Deliverable 2.2 – Use Case



Project group 2 | CS386 Spring 2017 | [GitHub Link](https://github.com/jjg297/CS386-Software-Project)

College toolkit

Jesus Garcia

Matthew Burns

Vincent Mcivor

2017

Project group 2 | CS386 Spring 2017 | [GitHub Link](https://github.com/jjg297/CS386-Software-Project)

College toolkit

Jesus Garcia

Matthew Burns

Vincent Mcivor

**College Toolkit  
Use-Case: Class Information**

1. **Brief Description**

The user of the toolkit will be able to use the app to access relevant information about their classes such as class times, test dates, and office hours.

1. **Actor Brief Descriptions**
   1. **User – The actor who will be using and interacting with the app**
2. **Preconditions**

The user must have access to the application and have used one of the available methods to retrieve class information, either through manual input, extraction from a syllabus, or linking to a website such as BBLearns.

1. **Basic Flow of Events**
2. The use case begins when the user opens the class info section
3. The application will show a list of the user’s classes
4. The user can select which class they would like to view.
5. The application will display a screen with relevant information in a spreadsheet-type format.
6. The user can select to go back to the class list to view a different class.
7. The use case ends when the user returns from the class list to the home page of the application
8. **Alternative Flows**
   1. **The application will display blank spaces or an “incomplete data” message**

If in step 4 of the basic flow the application does not have all the proper information entered from the precondition step, then

1. The application will display an option for the user to enter the missing data at that time.
2. The user may either choose to enter the missing data or decline to do so.
3. The use case resumes at step 3.
4. **Key Scenarios**
   1. **Unlimited class list**
5. The app can hold as many classes as the user as information to enter for.
6. **Post-conditions**
   1. **Viewing the information has no effect on any other function of the application**
   2. **Information can only be viewed and not edited, with the exception of entering missing data.**
7. **Special Requirements**

The app must be able to recognize and extract the relevant data from a document such as a class syllabus through such techniques as specific word or phrase recognition.

**College Toolkit  
Use-Case: Sending Notes**

1. **Brief Description**

The user of the toolkit will be able to take notes in the app, and have a built in index similar to common pdf set-ups. Then they will be able to email this as either a pdf or a text document to any email.

1. **Actor Brief Descriptions**
   1. **User – The actor in this case, who will be interacting with the application.**
2. **Preconditions**

The user must have the main application downloaded and has the app storing their email.

1. **Basic Flow of Events**
2. The use case begins when the user opens the Note Scheduler.
3. The application will display the note interface.
4. User interfaces with the specific note section.
5. The app opens a new screen with the note index page.
6. User can select the send notes button.
7. Pops open a page that allows for the entry of more than the default email
8. User can select pdf or txt format.
9. The user hits send and then the email is sent to the entered emails.
10. The page is closed and you are returned to the app page.
11. **Alternative Flows**
    1. **Email sending is cancelled**

If in step 6 of the basic flow the user doesn’t want to finish the event, then

1. The application deletes all entered information
2. The application stops displaying the send email screen.
3. The use case returns to step 4
   1. **The wrong fields are filled in**

If in step 6 of the basic flow the user doesn’t have a default email, or enters in a bad email, then

1. The application will display an error informing the user that they must correct it.
2. Returns to step 5
3. **Key Scenarios**
   1. **The app can handle a limitless number of emails.**
4. The app can place an essentially limitless amount of emails in one sending.
5. **Post-conditions**
   1. **The email is sent from the system to the entered emails.**
   2. **Replying to the emails does nothing.**
6. **Special Requirements**

The app needs to keep a log of sent messages for the backend.

The app must not corrupt the files on delivery.

**College Toolkit  
Use-Case: Put Event in Scheduler**

1. **Brief Description**

The user of our application will want to, as determined from our consumer discoveries, plan their week with our Scheduling applet. This use-case will be the order of operations necessary to put an event in our scheduling app.

1. **Actor Brief Descriptions**

## User – The only actor in this use-case will be the user of application.

## Application – The application is what the user will be interacting with.

1. **Preconditions**

The User has already downloaded the main application and has allowed for notifications to be pushed to their phone/email.

1. **Basic Flow of Events**
2. The use case begins when the user opens the Scheduler applet.
3. The application will display the Calendar (or whatever list ends up being implemented for the scheduler applet).
4. The user presses a New Event button to create a new event.
5. The application will display a new screen for the event
6. The user will fill in fields pertaining to Name, Date and Time, Length (optional), Description (optional), etc.
7. The application will then place the created event on the “Calendar.”
8. When the date and time of the event occurs, the user will receive a notification about the event.
9. **Alternative Flows**
   1. **The user cancels the event creation**

If in step 3 of the basic flow the user doesn’t want to finish the event, then

1. The application will cancel the event creation.
2. The application will stop displaying the new event screen.
3. The use case returns to step 4.2
   1. **The user doesn’t fill in the right fields**

If in step 3 of the basic flow the user doesn’t fill in the Name or Date/Time fields, then

1. The application will display and error informing the user that they must fill in said fields.

2. The use case returns to step 4.5

1. **Subflows** 
   1. **The user wants to edit the created event**
2. The application will display a field that is similar to the new event but will have the appropriate fields already populated with the existing information.
3. The user can edit any of the fields, assuming the condition of (5.2) are met.
4. The user presses the Save button.
5. The application will update the fields for this update.
6. The use case returns to step 4.6.
7. **Key Scenarios**
   1. **The Application can handle a limitless amount of events**
8. The application can place an essentially limitless amount of events on the same time-span.
9. The calendar will go on for several years and perhaps even update itself for hundreds of years in the future.
10. **Post-conditions**
    1. **The event is saved and will present a notification when the time comes.**
    2. **The user can go back and edit the events even if they have already occurred**
11. **Special Requirements**

The scheduler needs to be able to handle multiple events on the same time-span.

The scheduler must keep track of at least 5 years into the future.

The scheduler must be able to create events for time down to the minutes.

The scheduler must be able to present notifications in a timely manner, perhaps even be able to edit this time prior to event for notification.

**Group participation:**

Jesus: First Use Case

Matt: Second Use Case

Vincent: Third Use Case