**CS321 Project Report 2 (Group 8)**

*Project Requirements*

**TEAM MEMBERS**

Bailey J. Knight, Christopher Fechter, Eric M. Sung, Kyungjae Lee

**PROJECT NAME**

Meeting Scheduler

**MOTIVATION**

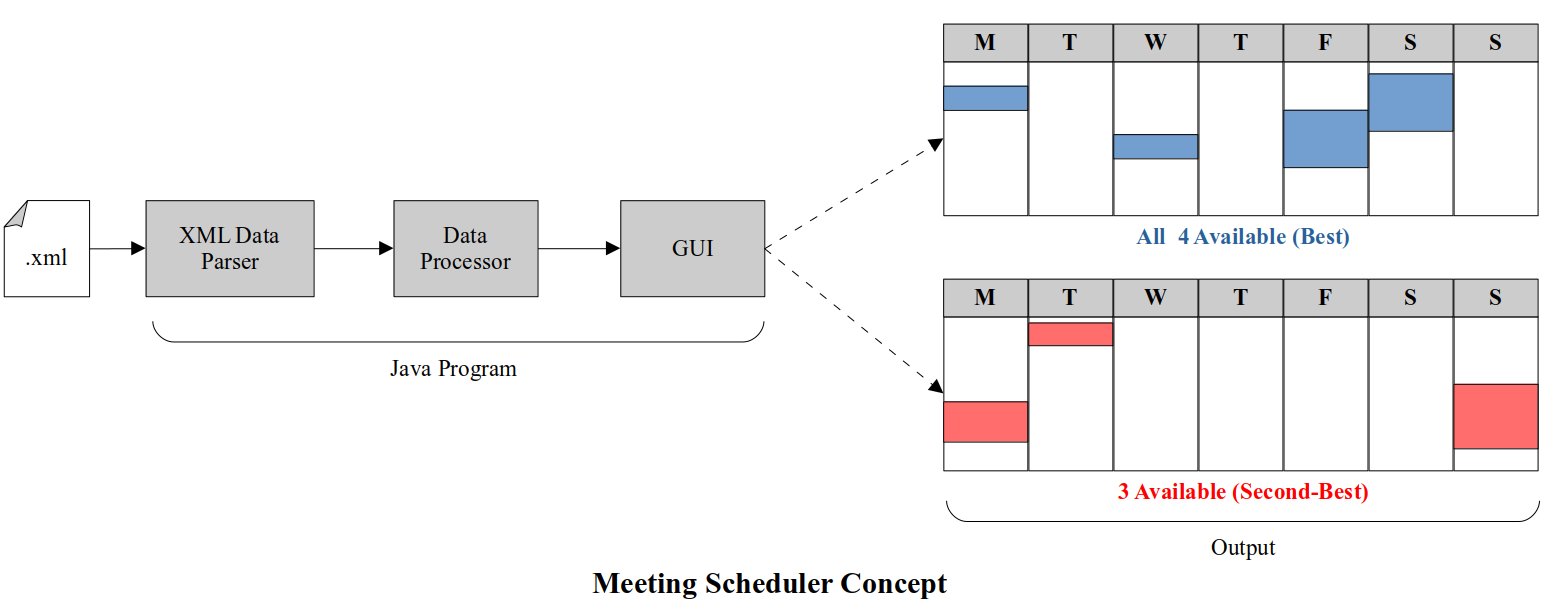
College students often have busy schedules, which more often than not, have little flexibility when it comes to coordinating more events within them. This applies heavily to group projects that arise spontaneously from new classes each semester. It would be nice to have a method of inputting all of the members’ schedules of a particular group into some program, which can take those and convert them into one unified schedule that shows the available times of everyone without much hassle. This is what we’re building our software for.

**PROJECT DESCRIPTION**

The program we are developing will be able to take a group’s schedules, and if time permits, maybe even more students, so that it can output all of the free time for a specific set of students, helping them collaborate easier. Initially for our real-world data set, the plan is to use our personal weekly college schedules from the members of our group, and if necessary, we can ask other students to provide their own schedules to add to the pool of options.

**CONCEPT**

This program is expected to read in the schedules of students from an XML file, in which the file specifies the times of the week where the students may be busy for any reason. Once the data has been read, the program will generate a one-week schedule with 7 blocks of information, for each of the 7 days of the week. Every block will have the time presented from the hours 8 am to 10 pm. The time slots for each day where all students are available will be highlighted in a specific color like blue, outlining when all members are free. This could potentially also show the second-best time slots of the day too, which might be represented in a different color, such as red. Besides this display, the GUI might also allow for filtering group members, to exclude or include specific people, or to see someone’s individual availability.



**MEETING THE CRITERIA (REQUIREMENTS)**

As for meeting the criteria for the required components of the project, our program’s model-view-control will utilize the information placed in the objects created for the students and display it in a generated table, depending on the specified filters that a user might change in the program. The set of options for a user to interact with will be displayed to the right of the table. Whenever a user changes a filter, such as disabling a group member or toggling the usage of second-best time slots, the table view will update to show the correct schedule. This part is considered the GUI as the user can interact with these specific controllers to alter the view presented on the table. For the data model, each student will be considered as an object that holds information for their name and each of the days of the week’s individual schedules of availability. The data will be loaded into the model from the XML file previously mentioned. The GUI will add the new data and display the schedule based on the toggled filters. The XML file is the data source we’re using. The results of the program will allow different displays of the schedule with filters, potentially second-best alternatives for 3 groups members, checking to see if a certain time frame is available, like Monday from 1pm-2pm, and the best times for meetings that recur throughout the week.

**USE CASES**

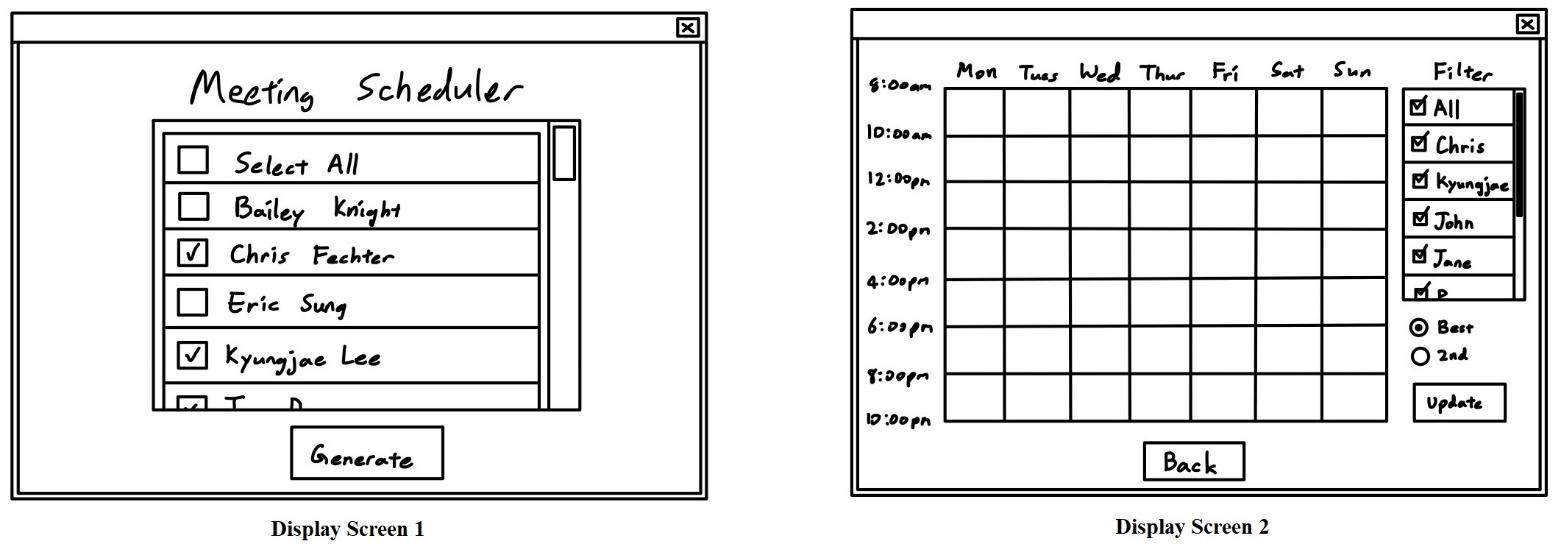
**Use Case 1**

| Name | Generate Base Schedule |
| --- | --- |
| Brief Description | This describes the initial setup of the program, in which the default schedule is generated from the set of students’ schedules provided by an XML file before running the program. The default schedule has every student shown and the best times where they are all available displayed. |
| Precondition | * The XML file with student schedules is provided   + More than one student schedule is included in the file * The program is executed with the file |
| Basic Flow | 1. The user is brought to the first GUI display where all students are enabled by default. 2. The user clicks the “generate” button on the GUI which makes the program transition to the second screen where the base schedule is generated. 3. The schedule displays to the user the best times of availability for all of the group members based on which were selected. |
| Alternate Flows | In step 1, the user can deselect some students in order to generate a different base schedule from the default.  In step 1, the user can deselect all students, which will disable the “generate” button from generating and displaying the schedule. |
| Exception Flows | In step 2, if the user selects the “generate” button, but no students are selected, the schedule will not be allowed to generate.  In step 2, if the user selects the “generate” button, but only one student is selected, the schedule will not be allowed to generate. |
| Post Conditions | ●     The base schedule is displayed on the 2nd screen |

**Use Case 2**

|  |  |
| --- | --- |
| Name | Generate Altered Schedule |
| Brief Description | This use case describes when the user wants to generate a new schedule based on specific filters that the user has selected. |
| Precondition | * A base schedule has been already generated |
| Basic Flow | 1. The user selects the filter to show second best times on the schedule. 2. The user clicks the “The user enables or disables certain student’s that are presented in the base schedule. 3. The user sets a specific time frame where they would like to see the available times. 4. The user clicks the “generate” button to make a new altered schedule with all of the new conditions selected. |
| Alternate Flows | In step 1, the user can choose not to select the second-best times and change other attributes.  In step 2, the user can leave all of the students enabled while changing other attributes.  In step 3, the user can leave the time frame alone while changing other attributes. |
| Exception Flows | In steps 1, 2, or 3, if none of the filters are changed, then the “generate” button will do nothing to the base schedule, and no alternate schedule will be formed. |
| Post Conditions | The altered schedule is displayed on the 2nd screen |

**SKETCHES OF DISPLAY SCREENS**



**LIST OF ESSENTIAL AND ENHANCEMENT ITEMS**

Essential Items

* Student Class. Students will be a data type-
  + This Class will store data about the student like their name and their schedule
  + This class will contain a method that compares the schedule of this student to other students and returns what periods of time are valid
* Driver Class
  + The Driver will have the GUI that controls how the program is operated. This GUI will allow the user to select students and compare their schedules with each other, to display the times of the week that are suitable meeting times for each person.
  + The Driver will use the data interpreter to load saved students.
* Data Interpreter (Possibly its own class, possibly part of the driver)
  + The data interpreter will take in values from a file and generate students that will be used by the program.

Enhancement Items:

* Adding Student Schedules to the data file through the GUI
* Displaying the second or third best schedule that accommodates group members
* Allowing group members to set soft preferences for which time slots they prefer
* From Several group members, finding which combinations/groups would be able to have viable meetings

**DOMAIN MODEL**

|  |  |
| --- | --- |
| Name | Student |
| Description | It holds information about a student, including their name and schedule. |
| Responsibility (or Interface) | * Contains the name of the student and the student’s schedule * Compares one student’s schedule with another student’s schedule |

|  |  |
| --- | --- |
| Name | Driver |
| Description | It extracts user data with the Data Interpreter and initiates the GUI with the data. |
| Responsibility (or Interface) | * Calls the Data Interpreter class to get the data from the XML files * Instantiates the Display class with the data |

|  |  |
| --- | --- |
| Name | Data Interpreter |
| Description | It reads in student information and converts it into usable data |
| Responsibility (or Interface) | * Reads and parses data from an XML file * Creates Student objects from the data |

|  |  |
| --- | --- |
| Name | Display |
| Description | It assembles the GUI of the program and manages the communication with the Backend class |
| Responsibility (or Interface) | * Builds GUI from with Swing components |

**WALKTHROUGH**

Following two test cases show the basic operations of “Meeting Scheduler”.

**Test Case 1 (for Use Case 1)**

* Precondition
  + Input file (.xml) contains schedules of students in a pre-defined format.
* Procedure
  + Step 1: Run the “Meeting Scheduler” program, this will start by running the driver, which will then call the data interpreter to create several students.
  + Step 2: “Display Screen 1” pops up (All students are enabled by default), this is done with the Display, which manages the GUI.
  + Step 3: Click the “Generate” button
* Expected Results
  + Transition to “Display Screen 2” that displays the common availability of all the students

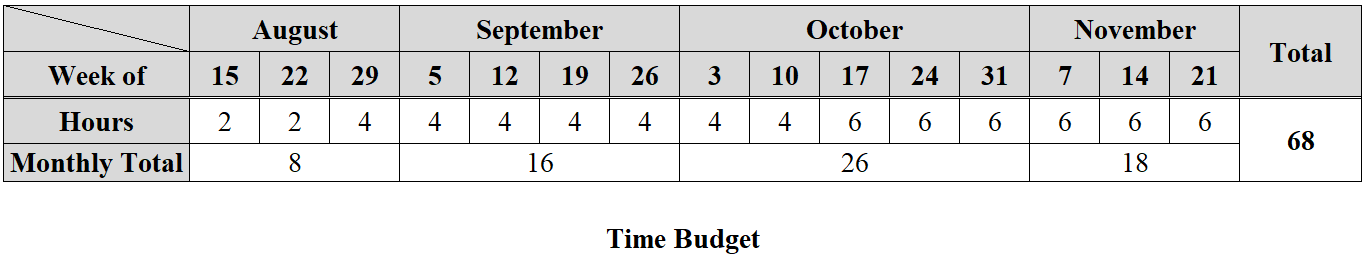
**Test Case 2 (for Use Case 2)**

* Precondition
  + Input file (.xml) contains schedules of students (more than three) in a pre-defined format.
  + The user runs the “Meeting Scheduler’ program
  + The user is at “Display Screen 2” after generating a schedule from “Display Screen 1”
* Procedure
  + Step 1: Disable one student (e.g., Kyungjae Lee)
  + Step 2: Click the “Generate” button
* Expected Results
  + Step 1: Disabled student does not appear in the new schedule

**TIME BUDGET**

Last Updated: September 23, 2022

Following table shows the hours spent (or planned to be spent) by the group.



* Time Spent: 20 hours
* Time Left: 48 hours