Course Site Generator™

Software Design Description

Author: Raymond Huang

Debugging Enterprises™

October 2018

Version 1.0

Abstract: This document describes the software design for Course Site Generator, a tool to automate the process of building and updating a course Web site.

Based on IEEE Std 1016™ – 2009 document format

1 Introduction

This is the Software Design Description (SDD) for the ***Course Site Generator*** application. Note that this document format is based on the IEEE Standard 1016-2009 recommendation for software design.

1.1 Purpose

This document is to serve as the blueprint for the construction of the ***Course Site Generator*** application. This design will use UML class diagrams to provide complete detail regarding all packages, classes, instance variables, class variables, and method signatures needed to build the application. In addition, UML Sequence diagrams will be used to specify object interactions post-initialization of the application, meaning in response to user interactions or timed events.

1.2 Scope

For this project the goal is for instructors to easily make and update course Web sites. There will be a common structure to the pages and so there are limitations on customization, but the site should be usable for instructors teaching courses in any department at any University. The following frameworks, Desktop Java Framework, jTPS and Properties Manager, will be used to assist in the development of this application. Java will be the target language for this software design.

1.3 Definitions, acronyms, and abbreviations

Class Diagram – A UML document format that describes classes graphically. Specifically, it describes their instance variables, method headers, and relationships to other classes.

Desktop Java Framework (DJF) – a framework designed to help developers design other desktop java applications.

Document Object Model (DOM) – a tree data structure maintained by the browser that contains all content for the currently loaded Web page.

Framework – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

GUI – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

HyperText Markup Language – a markup language used to describe Web pages. Web pages are text files encoded in HTML that can employ JavaScript and Stylesheets to build and style content.

IEEE – Institute of Electrical and Electronics Engineers, the “world’s largest professional association for the advancement of technology”.

Java – A high-level programming language that uses a virtual machine layer between the Java application and the hardware to provide program portability.

JavaScript – the default scripting language of the Web, JavaScript is provided to pages in the form of text files with code that can be loaded and executed when a page loads so as to dynamically generate page content in the DOM.

jTPS- a framework designed to simplify transactions within an application

Properties Manager- a framework designed to load application properties from XML documents

Sequence Diagram – A UML document format that specifies how object methods interact with one another

Stylesheet – a static text file employed by HTML pages that can control the colors, fonts, layout and other style components in a Web page.

UML – Unified Modeling Language, a standard set of document formats for designing software graphically.

Use Case Diagram – A UML document format that specifies how a user will interact with a system.

1.4 References

IEEE Std 830TM-1998 (R2009) – IEEE Recommended Practice for Software Requirements Specification

Course Site Generator™ SRS – Debugging Enterprises' Software Requirements Specification for the Course Site Generator application.

1.5 Overview

This Software Design Description document provides a working design for the ***Course Site Generator*** software application as descripted in the Course Site Generator Software Requirements Specification. Note that all parties in the implementation stage must agree upon all connections between components before proceeding with the implementation stage. Section 2 of this document will provide the Package-Level Viewpoint, specifying the packages and frameworks to be designed. Section 3 will provide the Class-Level Viewpoint, using UML Class Diagrams to specify how the classes should be constructed. Section 4 will provide the Method-Level System Viewpoint, describing how methods will interact with one another. Section 5 provides deployment information like file structures and formats to use. Section 6 provides a Table of Contents, an Index, and References. Note that all UML Diagrams in this document were created using the VioletUML editor.

2 Package-Level Design Viewpoint

As mentioned, this design will encompass both the Course Site Generator application while utilizing the Desktop Java Framework, jTPS and Properties Manager. During construction, we will be utilizing the Java API to provide services. The following are descriptions of the components to be built, as well as how the Java API will be used to build them.

2.1 Course Site Generator Overview

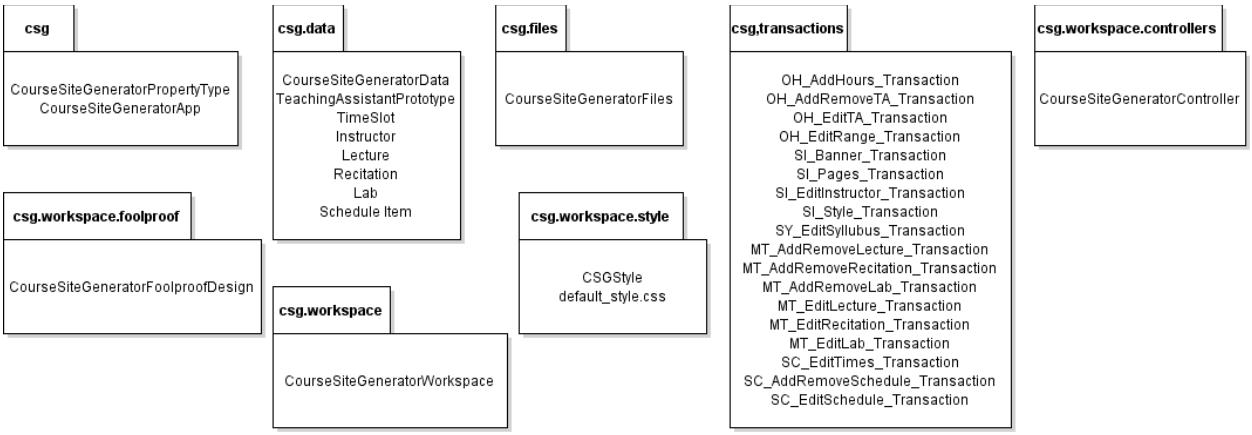


Figure 2.1: Design Packages Overview

2.2 Java API Usage

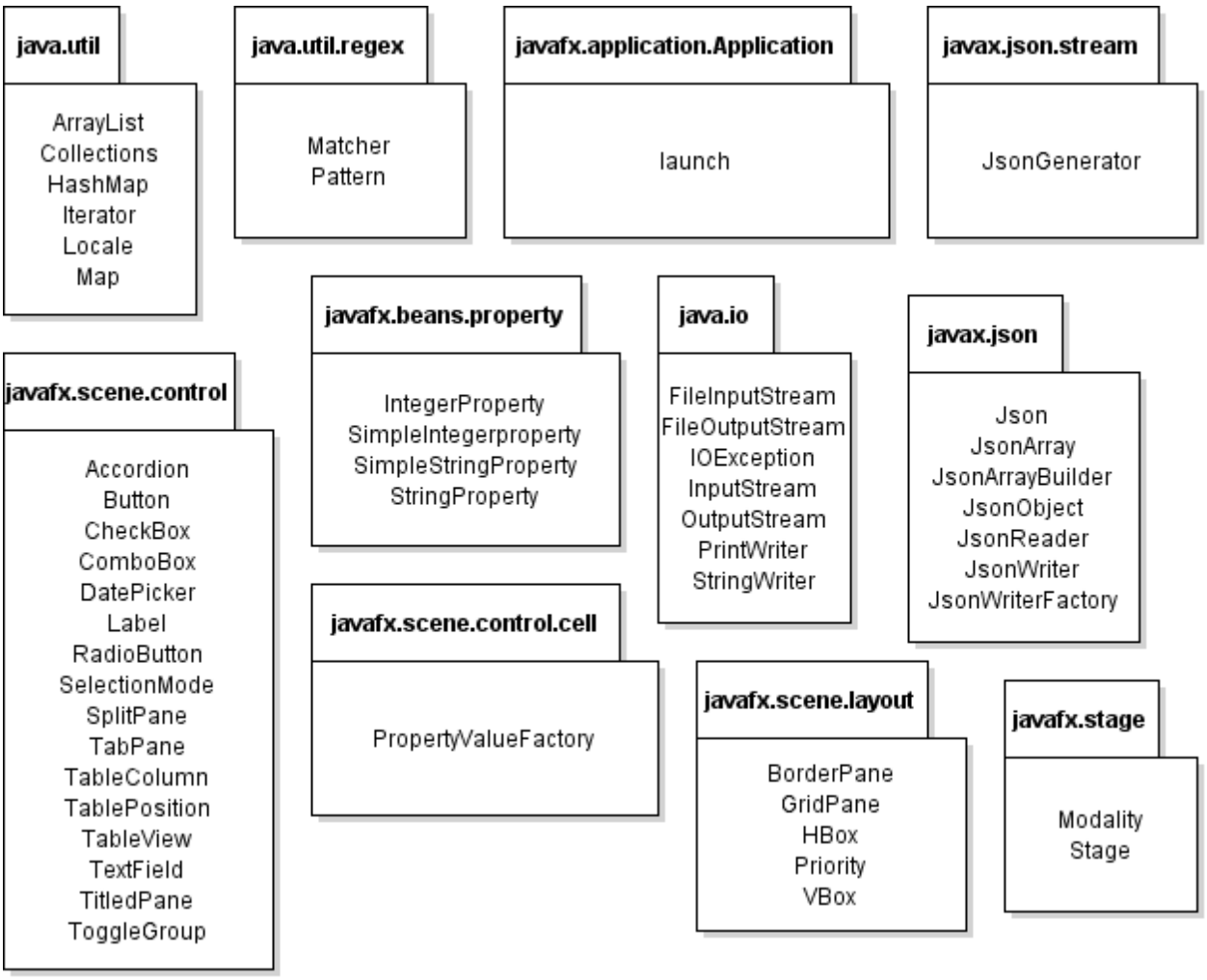


Figure 2.2 Java API Classes and Packages To Be Used

2.3 Java API Usage Descriptions

Tables 2.1-2.11 below summarize how each of the classes will be used

Class/Interface	Use
ArrayList	For storing objects that will be inserted into tables
Collections	For sorting information in ArrayLists
HashMap	For storing data in the Time Slots class
Iterator	For going through each element in the ArrayLists
Locale	For defining the language of the application
Map	For implementing HashMap

Table 2.1: Uses for classes in the Java API's java.util package

Class/Interface	Use
Matcher	For checking if a Pattern instance is a valid email
Pattern	For creating a Pattern instance from a regular expression

Table 2.2: Uses for classes in the Java API's java.util.regex package

Class/Interface	Use
launch	For starting the starting the Java application

Table 2.3: Uses for the classes in the Java API's javafx.application.Application package

Class/Interface	Use
JsonGenerator	For allowing the ability to write to Json files

Table 2.4: Uses for the classes in the Java API's javax.json.stream package

Class/Interface	Use
Accordion	For putting multiple TitledPanes in a row
Button	For setting up buttons in the GUI
CheckBox	For the Pages section of the Site tab
ComboBox	For selecting options in the GUI
DatePicker	For picking a date in the Schedule tab
Label	For labeling various elements of the application
RadioButton	For selecting TA type in Office Hours
SelectionMode	For getting selected elements in tables
SplitPane	For designing the layout for the GUI
TabPane	For creating the various tabs in the GUI
TableColumn	For getting the column selected in a table
TablePosition	For getting the numerical value of a selected table element
TableView	For translating data into a viewable table
TextField	For getting text input from the user
TitledPane	For creating expandable panes
ToggleGroup	To ensure that only one radio button is selected in a group

Table 2.5: Uses for the classes in the Java API's javafx.scene.control package

Class/Interface	Use
IntegerProperty	For creating Integers that can be used by TableView
SimpleIntegerProperty	For constructing IntegerProperties
SimpleStringProperty	For constructing StringProperties
StringPreoperty	For creating Strings that can be used by TableView

Table 2.6: Uses for the classes in the Java API's javafx.beans.property package

Class/Interface	Use
FileInputStream	For reading input from a json file
FileOutputStream	For storing output to a json file
IOException	For catching exceptions thrown while reading/writing to a file
InputStream	For creating methods to read from input
OutputStream	For creating methods to write to the output
PrintWriter	For writing data to a json file
StringWriter	For collecting data which is used to construct a string

Table 2.7: Uses for the classes in the Java API's java.io package

Class/Interface	Use
Json	For getting methods related to json files
JsonArray	For storing data into json arrays
JsonArrayBuilder	For building the arrays to used to store data
JsonObject	For creating a json object to save to a json file
JsonReader	For reading data from a json file
JsonWriter	For writing data to a json file
JsonWriterFactory	For helping construct a Json writer to write to a json object

Table 2.8: Uses for the classes in the Java API's javax.json package

Class/Interface	Use
PropertyValueFactory	For binding an ObservableList to a Table Column

Table 2.9: Uses for the classes in the Java API's javafx.scene.control.cell package

Class/Interface	Use
BorderPane	For putting elements into a pane based on position
GridPane	For creating the edit TA dialog
HBox	For storing elements in a horizontal box
Priority	For ensuring that the table will always fit the information
VBox	For storing elements in a vertical box

Table 2.10: Uses for the classes in the Java API's javafx.scene.layout package

Class/Interface	Use
Modality	For ensuring that the user stays on the current window
Stage	For creating the baseline window for the application

Table 2.11: Uses for the classes in the Java API's javafx.stage package

3 Class-Level Design Viewpoint

As mentioned, this design will include Desktop Java Framework, Properties Manager, jTPS along with Course Site Generator. Due to the complexity of the project, the classes will be presented using a series of diagrams from overview diagrams to detailed ones. Several images will be supplied for larger diagrams due to the size of the diagrams.

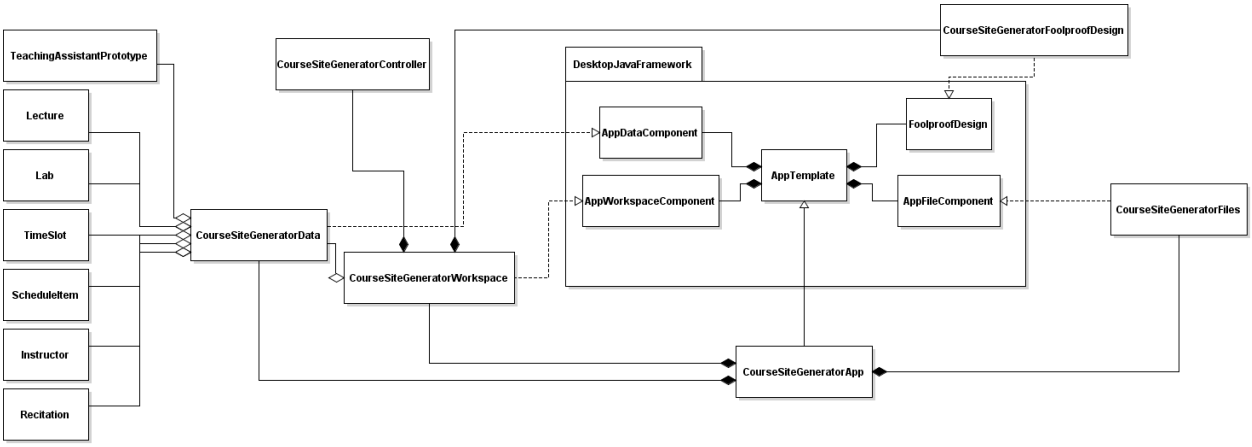


Figure 3.1: CourseSiteGenerator Overview UML Class Diagram

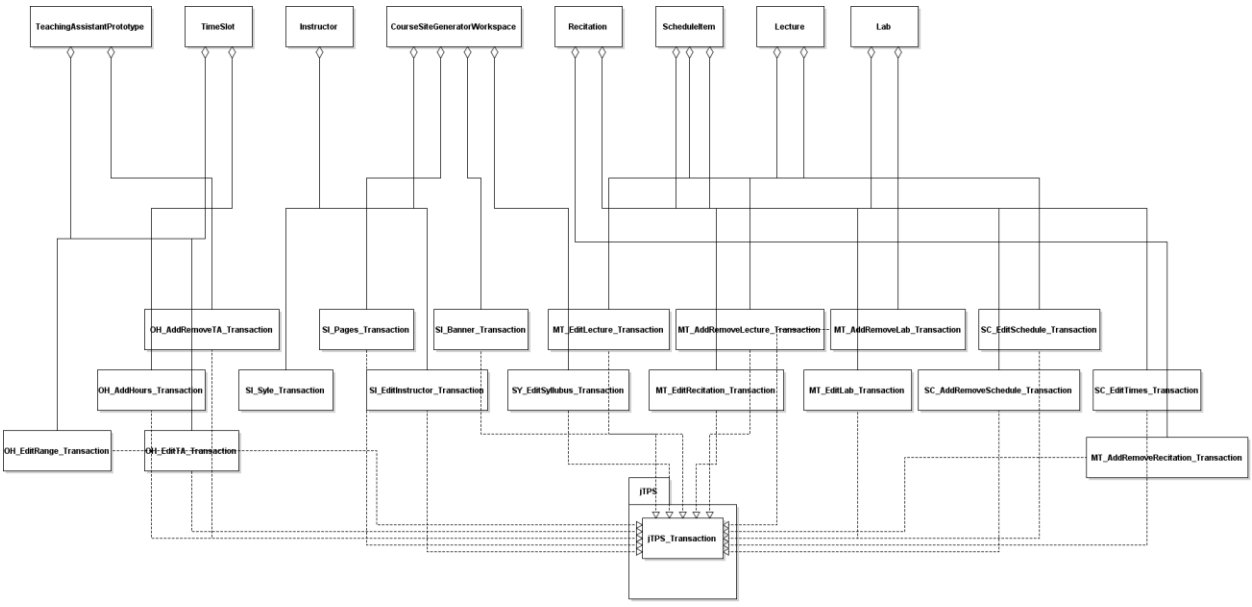
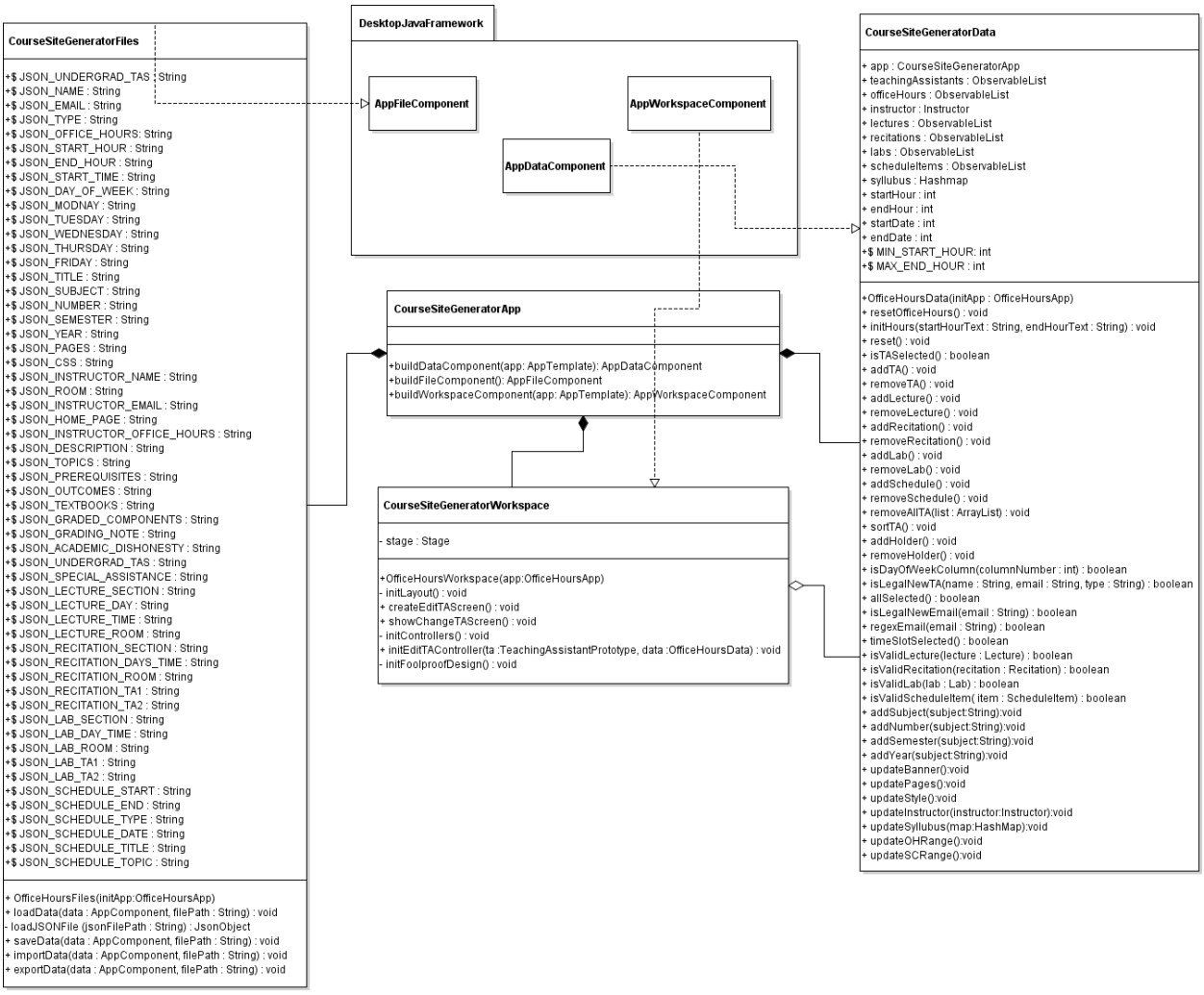
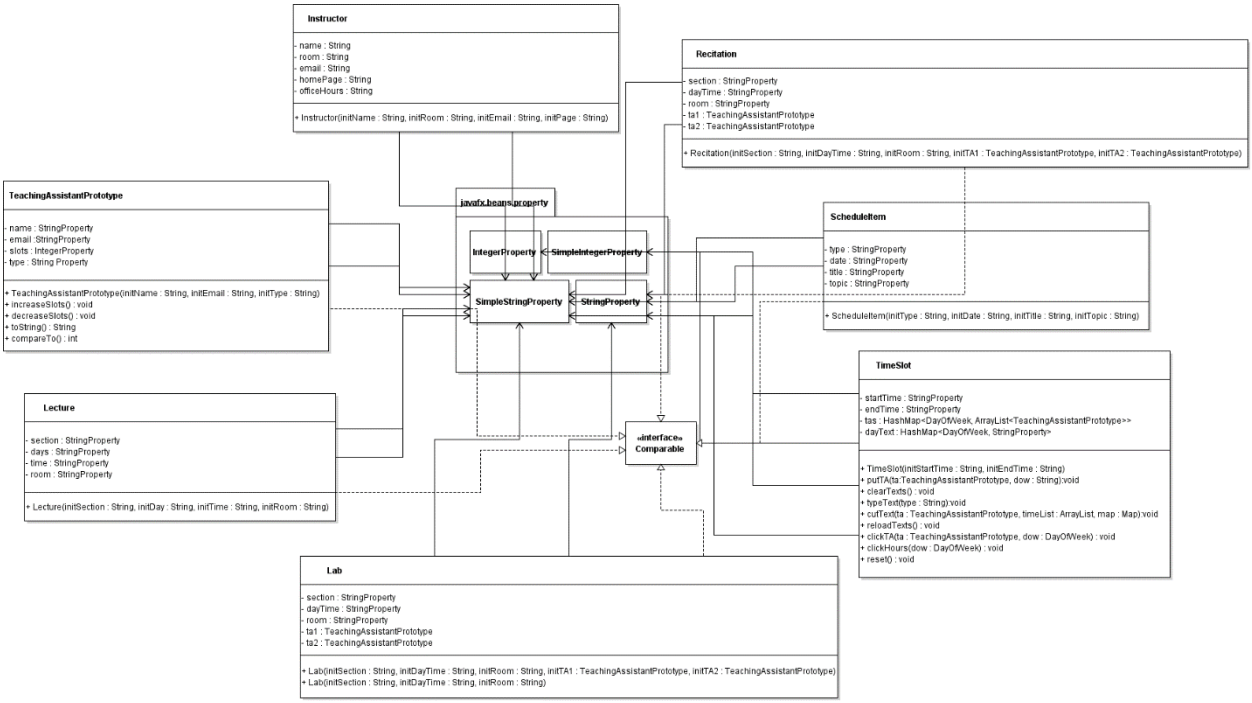


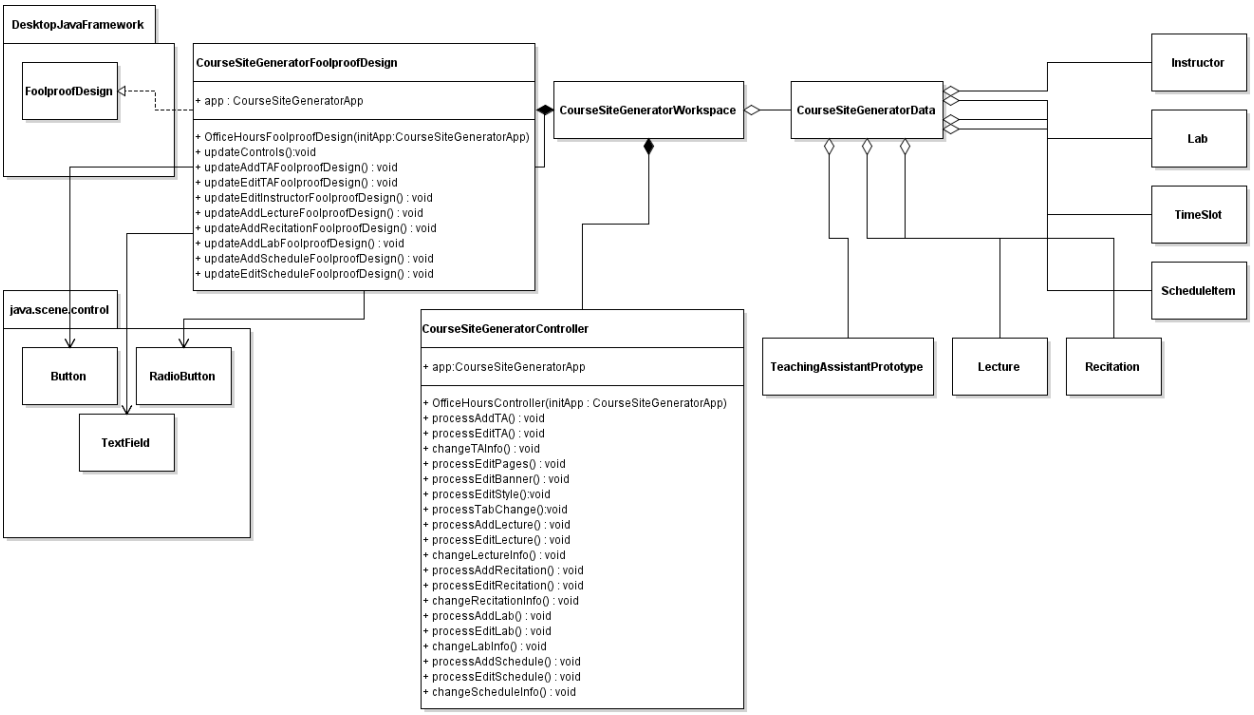
Figure 3.2: CourseSiteGenerator Transaction Overview UML Class Diagram



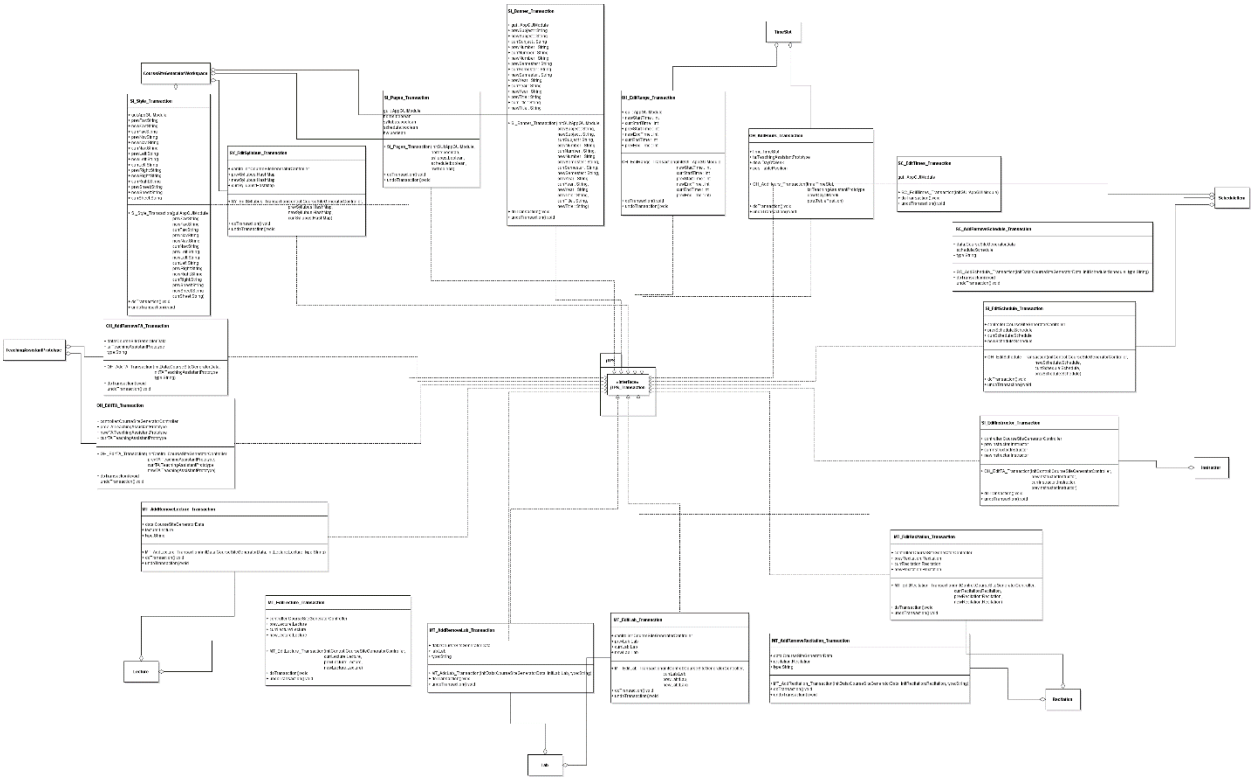
3.3: Detailed CourseSiteGeneratorApp, CourseSiteGeneratorFiles, CourseSiteGeneratorData and CourseSiteGeneratorWorkspace UML Class Diagram



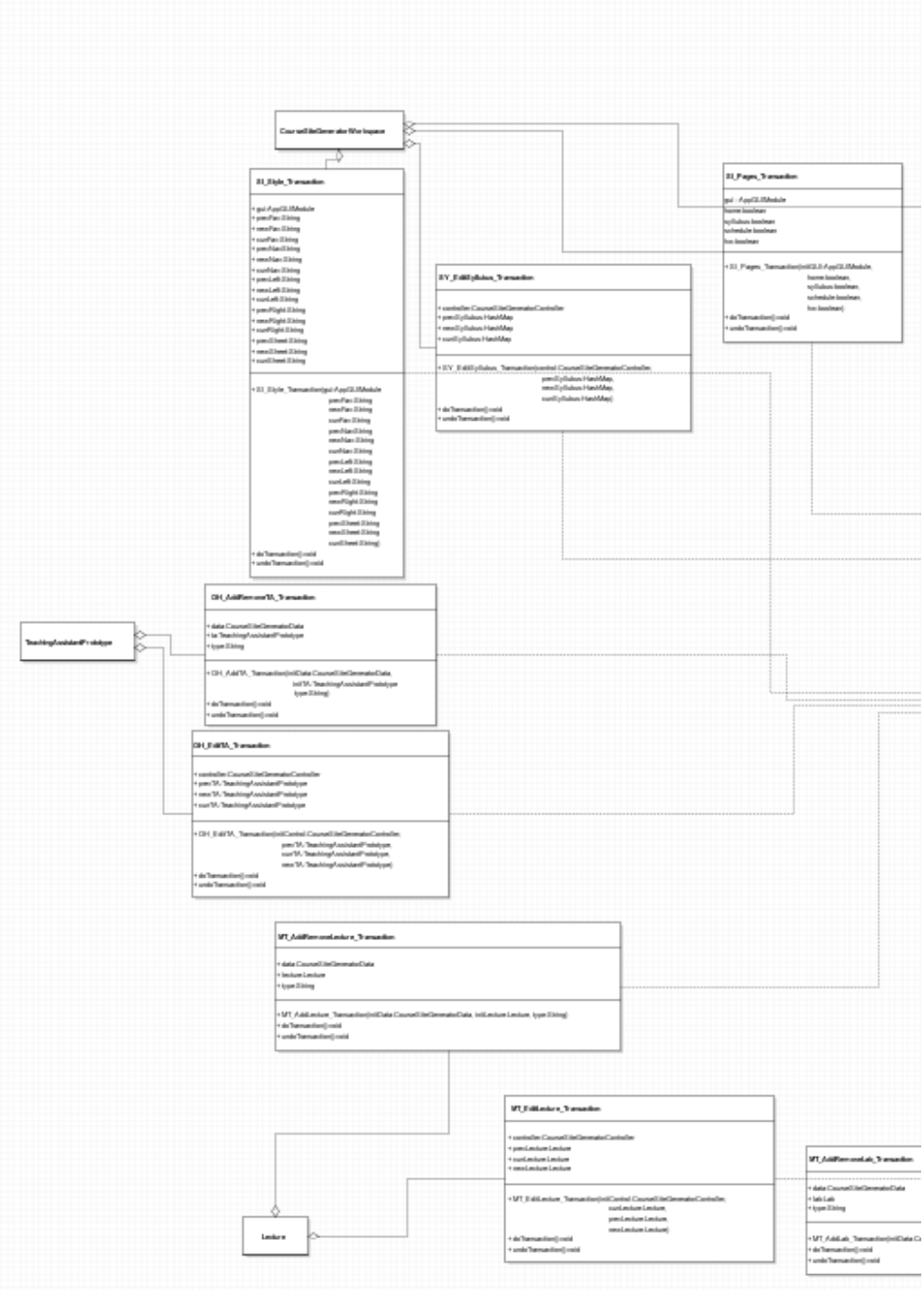
3.4: Detailed TeachingAssistantPrototype, TimeSlot, Lecture, Lab, ScheduleItem, Instructor and Recitation UML Class Diagram



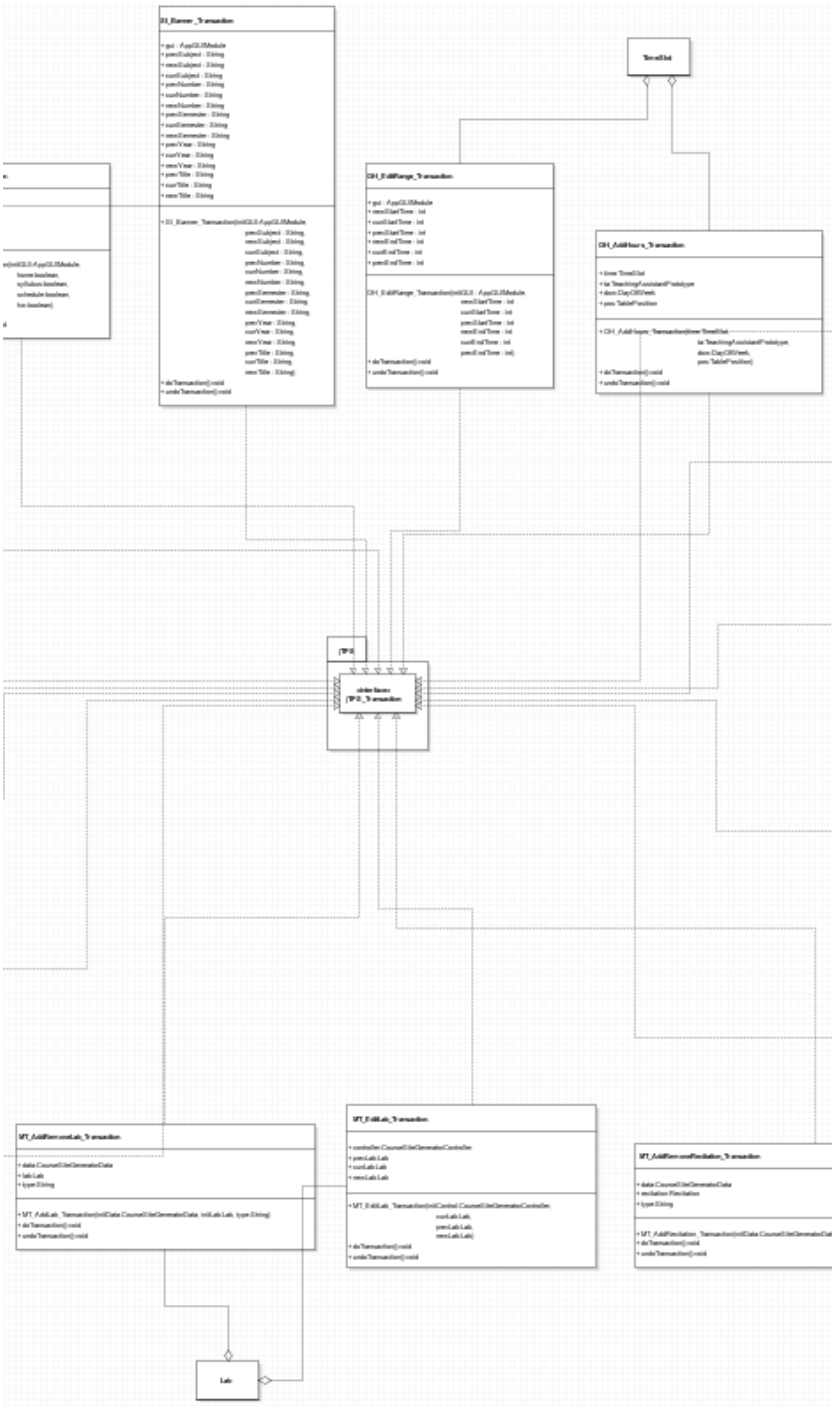
3.5 Detailed CourseSiteGeneratorFoolproofDesign and CourseSiteGeneratorController UML Class Diagram



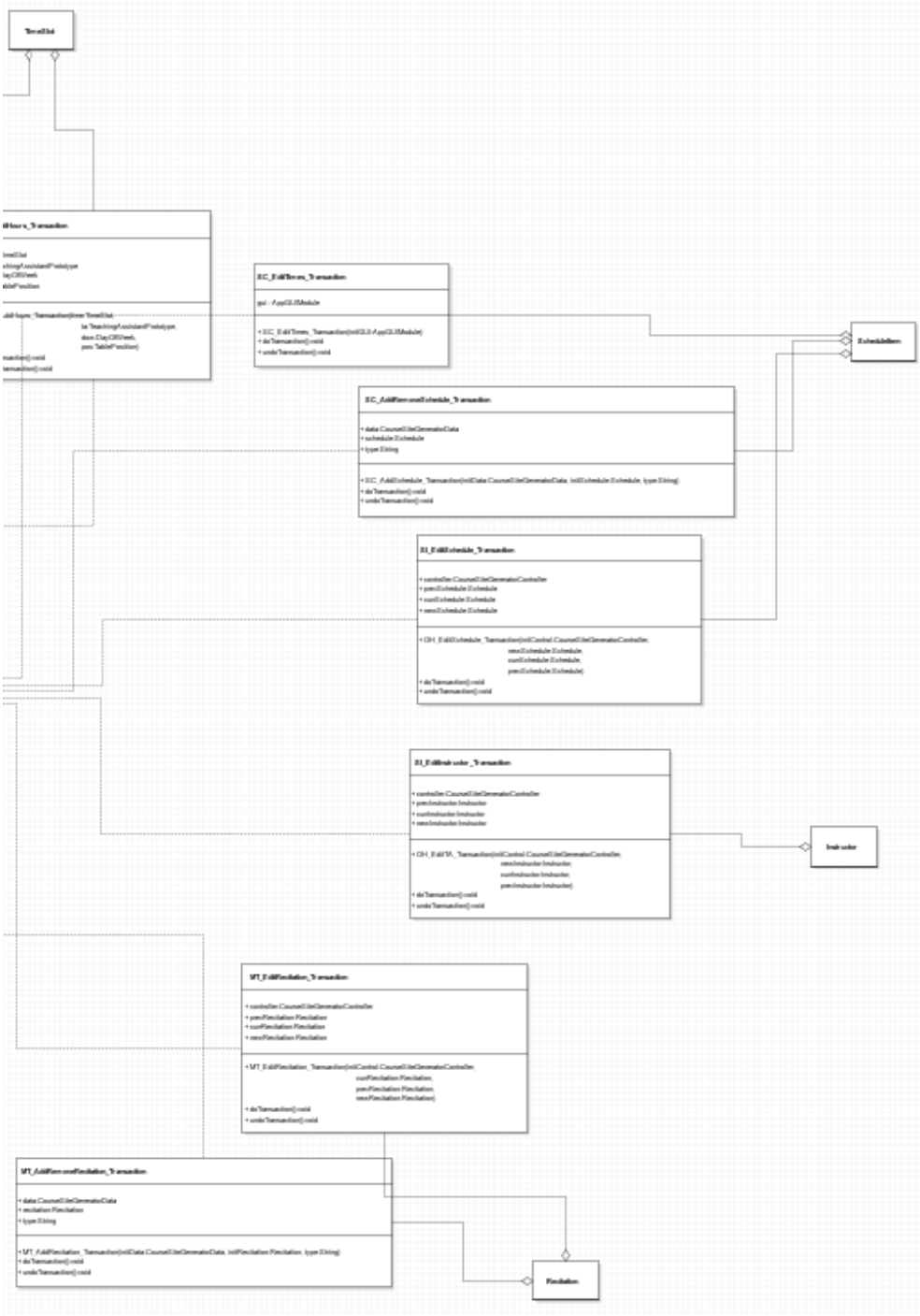
3.6 Detailed OH_EditRange_Transaction, OH_EditTA_Transaction, OH_AddHours_Transaction, OH_AddRemoveTA_Transaction, SI_Pages_Transaction, SI_EditInstructorTransaction, SI_Banner_Transaction, SI_Style_Transaction, SY_EditSyllubus_Transaction, MT_EditLecture_Transaction, MT_AddRemoveLecture_Transaction, MT_EditRecitation_Transaction, MT_AddRemoveLab_Transaction, MT_AddRemoveRecitation_Transaction, MT_AddRemoveLab_Transaction, SC_AddSchedule_Transaction, SC_EditSchedule_Transaction, SC_EditTimes_Transaction



3.6.1 Detailed OH_EditRange_Transaction, OH_EditTA_Transaction, OH_EditHours_Transaction, OH_AddTA_Transaction, SI_Pages_Transaction, SI_EditInstructor_Transaction, SI_Banner_Transaction, SI_Style_Transaction, SY_EditSyllubus_Transaction, MT_EditLecture_Transaction, MT_AddLecture_Transaction, MT_EditRecitation_Transaction, MT_AddRemoveLab_Transaction, MT_AddRecitation_Transaction, MT_EditLab_Transaction, SC_AddSchedule_Transaction, SC_EditSchedule_Transaction, SC_EditTimes_Transaction Part 1



3.6.2 Detailed OH_EditRange_Transaction, OH_EditTA_Transaction, OH_EditHours_Transaction, OH_AddTA_Transaction, SI_Pages_Transaction, SI_EditInstructor_Transaction, SI_Banner_Transaction, SI_Style_Transaction, SY_EditSyllubus_Transaction, MT_EditLecture_Transaction, MT_AddLecture_Transaction, MT_EditRecitation_Transaction, MT_AddRemoveLab_Transaction, MT_AddRecitation_Transaction, MT_EditLab_Transaction, SC_AddSchedule_Transaction, SC_EditSchedule_Transaction, SC_EditTimes_Transaction Part 2



3.6.3 Detailed OH_EditRange_Transaction, OH_EditTA_Transaction, OH_EditHours_Transaction, OH_AddTA_Transaction, SI_Pages_Transaction, SI_EditInstructor_Transaction, SI_Banner_Transaction, SI_Style_Transaction, SY_EditSyllubus_Transaction, MT_EditLecture_Transaction, MT_AddLecture_Transaction, MT_EditRecitation_Transaction, MT_AddRemoveLab_Transaction, MT_AddRecitation_Transaction, MT_EditLab_Transaction, SC_AddSchedule_Transaction, SC_EditSchedule_Transaction, SC_EditTimes_Transaction Part 3

4 Method-Level Design Viewpoint

Now that the general architecture of the classes has been determined, it is time to specify how data will flow through the system. The following UML Sequence Diagrams describe the methods called within the code to be developed in order to provide the appropriate event responses.

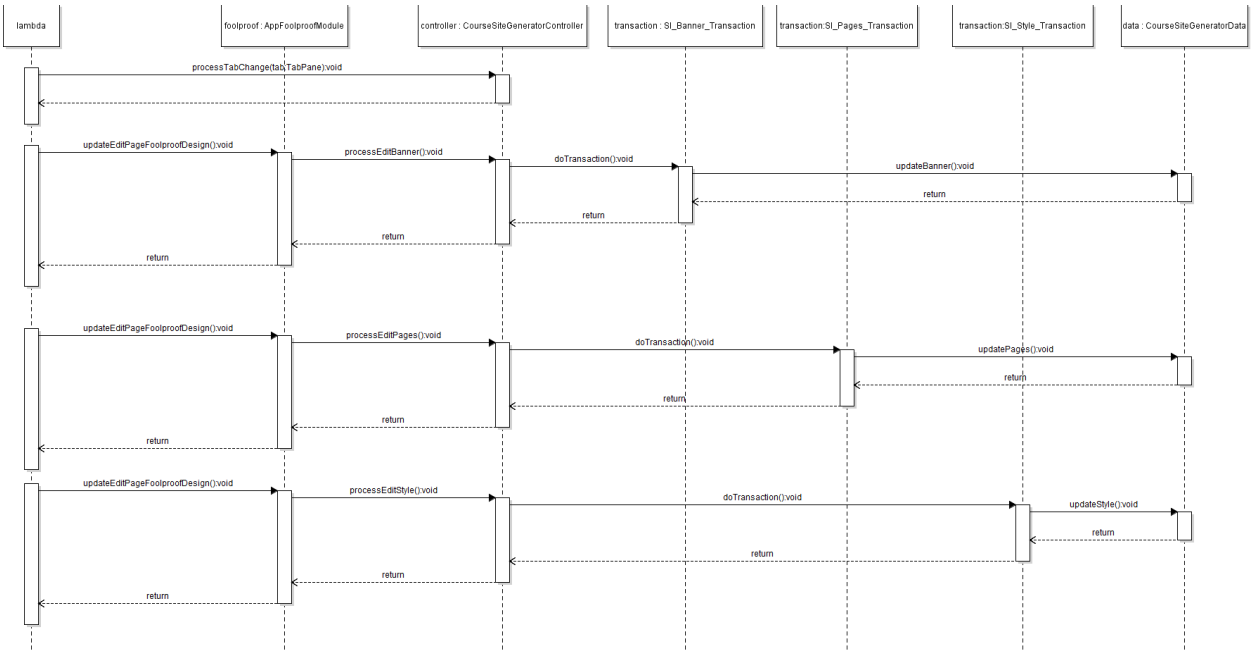


Figure 4.1: Edit Page Details UML Sequence Diagrams

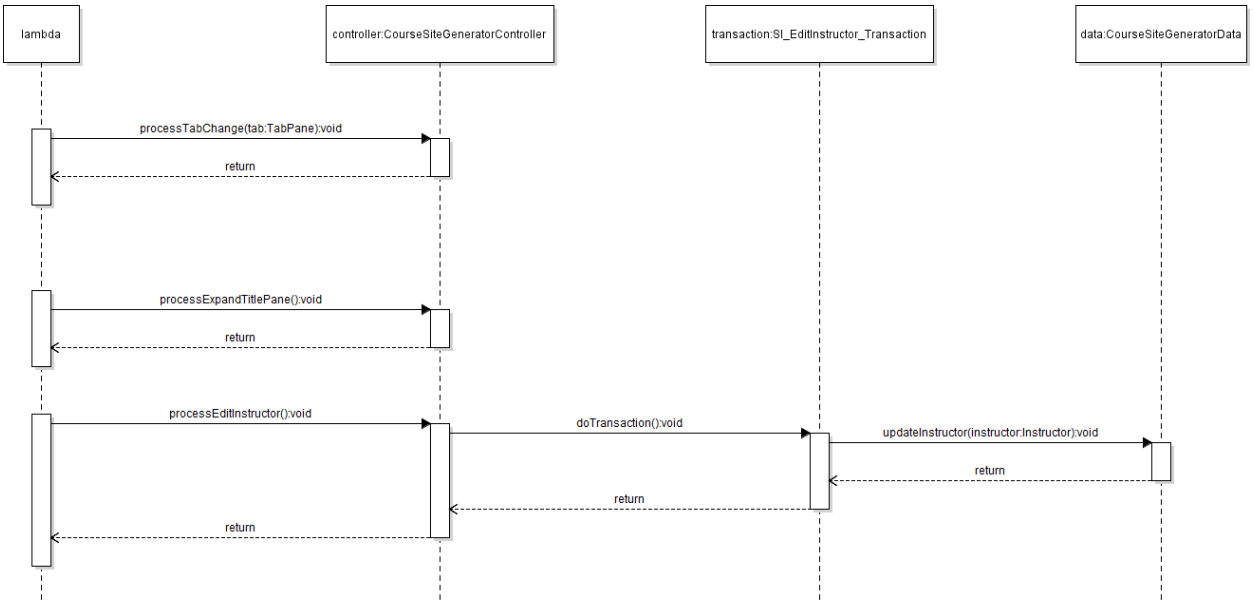


Figure 4.2: Edit Instructor UML Sequence Diagrams

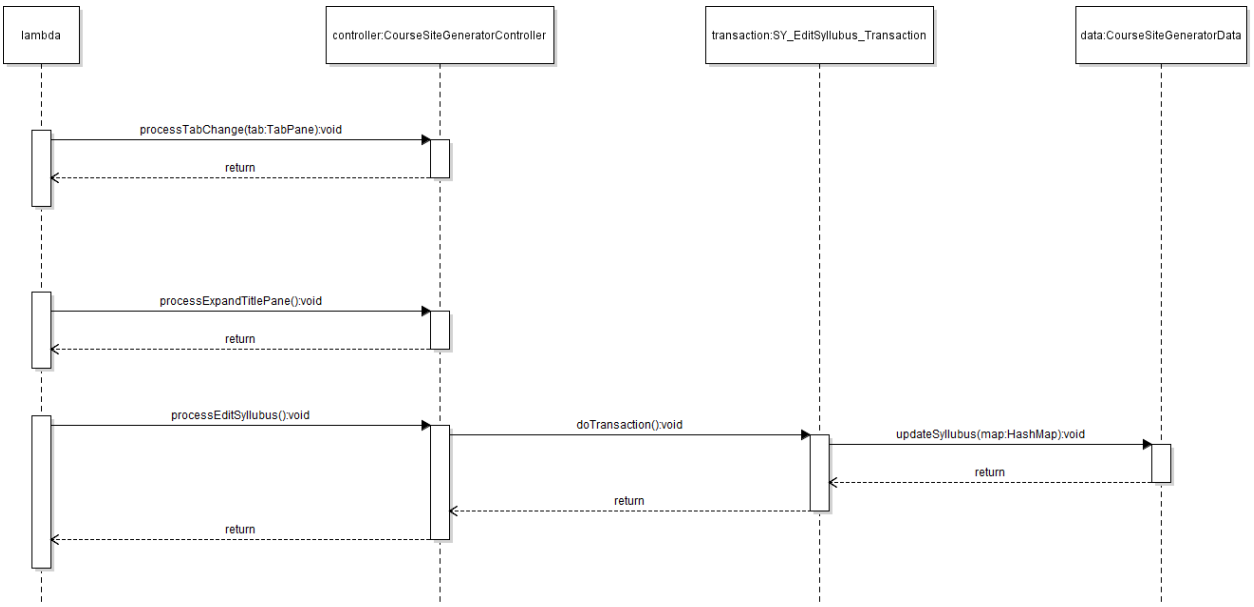


Figure 4.3: Edit Syllabus Details UML Sequence Diagrams

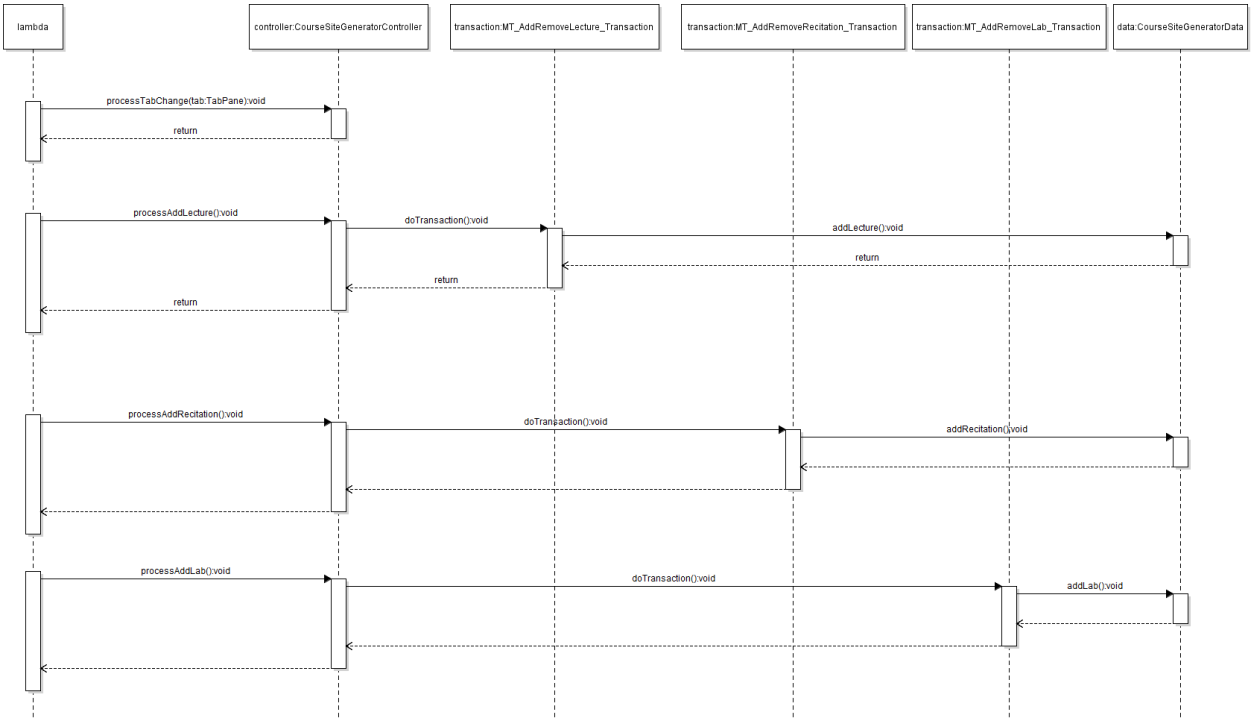


Figure 4.4: Add Lecture/Recitation/Lab UML Sequence Diagrams

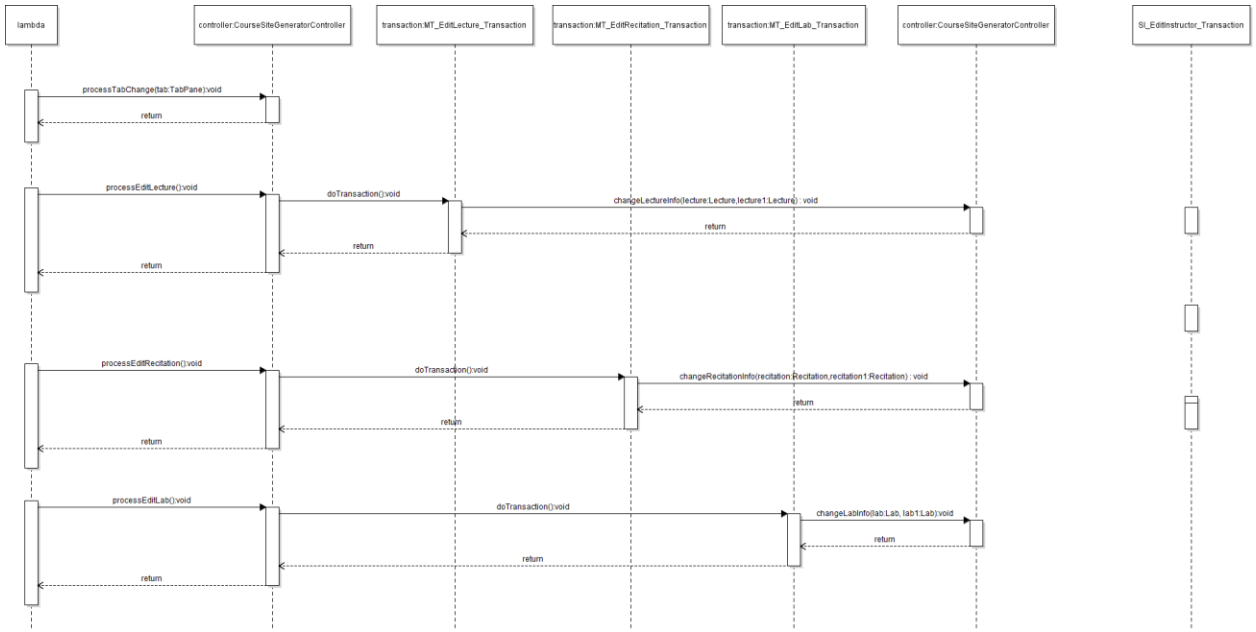


Figure 4.5: Edit Lecture/Recitation/Lab UML Sequence Diagrams

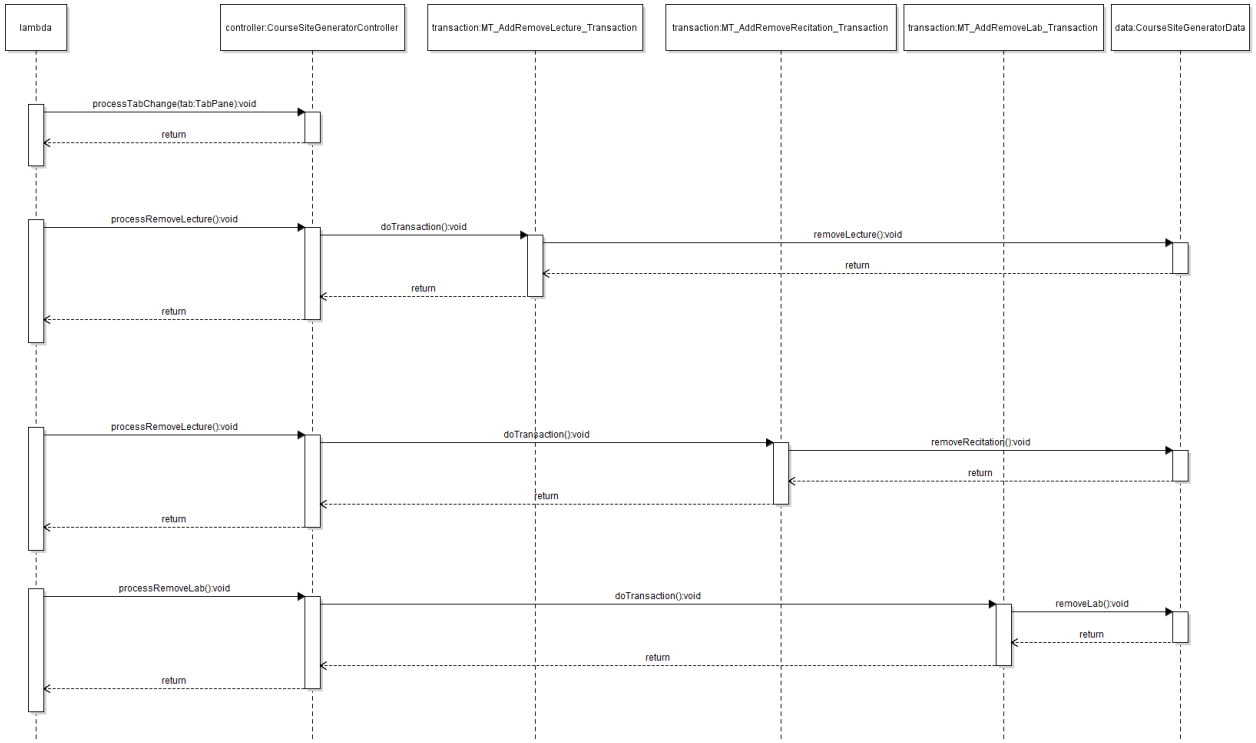


Figure 4.6: Remove Lecture/Recitation/Lab UML Sequence Diagrams

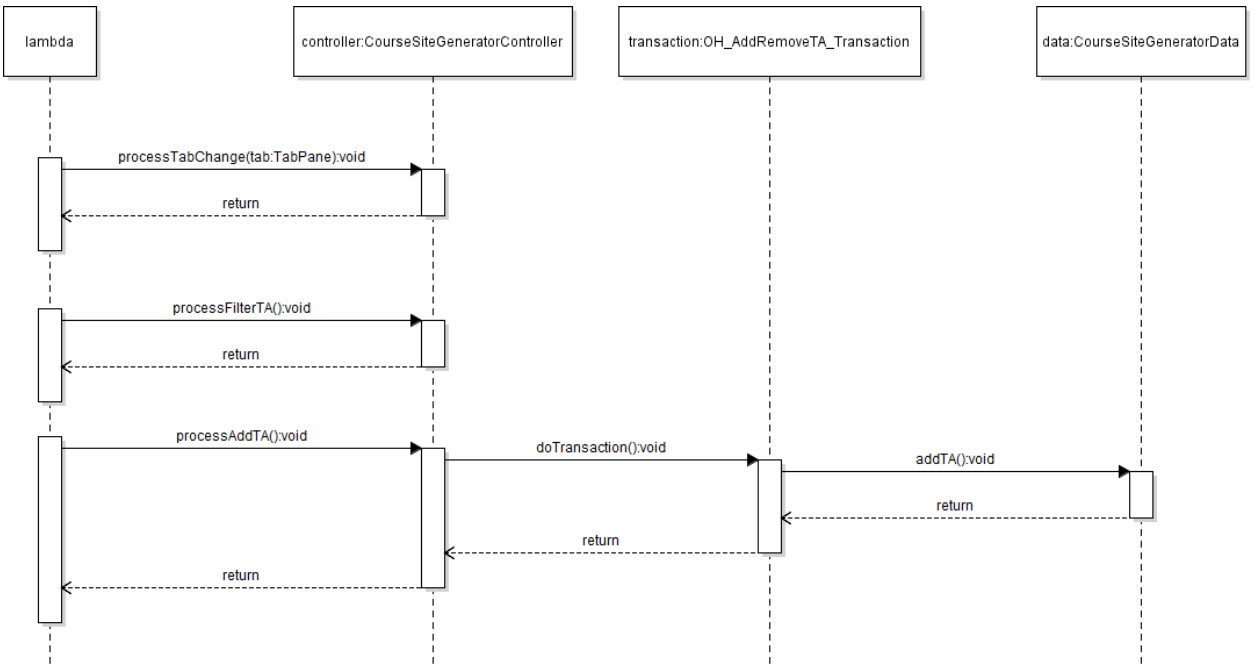


Figure 4.7: Add TA UML Sequence Diagrams

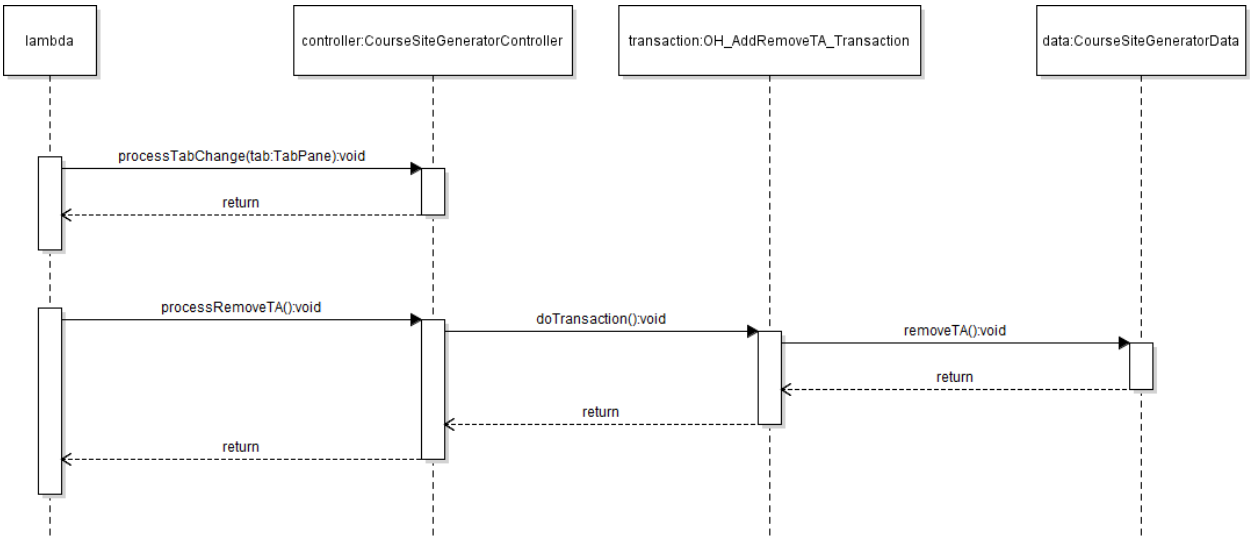


Figure 4.8: Remove TA UML Sequence Diagrams

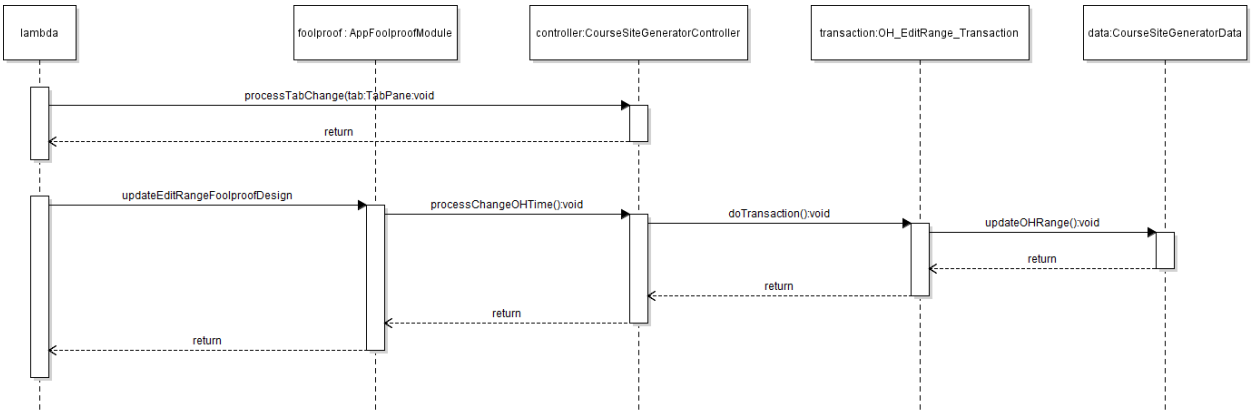


Figure 4.9: Select Time Range UML Sequence Diagrams

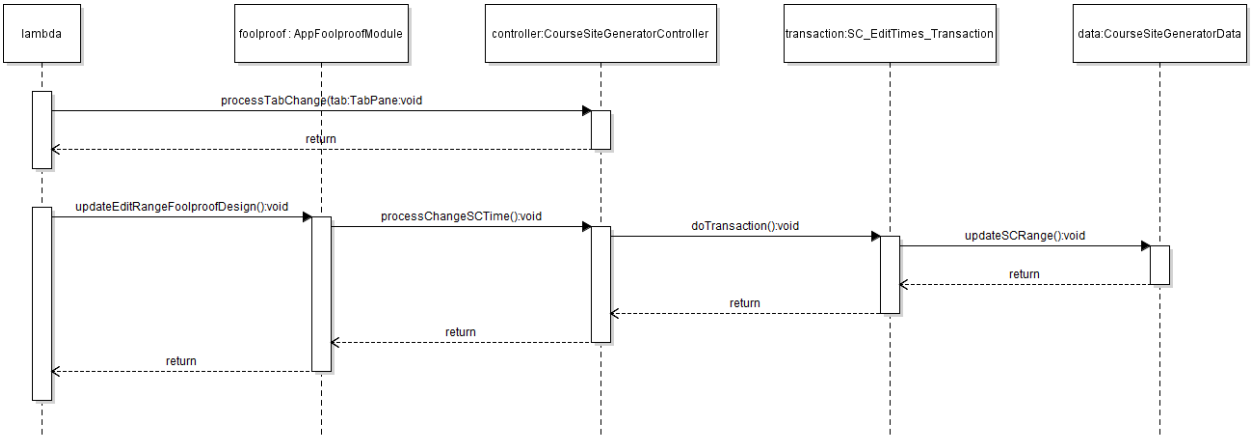


Figure 4.10: Edit Start and End Dates UML Sequence Diagrams

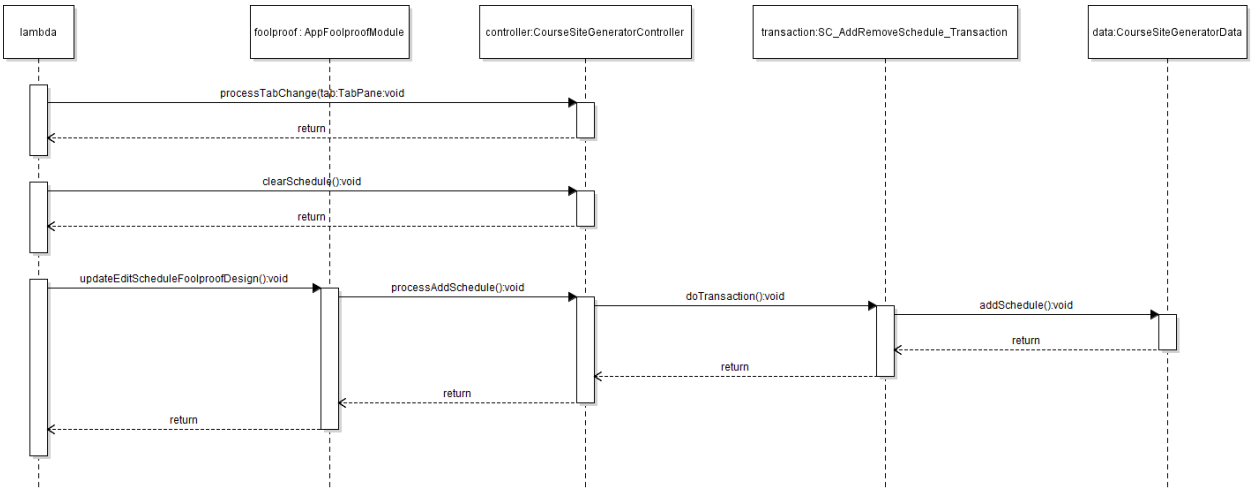


Figure 4.11: Add Schedule Item UML Sequence Diagrams

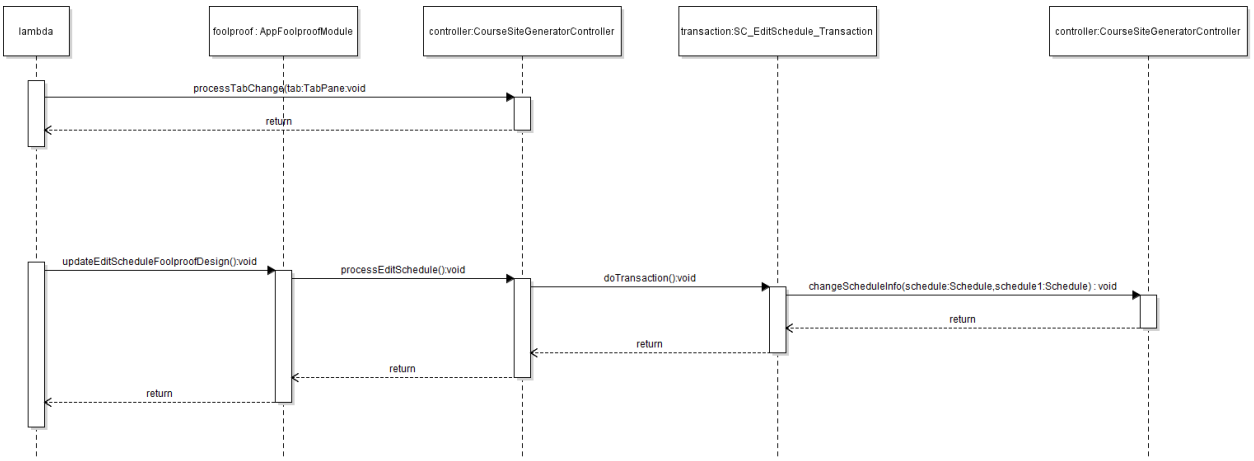


Figure 4.12: Edit Schedule Item UML Sequence Diagrams

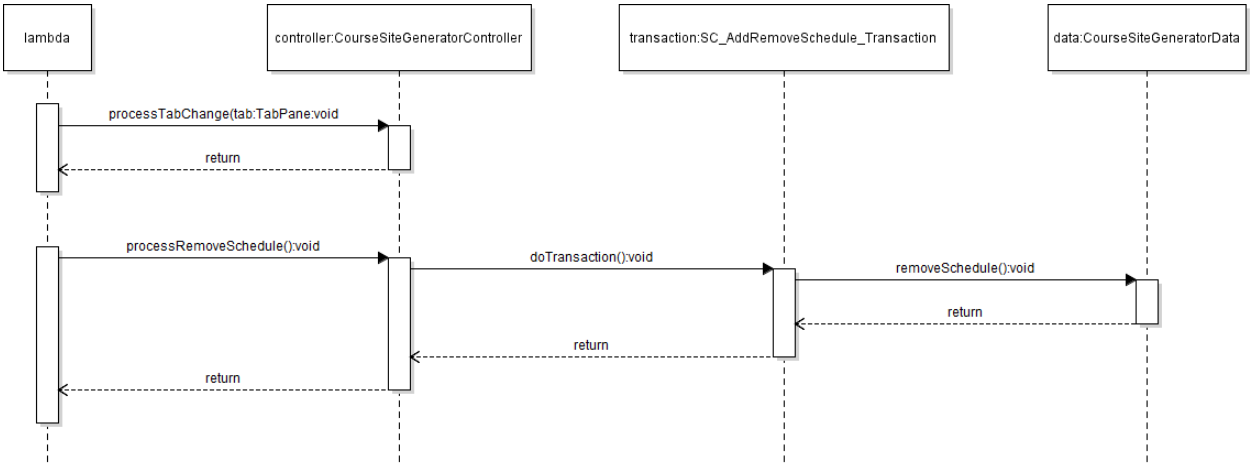


Figure 4.13: Remove Schedule Item UML Sequence Diagrams

5 File Structure and Formats

Note that the Desktop Java Framework will be provided inside DesktopJavaFramework.jar, a Java ARchive file that will encapsulate the entire framework. Accordingly, jTPS and PropertiesManager will also be provided through jTPS.jar and PropertiesManager.jar. This should be imported into the necessary project for the CourseSiteGenerator application and will be included in the deployment of a single, executable JAR file titled CourseSiteGenerator.jar. Note that all necessary data and art files must accompany this program. The application will pull necessary images and HTML files from folder within the project. This application also utilizes JSON and XML file types. JSON files are used in this application to save data for each course site generated. XML file types are used to store property types of the application itself.

For each page generated, three separate folders should be created for files. These folders should be named CSS, images and JS. The images folder will contain all the needed image files for the web page, the CSS folder will contain the CSS files while the JS folder will contain the necessary JavaScript files.

Images and icons for the application will have to be saved in the icons folder within the images folder. HTML files will also be accompanied with the application in the web folder within the app_data folder

RecentWork.txt provides all the current sites that are available to work on. This file is a raw text file that lists all file names, with the appropriate directory path to retrieve the file. This file will be stored in the recent folder within the app_data folder.

File1-folder/File1.json

File2-folder/File2.json

…

We can describe these values as follows:

FileX – a String for the name of the file that appears in the application

-folder – the folder that the file exists in

FileX.json- the actual name of the file within the directory

language_properties_EN.xml is one example of various language files used in Course Site Generator. To add additional languages to the application, you would create additional files based on the language you want to add. XML files will be stored in the properties folder within the app_data folder

<property name=" TAG_HERE" value="String Value"></property>

TAG_HERE- the tag attached to the Node that the value will be associated with

String Value- the actual text that will appear on the application, change this value based on the language that you want represented

app_properties.xml is another XML file used by the application to load its properties. This file will contain file paths for various elements along with Boolean values for toolbars to include to which images to load into the application. XML files will be stored in the properties folder within the app_data folder.

<property name=" TAG_HERE" value="String Value"></property>

TAG_HERE- a tag attached to the property value that will be retrieved by the Properties Manager framework

String Value- a String value representing a file path, Boolean or file name based on the actual content needed

To save the current site being worked on, JSON files are used by the application. This application saves data by storing them with an appropriate tag and data for all mutable information. To load the information, the tags are used again to identify the appropriate data to be retrieved. JSON files will be stored in the work folder in the application.

```
{
    "subject_tag":"data",
    "number_tag":"data",
    "semester_tag":"data",
    "year_tag":"data",
    "title_tag":"data,
    "page_tag":[
        {
            "home_tag":"data",
            "syllabus_tag":"data",
            "schedule_tag":"data",
            "hw_tag":"data"
        }
    ],
    "style_tag":[
```

{

“fav_tag”:”data”,

“nav_tag”:”data”,

“left_tag”:”data”,

“right_tag”:”data”

}

],

“instructor_tag”:[

{

“name_tag”:”data”,

“room_tag”:”data”,

“email_tag”:”data”,

“home_tag”:”data”,

“hours_tag”:”data”,

}

],

“syllabus_tag”:[

{

“desciption_tag”:”data”,

“topics_tag”:”data”,

“prerequisites_tag”:”data”,

“outcomes_tag”:”data”,

“textbooks_tag”:”data”,

“graded_tag”:”data”,

“grading_tag”:”data”,

“academic_tag”:”data”,

“assistance_tag”:”data”,

}

],

“lectures_tag”:[

{

“section_tag”:”data”,

“day_tag”:”data”,

“time_tag”:”data”,

“room_tag”:”data”

}

…

],

“recitation_tag”:[

{

“section_tag”:”data”,

“day_tag”:”data”,

“room_tag”:”data”,

“ta1_tag”:”data”,

“ta2_tag”:”data”

}

…

],

“lab_tag”:[

{

“section_tag”:”data”,

“day_tag”:”data”,

“room_tag”:”data”,

“ta1_tag”:”data”,

“ta2_tag”:”data”

}

…

],

“start_tag”:”data”,

“end_tag”:”data”,

“ta_tag”:[

{

“name_tag”:”data”,

“email_tag”:”data”,

“type_tag”:”data”

}

…

],

“oh_tag”:[

{

“time_tag”:”data”,

“day_tag”:”data”,

“name_tag”:”data”

}

…

],

“start_date_tag”:”data”,

“end_date_tag”:”data”,

“schedule_tag”:[

{

“section_tag”:”data”,

“date_tag”:”data”,

“title_tag”:”data”,

“topic_tag”:”data”

}

…

]

}

subject_tag- tag for the data of the subject combobox

number_tag- tag for the data of the number combobox

semester_tag- tag for the data of the semester combobox

year_tag- tag for the data of the year combobox

title_tag- tag for the data of the title textbox

page_tag- tag for the data of the page section of the application

home_tag- tag for the home checkbox

syllabus_tag- tag for the syllabus checkbox

schedule_tag- tag for the schedule checkbox

hw_tag- tag for the hw checkbox

style_tag- tag for the style section of the application

fav_tag- tag for the image file of the icon

nav_tag- tag for the image file of the icon

left_tag- tag for the image file of the icon

right_tag- tag for the image file of the icon

instructor_tag- tag for the instructor section of the application

name_tag- tag for the name data to be retrieved

room_tag- tag for the room data to be retrieved

email_tag- tag for the email data to be retrieved

home_tag- tag for the home data to be retrieved

hours_tag- tag for the hours data to be retrieved

syllabus_tag- tag for the syllabus section of the application

description_tag- tag for the description data

topics_tag- tag for the topics data

prerequisites_tag- tag for the prerequisites data

outcomes_tag- tag for the outcome sdata

textbooks_tag- tag for the textbooks data

graded_tag- tag for the graded data

grading_tag- tag for the grading data

academic_tag- tag for the academic data

assistance_tag- tag for the assistance data

lectures_tag- the tag for the lectures section of the application

section_tag- tag for the section data

day_tag- tag for the day data

time_tag-tag for the time data

recitation_tag- the tag for the recitation section of the application

ta1_tag- tag for the first ta data

ta2_tag- tag for the second ta data

lab_tag- tag for the lab section of the application

start_tag- tag for the start time data

end_tag- tag for the end time data

ta_tag- tag for the ta section of the application

type_tag- tag for the type data

oh_tag- tag for the office hours section of the application

start_date_tag- tag for the start date data

end_date_tag- tag for the end date data

schedule_tag- tag for the schedule section of the application

date_tag- tag for the date data

title_tag- tag for the title data

topic_tag- tag for the topic data

6 Supporting Information

Note that this document should serve as a reference for those implementing the code, so we'll provide a table of contents to help quickly find important sections.

6.1 Table of Contents

6.2 Appendixes

N/A