# The *Course Site Generator* ™
# Software Requirements Specification


**Author:** Richard McKenna
Debugging Enterprises™

**Based on IEEE Std 830™-1998 (R2009) document format**

# 1 Introduction

University courses are required to provide course materials at the start of a term that help students understand course requirements and plan their semesters. These come in the form of a course syllabus and schedule that list information like course policies and dates for exams and assignments. Many times instructors find the most convenient means to disseminate this information is by way of a course Web site. Course Web sites are typically published at the start of a semester and are updated as it progresses. Such a site keeps the students up to date on deadlines and provides a place for instructors to distribute things like lecture slides and assignments.

Places like Stony Brook University's Computer Science Department have long required a Course Web Site for each taught course, and so every semester, the instructors teaching these courses generate and update this content. This process can be time consuming and tedious. In addition, many times instructors have more important things to do than build beautiful sites, and the result is each course Web site looks different, making it difficult for students to find the content they are after as they navigate differently arranged structures.

But why use course sites at all? Why not just use a tool like Blackboard for organizing course content? Well, sites like that have their own difficulties. They require time consuming login and navigation processes, they do many things, and so many things interfere with the quick retrieval for what a student is looking for, and they are general purpose sites, and so are cluttered with many things a course isn't even using. For an instructor, building and maintaining a custom course Web page still provides the best service to its students.

The ***Course Site Generator*** application intends to automate the process of building and updating a course Web site in one easy to use tool. The sites produced by this application will look good and will be customizable in many different ways but will exist within a common site and page structure.

## 1.1 **Purpose**

The purpose of this document is to specify how our ***Course Site Generator*** program should look and operate. The intended audience for this document is all the members of the development team, from the instructors to the software engineers and designers. This document serves as an agreement among all parties and as a reference for how the site creation tool should ultimately be constructed. Upon completing the reading of this document, one should clearly visualize how the application will look and operate as well as understand the way a generated site is constructed.

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

## 1.5 Overview

This SRS will clearly define how the ***Course Site Generator*** application should look and operate. Note that this is not a software design description (SDD), which would design how to construct the software using UML. This document does not specify how to build the appropriate technologies, it is simply an agreement concerning what to build. Section 2 of this document will provide the context for the project and specify all the conceptual design. Section 3 will present how the user interface should be laid out. Section 4 provides a Table of Contents, an Index, and References.

## 2 Overall description

Here you are a course instructor preparing content for your new semester and it's time to create and upload your course Web page. Perhaps you've taught this course before and so you have an old Web page you can modify, but still you spend quite a bit of time editing HTML such that everything is updated properly. Then, as you further make course decisions you realize you're going to need to add a lecture to your schedule, which requires quite a bit of HTML refactoring, again, it's time consuming and error prone. When it's all said and done you've spent a large amount of time authoring a site that doesn't look so great and will be difficult to keep updated without errors as the semester goes along. This is a job for our ***Course Site Generator***.

### 2.1 Product perspective

Our site generator will automate the process of creating our course Web site to greatly reduce the amount of time needed in planning a semester and update it as it progresses. In order to do that our application will provide a Web site template that all generated content will use where user entered data can be store in JSON files that can then be loaded using JavaScript.

#### 2.1.1 System Interfaces

The ***Course Site Generator*** will be a traditional workspace-type application that will let the user create a new course site and export a full generated site as needed. The first important thing to understand is what may go into a course site. Not all courses use the same content, or even the same pages for their sites, but some things, like the navigation bar structure, will be uniform. The following course site provides an example of what a course site might look like. Note that it has all four pages (Home, Syllabus, Schedule, HWs) that our application will be able to generate.

**CSE 219: http://www.cs.stonybrook.edu/~cse219/index.html**

The particularly important things our application will do is provide a convenient means for editing data that will end up in JSON files loaded for these pages. It is important to understand the structure of the various JSON files that need to be build. In this respect there are five types of JSON files that will need to be generated, and so our application will need to provide a means to edit the content destined for these files, with examples listed below:

- http://www.cs.stonybrook.edu/~cse219/js/OfficeHoursData.json
- http://www.cs.stonybrook.edu/~cse219/js/PageData.json
- http://www.cs.stonybrook.edu/~cse219/js/ScheduleData.json
- http://www.cs.stonybrook.edu/~cse219/js/SectionsData.json
- http://www.cs.stonybrook.edu/~cse219/js/SyllabusData.json

It is important that we understand these JSON file formats and understand the data they hold. Note that JavaScript files already exist for properly loading this data into their necessary pages (OfficeHoursBuilder.js, PageBuilder.js, ScheduleBuilder.js, SectionsBuilder.js, and SyllabusBuilder.js).

### 2.1.2 **User Interfaces**

Our site editing program will be a desktop application and so will make use of a mouse and keyboard for user input. Figure 2.2 below summarizes the ways with which the user will interact with our ***Course Site Generator*** application, which will be further detailed using UML Use Case diagrams. These Use-Case diagrams should be fed as input directly into Section 3.1, external interfaces, which is where the design of the user interface is specified. Here is the full list of UML Use-Case Diagrams:

| Use Case | UI Context | Use Case |
|---|---|---|
| 2.1 | Welcome Dialog & File Toolbar | Create New Course Site |
| 2.2 | Welcome Dialog & File Toolbar | Load Course Site |
| 2.3 | File Toolbar | Save Course Site |
| 2.4 | File Toolbar | Close Course Site |
| 2.5 | File Toolbar | Export & View Course Site |
| 2.6 | File Toolbar | Exit |
| 2.7 | Undo Toolbar | Undo |
| 2.8 | Undo Toolbar | Redo |
| 2.9 | Info Toolbar | About |
| 2.10 | Info Toolbar | Change Language |
| 2.11 | Info Toolbar | Help |
| 2.12 | Site Pane | Edit Page Details |
| 2.13 | Site Page | Edit Instructor |
| 2.14 | Syllabus Pane | Edit Syllabus Details |
| 2.15 | Meeting Times Pane | Add Lecture/Recitation/Lab |
| 2.16 | Meeting Times Pane | Edit Lecture/Recitation/Lab |
| 2.17 | Meeting Times Pane | Remove Lecture/Recitation/Lab |
| 2.18 | Office Hours Pane | Filter TAs |
| 2.19 | Office Hours Pane | Add TA |
| 2.20 | Office Hours Pane | Edit TA |
| 2.21 | Office Hours Pane | Remove TA |
| 2.22 | Office Hours Pane | Select Time Range |
| 2.23 | Office Hours Pane | Toggle TA Office Hours |
| 2.24 | Schedule Pane | Edit start and end dates |
| 2.25 | Schedule Pane | Add Schedule Item |
| 2.26 | Schedule Pane | Edit Schedule Item |
| 2.27 | Schedule Pane | Remove Schedule Item |

**Figure 2.1: Overview of Use-Case Diagrams**

**Use Case 2.1: Create New Course Site**

| | |
|---|---|
| Use-Case: | Create New Course Site |
| Primary Actor: | Instructor |
| Goal in Context: | The user wishes to make a new course Web site |
| Preconditions: | The application has been started and the user is viewing either the welcome dialog or the main user interface |
| Scenario: | • User is viewing the Course Site Generator<br>• User clicks on the New button<br>• User is prompted to save first if work is unsaved. Data is initialized and the user interface is reset with default values for a new site |
| Exceptions: | This button should always be enabled. |
| Priority: | • Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Used every time the user makes a new site |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.2: Load Course Site**

| | |
|---|---|
| Use-Case: | Load Course Site |
| Primary Actor: | Instructor |
| Goal in Context: | The user wishes to load and edit a course Web site that has already been created |
| Preconditions: | The application has been started and the user is viewing either the welcome dialog or the main user interface |
| Scenario: | • User is viewing the Course Site Generator<br>• If on welcome dialog, user clicks recent file, if in main use, user clicks load button and then selects file to load via load dialog. Program proceeds to load site data into UI and present it for editing |
| Exceptions: | Should the user select a file that is not loadable because it is in the incorrect format, the program should not crash, but instead should display a dialog message telling the user what happened |
| Priority: | Essential, must be implemented |
| When available: | Second Benchmark |
| Frequency of use: | Used every time the user edits and existing site |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.3: Save Course Site**

| | |
|---|---|
| Use-Case: | Save Course Site |
| Primary Actor: | Instructor |
| Goal in Context: | The user wishes to save data for the site that is currently being edited |
| Preconditions: | The user is currently editing an existing site and has made at least one change |
| Scenario: | • User clicks on Save button (which would have to be enabled at the time) Site is saved to its current file (note that when a site is created for the first time the file should be created at that time)<br>• User can then proceed editing site |
| Exceptions: | This button should only be enabled if the user has made a change since last loading or saving |
| Priority: | Essential, must be implemented |

| | |
|---|---|
| When available: | Second Benchmark |
| Frequency of use: | Used many times per editing session |
| Open Issues: | Size, location, and style of buttons should be finalized by UI designer |

**Use Case 2.4: Close Course Site**

| | |
|---|---|
| Use-Case: | Close Course Site |
| Primary Actor: | Instructor |
| Goal in Context: | The user wishes to close the current course site |
| Preconditions: | The user is currently editing an existing site |
| Scenario: | • User is editing a site<br>• User presses the close button.<br>• If the user has not saved since last edit, user will be prompted to save. User will either press Yes (to save), No (to not save), or Cancel (to cancel close). If user selects Yes or No the site will be closed and the workspace will be hidden. |
| Exceptions: | This button should only be enabled if a site is being edited |
| Priority: | Essential, must be implemented |
| When available: | Second Benchmark |
| Frequency of use: | Used many times per editing session |
| Open Issues: | Size, location, and style of buttons should be finalized by UI designer |

**Use Case 2.5: Export & View Course Site**

| | |
|---|---|
| Use-Case: | Export & View Course Site |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to export the site and see how it looks in a Web browser |
| Preconditions: | The user is editing a site |
| Scenario: | • User edits a site<br>• User clicks on Export button, which should export full site to export directory and then open a dialog that contains a Web View of the exported site.<br>• User may browse the site in the dialog<br>• User presses X in dialog to close and return to editing |
| Exceptions: | If site has never been saved export button should be disabled |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Used many times per session to view exported site |
| Open Issues: | Size, location, and style of dialog should be finalized by UI designer |

**Use Case 2.6: Exit Application**

| | |
|---|---|
| Use-Case: | Exit Application |
| Primary Actor: | Instructor |
| Goal in Context: | The user is done working and wishes to exit the program |
| Preconditions: | N/A |
| Scenario: | • User presses Exit button.<br>• If the user has unsaved work the application will prompt the user to save, so the user will specify whether to save or not. If user presses Yes, application will save work to current file and exit. If user presses No, application will simply exit. If user |

| | |
|---|---|
| | Presses cancel, application will return to editing and not close. |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Used once per session |
| Open Issues: | Size, location, and style of stats display should be finalized by UI designer |

**Use Case 2.7: Undo**

| | |
|---|---|
| Use-Case: | Undo |
| Primary Actor: | Instructor |
| Goal in Context: | Lets the user Undo an edit |
| Preconditions: | User is editing a site and has made at least one change |
| Scenario: | • User edits site data<br>• User presses Undo, which restores application to state before previous edit |
| Exceptions: | Button should only be enabled if at least one edit has been applied since load time |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.8: Redo**

| | |
|---|---|
| Use-Case: | Redo |
| Primary Actor: | Instructor |
| Goal in Context: | Lets the user Redo an Undone edit |
| Preconditions: | User is editing a site and has made at least one change and then pressed Undo |
| Scenario: | • User edits site data<br>• User presses undo at least one time<br>• User decides to keep previous version so presses Redo |
| Exceptions: | Button should only be enabled if at least one edit has been undone and is still redo-able on the transaction stack |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.9: About**

| | |
|---|---|
| Use-Case: | About |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to learn a little about the app |
| Preconditions: | Application has started |
| Scenario: | • User is viewing user interface with any tab activated<br>• User clicks About button<br>• User views dialog that pops up that has information about the app<br>• User clicks Ok to close dialog |
| Exceptions: | N/A |

| | |
|---|---|
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.10: Change Language**

| | |
|---|---|
| Use-Case: | Change Language |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to change the language used for text in the application's user interface (only requires 2 languages, English and another) |
| Preconditions: | A Course Web site is being edited |
| Scenario: | • User is viewing the UI<br>• User clicks on the Change Language button, which opens up a dialog with a Combo Box for choosing a language.<br>• User selects the language and presses Ok, which changes all the UI text to the selected language. |
| Exceptions: | Note, this will not change the language of the course site (i.e. the data), this only changes the language used by the Course Site Generator user interface |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.11: Help**

| | |
|---|---|
| Use-Case: | Help |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to learn more about how to use the app |
| Preconditions: | Application has started |
| Scenario: | • User is viewing user interface with any tab activated<br>• User clicks Help button<br>• User views dialog that pops up that has information about how to use the app<br>• User clicks Ok to close dialog |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Use Case 2.12: Edit Page Details**

| | |
|---|---|
| Use-Case: | Edit Page Details |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to edit information that affects all site pages |
| Preconditions: | The user is editing a site |
| Scenario: | • User is viewing UI<br>• User clicks on Site tab |

| | |
|---|---|
| | • User selects appropriate Subject, Class Number, Semester, and Year from drop down boxes or Types values in if not available (typed in values should then be saved as settings so as to appear as options for future sites).<br>• User clicks on Pages checkboxes to select applicable export pages<br>• User clicks on Style buttons to select site images<br>• User selects appropriate CSS style file from combo box |
| Exceptions: | At least one check box must be selected else there is nothing to export |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of dialog and button should be finalized by UI designer |

**Use Case 2.13: Edit Instructor**

| | |
|---|---|
| Use-Case: | Edit Instructor |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to enter data for the course instructor, for which there is only one |
| Preconditions: | The user is editing a site |
| Scenario: | • User is viewing UI<br>• User clicks on Site tab<br>• User enters instructor name, email, room, and home page in text fields<br>• User clicks expand (‘+’) button to view office hours text area. This should add the office hours text area to the UI such that it may be viewed and edited. It would also change the “+” to a “-” to represent collapse.<br>• User enters JSON-formatted description of instructor office hours in text area (see PageData.json)<br>• User clicks collapse (‘-’) button to hide office hours text area |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of dialog and button should be finalized by UI designer |

**Use Case 2.14: Edit Syllabus Details**

| | |
|---|---|
| Use-Case: | Edit Syllabus Details |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to enter data for the course syllabus |
| Preconditions: | The user is editing a site |
| Scenario: | • User is viewing UI<br>• User clicks on Syllabus tab<br>• User clicks expand (‘+’) button next to Description or Topics or whichever section they wish to edit to view corresponding label. This should add the corresponding text area to the UI such that it may be viewed and edited. It would also change the “+” to a “-” to represent collapse for the corresponding text area.<br>• User enters JSON-formatted data in text area (see SyllabusData.json)<br>• User clicks collapse (‘-’) button to hide corresponding text area |
| Exceptions: | N/A |

| | |
|---|---|
| Priority: | Essential, must be implemented |
| When available: | Second Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of dialog and button should be finalized by UI designer |

**Use Case 2.15: Add Lecture/Recitation/Lab**

| | |
|---|---|
| Use-Case: | Add Lecture/Recitation/Lab |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to Add Lecture, Recitation, and Lab sections |
| Preconditions: | The user is editing a site |
| Scenario: | • User is viewing UI<br>• User clicks on Meeting Times tab<br>• User clicks on Add button (‘+’) according to appropriate course component section (lecture, recitation, or lab). This will add a new row to the appropriate table with all ‘?’ values for data |
| Exceptions: | Duplicate data is allowed in these tables |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of dialog and button should be finalized by UI designer |

**Use Case 2.16: Edit Lecture/Recitation/Lab**

| | |
|---|---|
| Use-Case: | Edit Lecture/Recitation/Lab |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to change data for a lecture, recitation, or lab section |
| Preconditions: | The user is editing a site |
| Scenario: | • User is viewing UI<br>• User clicks on Meeting Times tab<br>• User clicks on table cell containing data to change<br>• User types in updated information and presses Enter. This will update data in table and select the item in the row |
| Exceptions: | Duplicate data is allowed in these tables |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button & screens should be finalized by UI designer |

**Use Case 2.17: Remove Lecture/Recitation/Lab**

| | |
|---|---|
| Use-Case: | Remove Lecture/Recitation/Lab |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to remove a lecture, recitation, or lab section |
| Preconditions: | There is at least on lecture, recitation, or lab section in a table |
| Scenario: | • User is viewing UI<br>• User clicks on Meeting Times tab, which has at least on existing section |

| | |
|---|---|
| | • User selects an existing table from appropriate table. This would enable the remove button (‘-‘)<br>• User presses the remove button, which will remove it from the table |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Size, location, and style of button & screens should be finalized by UI designer |

**Use Case 2.18: Filter TAs**

| | |
|---|---|
| Use-Case: | Filter TAs |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to view a type of teaching assistant |
| Preconditions: | At least one teaching assistant has been added |
| Scenario: | • User is viewing UI<br>• User clicks on Office Hours tab<br>• User clicks Radio Button to filter TAs as desired |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Appropriate size, location, and style of buttons & Screens to be determined by UI designer |

**Use Case 2.19: Add TA**

| | |
|---|---|
| Use-Case: | Add TA |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to add a TA |
| Preconditions: | N/A |
| Scenario: | • User is viewing UI<br>• User clicks on Office Hours tab<br>• User clicks Radio Button to filter TAs as desired<br>• User enters name and email of TA in text fields<br>• User types ENTER or presses Add TA button. This will add the new TA if it qualifies as a valid name/email combo |
| Exceptions: | Must be a unique name and email with a valid email format. |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session. |
| Open Issues: | Size, location, and style of Screen should be finalized by UI designer |

**Use Case 2.20: Edit TA**

| | |
|---|---|
| Use-Case: | Edit TA |
| Primary Actor: | Instructor |
| Goal in Context: | Player wishes to change TA information |
| Preconditions: | At least one TA has been added |

| | |
|---|---|
| Scenario: | • User is viewing UI<br>• User double clicks on a TA in the TA table. This opens a dialog.<br>• User updates name, email, and type.<br>• User presses Ok to confirm all changes. |
| Exceptions: | Name and email must still be valid |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.21: Remove TA**

| | |
|---|---|
| Use-Case: | Remove TA |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to remove a TA |
| Preconditions: | At least one TA exists |
| Scenario: | • User is viewing UI<br>• User clicks on Office Hours tab<br>• User clicks on TA in table to select it<br>• User clicks on remove button (‘-’) to remove the TA, which will also remove all of its office hours. |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.22: Select Time Range**

| | |
|---|---|
| Use-Case: | Select Time Range |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to change the office hours time range |
| Preconditions: | N/A |
| Scenario: | • User is viewing UI<br>• User clicks on Office Hours tab<br>• User selects start time from drop down combo box<br>• User selects end time from drop down combo box. Note that these two actions will affect presentation of table, but should not delete office hours data for TAs in case user decides to change times again |
| Exceptions: | End Time cannot be before Start Time. Foolproof design should prevent this. |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.23: Toggle TA Office Hours**

| | |
|---|---|
| Use-Case: | Toggle TA Office Hours |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to add or remove a TA to a time slot |
| Preconditions: | At least one TA exists |
| Scenario: | • User is viewing UI<br>• User clicks on Office Hours tab<br>• User clicks on TA in table to select it<br>• User clicks on time slot to add or remove TA from that time slot |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.24: Edit start and end dates**

| | |
|---|---|
| Use-Case: | Edit start and end dates |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to specify when course will start and end |
| Preconditions: | N/A |
| Scenario: | • User is viewing UI<br>• User clicks on Schedule Tab<br>• User selects starting Monday from date chooser<br>• User selects ending Friday from date chooser. Note that changing dates should not remove any existing schedule items that may not be within those dates as one may choose to change them later. |
| Exceptions: | End Data cannot be before Start Date. Foolproof design should prevent this. |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.25: Add Schedule Item**

| | |
|---|---|
| Use-Case: | Add Schedule Item |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to add a Holiday or Lecture or HW or Recitation or Lab or Reference to the schedule |
| Preconditions: | At least one week of time already is in schedule for desired dates |
| Scenario: | • User is viewing UI<br>• User clicks on Schedule Tab<br>• If item is selected in table, user clicks Clear button, which unselects all items in table, clears out all data in entry fields, and makes Add button say “Add”<br>• User selects type of schedule item from drop down combo box<br>• User selects date of schedule item from date picker. Title, Topic, and Link should be entered in text field.<br>• User presses Add to add to table |
| Exceptions: | Date of item must be in range, foolproof design must ensure this |

| | |
|---|---|
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.26: Edit Schedule Item**

| | |
|---|---|
| Use-Case: | Edit Schedule Item |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to edit an existing Holiday or Lecture or HW or Recitation or Lab or Reference to the schedule |
| Preconditions: | At least one week of time already is in schedule for desired dates |
| Scenario: | • User is viewing UI<br>• User clicks on Schedule Tab<br>• User selects existing schedule item from table, which will load all data from that item into controls and change Add button to say "Update"<br>• User selects type of schedule item from drop down combo box<br>• User selects date of schedule item from date picker. Title, Topic, and Link should be entered in text field.<br>• User presses Update to commit changes to table |
| Exceptions: | Date of item must be in range, foolproof design must ensure this |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

**Use Case 2.27: Remove Schedule Item**

| | |
|---|---|
| Use-Case: | Remove Schedule Item |
| Primary Actor: | Instructor |
| Goal in Context: | User wishes to remove an existing schedule item |
| Preconditions: | User is editing a Course Site |
| Scenario: | • User is viewing UI<br>• User clicks on Schedule tab<br>• User selects a schedule item from the table<br>• User presses remove button ("-") to remove item from table |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Third Benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | Precise size, location, and style of window should be finalized by UI designer. |

### i. Hardware Interfaces

The application should be runnable on any platform that supports Java, but would require a keyboard and mouse.

### ii. Software Interfaces

***Course Site Generator*** will be developed using the Java language. Note that since it is a traditional workspace-type application, it may be best to use the Desktop Java Framework. Note that the site exported by this application should be able to be deployed to any Web server and work via any Web browser, namely Chrome and Firefox. Note that it is recommended that Web Server for Chrome and the Chrome browser be used in cooperation to test exporting.

### iii. Communications Interfaces

Note that this editing application will operate locally. There will be no networking requirements.

### iv. Memory Constraints

This application uses a manageable amount of user provided data so this should not be a concern.

### v. Operations

We must make sure the data exported to JSON files use the proper format according to current Web page requirements per the example files given.

### vi. Site Adaptation Requirements

N/A

## b. Product functions

N/A.

### c. User characteristics

The editor should aim to be as user friendly as possible, using the principles of foolproof design as well as sound UI design principles.

### d. Constraints

N/A

### e. Assumptions and dependencies

N/A

### f. Apportioning of the Requirements

N/A

## 3 Specific requirements

The Course Site Generator application will require a number of user interface contexts and components including:

- Welcome Dialog
- File Toolbar
- Undo Toolbar
- Info Toolbar
- Tabbed Pane
    - Site Pane
    - Syllabus Pane
    - Meeting Times Pane
    - TAs Pane
    - Schedule Pane

Note that the user should be free to navigate between these five panes as desired.

### 3.1 **External interfaces**

The following wireframe mockups provide a look at the types of controls and layout to be used for the User Interface. Note that the User Interface designer should select the appropriate icons for all buttons and should carefully choose color and font combinations that provide good contrast and attract the eye. There are six UI diagrams. Note that the User Interface designer should also consider additional dialogs for providing adequate feedback to the user as well as for navigation through the file system to select files and directories as part of certain use cases and for selecting the UI language.

**Figure 3.1 Welcome Dialog**

**Figure 3.2 Course Site Tab**

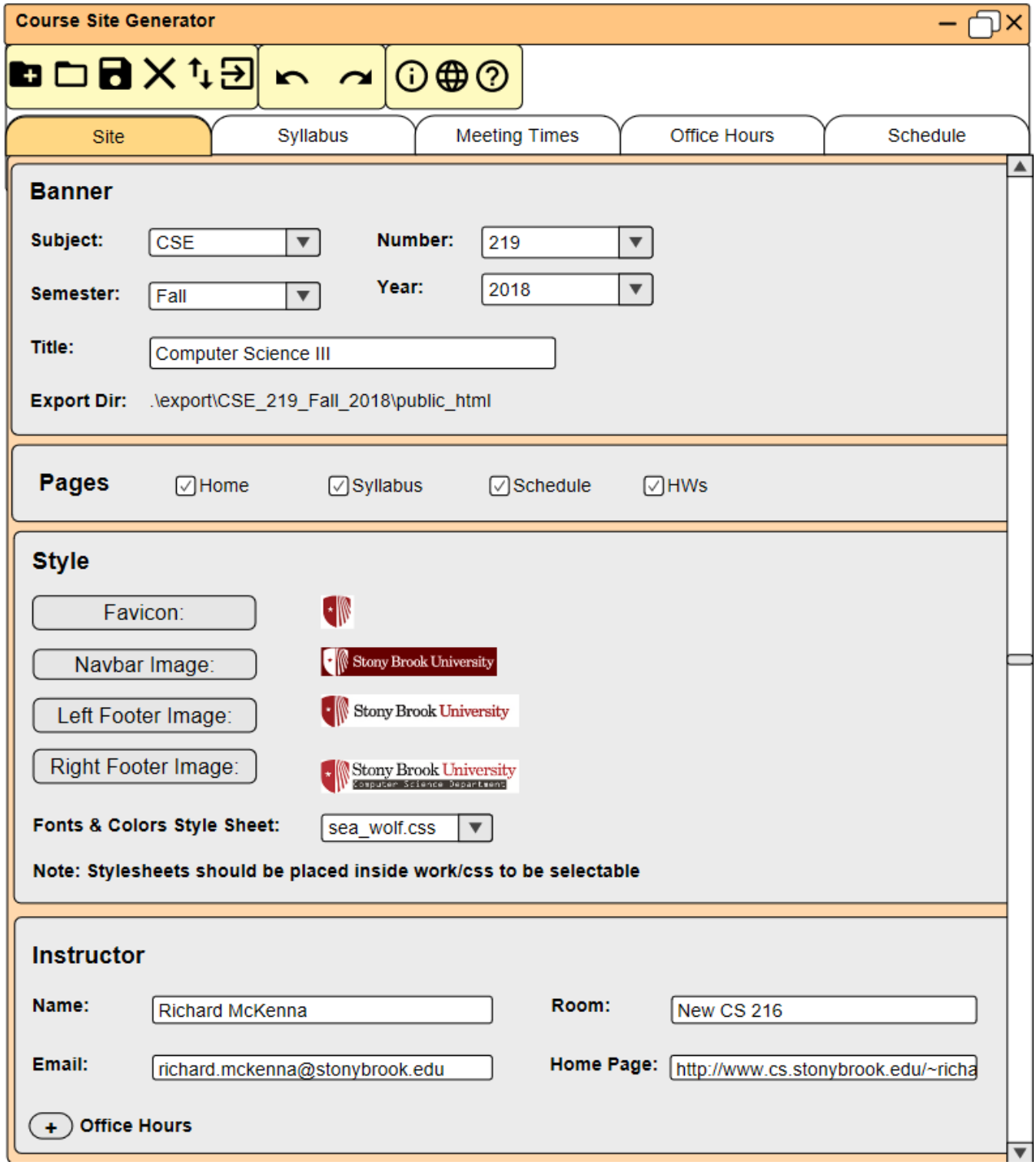


**Figure 3.3 Course Site Tab with Expanded Office Hours**

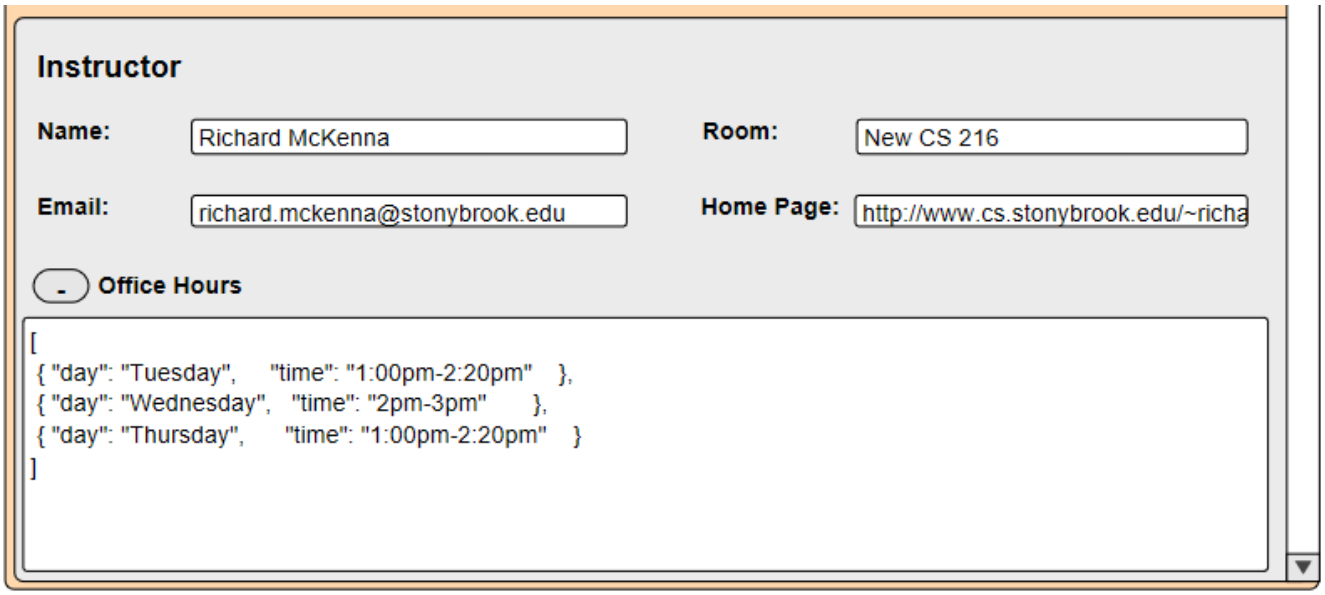


**Figure 3.4 Course Syllabus Tab**

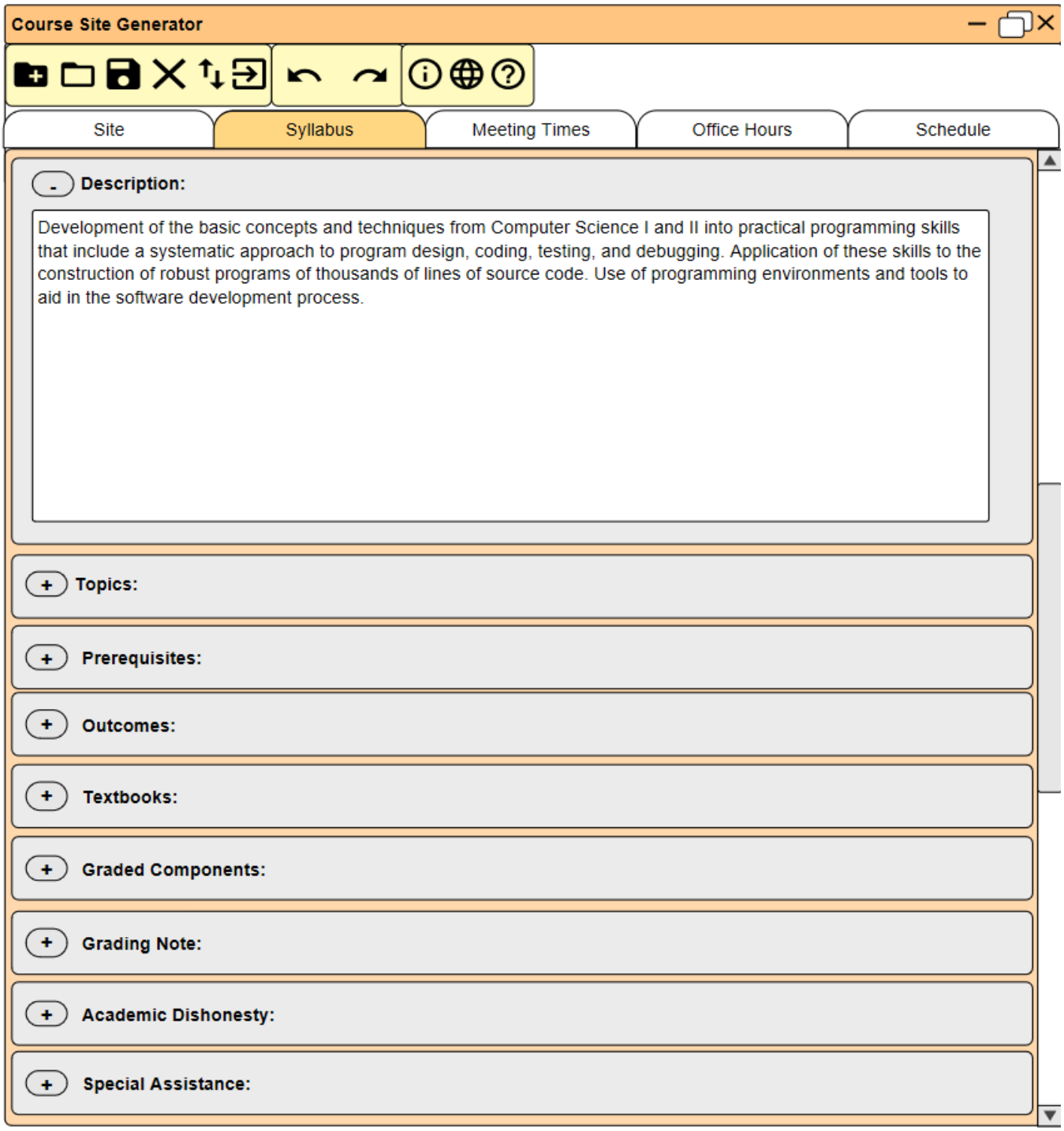


**Figure 3.5 Course Meeting Times Tab**

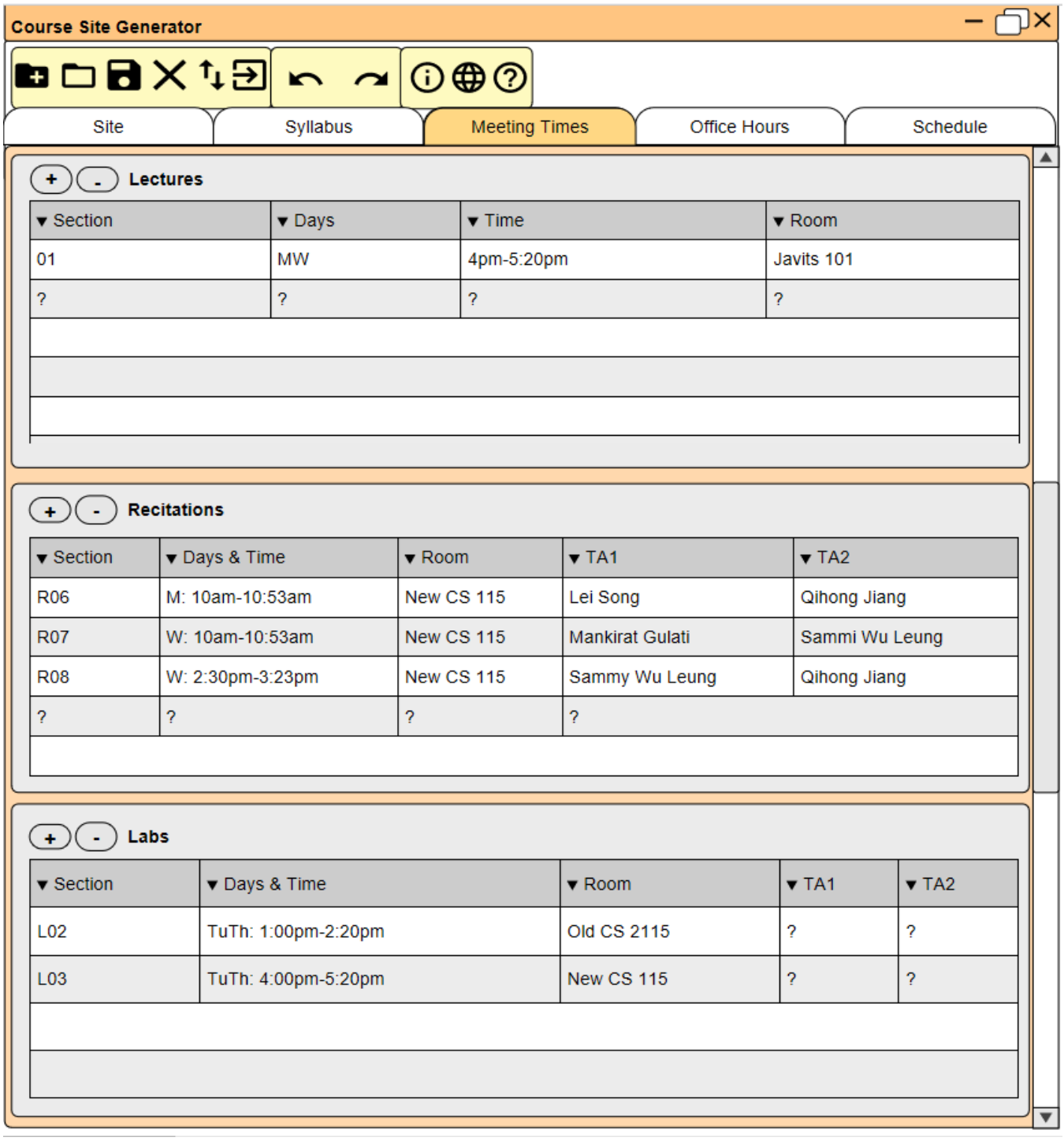


**Figure 3.6 Course Office Hours Tab**

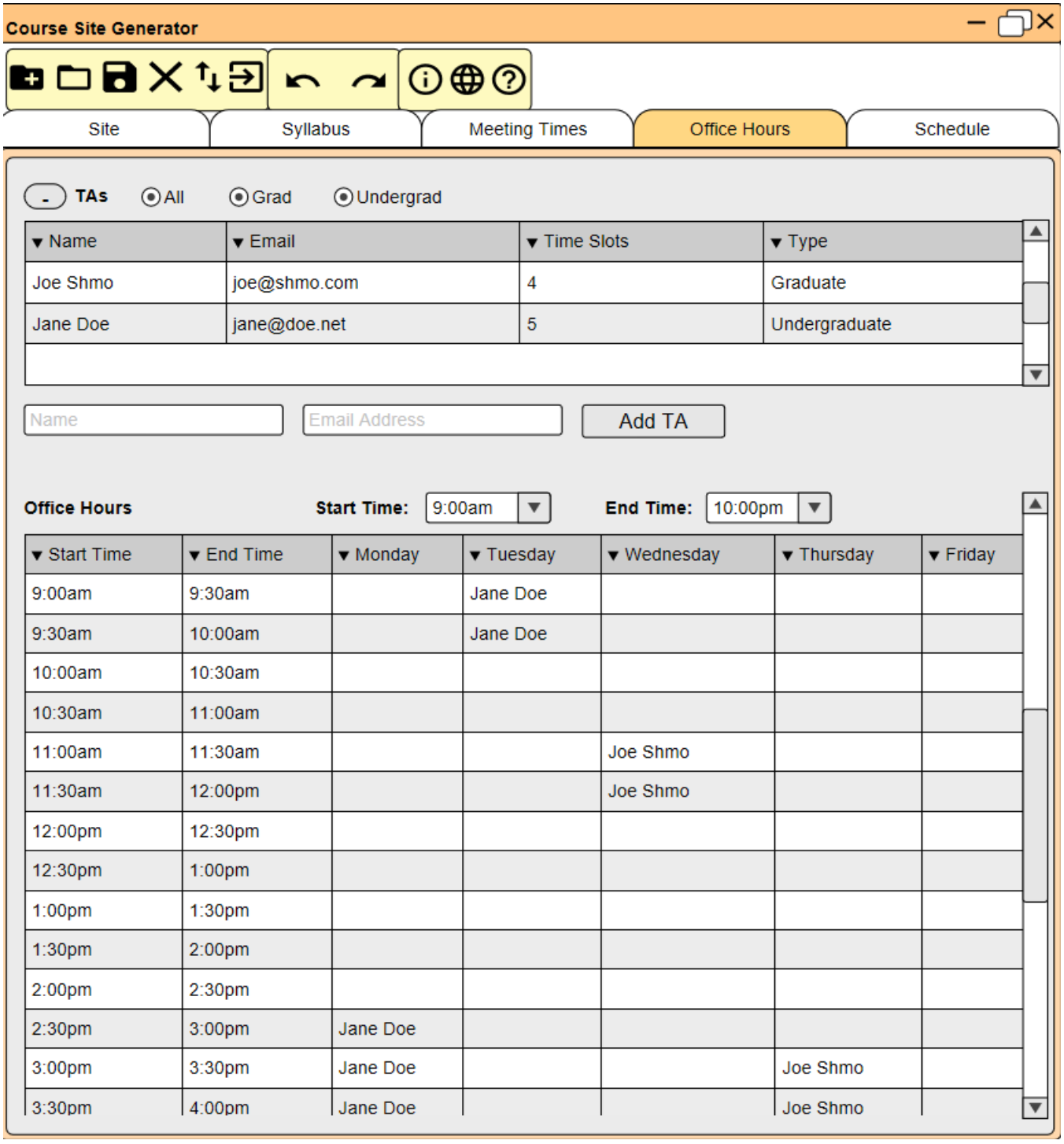


**Figure 3.7 Edit TA Dialog**

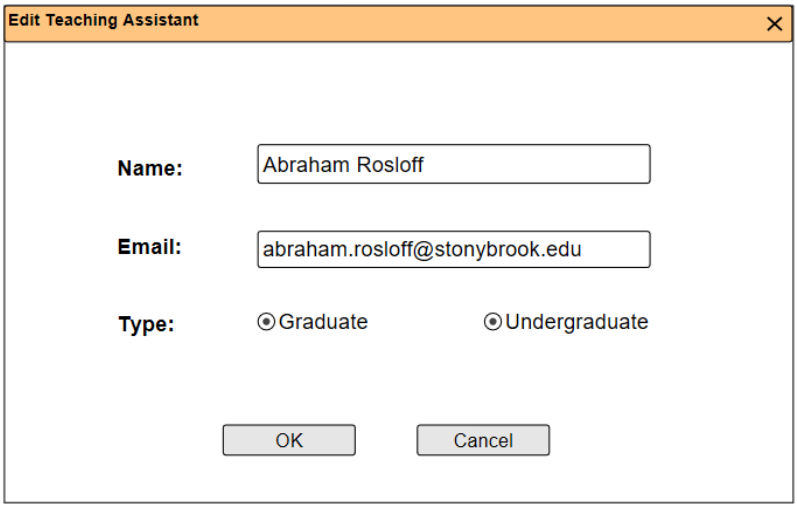
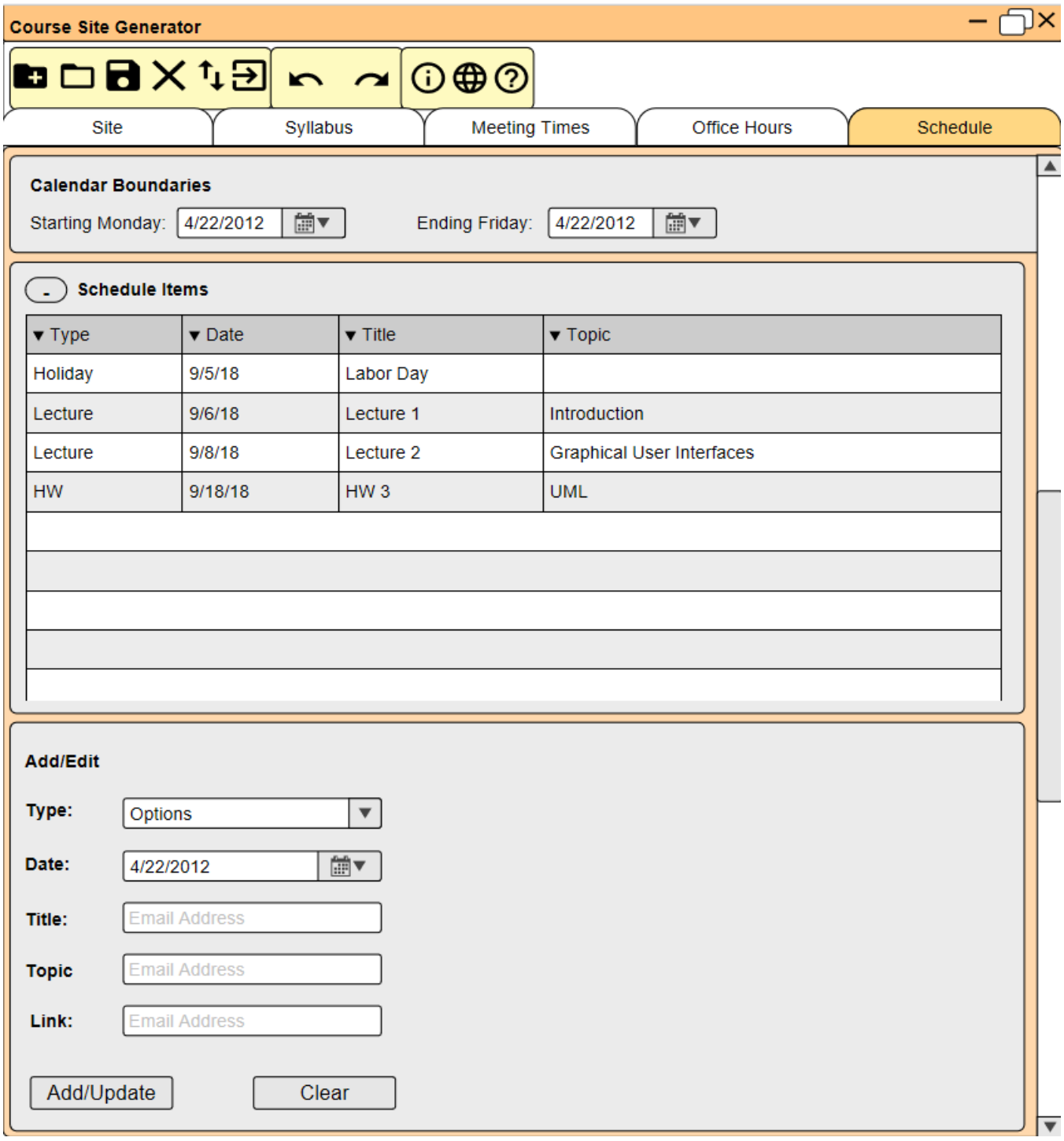


**Figure 3.8 Course Schedule Tab**



## 3.2 **Functions**

One of the important things to consider in our application is providing the appropriate feedback to the user. Users need feedback to enjoy their experience. This is typically done with visual cues like dialog boxes.

## 3.3 **Performance requirements**

N/A

## 3.4 **Logical database requirements**

N/A

## 3.5 **Design constraints**

JavaFX will be used because it effectively leverages each system's available rendering technologies and provides platform independence for personal computers.

## 3.6 **Software system attributes**

As professionals, all members of this project must take this project seriously. We are dedicated to producing robust software that exceeds the expectations of our customers. In order to achieve this level of quality, we should build a product with the following properties in mind:

**3.6.1 Reliability** – The program should be carefully planned, constructed and tested such that it behaves flawlessly for the end user. Bugs, including rendering problems, are unacceptable. In order to minimize these problems, all software will be carefully designed using UML diagrams and a Design to Test approach should be used for the Implementation Stage.

**3.6.2 Availability** –Customers may download and install the application for free.

**3.6.3 Security** – All security mechanisms will be addressed by future revisions

**3.6.4 Extensibility** – It is possible that more JavaScript/JSON widgets might be added to course sites in the future, so by providing an additional tab with suitable data and making changes to exporting methods, this should be considered during this deaign.

**1.6.5 Portability** – To start with, the app will target desktop Java applications.

**3.6.6 Maintainability** – Update mechanisms will be addressed by future revisions.

### 3.7 **Organizing the specific requirements**

Note that the application is simple enough that we need not worry about using an alternative arrangement of the content of this document. The specific requirements for this application already fit neatly into the sections listed in the IEEE's recommended SRS format.

### **3.8 Additional comments**

It is important to keep in mind that the UI designers and instructors should make updates to the themes and content as need to make something that looks great. It will be to their discretion to design all the interface controls in an effective, interactive style.

## 4 Supporting Information

Note that this document should serve as a reference for the designers and coders in the future stages of the development process, so we'll provide a table of contents to help quickly find important sections.

### 4.1 Table of contents



### 4.2 **Appendixes**

N/A