The Course Site Generator

Software Design Description

Author: Ishan Soni
Version 1.0
Abstract: This document describes the software design for Course Site Generator Application
Based on IEEE Std 1016TM-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.

1.3 Definitions, acronyms, and abbreviations

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”.

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.

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

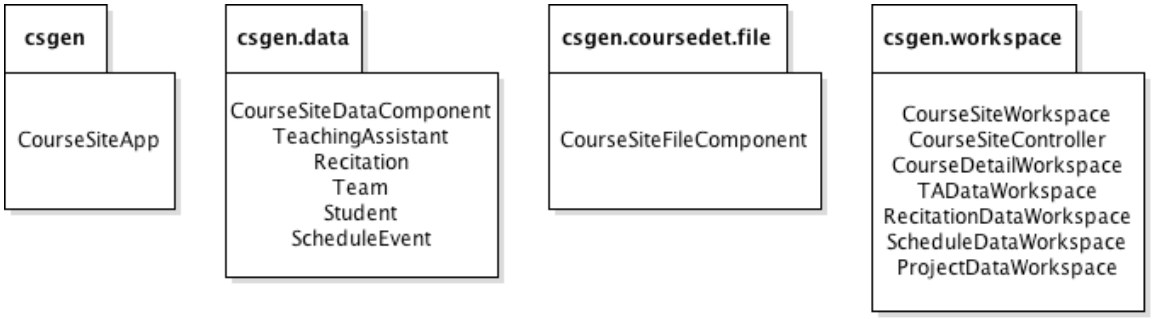
1.5 Overview

This Software Design Description document provides a working design for the Course Site Generator software application as described 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

2.1 Course Site Generator Overview

This figure below specifies all the required packages and classes for the implementation of this software. The classes implements and are extension of the DesktopJavaFramework for which the design is not included.



3 Class-Level Design Viewpoint

This section will show design for each class in detail. Figure 3.1 shows how all the different components of the application ties together as one.

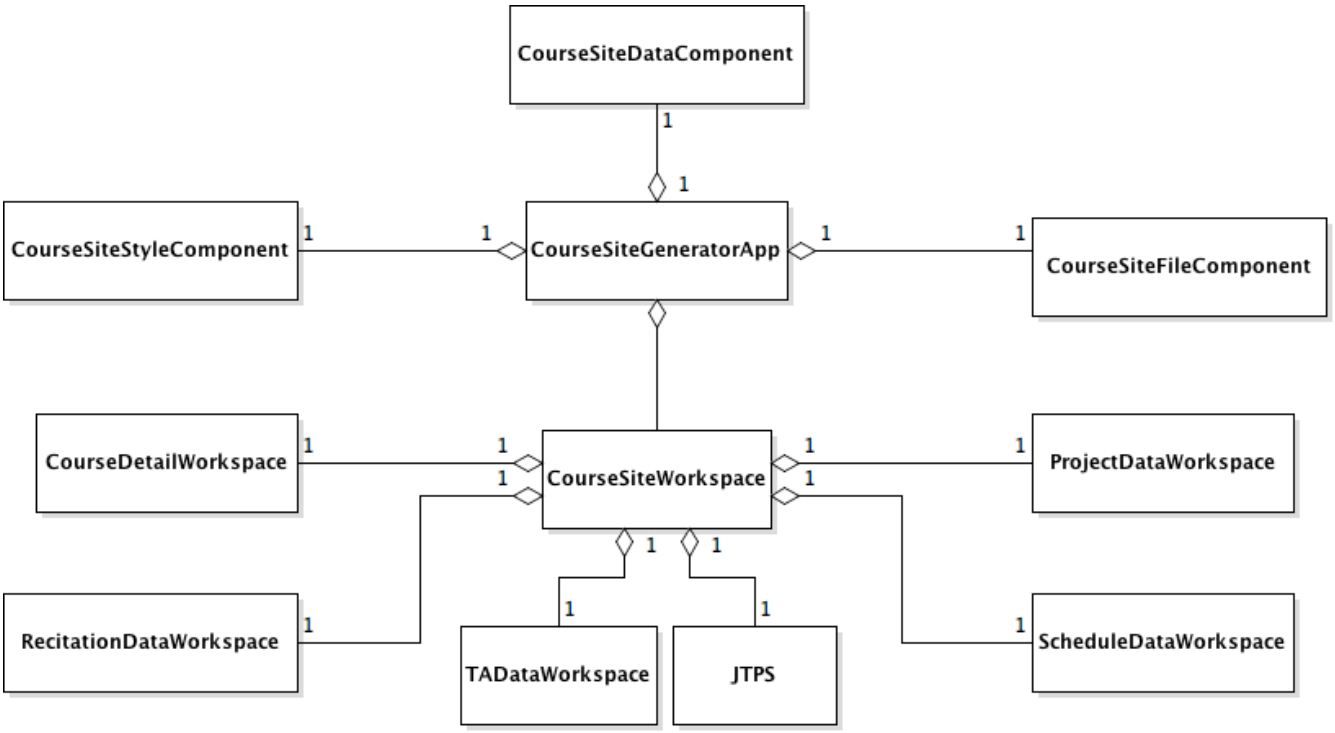


Figure 3.1

In Figure 3.2 the central class of the framework is being extended by the central class of the course site generator app.

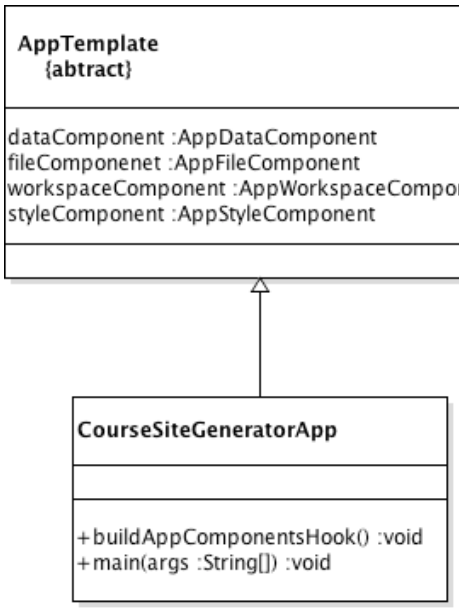
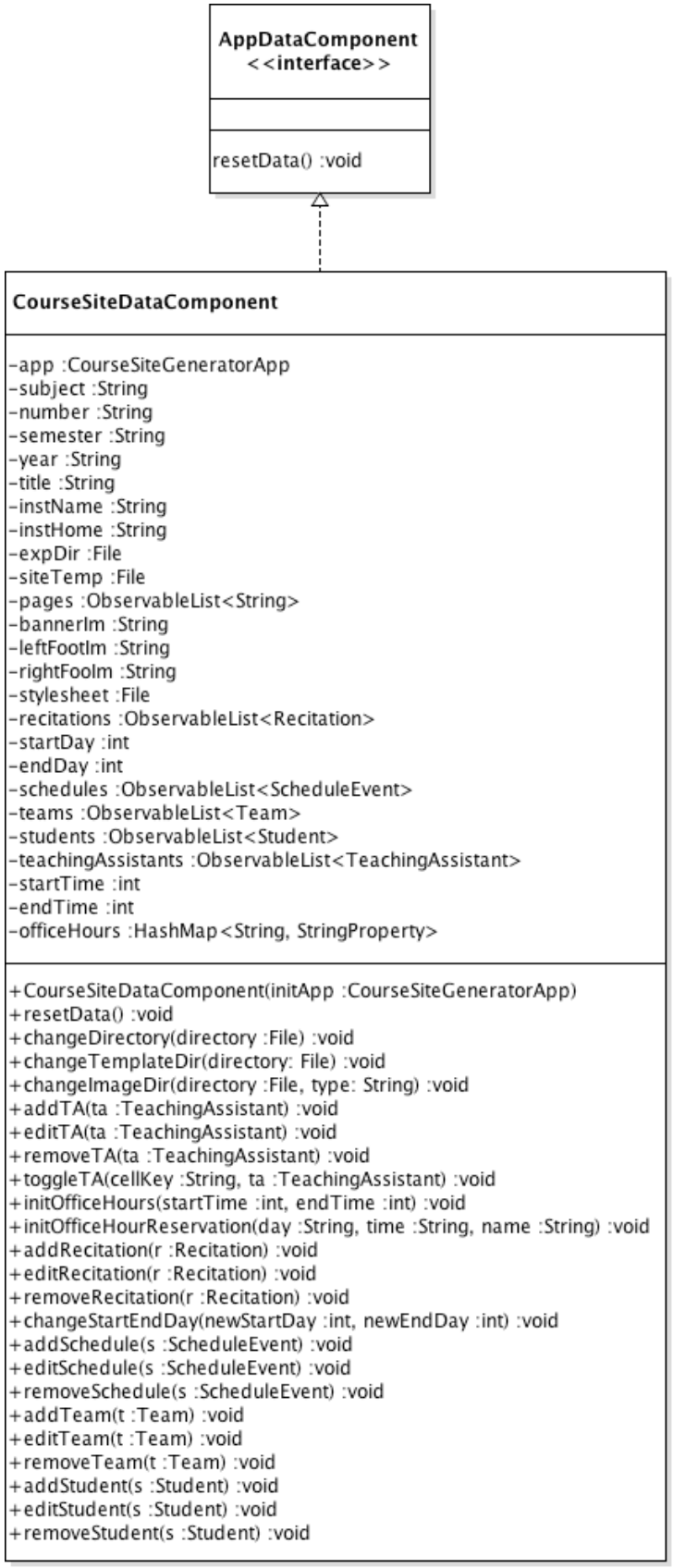


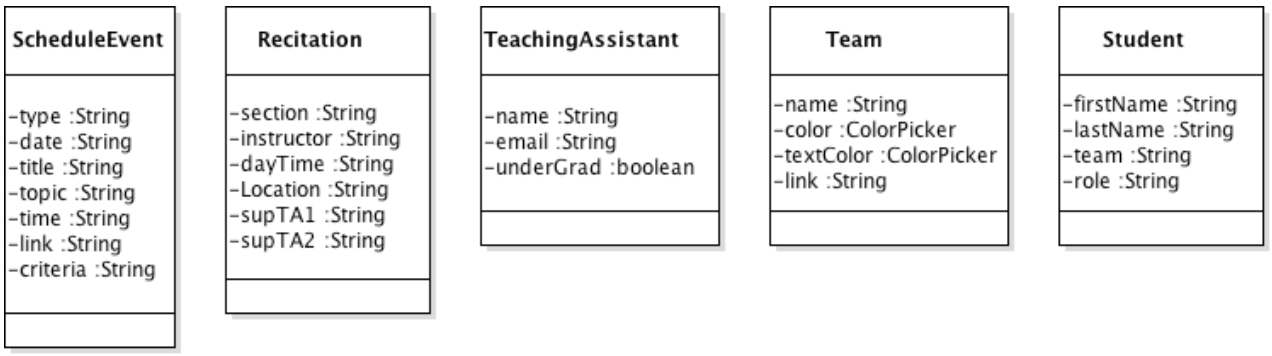
Figure 3.2

The figures below will show class designs for each of the different component of the application.

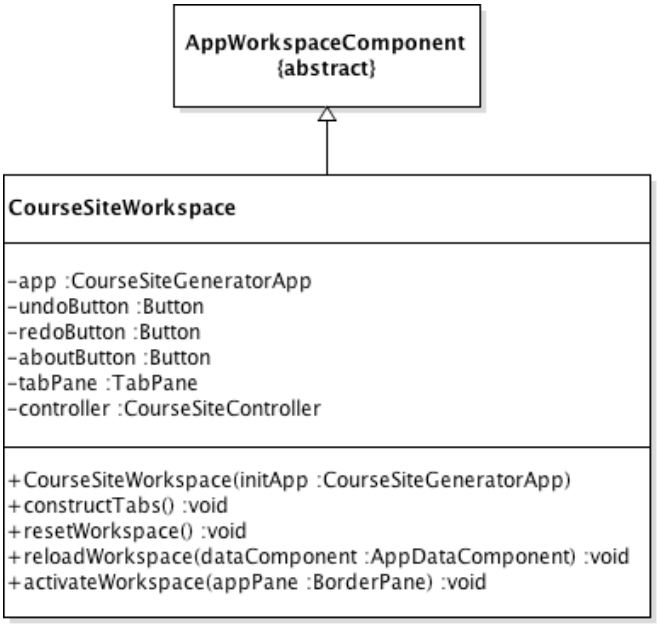
Data Component-



The figure above is the data component of the application, which holds all the important data that is displayed and contained in the application. The class implements an interface from DJFramework. The classes in the figure below are also composed in the data component class.



WorkspaceComponent-



Workspace component holds and constructs all the UI elements that are needed in the application. The following classes below constructs the UI elements for each of the tabs in the application.

UI elements for Course Detail Tab-

UI Elements for TAData and RecitationData tab-

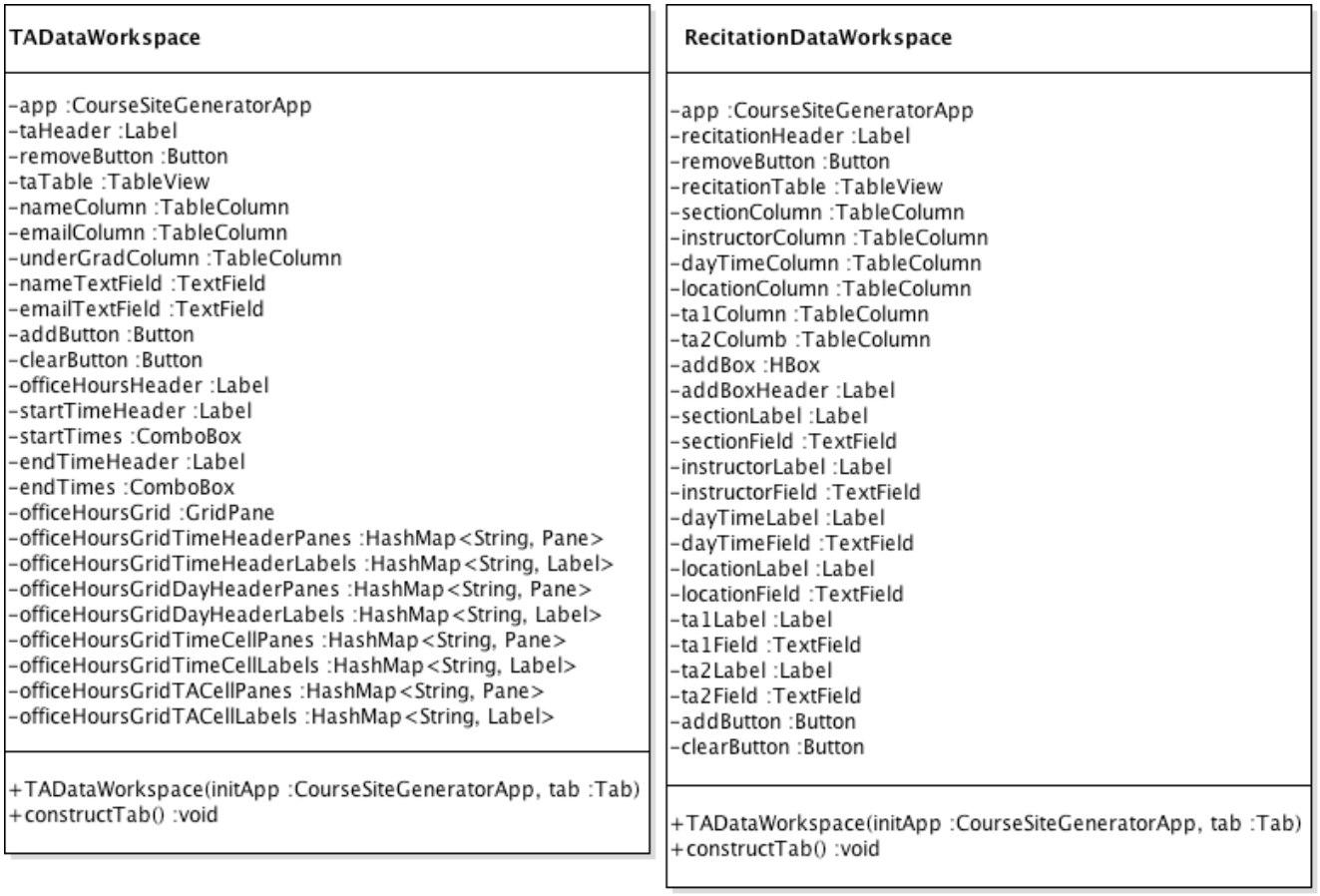
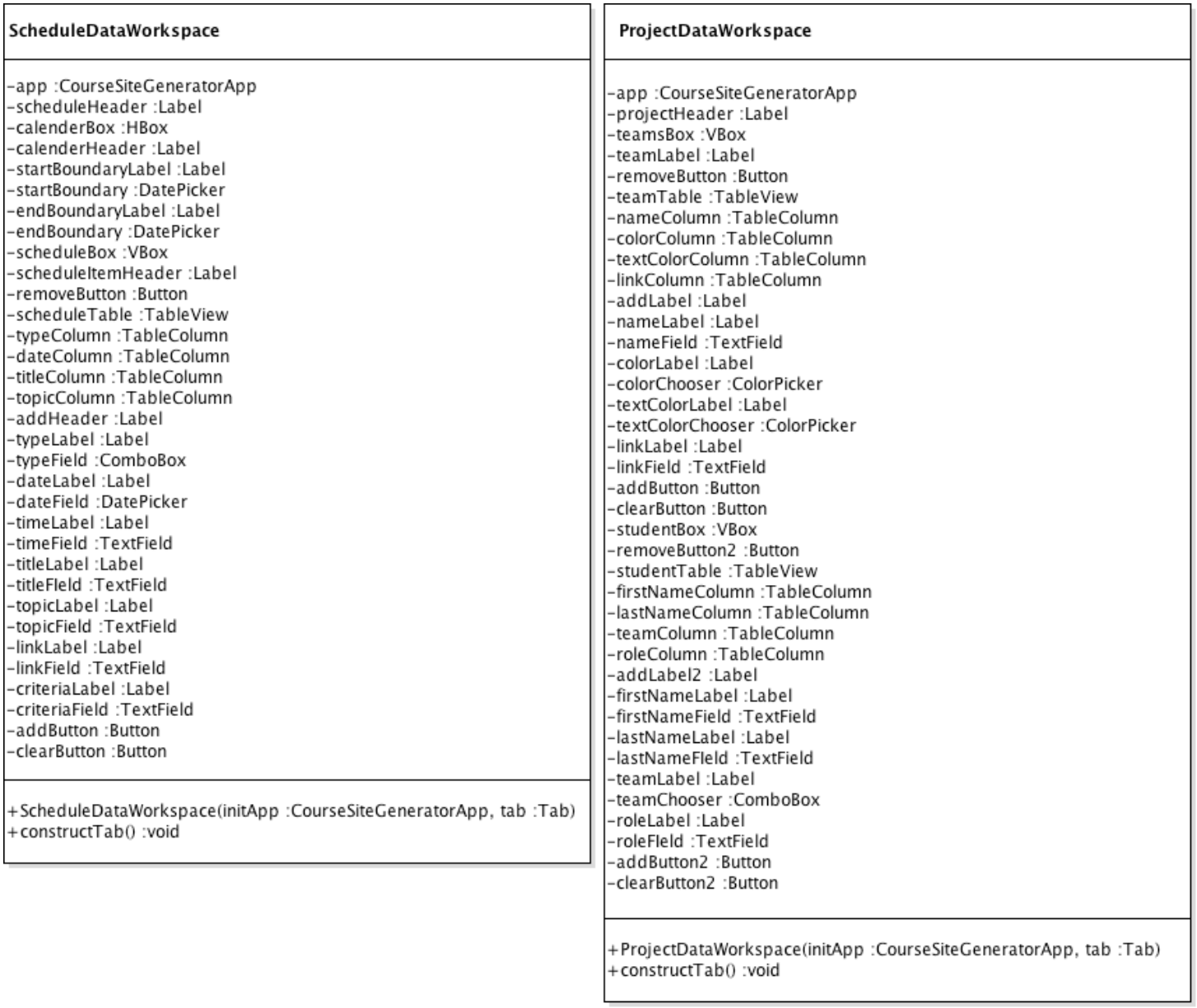
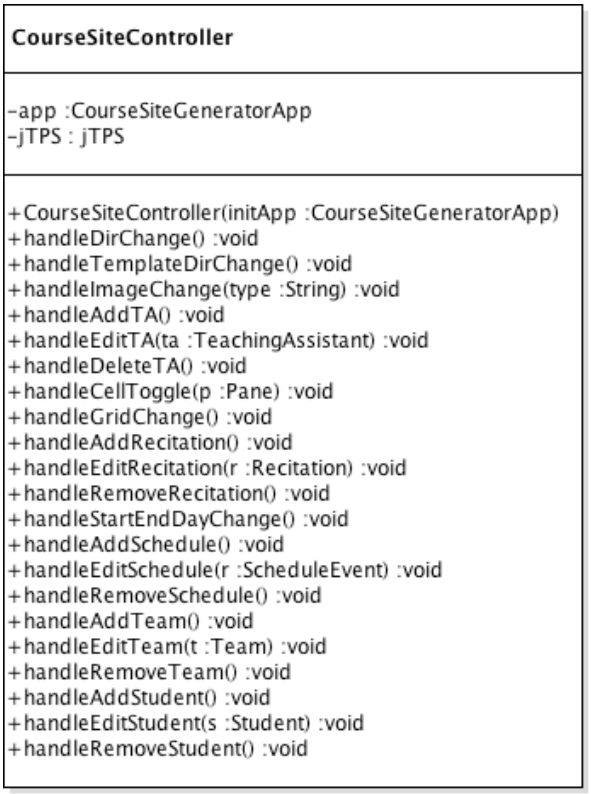


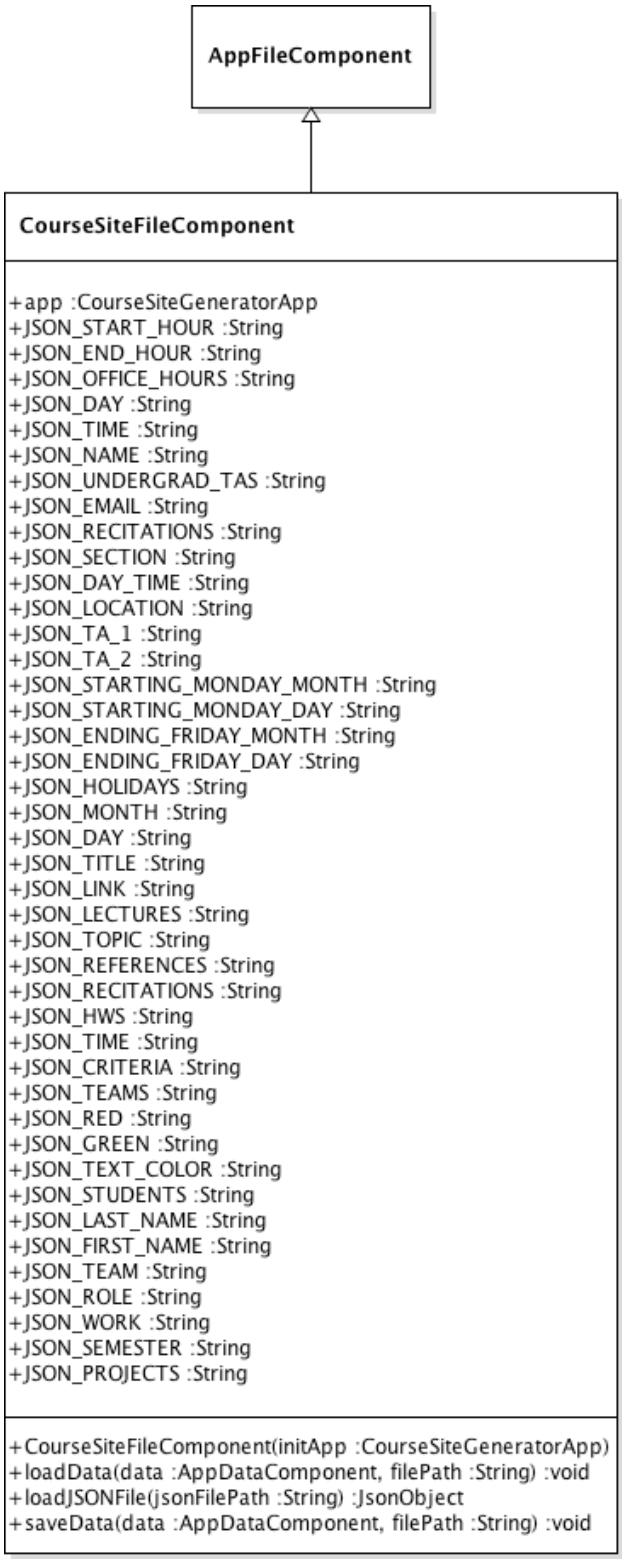
Figure for ScheduleData and ProjectData Tab-



The figure below shows CourseSiteController which holds the JTPS Framework and all the event handler controls for the UI.

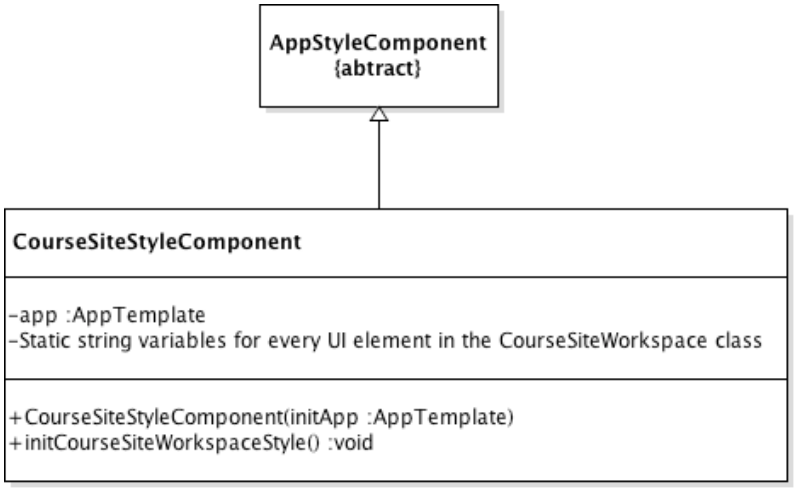


FileComponent-



The figure above shows the file component which holds all the information to load and save data from JSON files.

Style Component-



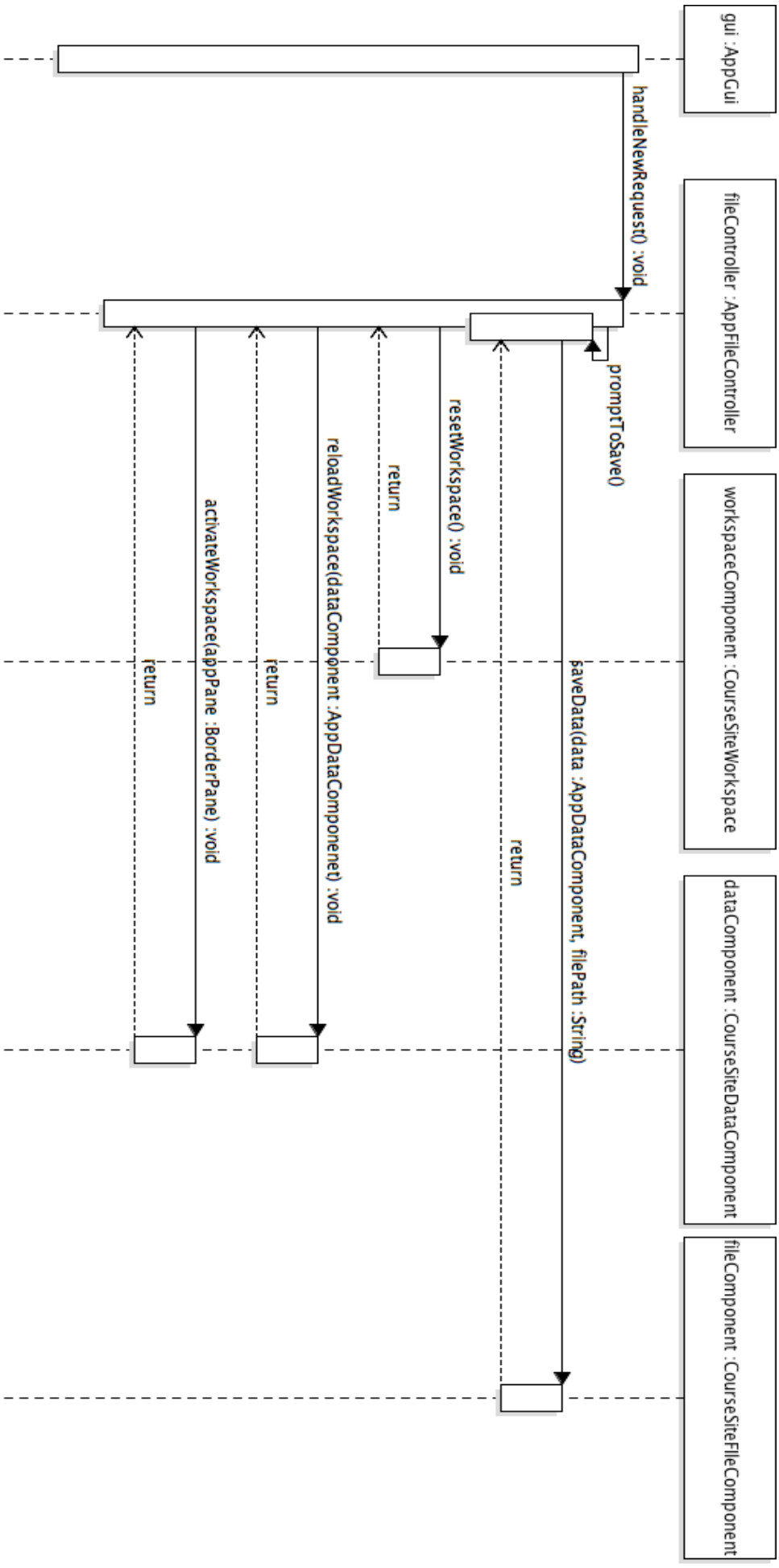
The figure above shows style component class which will hold the information to access styles from the provided css files. The fields for the class have been left out to keep the diagram short. The fields for the class above essentially consists of all the UI elements so they can be styled in the css files.

4 Method-Level Design Viewpoint

The figure below shows sequence diagrams for each use-cases stated in SRS.

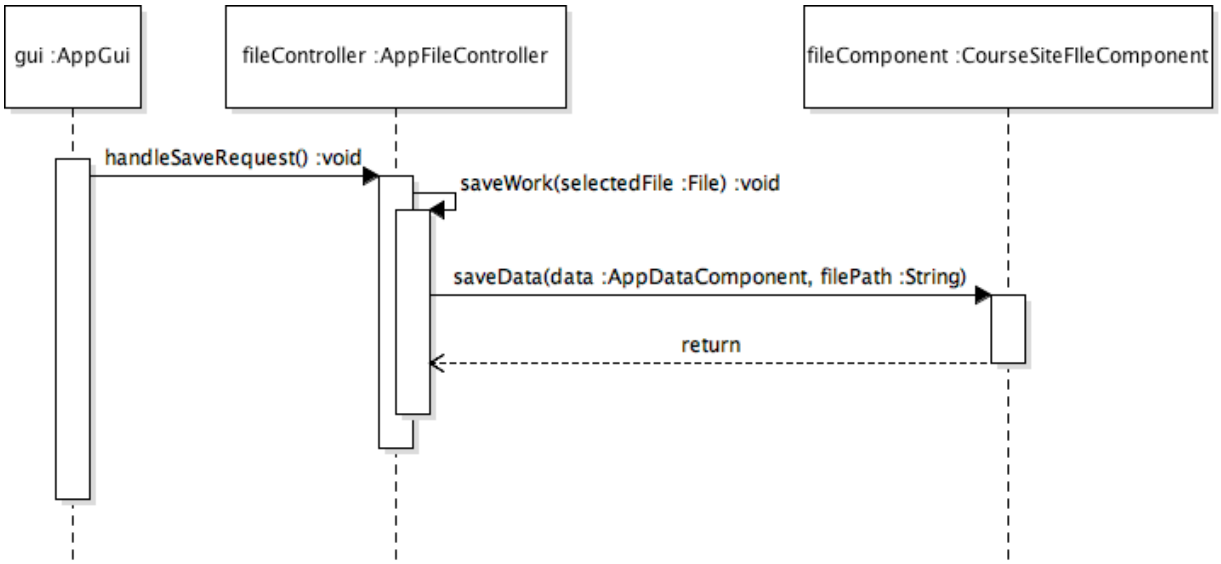
NOTE- The changes in Use cases 2.9, 2.10, 2.13, 2.15, and 2.19 take place once the work has been saved or exported which cannot be shown through a sequence diagram, so those use cases have been omitted.

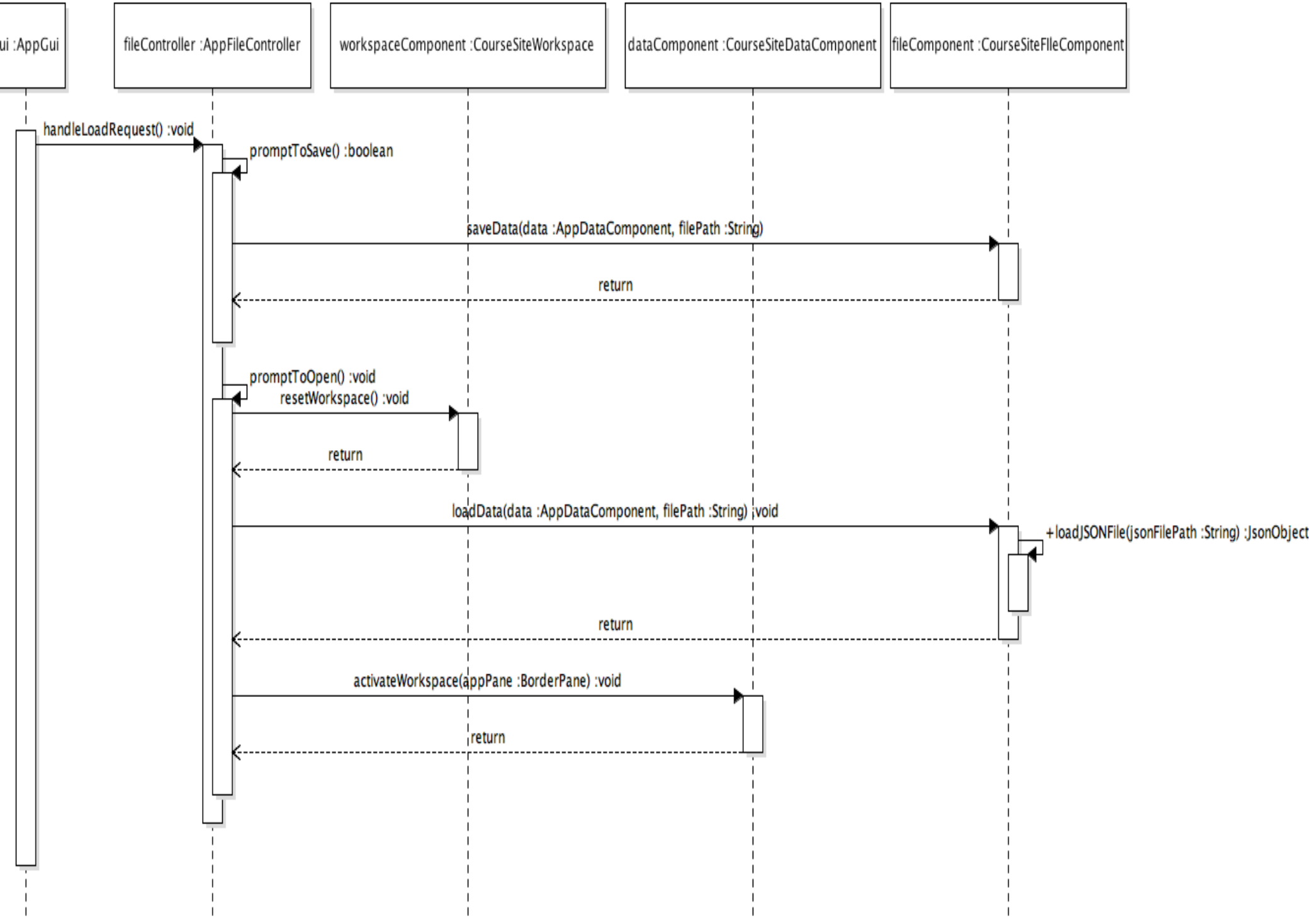
Use Case 2.1: Create New Course Site-



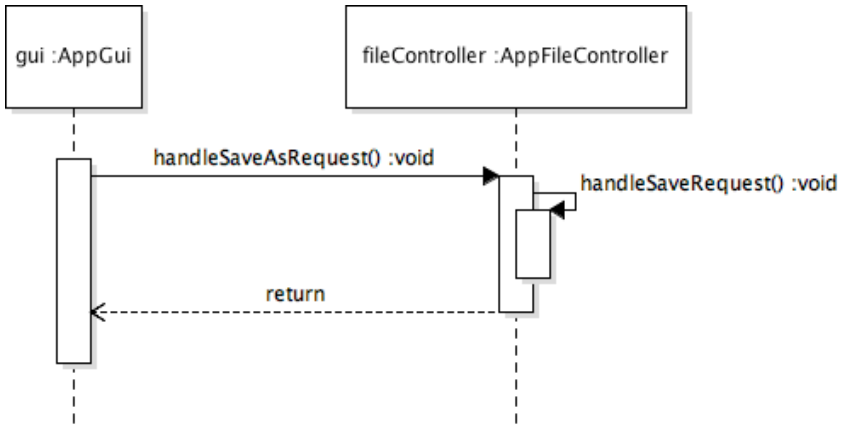
Use Case 2.2: Load Course Site

Use Case 2.3: Save Course Site-

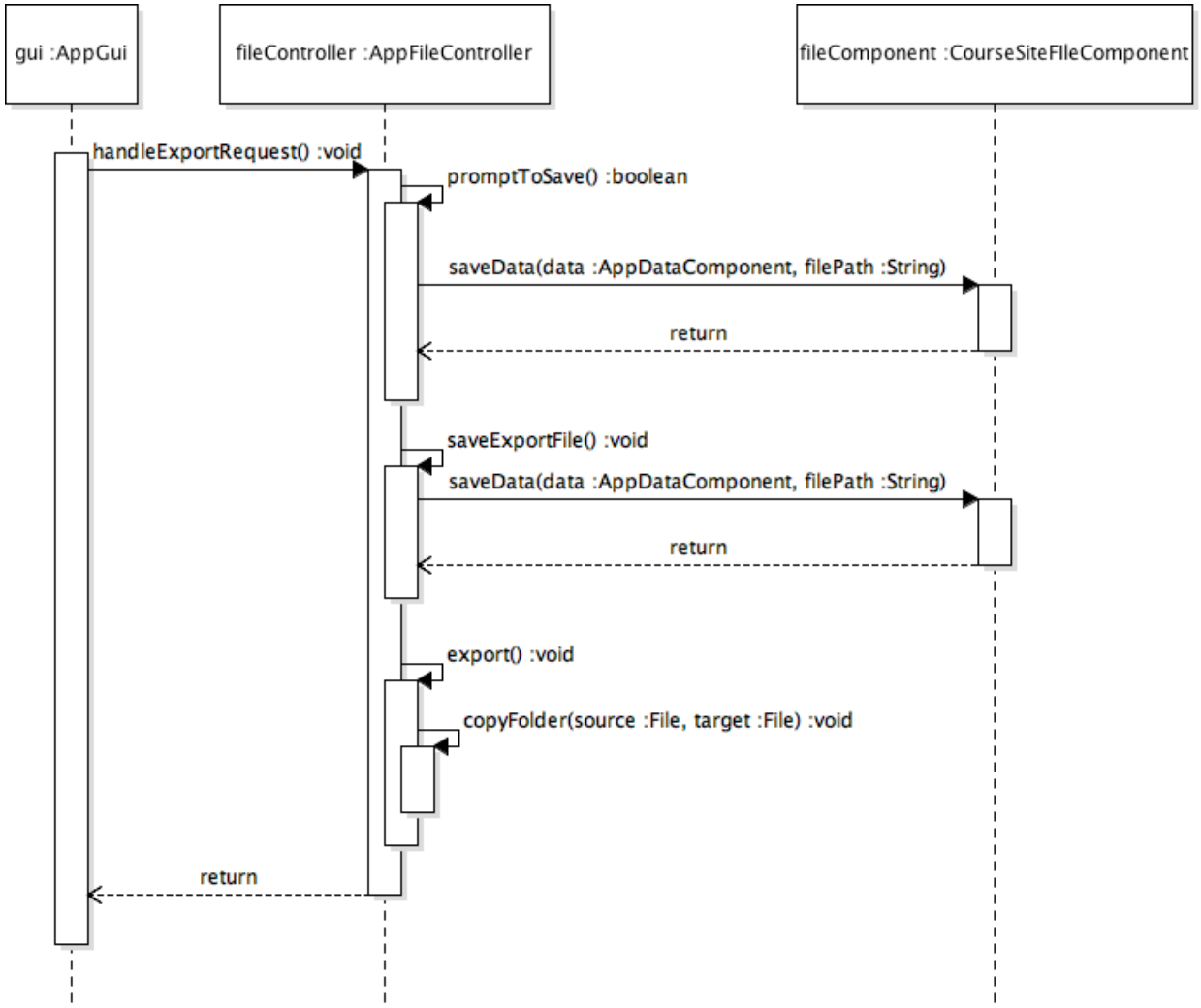



ui :AppGui
fileController :AppFileController
workspaceComponent :CourseSiteWorkspace
dataComponent :CourseSiteDataComponent
fileComponent :CourseSiteFIleComponent
handleLoadRequest() :void
promptToSave() :boolean
saveData(data :AppDataComponent, filePath :String)
return
promptToOpen() :void
resetWorkspace() :void
return
loadData(data :AppDataComponent, filePath :String) :void
+loadJSONFile(jsonFilePath :String) :JsonObject
return
activateWorkspace(appPane :BorderPane) :void
return

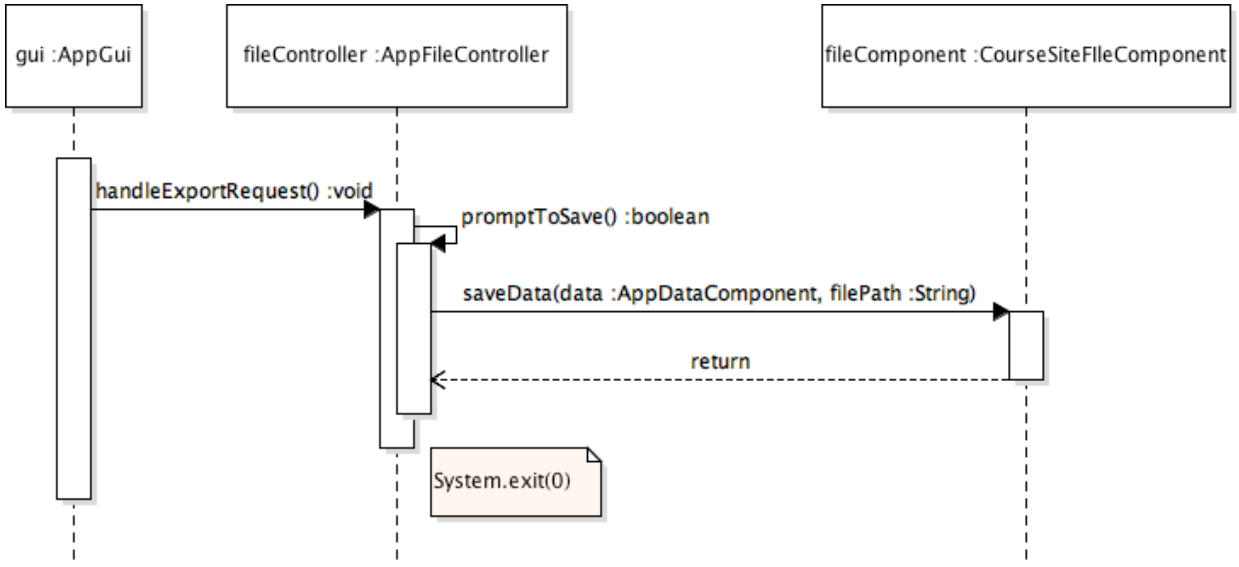
Use Case 2.4: Save As Course Site-



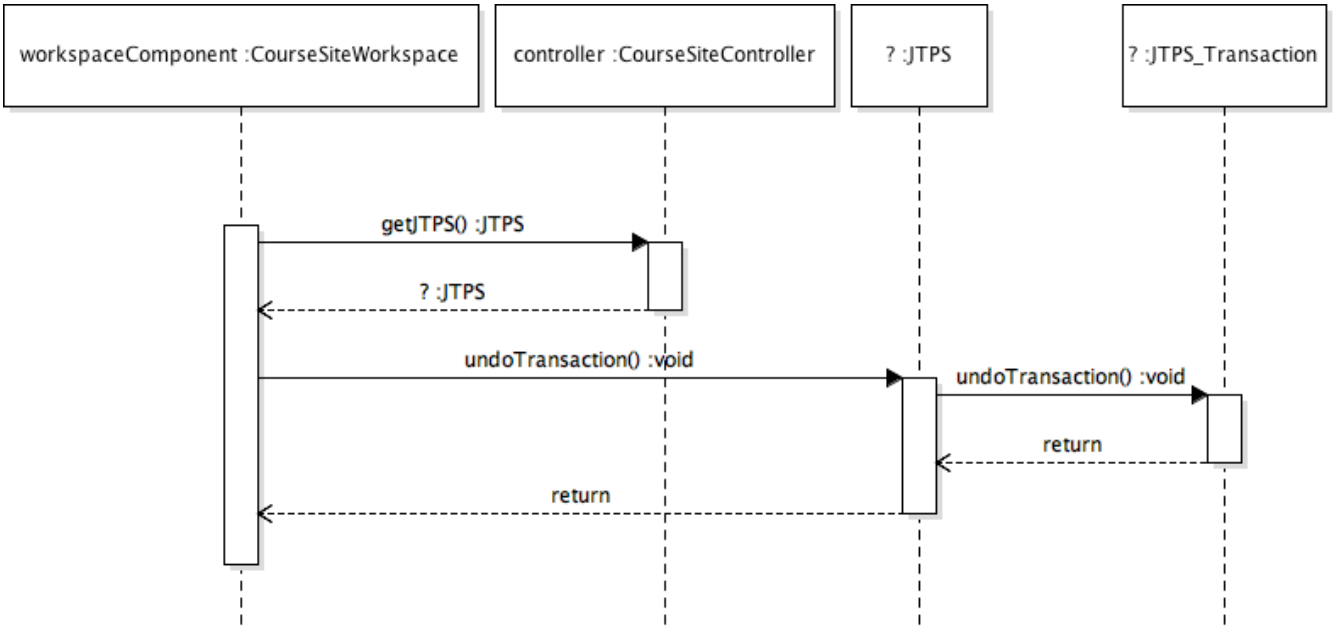
Use Case 2.5: Export Course Site-



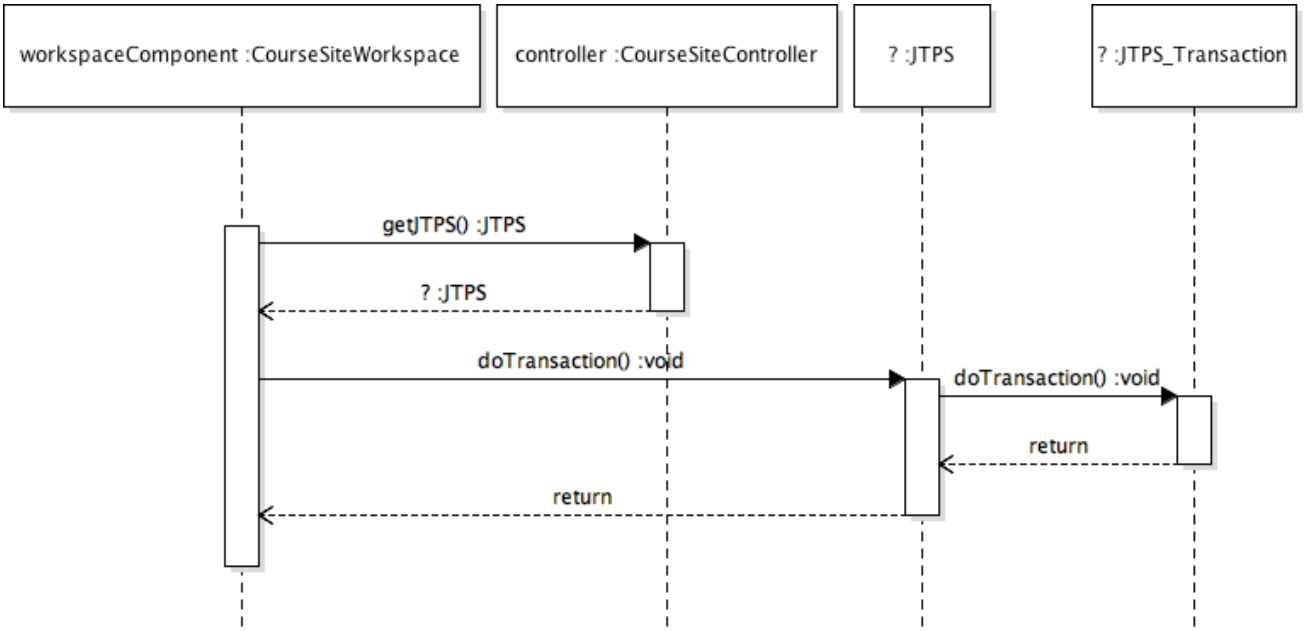
Use Case 2.6: Exit Application-



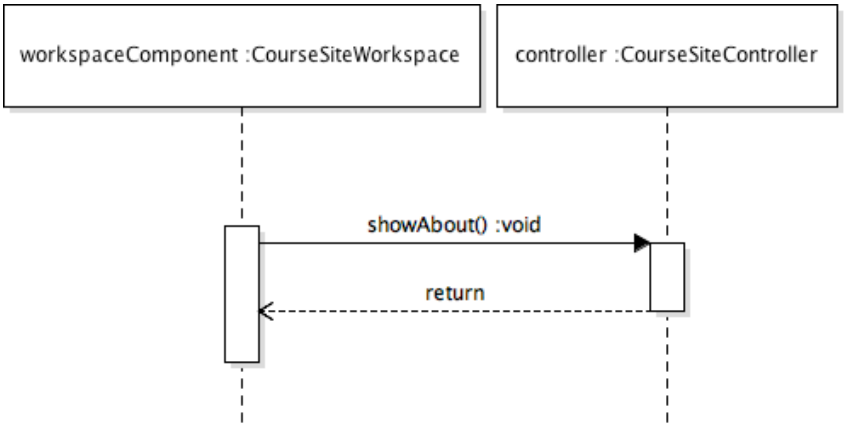
Use Case 2.7: Undo-



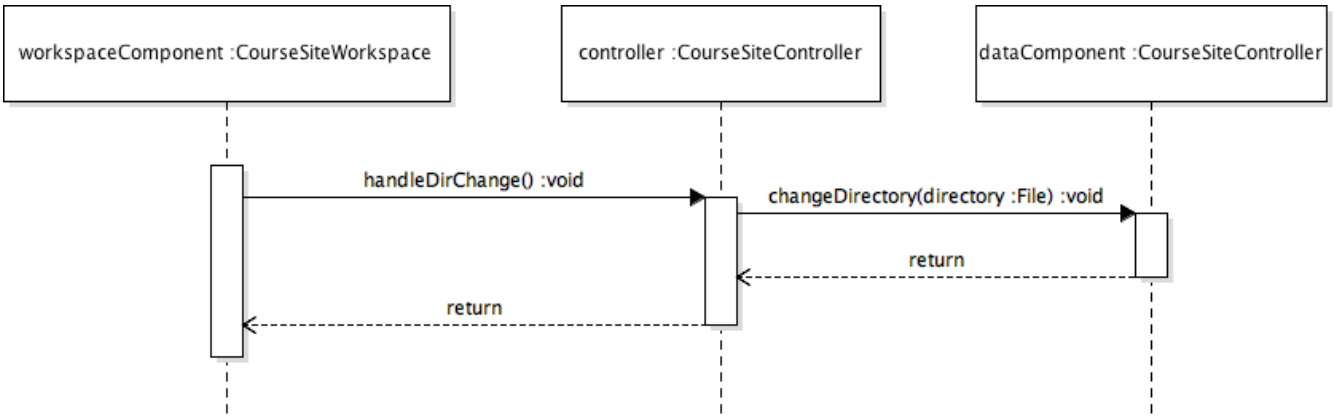
Use Case 2.8: Redo-



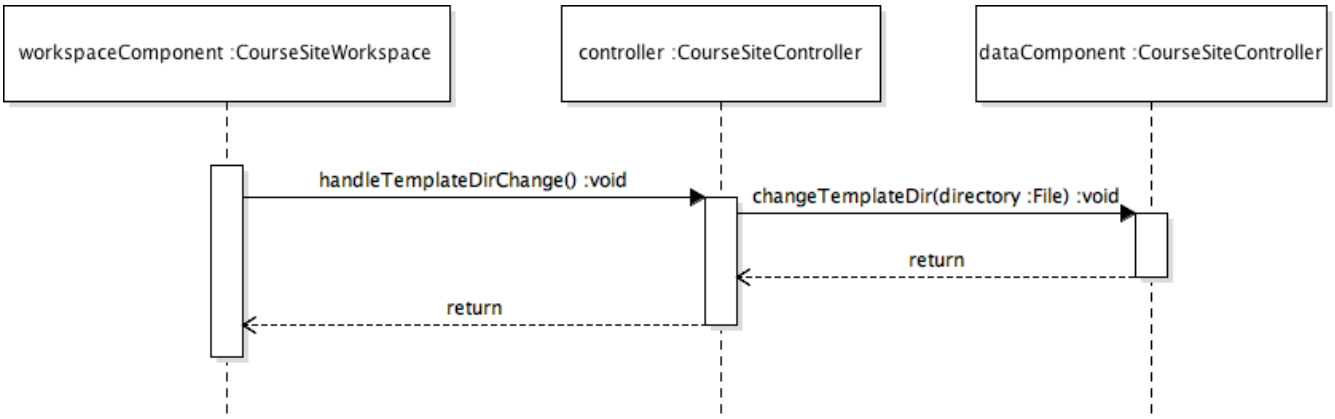
Use Case 2.9: About-



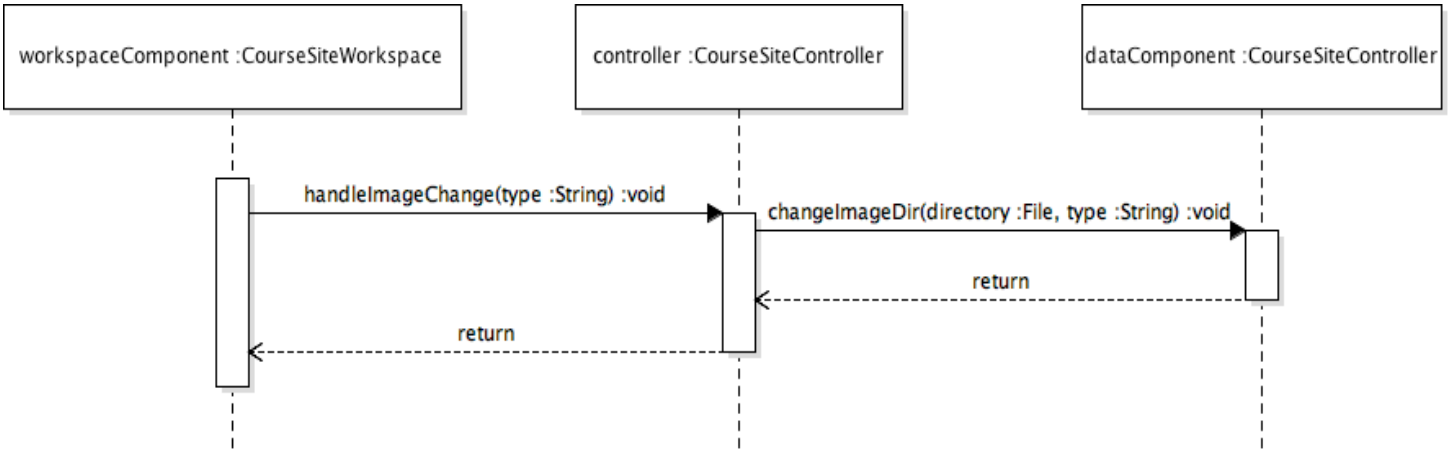
Use Case 2.11: Select Export Directory-



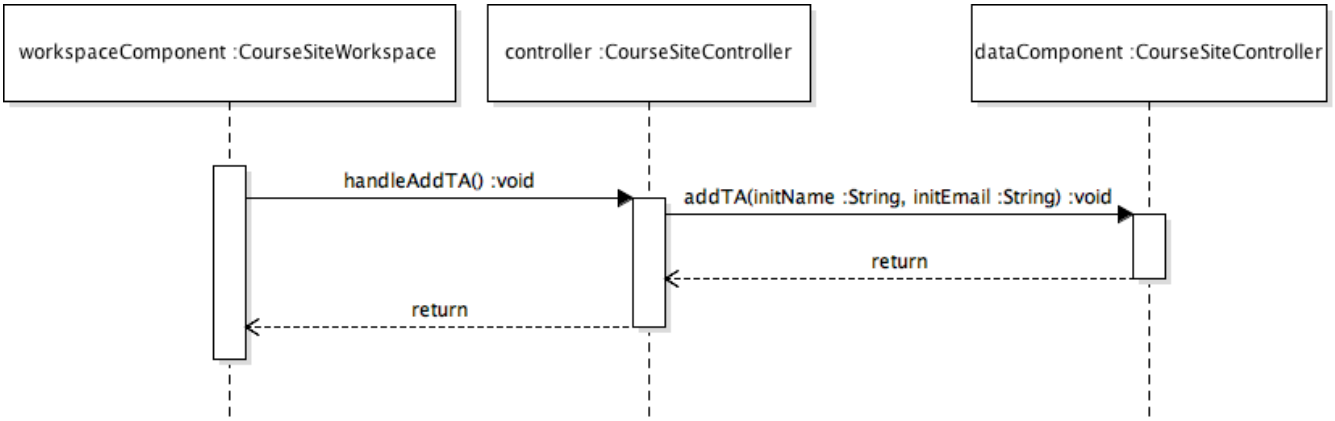
Use Case 2.12: Select Template Directory-



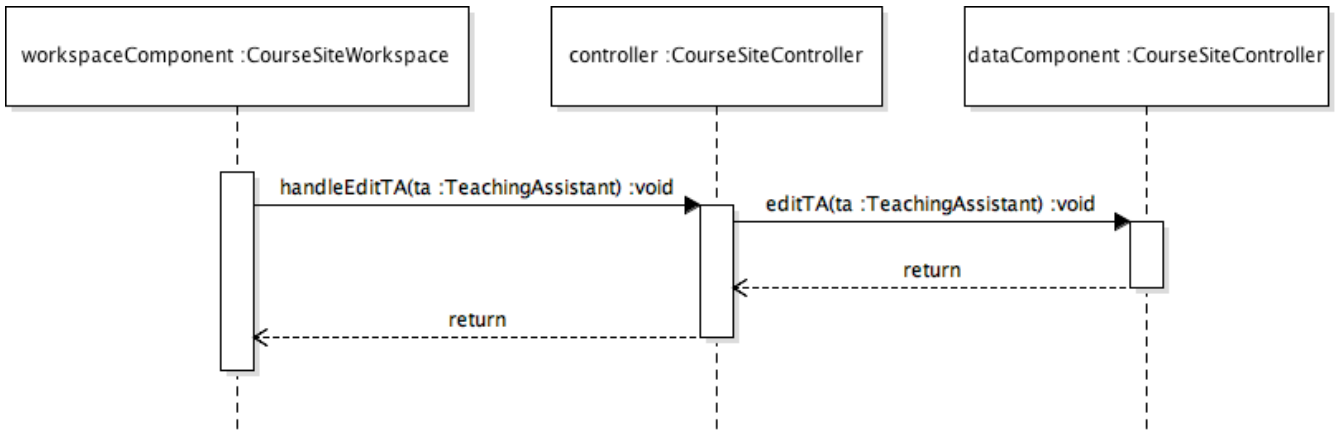
Use Case 2.14: Select Branding Images-



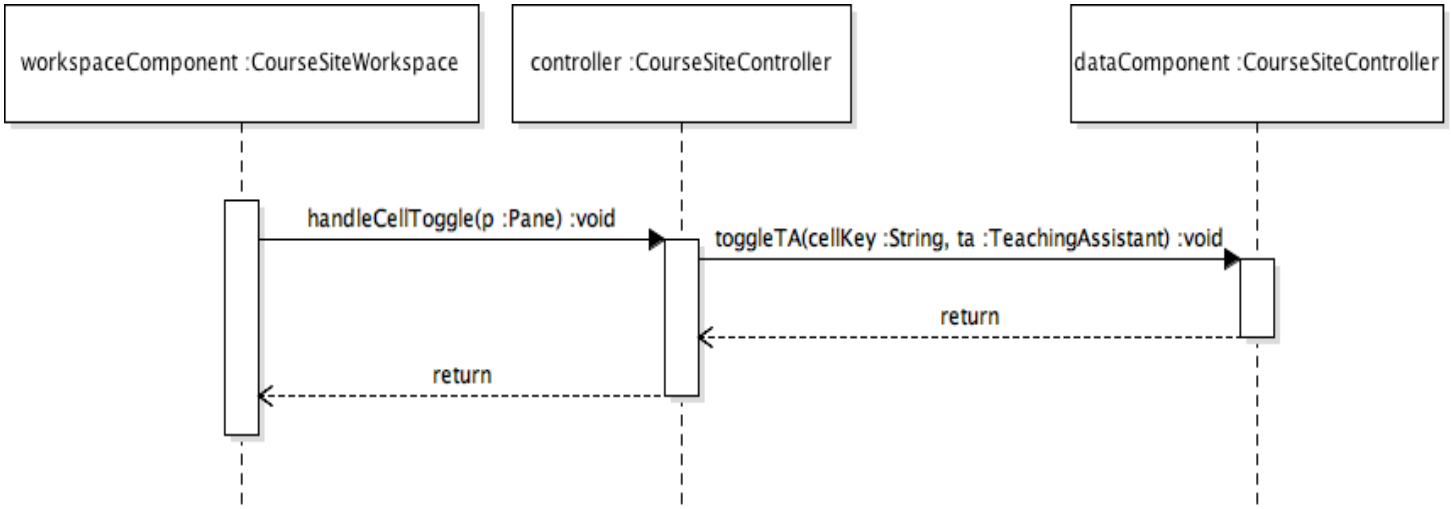
Use Case 2.16: Add Teaching Assistant-



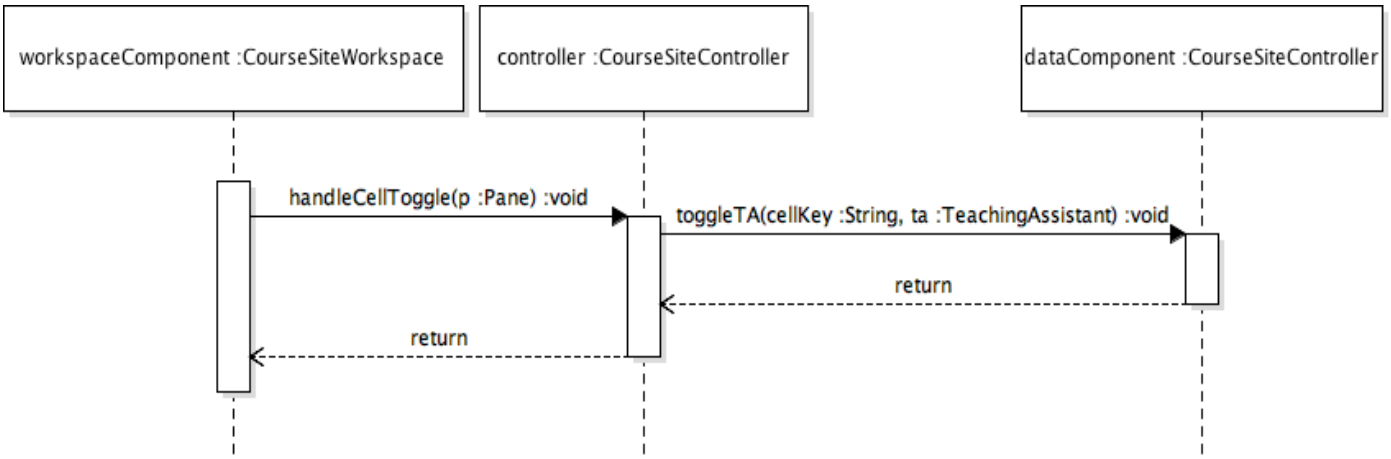
Use Case 2.17: Edit Teaching Assistant-



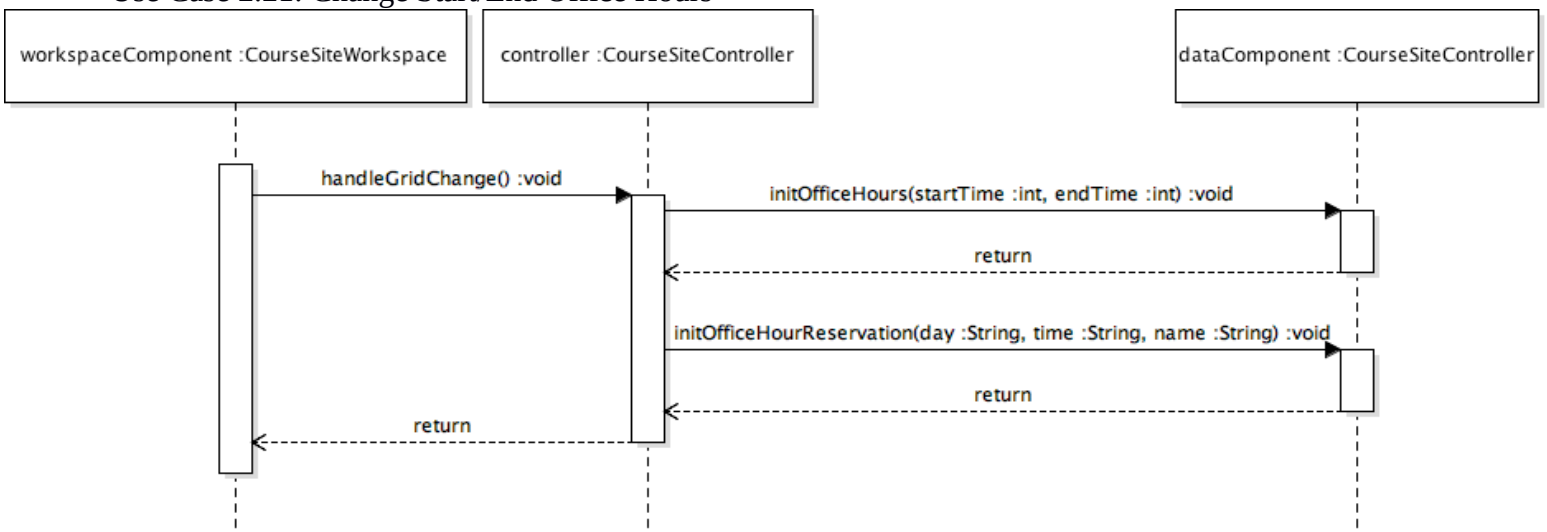
Use Case 2.18: Remove Teaching Assistant-



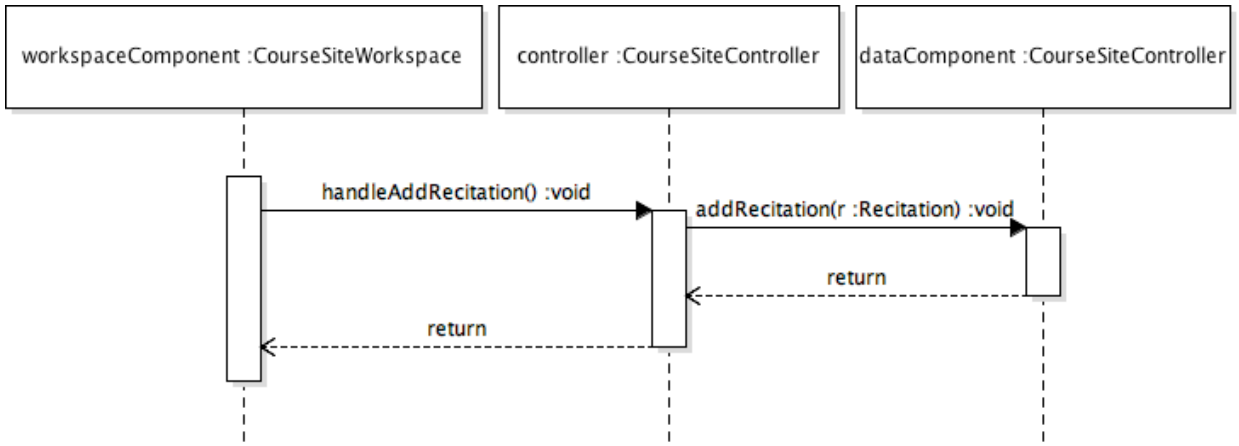
Use Case 2.20: Toggle TA Office Hours-



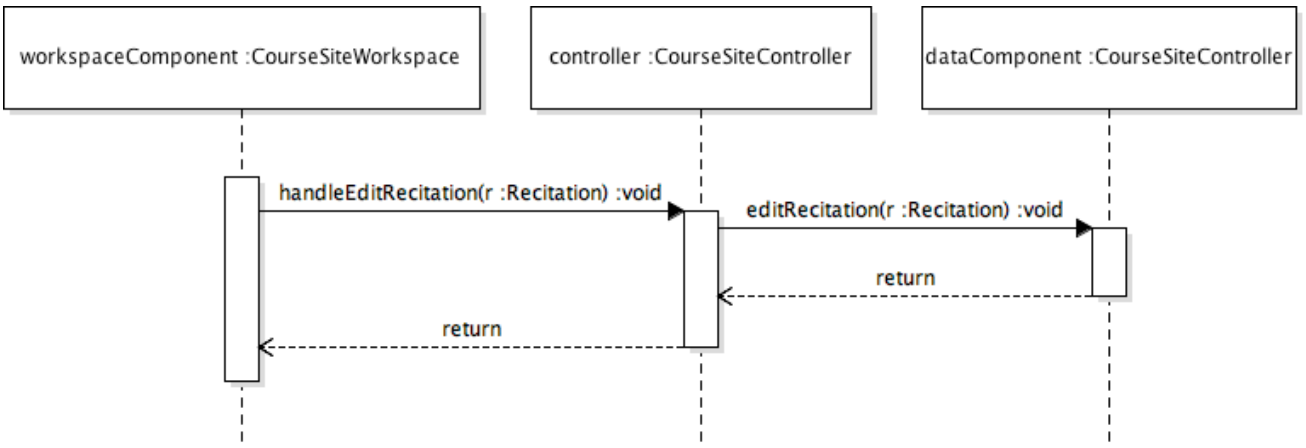
Use Case 2.21: Change Start/End Office Hours-



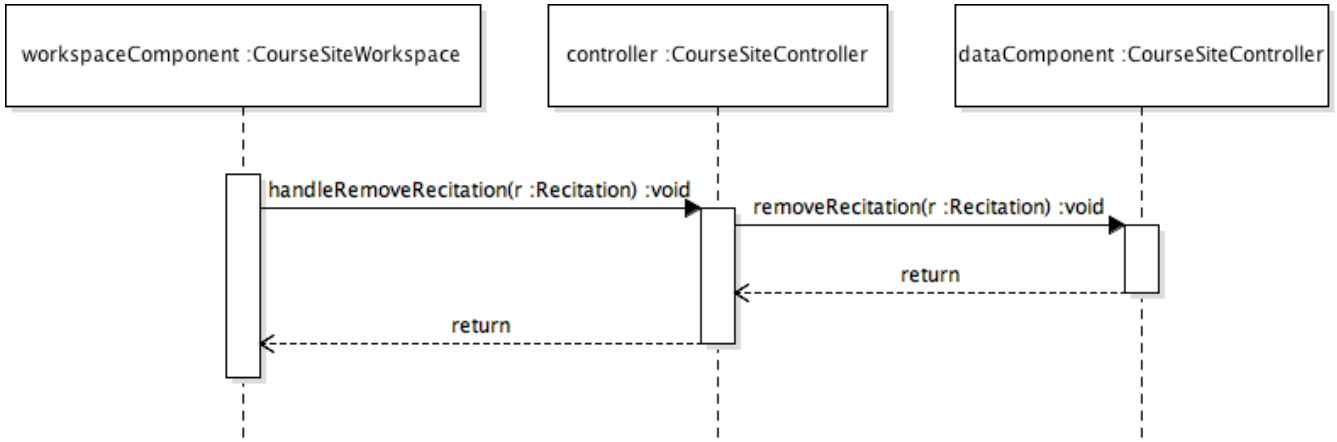
Use Case 2.22: Add Recitation-



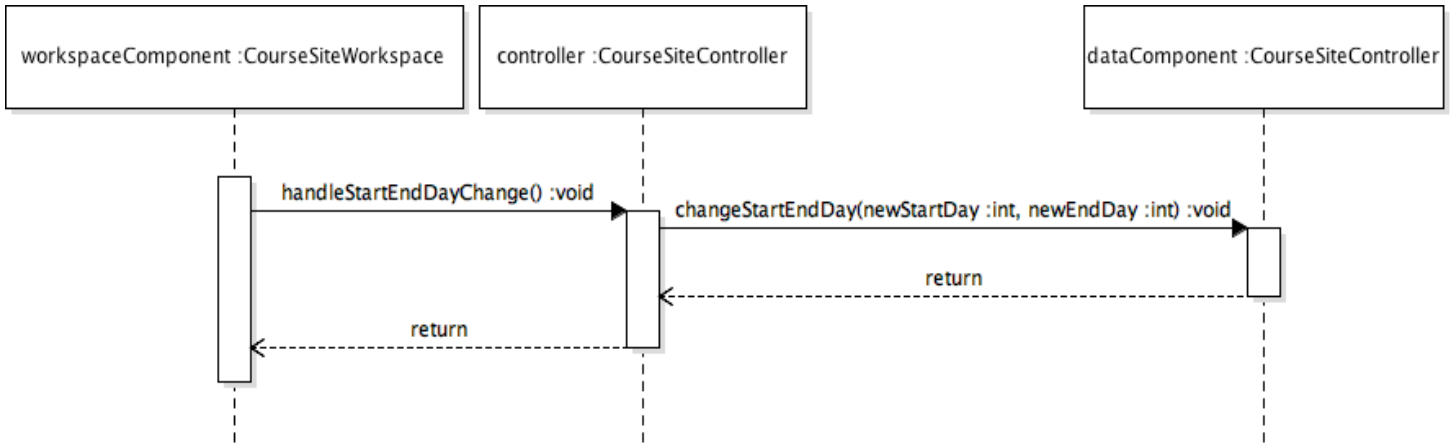
Use Case 2.23: Edit Recitation-



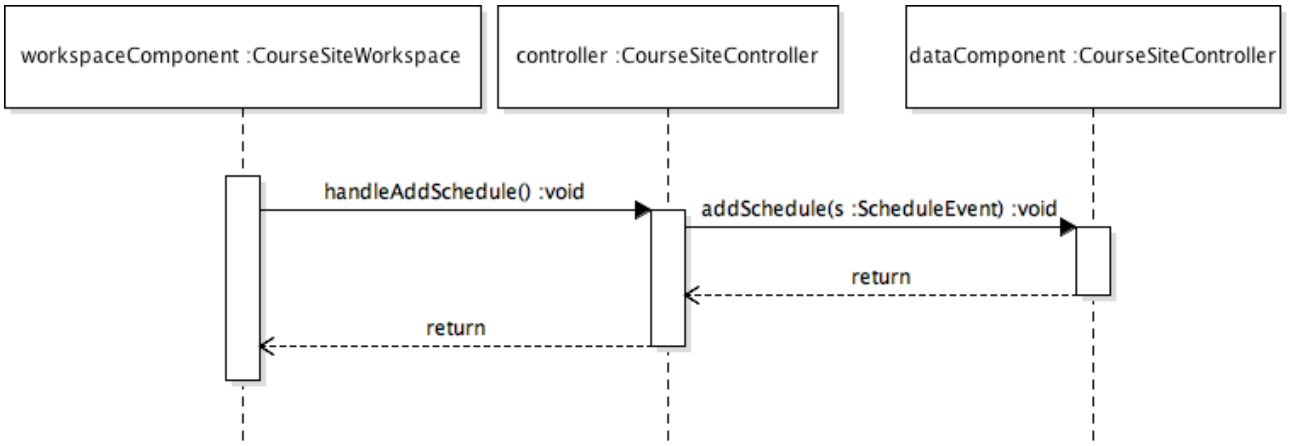
Use Case 2.24: Remove Recitation-



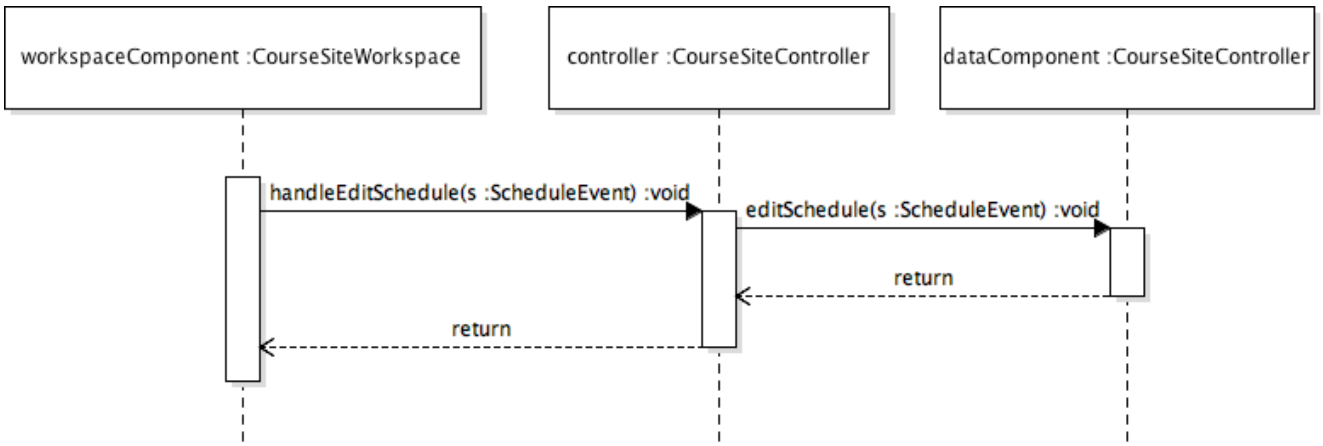
Use Case 2.25: Edit start and end dates-



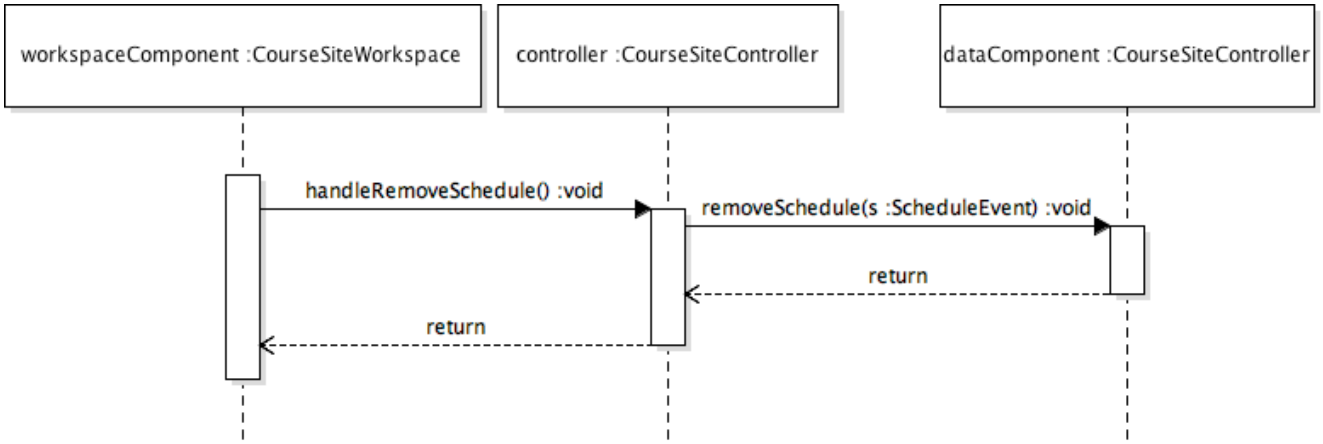
Use Case 2.26: Add Schedule Item-



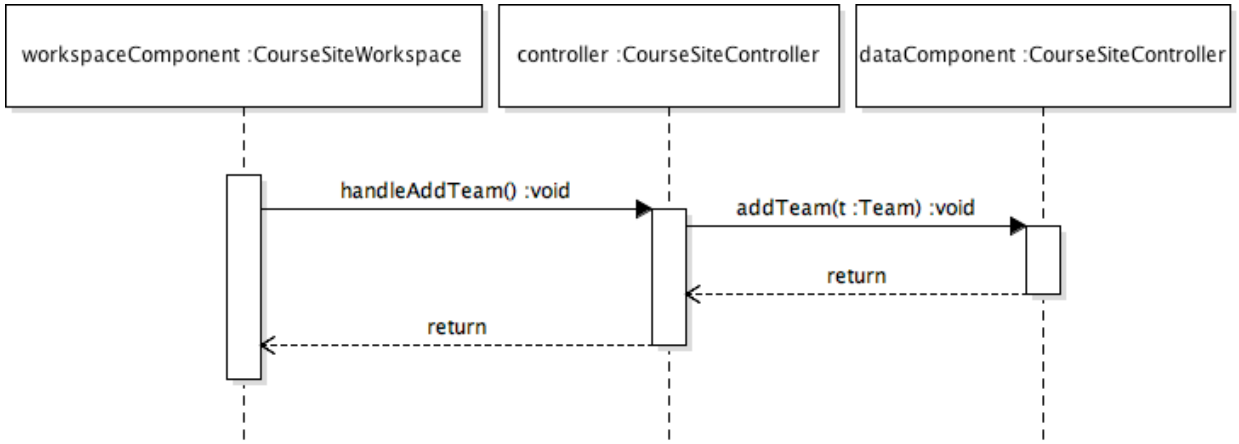
Use Case 2.27: Edit Schedule Item-



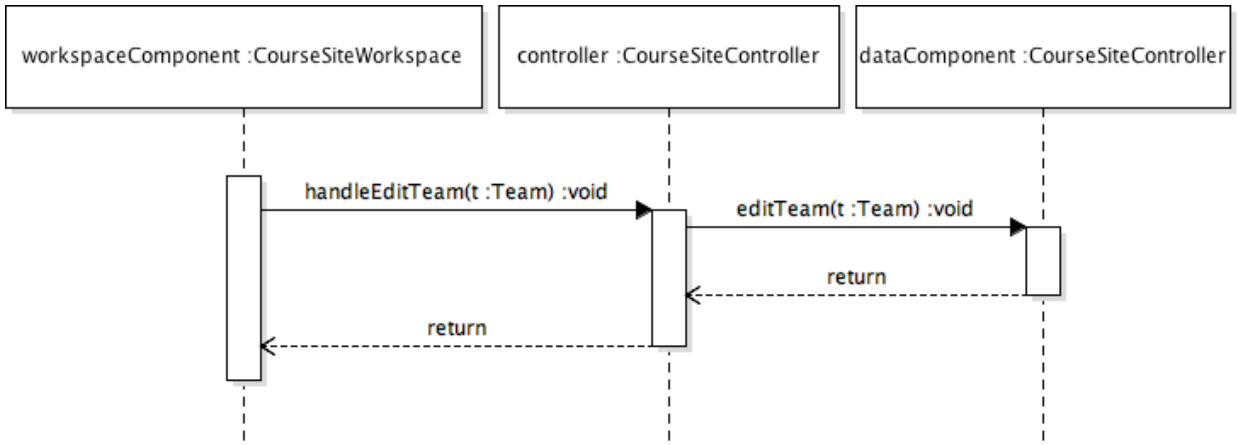
Use Case 2.28: Remove Schedule Item-



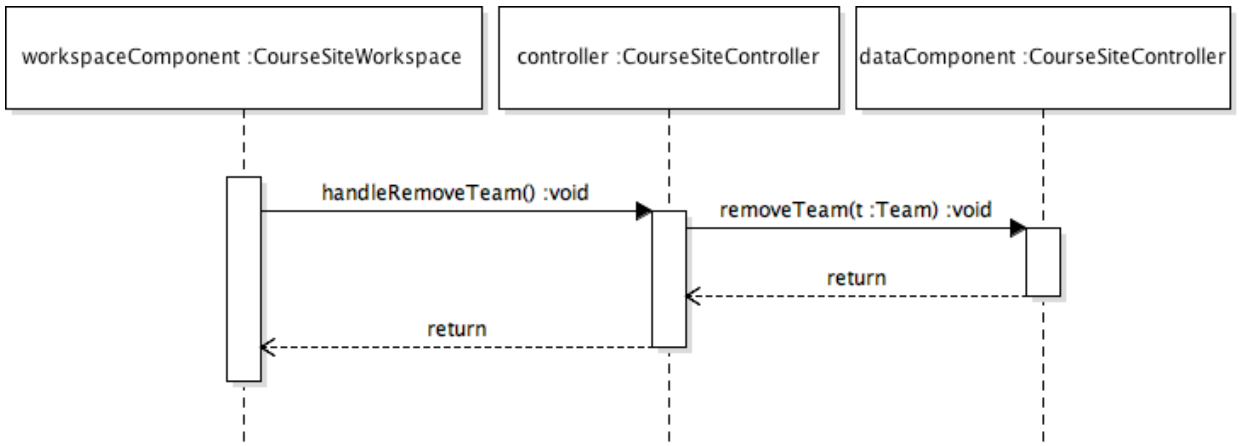
Use Case 2.29: Add Team-



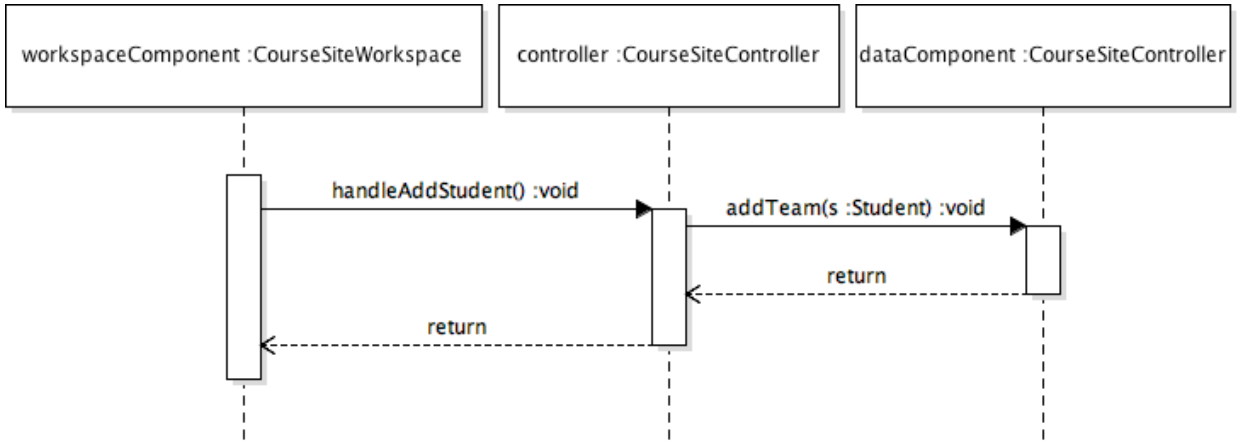
Use Case 2.30: Edit Team-



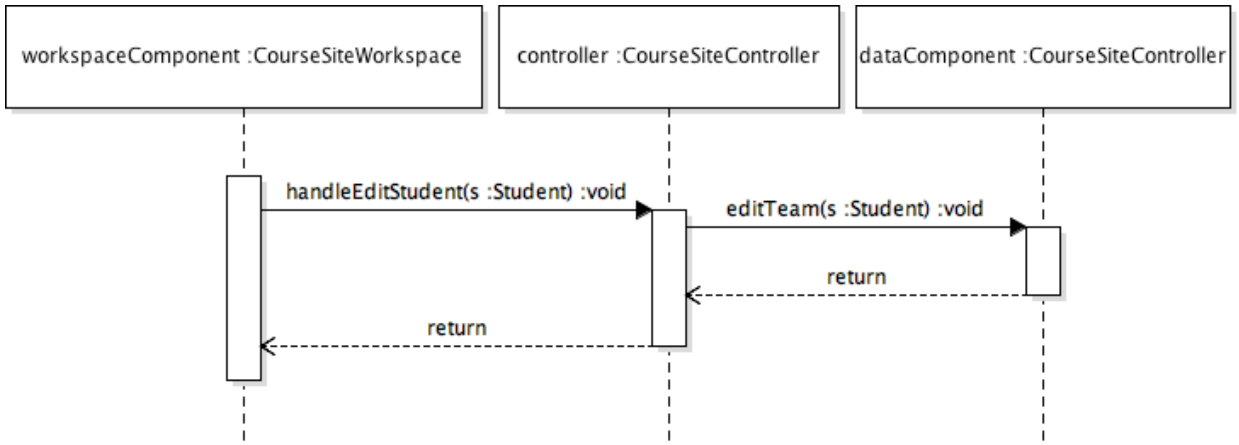
Use Case 2.31: Remove Team-



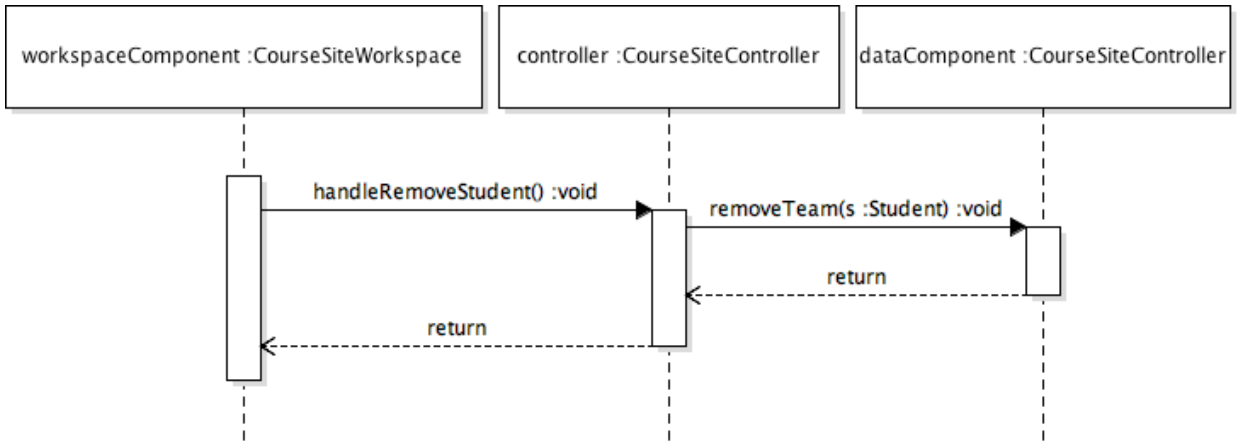
Use Case 2.32: Add Student-



Use Case 2.33: Edit Student-



Use Case 2.34: Remove Student-



5. File Structure and Formats

5.1 Directory Structure

The output website will contain:
CSS folder- this folder will include the stylesheets for the website template.
Images folder- Any images that might be used in the website itself.
JS folder- will contain all the JSON files generated from the software to display data on the website.
Index.html and other html files for each page that was selected in the website.

5.2 Data File Formats

JSON files will contain all the necessary data that will be displayed on the website such as TA names, office hours, Recitations, Calendar info, etc.

5.3 Additional Files

The javax.json-1.0.4.jar will be required as a library for this project.

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