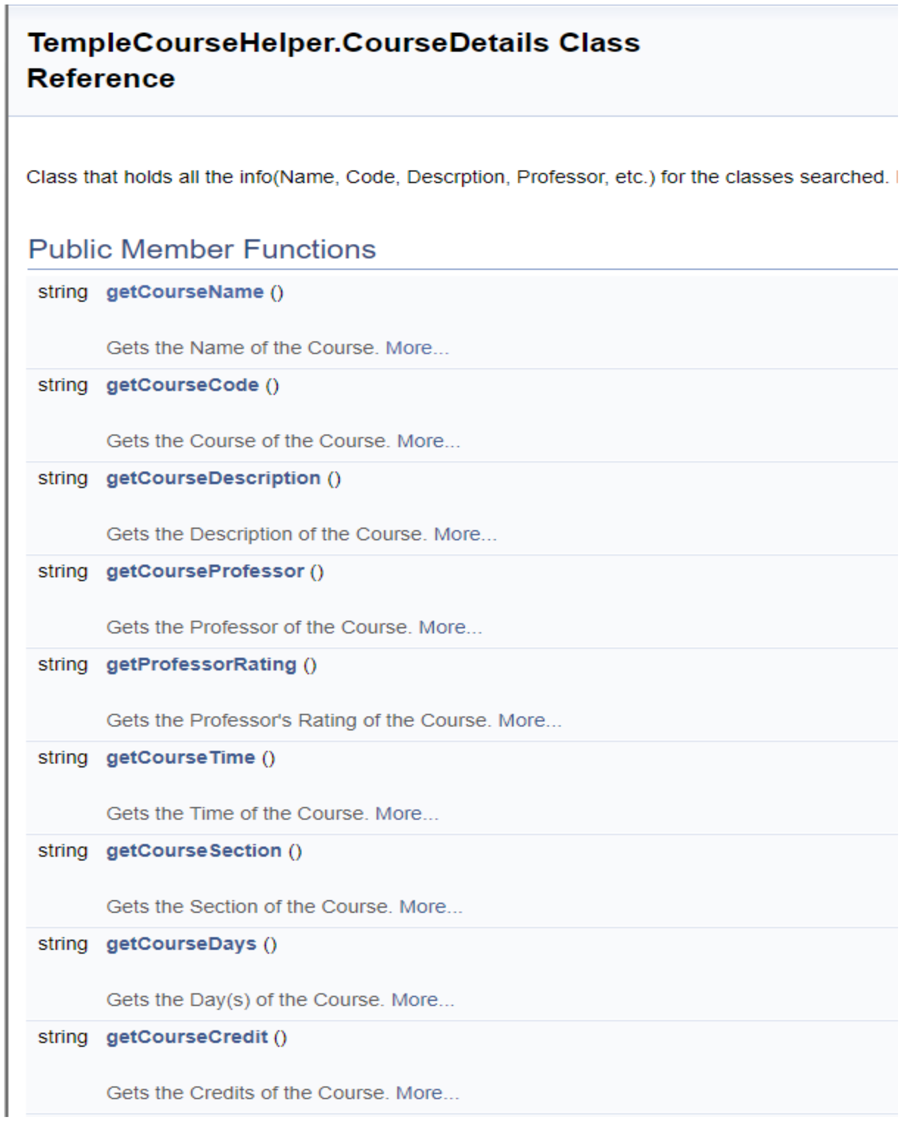


[detail level 1 2

▼ N TempleCourseHelper	
C CourseDetails	Class that holds all the info(Name, Code, Descrption, Professor, etc.) for the classes searched.
C DBConnector	The class that connects the Database with our program and updates it or pulls the info when needed.
C EmailBot	Email Bot Class that handles sending the email ot the user.
C frmMenu	frmMenu is the class for the GUI. When every button on the interface is pressed, its action is handled by this class.
C Worker	Worker Class that handles the web scrapping and calling other classes to handle the other jobs.(Updating Database, sending email to user, etc.)

Generated by doxygen 1.9.

CourseDetails Class

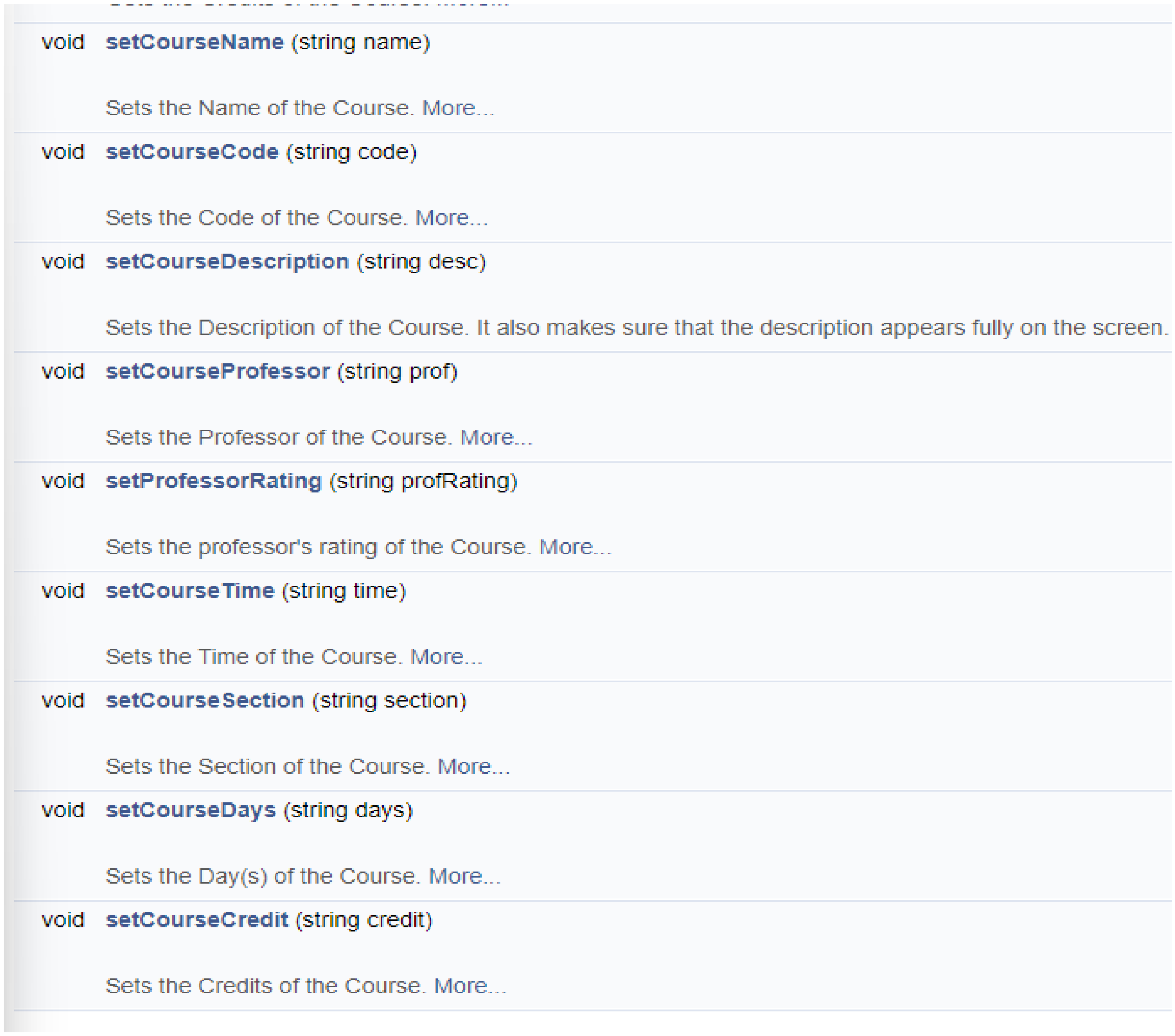


TempleCourseHelper.CourseDetails Class Reference

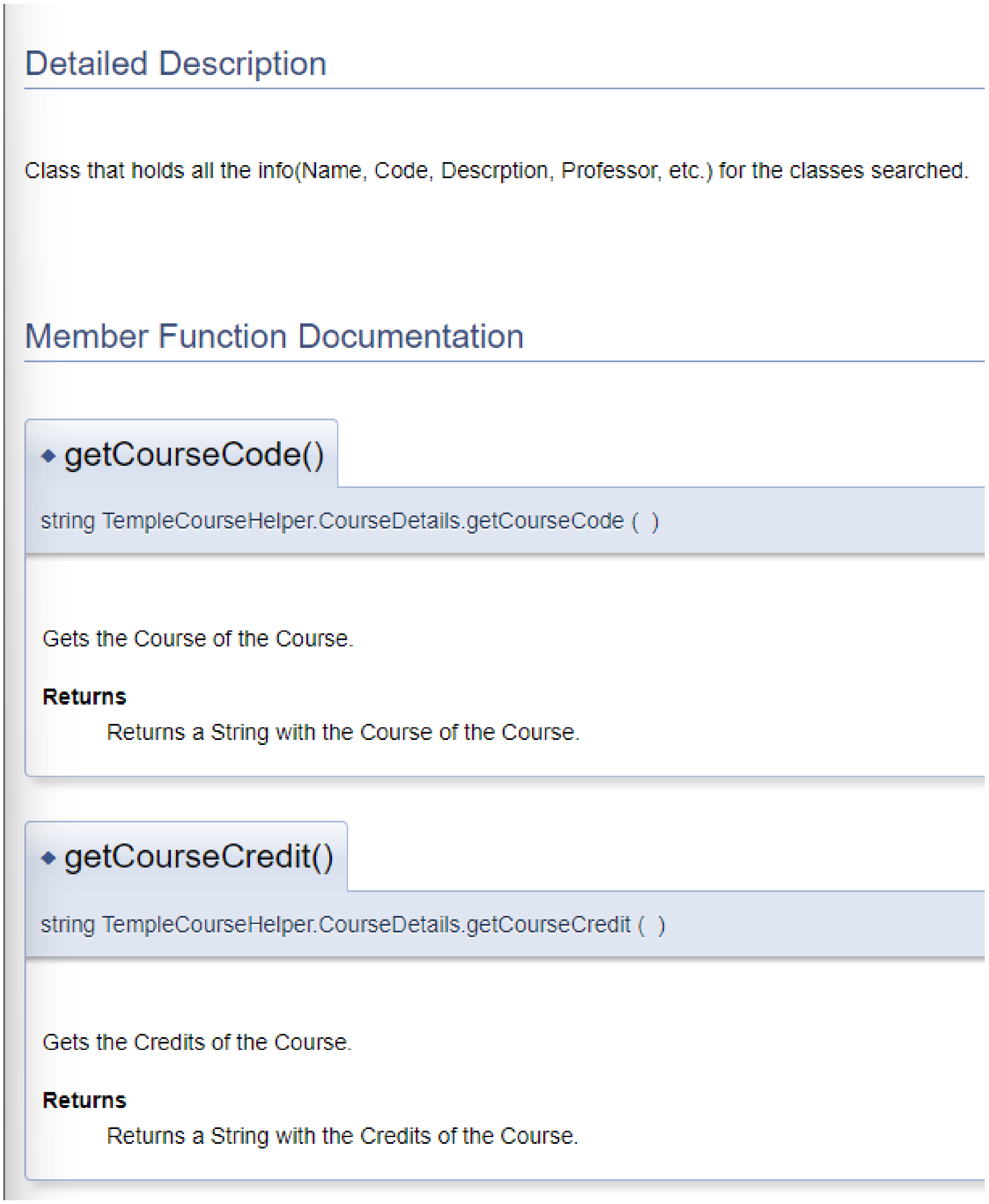
Class that holds all the info(Name, Code, Descrption, Professor, etc.) for the classes searched.

Public Member Functions

string	getCourseName ()
	Gets the Name of the Course. More...
string	getCourseCode ()
	Gets the Course of the Course. More...
string	getCourseDescription ()
	Gets the Description of the Course. More...
string	getCourseProfessor ()
	Gets the Professor of the Course. More...
string	getProfessorRating ()
	Gets the Professor's Rating of the Course. More...
string	getCourseTime ()
	Gets the Time of the Course. More...
string	getCourseSection ()
	Gets the Section of the Course. More...
string	getCourseDays ()
	Gets the Day(s) of the Course. More...
string	getCourseCredit ()
	Gets the Credits of the Course. More...



void	**setCourseName** (string name)
	Sets the Name of the Course. More...
void	**setCourseCode** (string code)
	Sets the Code of the Course. More...
void	**setCourseDescription** (string desc)
	Sets the Description of the Course. It also makes sure that the description appears fully on the screen.
void	**setCourseProfessor** (string prof)
	Sets the Professor of the Course. More...
void	**setProfessorRating** (string profRating)
	Sets the professor's rating of the Course. More...
void	**setCourseTime** (string time)
	Sets the Time of the Course. More...
void	**setCourseSection** (string section)
	Sets the Section of the Course. More...
void	**setCourseDays** (string days)
	Sets the Day(s) of the Course. More...
void	**setCourseCredit** (string credit)
	Sets the Credits of the Course. More...



Detailed Description

Class that holds all the info(Name, Code, Descrption, Professor, etc.) for the classes searched.

Member Function Documentation

◆ getCourseCode()

string TempleCourseHelper.CourseDetails.getCourseCode ( )

Gets the Course of the Course.

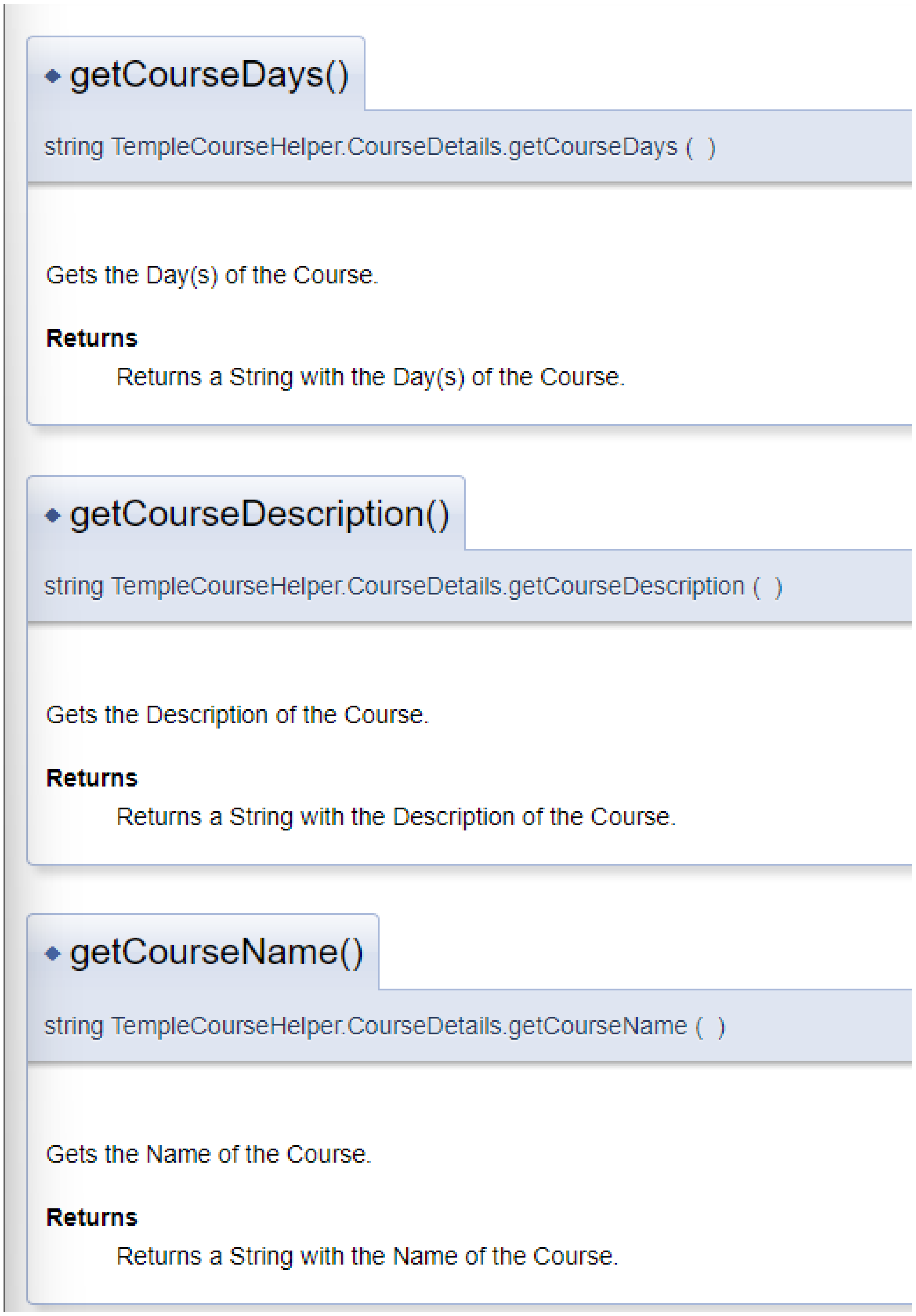
Returns
Returns a String with the Course of the Course.

◆ getCourseCredit()

string TempleCourseHelper.CourseDetails.getCourseCredit ( )

Gets the Credits of the Course.

Returns
Returns a String with the Credits of the Course.



◆ getCourseDays()

string TempleCourseHelper.CourseDetails.getCourseDays ( )

Gets the Day(s) of the Course.

Returns
Returns a String with the Day(s) of the Course.

◆ getCourseDescription()

string TempleCourseHelper.CourseDetails.getCourseDescription ( )

Gets the Description of the Course.

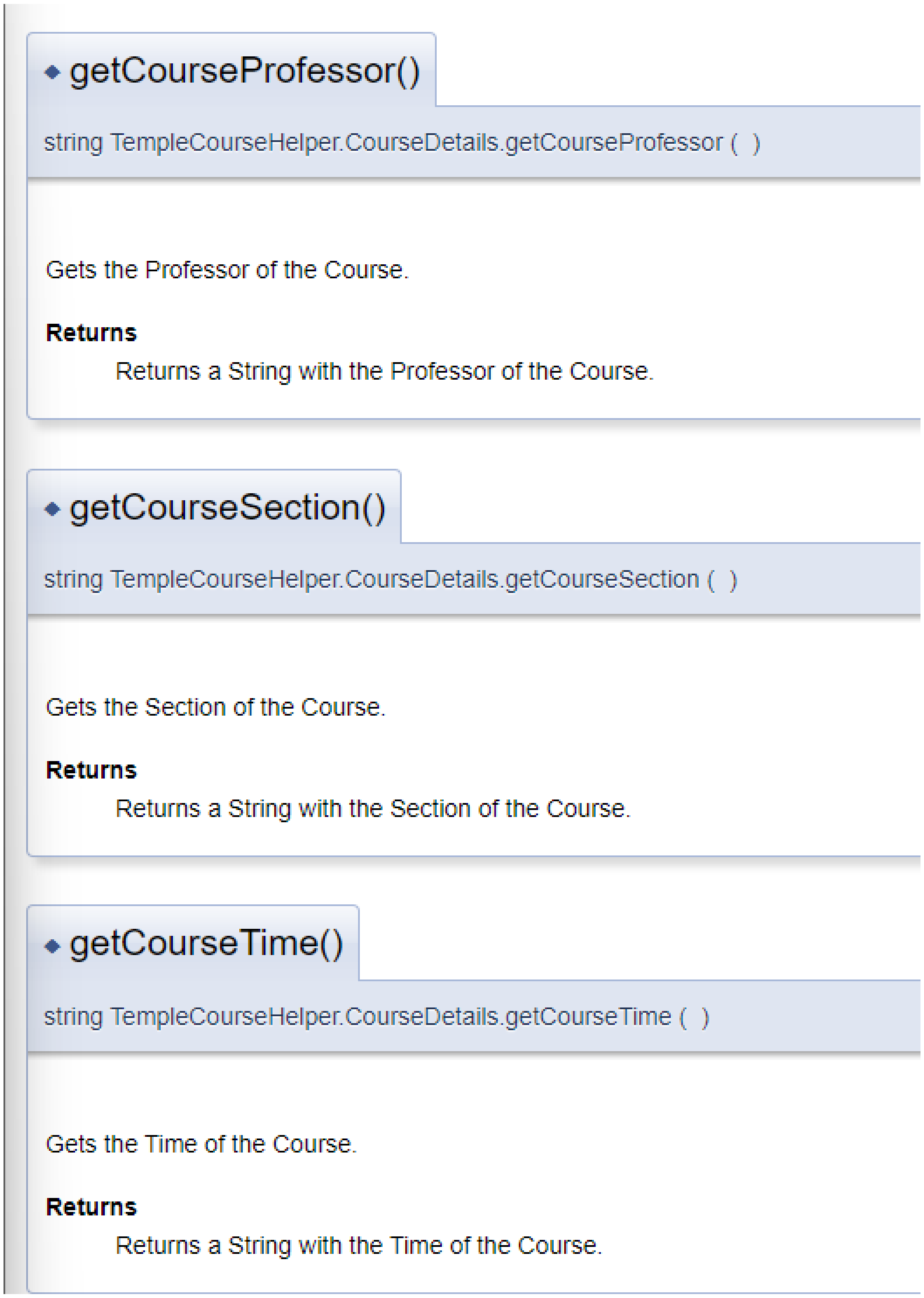
Returns
Returns a String with the Description of the Course.

◆ getCourseName()

string TempleCourseHelper.CourseDetails.getCourseName ( )

Gets the Name of the Course.

Returns
Returns a String with the Name of the Course.



◆ getCourseProfessor()

string TempleCourseHelper.CourseDetails.getCourseProfessor ( )

Gets the Professor of the Course.

Returns

Returns a String with the Professor of the Course.

◆ getCourseSection()

string TempleCourseHelper.CourseDetails.getCourseSection ( )

Gets the Section of the Course.

Returns

Returns a String with the Section of the Course.

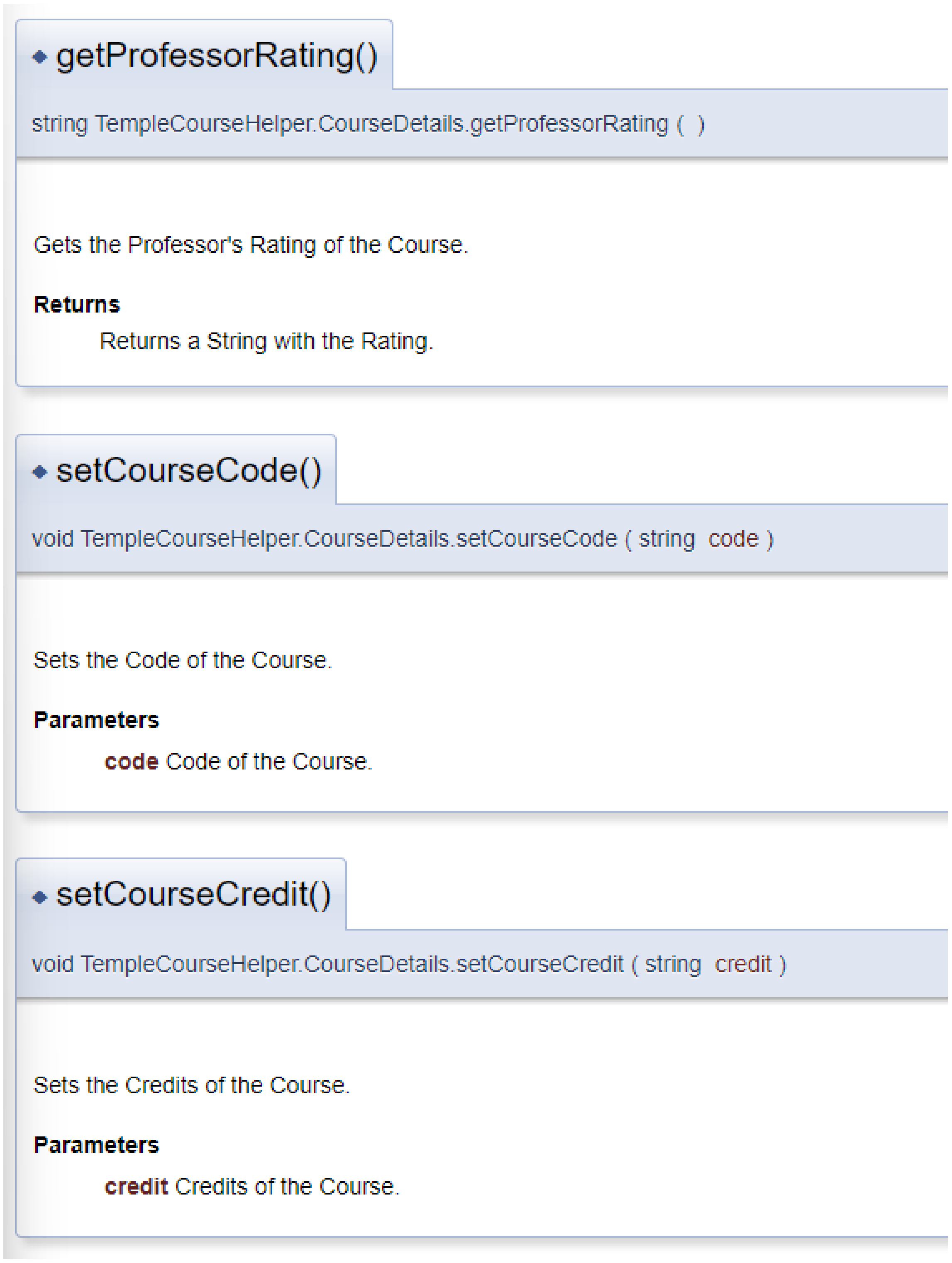
◆ getCourseTime()

string TempleCourseHelper.CourseDetails.getCourseTime ( )

Gets the Time of the Course.

Returns

Returns a String with the Time of the Course.



◆ getProfessorRating()

string TempleCourseHelper.CourseDetails.getProfessorRating ( )

Gets the Professor's Rating of the Course.

Returns

Returns a String with the Rating.

◆ setCourseCode()

void TempleCourseHelper.CourseDetails.setCourseCode (string code)

Sets the Code of the Course.

Parameters

code Code of the Course.

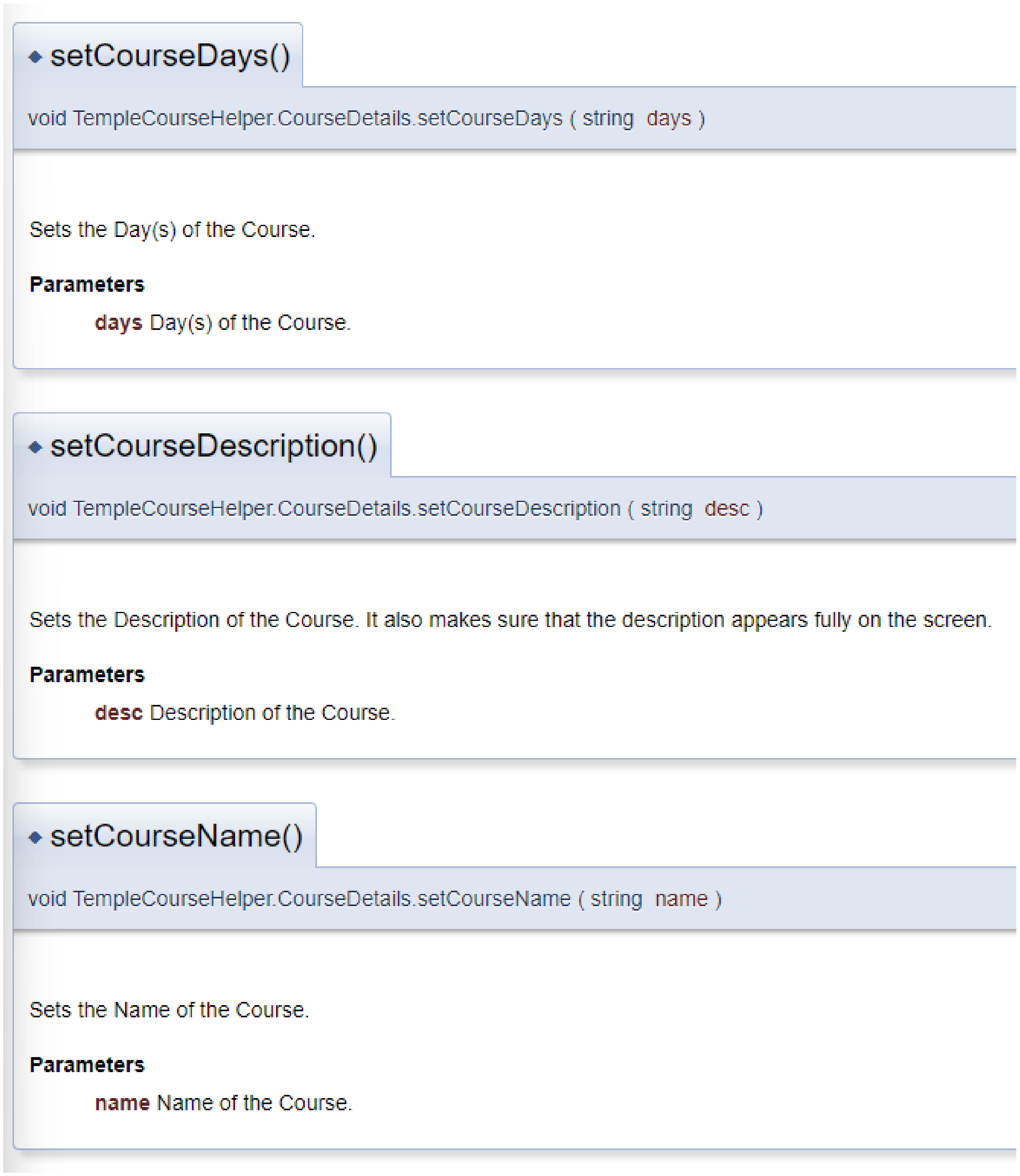
◆ setCourseCredit()

void TempleCourseHelper.CourseDetails.setCourseCredit (string credit)

Sets the Credits of the Course.

Parameters

credit Credits of the Course.



◆ setCourseDays()

void TempleCourseHelper.CourseDetails.setCourseDays (string days)

Sets the Day(s) of the Course.

Parameters

days Day(s) of the Course.

◆ setCourseDescription()

void TempleCourseHelper.CourseDetails.setCourseDescription (string desc)

Sets the Description of the Course. It also makes sure that the description appears fully on the screen.

Parameters

desc Description of the Course.

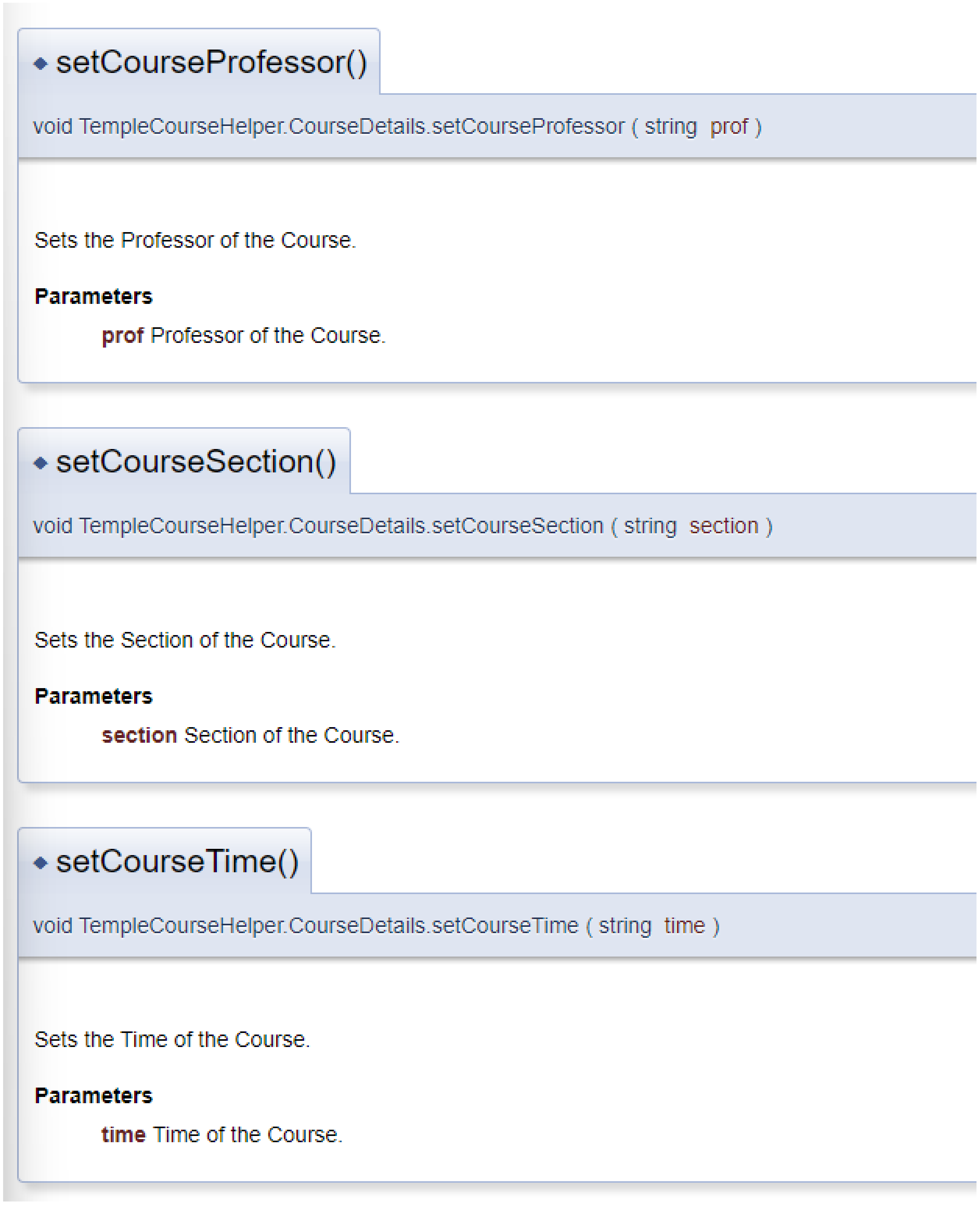
◆ setCourseName()

void TempleCourseHelper.CourseDetails.setCourseName (string name)

Sets the Name of the Course.

Parameters

name Name of the Course.



◆ setCourseProfessor()

void TempleCourseHelper.CourseDetails.setCourseProfessor (string prof)

Sets the Professor of the Course.

Parameters

prof Professor of the Course.

◆ setCourseSection()

void TempleCourseHelper.CourseDetails.setCourseSection (string section)

Sets the Section of the Course.

Parameters

section Section of the Course.

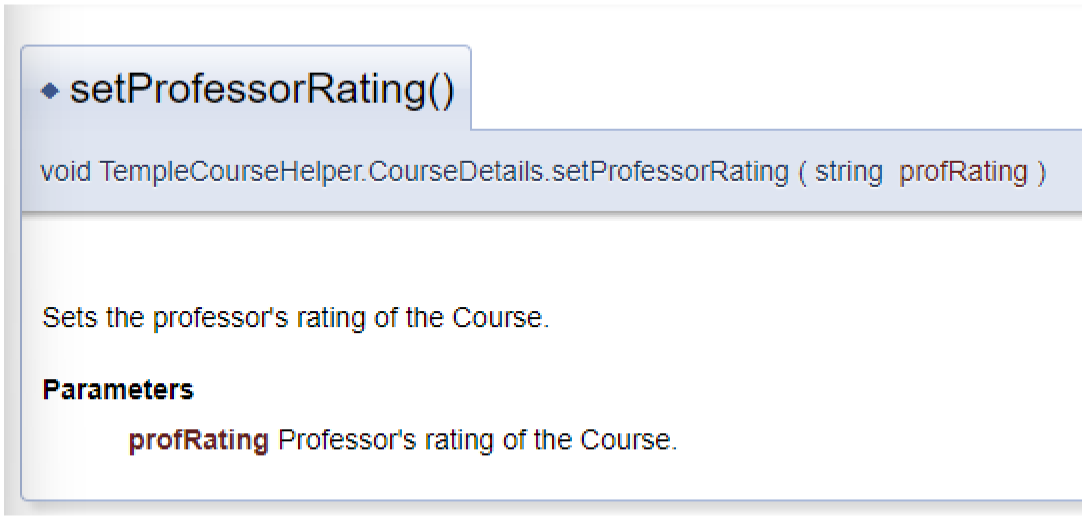
◆ setCourseTime()

void TempleCourseHelper.CourseDetails.setCourseTime (string time)

Sets the Time of the Course.

Parameters

time Time of the Course.



◆ setProfessorRating()

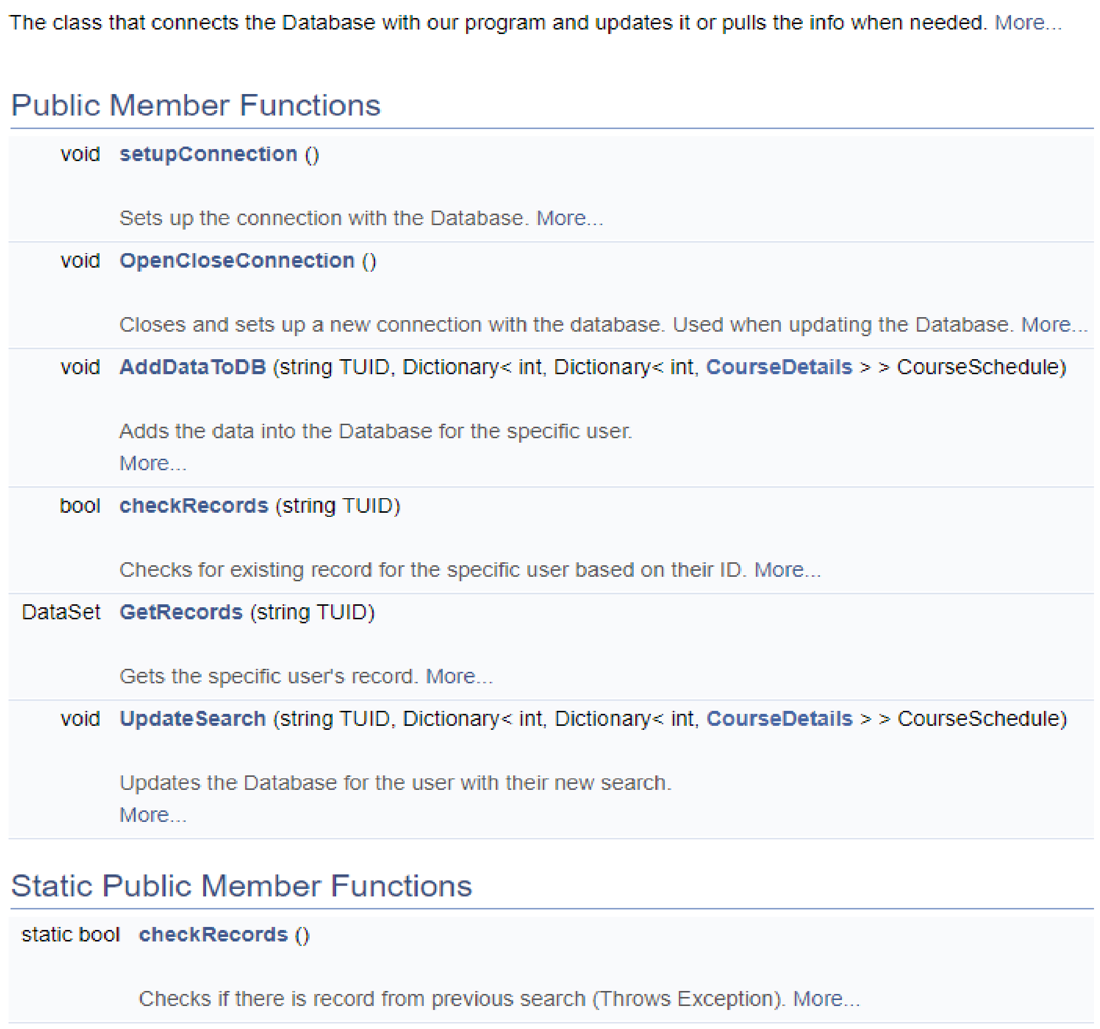
void TempleCourseHelper.CourseDetails.setProfessorRating (string profRating)

Sets the professor's rating of the Course.

Parameters

profRating Professor's rating of the Course.

DBConnector Class



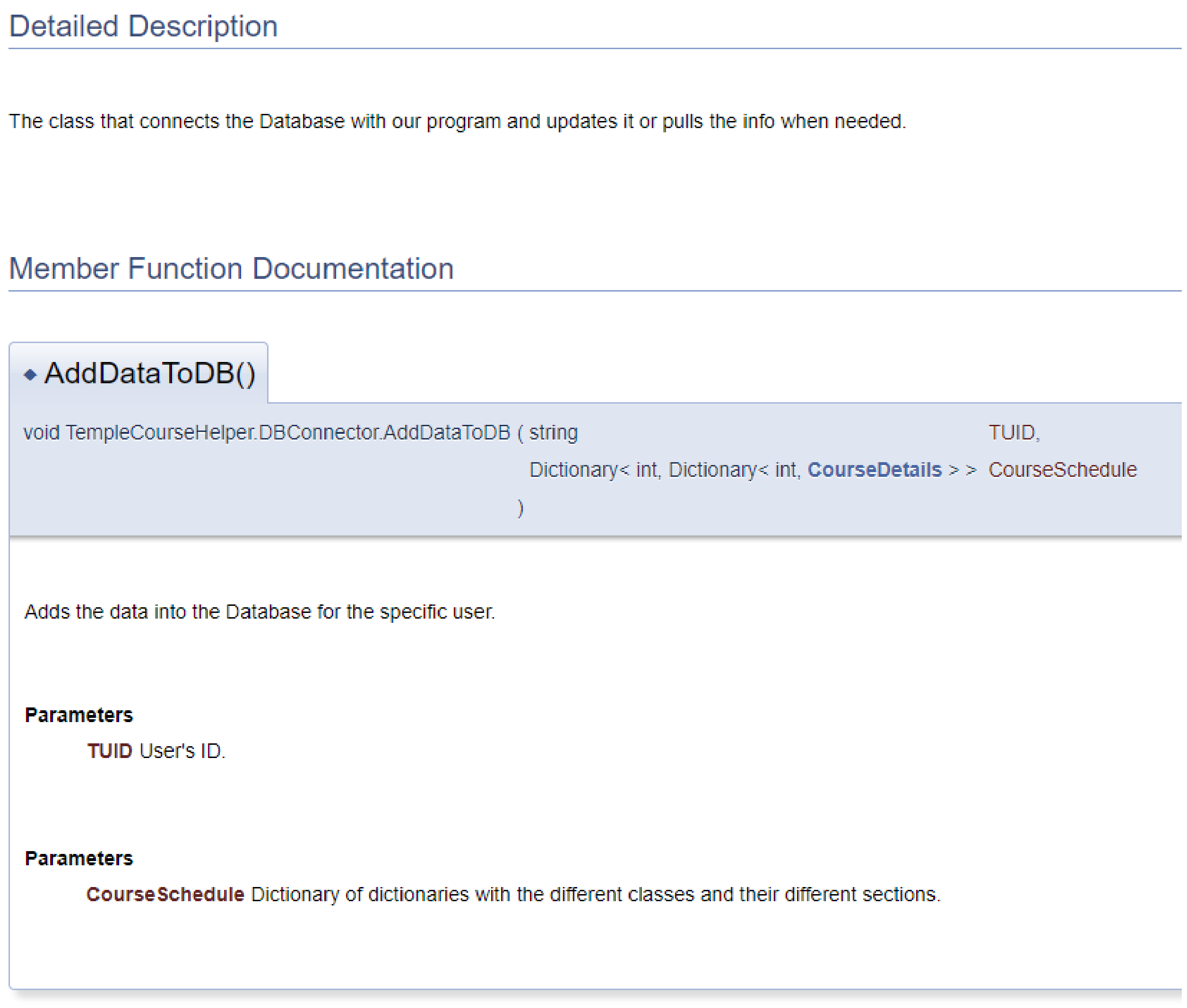
The class that connects the Database with our program and updates it or pulls the info when needed. More...

Public Member Functions

void	**setupConnection** ()
	Sets up the connection with the Database. More...
void	**OpenCloseConnection** ()
	Closes and sets up a new connection with the database. Used when updating the Database. More...
void	**AddDataToDB** (string TUID, Dictionary< int, Dictionary< int, **CourseDetails** > > CourseSchedule)
	Adds the data into the Database for the specific user. More...
bool	**checkRecords** (string TUID)
	Checks for existing record for the specific user based on their ID. More...
DataSet	**GetRecords** (string TUID)
	Gets the specific user's record. More...
void	**UpdateSearch** (string TUID, Dictionary< int, Dictionary< int, **CourseDetails** > > CourseSchedule)
	Updates the Database for the user with their new search. More...

Static Public Member Functions

static bool	**checkRecords** ()
	Checks if there is record from previous search (Throws Exception). More...



Detailed Description

The class that connects the Database with our program and updates it or pulls the info when needed.

Member Function Documentation

◆ AddDataToDB()

```
void TempleCourseHelper.DBConnector.AddDataToDB ( string                                                          TUID,
                                                  Dictionary< int, Dictionary< int, CourseDetails > >  CourseSchedule
                                                )
```

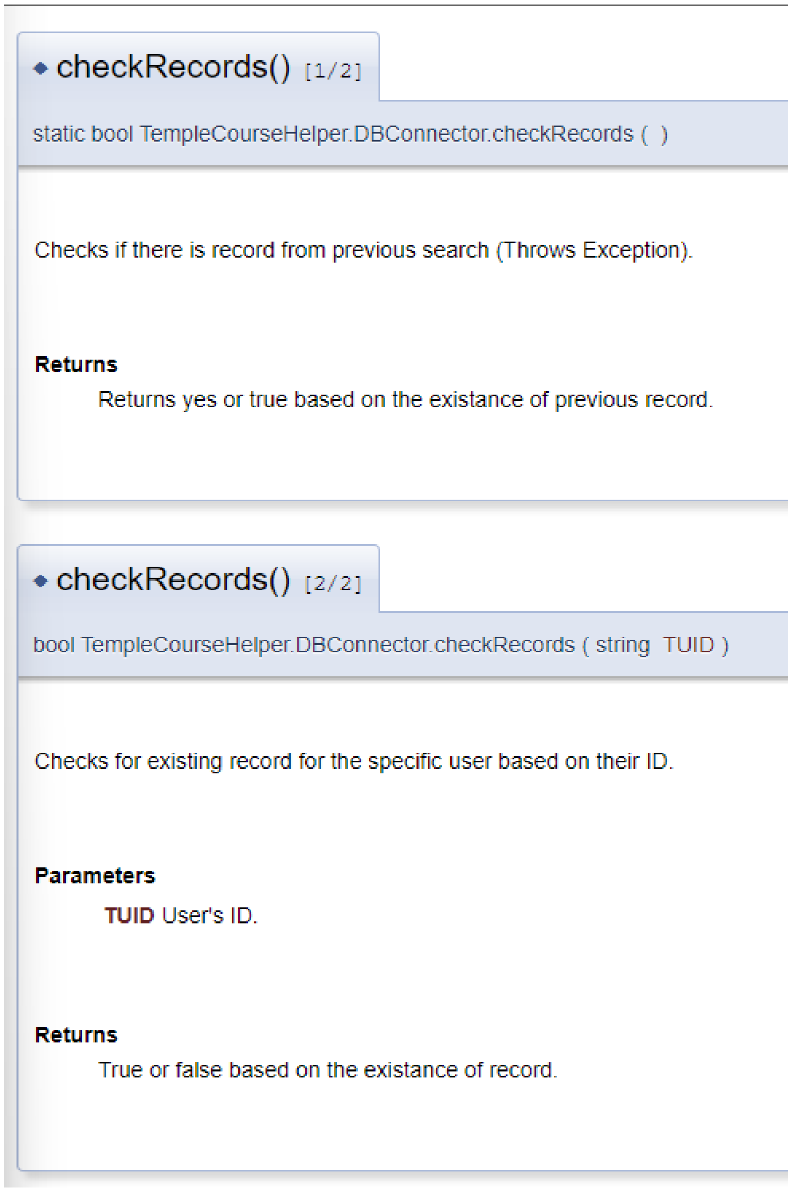
Adds the data into the Database for the specific user.

Parameters

TUID User's ID.

Parameters

CourseSchedule Dictionary of dictionaries with the different classes and their different sections.



◆ checkRecords() [1/2]

static bool TempleCourseHelper.DBConnector.checkRecords ( )

Checks if there is record from previous search (Throws Exception).

Returns

Returns yes or true based on the existance of previous record.

◆ checkRecords() [2/2]

bool TempleCourseHelper.DBConnector.checkRecords (string TUID)

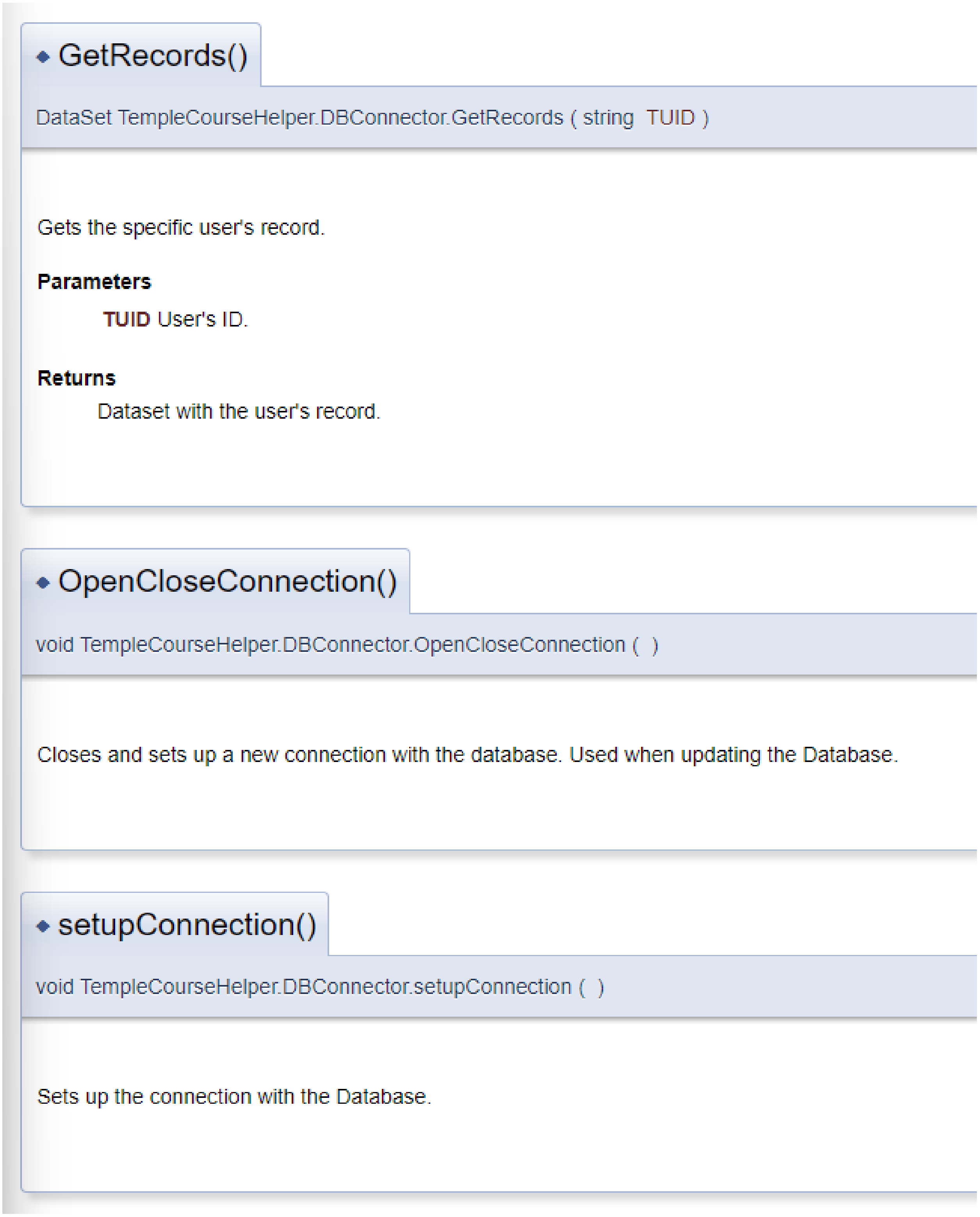
Checks for existing record for the specific user based on their ID.

Parameters

TUID User's ID.

Returns

True or false based on the existance of record.



◆ GetRecords()

DataSet TempleCourseHelper.DBConnector.GetRecords (string TUID)

Gets the specific user's record.

Parameters

TUID User's ID.

Returns

Dataset with the user's record.

◆ OpenCloseConnection()

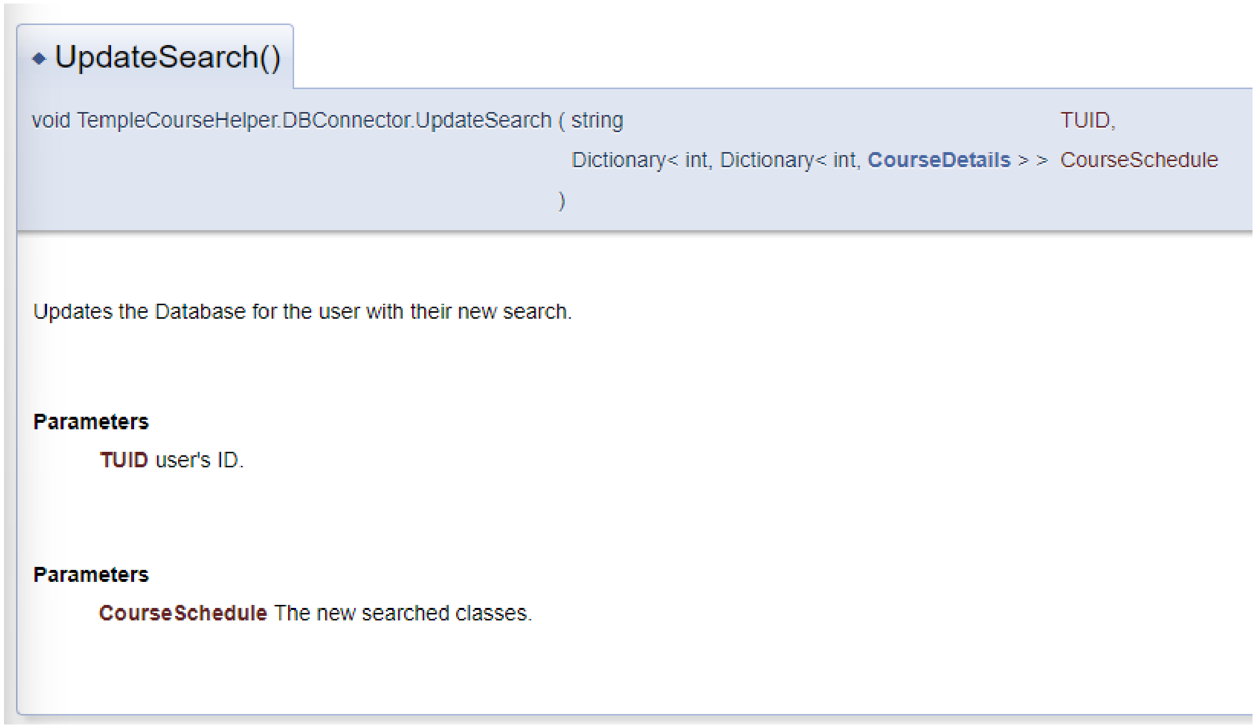
void TempleCourseHelper.DBConnector.OpenCloseConnection ( )

Closes and sets up a new connection with the database. Used when updating the Database.

◆ setupConnection()

void TempleCourseHelper.DBConnector.setupConnection ( )

Sets up the connection with the Database.



◆ UpdateSearch()

```
void TempleCourseHelper.DBConnector.UpdateSearch ( string                                              TUID,
                                                   Dictionary< int, Dictionary< int, CourseDetails > >  CourseSchedule
                                                 )
```

Updates the Database for the user with their new search.

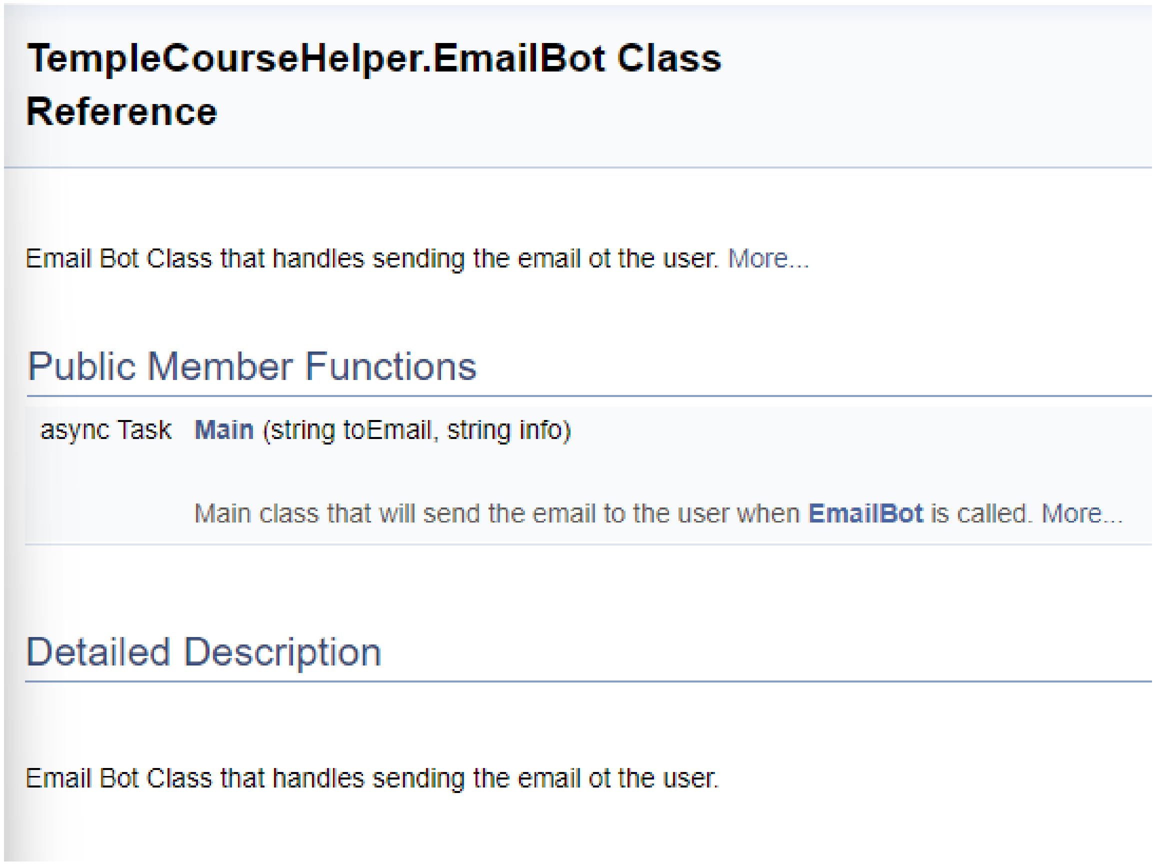
Parameters

TUID user's ID.

Parameters

CourseSchedule The new searched classes.

EmailBot Class



TempleCourseHelper.EmailBot Class Reference

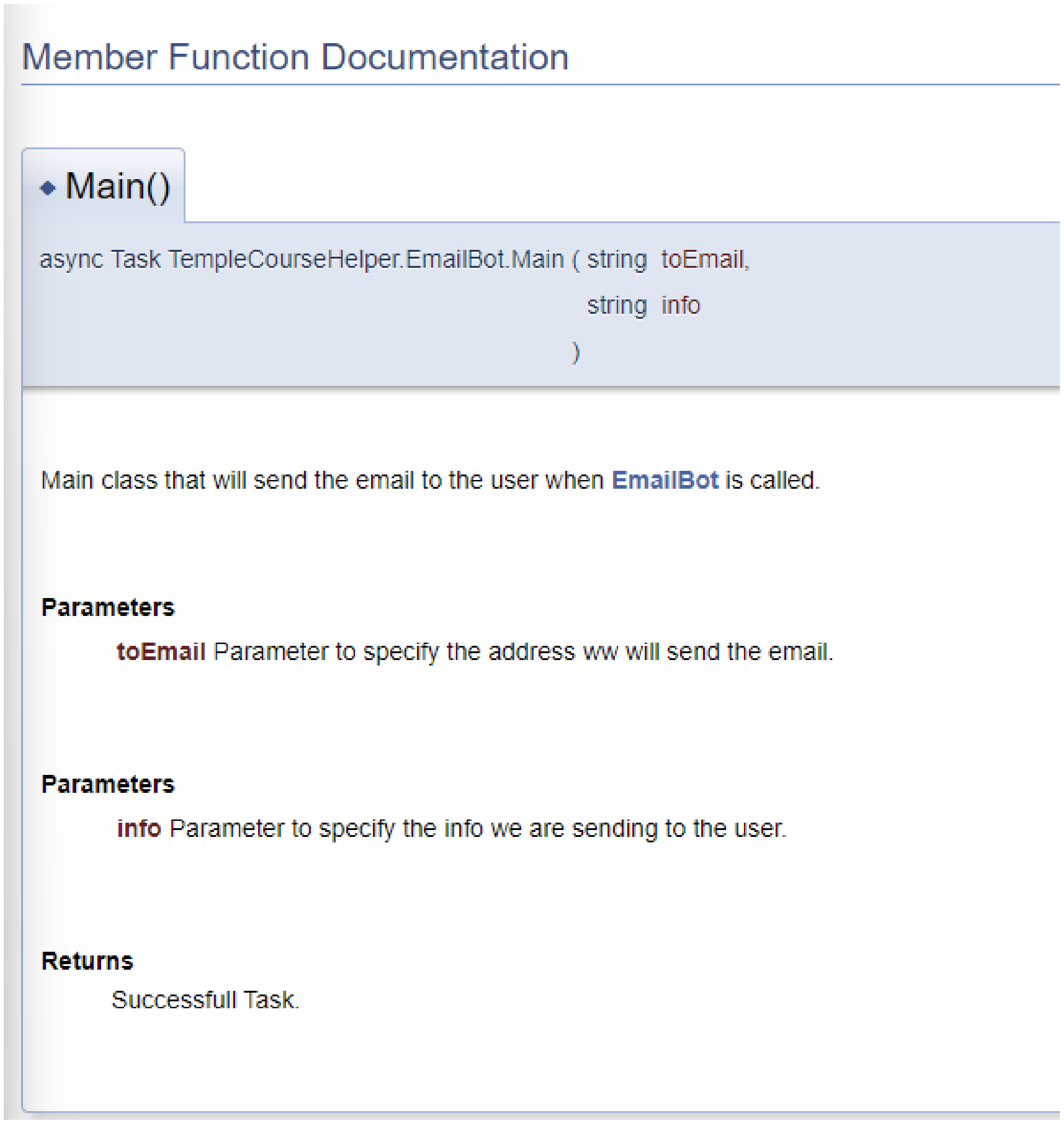
Email Bot Class that handles sending the email ot the user. More...

Public Member Functions

async Task	**Main** (string toEmail, string info)
	Main class that will send the email to the user when **EmailBot** is called. More...

Detailed Description

Email Bot Class that handles sending the email ot the user.



Member Function Documentation

◆ Main()

```
async Task TempleCourseHelper.EmailBot.Main ( string  toEmail,
                                              string  info
                                            )
```

Main class that will send the email to the user when **EmailBot** is called.

Parameters

toEmail Parameter to specify the address ww will send the email.

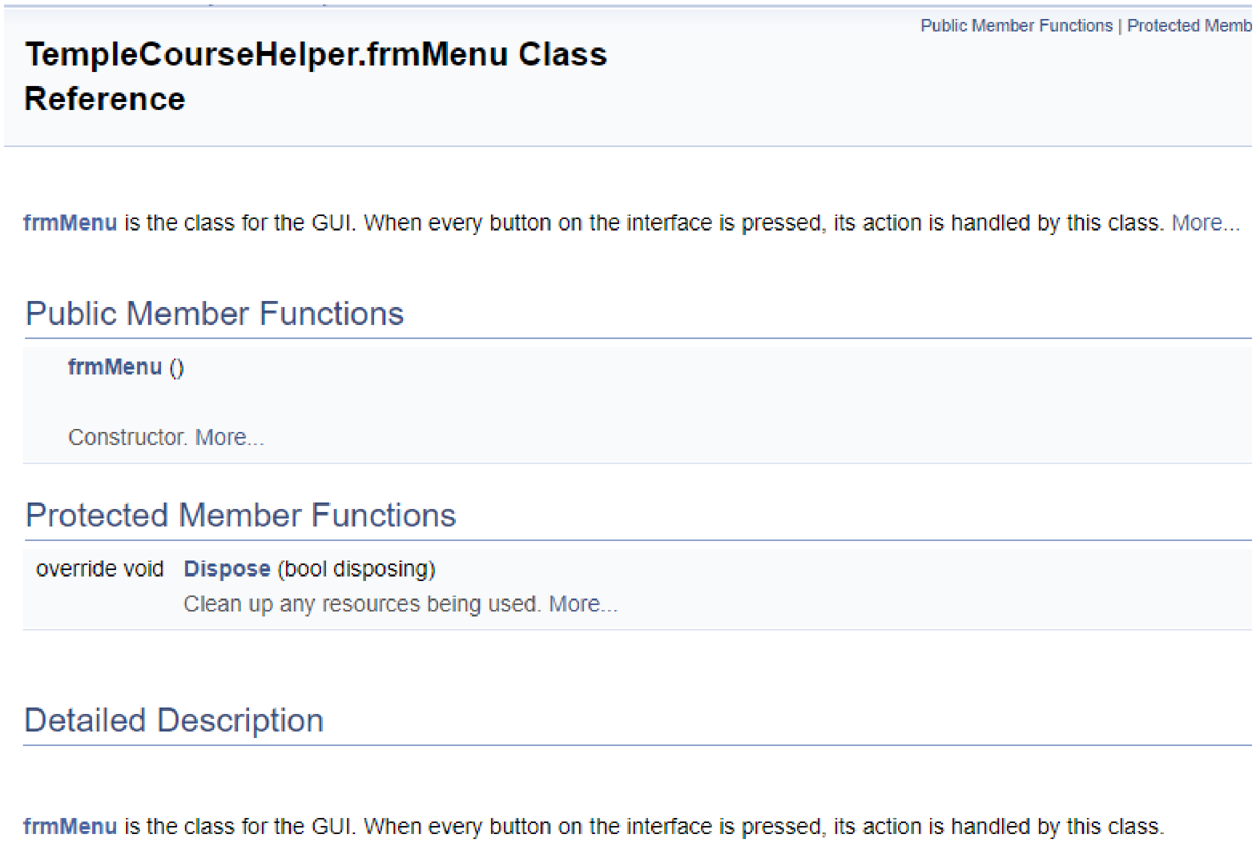
Parameters

info Parameter to specify the info we are sending to the user.

Returns

Successfull Task.

frmMenu Class



Public Member Functions | Protected Memb

TempleCourseHelper.frmMenu Class Reference

frmMenu is the class for the GUI. When every button on the interface is pressed, its action is handled by this class. More...

Public Member Functions

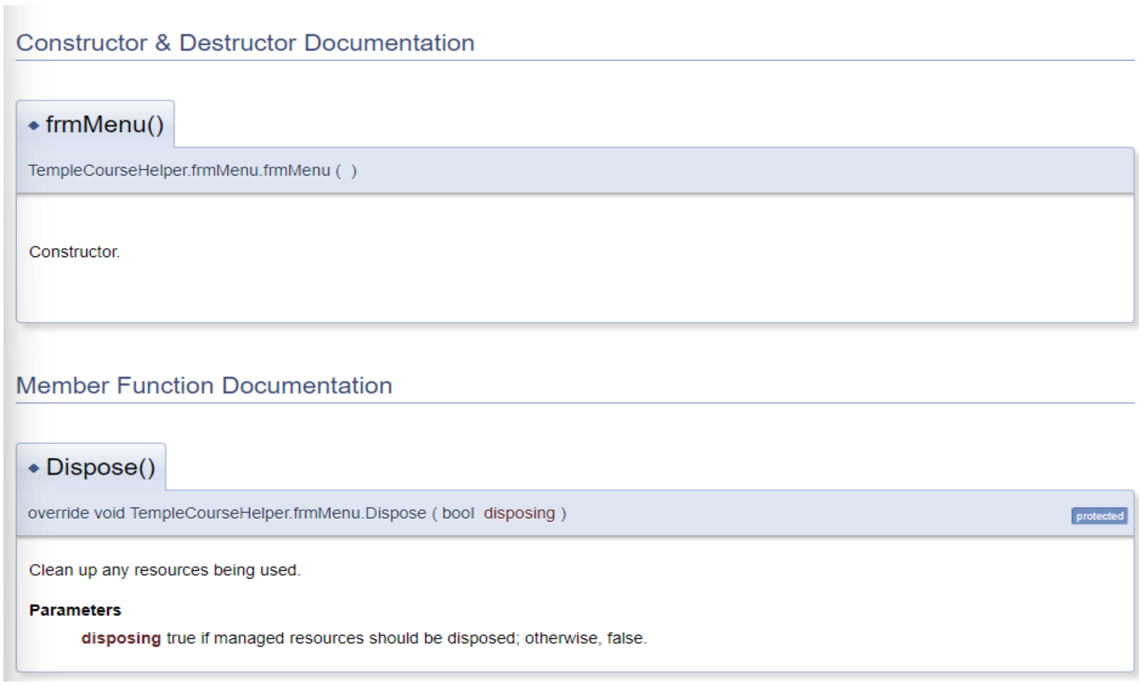
frmMenu ()	
Constructor. More...	

Protected Member Functions

override void	**Dispose** (bool disposing)
	Clean up any resources being used. More...

Detailed Description

frmMenu is the class for the GUI. When every button on the interface is pressed, its action is handled by this class.



Constructor & Destructor Documentation

◆ frmMenu()

```
TempleCourseHelper.frmMenu.frmMenu ( )
```

Constructor.

Member Function Documentation

◆ Dispose()

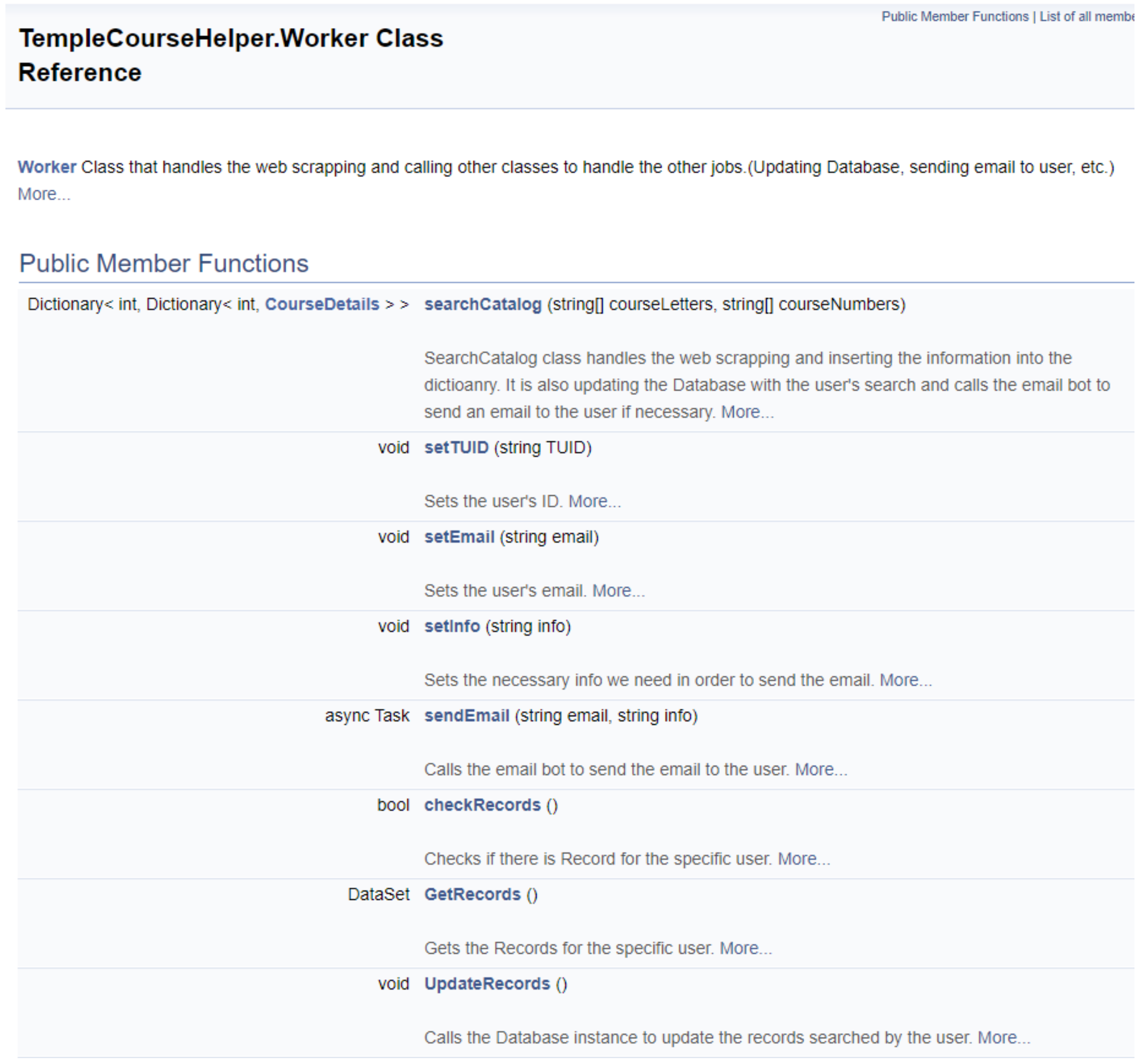
```
override void TempleCourseHelper.frmMenu.Dispose ( bool  disposing )
```
protected

Clean up any resources being used.

Parameters

disposing	true if managed resources should be disposed; otherwise, false.

WorkerClass



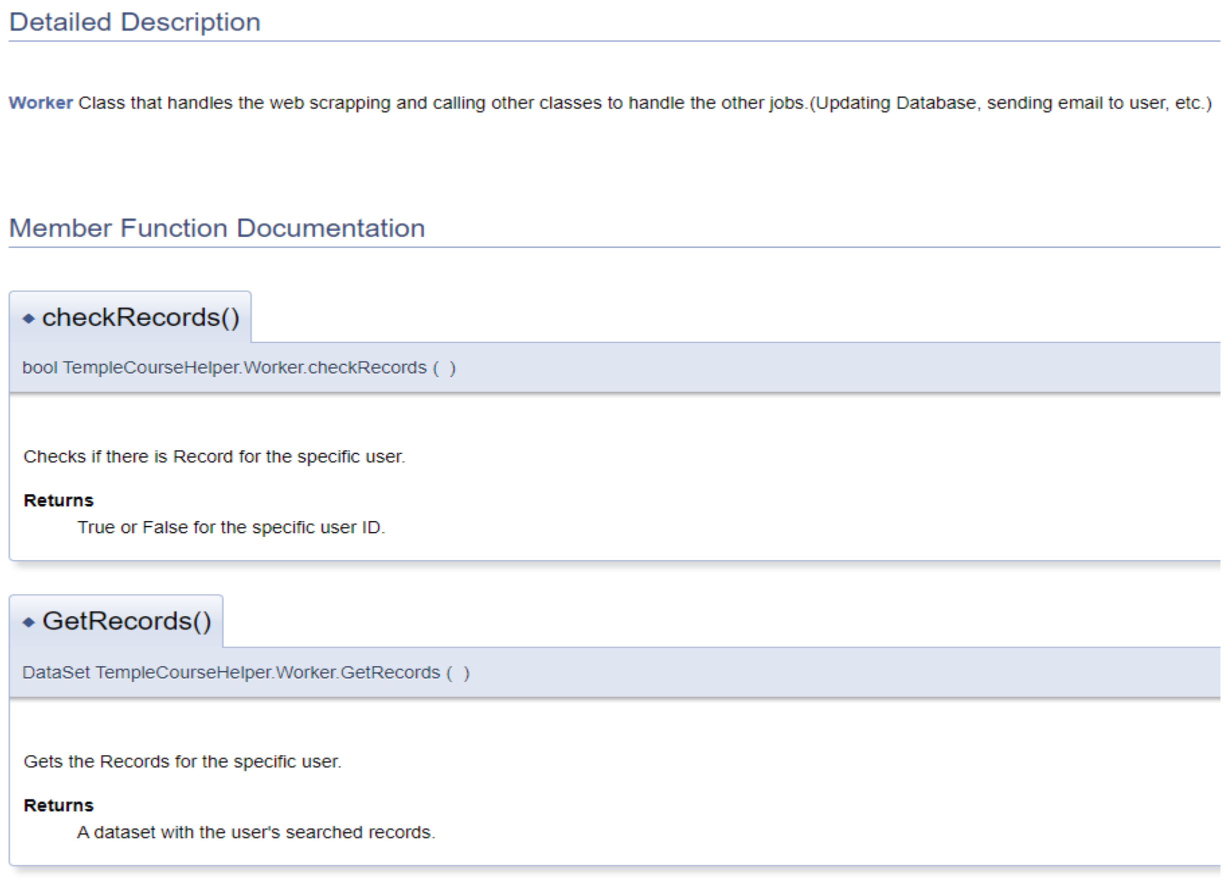
Public Member Functions | List of all membe

TempleCourseHelper.Worker Class Reference

Worker Class that handles the web scrapping and calling other classes to handle the other jobs.(Updating Database, sending email to user, etc.) More...

Public Member Functions

Dictionary< int, Dictionary< int, CourseDetails > >	searchCatalog (string[] courseLetters, string[] courseNumbers) SearchCatalog class handles the web scrapping and inserting the information into the dictioanry. It is also updating the Database with the user's search and calls the email bot to send an email to the user if necessary. More...
void	setTUID (string TUID) Sets the user's ID. More...
void	setEmail (string email) Sets the user's email. More...
void	setInfo (string info) Sets the necessary info we need in order to send the email. More...
async Task	sendEmail (string email, string info) Calls the email bot to send the email to the user. More...
bool	checkRecords () Checks if there is Record for the specific user. More...
DataSet	GetRecords () Gets the Records for the specific user. More...
void	UpdateRecords () Calls the Database instance to update the records searched by the user. More...



Detailed Description

Worker Class that handles the web scrapping and calling other classes to handle the other jobs.(Updating Database, sending email to user, etc.)

Member Function Documentation

◆ checkRecords()

```
bool TempleCourseHelper.Worker.checkRecords ( )
```

Checks if there is Record for the specific user.

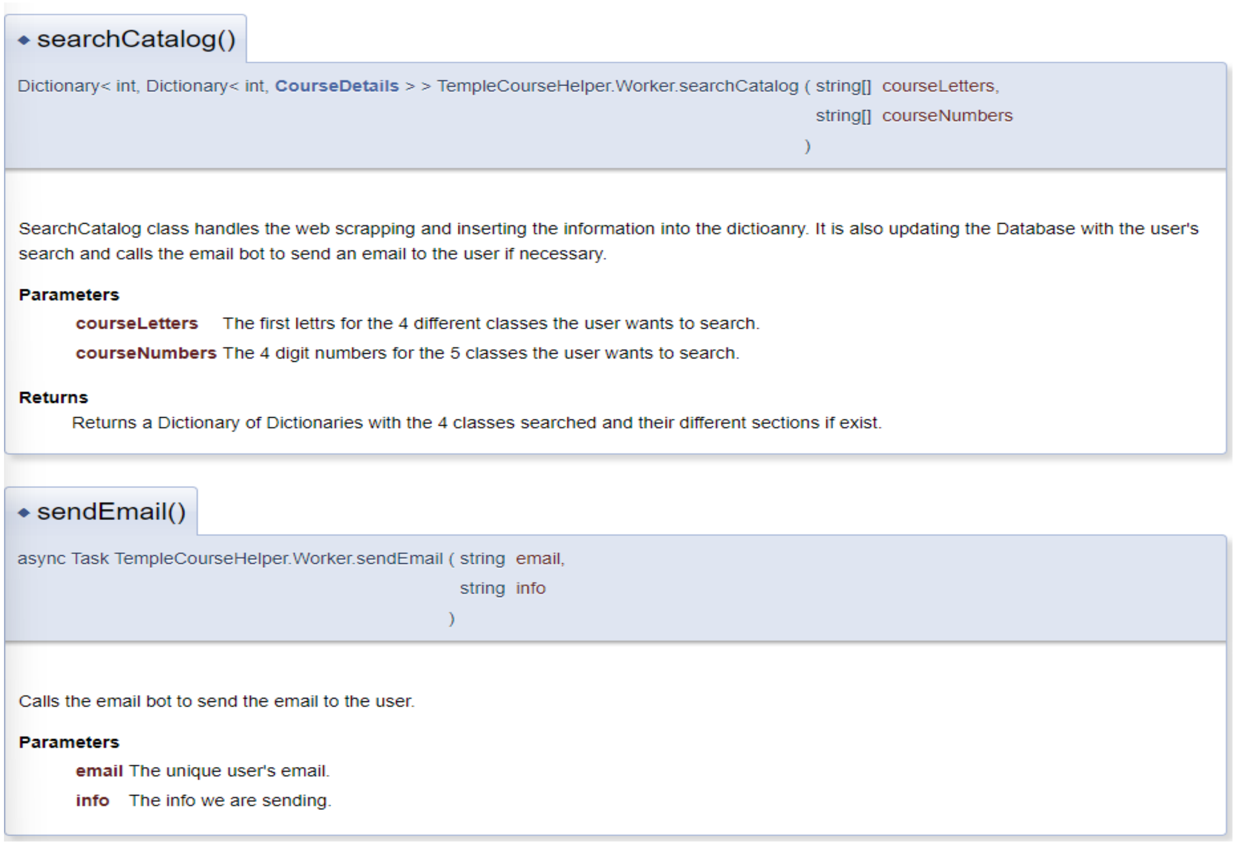
Returns
True or False for the specific user ID.

◆ GetRecords()

```
DataSet TempleCourseHelper.Worker.GetRecords ( )
```

Gets the Records for the specific user.

Returns
A dataset with the user's searched records.



◆ searchCatalog()

```
Dictionary< int, Dictionary< int, CourseDetails > > TempleCourseHelper.Worker.searchCatalog ( string[]  courseLetters,
                                                                                               string[]  courseNumbers
                                                                                             )
```

SearchCatalog class handles the web scrapping and inserting the information into the dictioanry. It is also updating the Database with the user's search and calls the email bot to send an email to the user if necessary.

Parameters

courseLetters	The first lettrs for the 4 different classes the user wants to search.
courseNumbers	The 4 digit numbers for the 5 classes the user wants to search.

Returns

Returns a Dictionary of Dictionaries with the 4 classes searched and their different sections if exist.

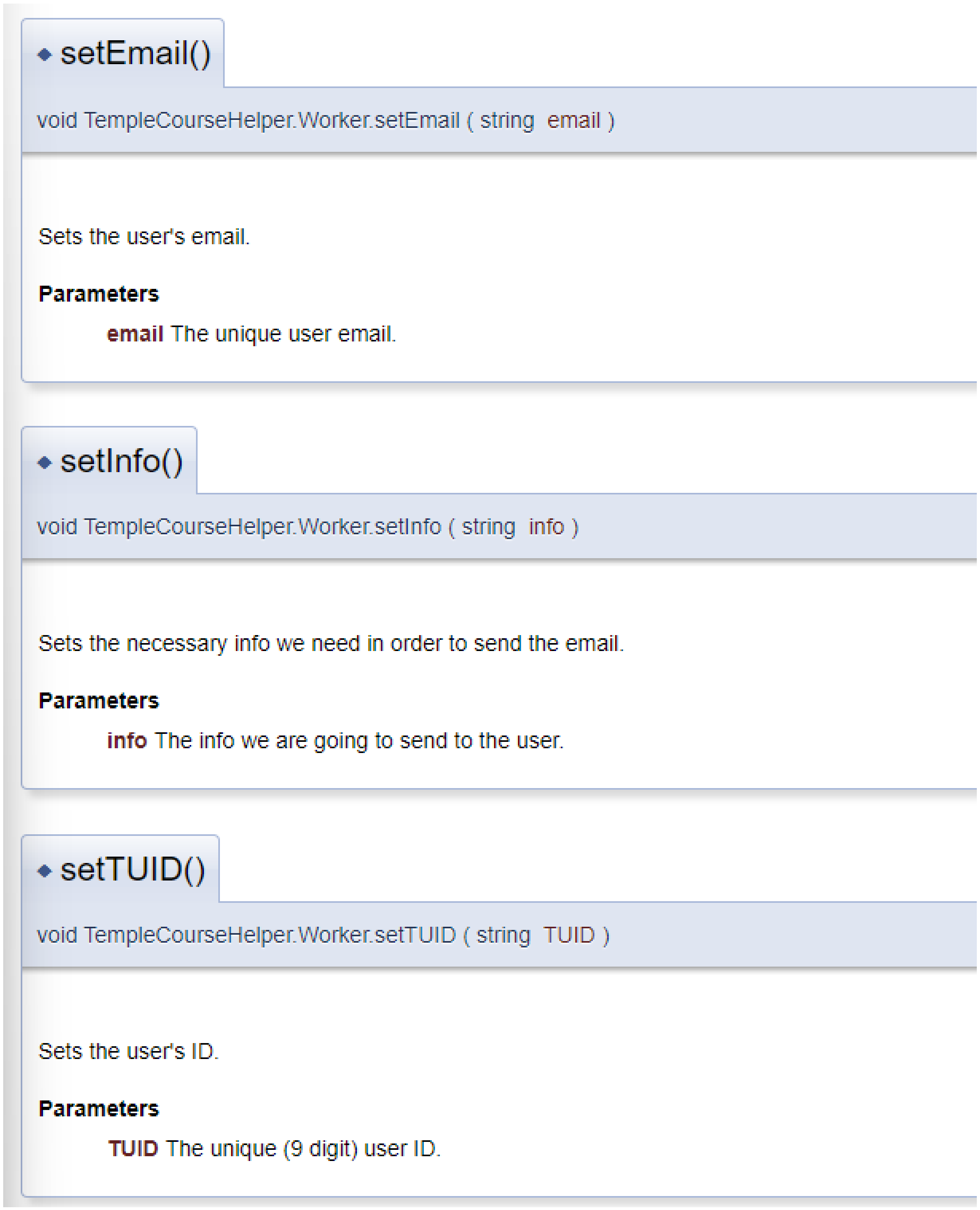
◆ sendEmail()

```
async Task TempleCourseHelper.Worker.sendEmail ( string  email,
                                                 string  info
                                               )
```

Calls the email bot to send the email to the user.

Parameters

email	The unique user's email.
info	The info we are sending.



◆ setEmail()

void TempleCourseHelper.Worker.setEmail (string email)

Sets the user's email.

Parameters

email The unique user email.

◆ setInfo()

void TempleCourseHelper.Worker.setInfo (string info)

Sets the necessary info we need in order to send the email.

Parameters

info The info we are going to send to the user.

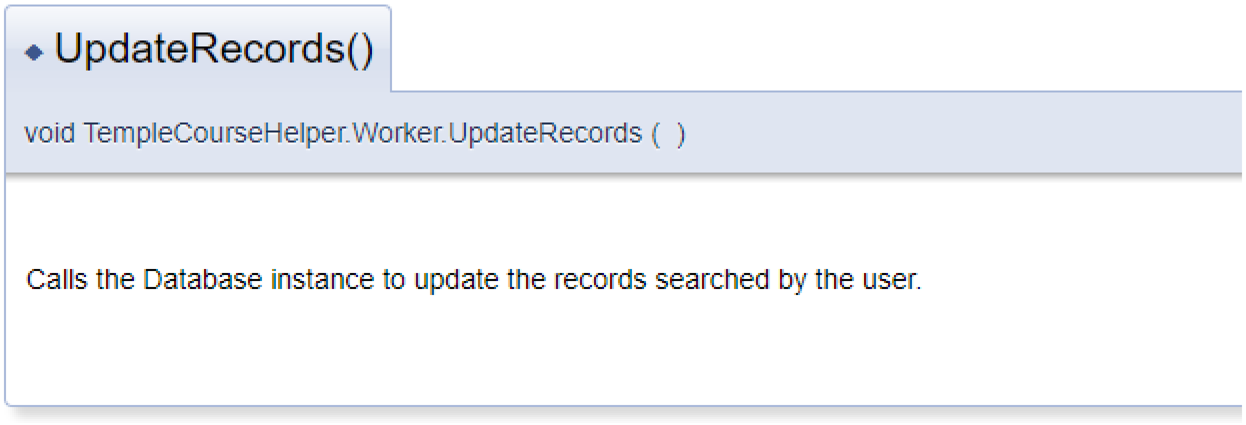
◆ setTUID()

void TempleCourseHelper.Worker.setTUID (string TUID)

Sets the user's ID.

Parameters

TUID The unique (9 digit) user ID.



◆ UpdateRecords()

void TempleCourseHelper.Worker.UpdateRecords ( )

Calls the Database instance to update the records searched by the user.

Project Progress

Week 2 Progress

Sprint Goal: The group will try and finish everything that is included in the Sprint Backlog this week. These things include the search bot that will get the course information from Coursicle and the connection to our database.

Backlog Features

- Bot goes online and searches for classes, and picks the first section.
- Bot stores all the data of the 4 classes searched into a dictionary and displays it to GUI as well as saving the information to a Database.
- Database has been added to store info.
- GUI has more functionality to allow for better user choice/mobility.

Tasks in Sprint	Task Status at end of Sprint	Assigned To
- Bot goes online and searches for classes, picks the first section.	Completed	Arthur
- Bot stores all the data of the 4 classes searched into a dictionary and displays it to GUI as well as saving the information to a Database.	Completed	Arthur/Sofia
- Database has been added to store info.	Completed	Sofia/ Roberto
- GUI has more functionality to allow for better user choice/mobility.	Completed	Arthur/Roberto/Rachel

Week 3 Progress

Sprint Goal: The group's goal for this Sprint is to implement all of our features. At first, we want to make sure that the bot will show all the available sections for every class searched. Then we want to make sure that we have implemented our Email Bot and works correctly. Finally, along with the updates inside the DBConnector class we want to stabilize our code and make sure the program does not crash and runs error-free.

Estimated velocity: 8+8+3+8=27

Effective Velocity: 8+5+3+8=24

Backlog Features:

- Bot will give the user a choice of what section to choose during the initial search.
- DBConnector will have new functions to check previous searches by users and update them.
- Stabilize code to make sure the code doesn't crash or cause errors.
- Implement an email bot.

Tasks in Sprint	**Task Status at end of Sprint**	**Assigned To**
- Bot will show user all the available sections for all the classes searched. (L-8)	Completed	Arthur
- DBConnector will have new functions to check previous searches by the user and update. (L-8)	Not fully Completed	Sofia
- Stabilize code to make sure the program doesn't crash or cause any errors. (S-3)	Completed	Arthur/Roberto
- Implementation of email bot Class. (L-8)	Completed	Rachel

Week 4 Progress (Final Week)

Sprint Goal: The group's goal for this Sprint is to complete everything was not completed from our previous Sprint, which was the new functions of the DBConnector Class, and complete everything we need for that class. Also, we want the automated Testing to be completed. Finally, we want to make sure, once more, that the code does not crash or cause any errors while running, and finalize and add comments on everything, in order to be ready for Demo.

Estimated velocity: 5+3+5+5+3=21

Effective Velocity: 5+3+5+5+3=21

Backlog Features:

- DBConnector will have new functions to check previous searches by user and update.
- DBConnector Class will be completed.
- Automated Testing will be implemented.
- Stabilize code to make sure the code doesn't crash or cause errors.
- Finalize all classes and add comments.

Tasks in Sprint	Task Status at end of Sprint	Assigned To
- DBConnector Class will be completed (M-5)	Completed	Sofia
- DBConnector will have new functions to check previous searches by the user and update. (S-3)	Completed	Sofia
- Stabilize code to make sure the program doesn't crash or cause any errors. (M-5)	Completed	Arthur/Rachel
- Automated Testing will be implemented. (M-5)	Completed	Roberto
- Finalize all classes and add comments. (S-3)	Completed	Arthur/Rachel/Sofia/Roberto