Juan Gonzalez  juan_gonzalez@ucsb.edu | Daniel Kang  dkang@ucsb.edu
Iris Moini-Nazeri  azadeh@ucsb.edu | Krishna Nanda  krishnananda@ucsb.edu
Arthur Phan  arthurphan@ucsb.edu

Slack:  https://allstars-space.slack.com/messages/CHK46GYDN/
Sprint 1 Trello Board:  https://trello.com/b/8ok9LoKk/sprint-1
Sprint 2 Trello Link:  https://trello.com/b/qrTOMOBA/sprint-2
Github:  https://github.com/dkang1617/cs48ProjectOrganized/tree/master
Travis CI:  https://travis-ci.org/dkang1617/cs48ProjectOrganized
Organized is an application that is looking to make the UCSB student’s life easier. The application has two main components: automatically adding course information associated
with a student into Google Calendar using the Google Calendar API and using Google Maps API to create an optimal route for traversing campus quickly and easily.

| What problem is the project solving (what is innovation, science, and/or new core technical advance)? |
| Organization of multiple classes, focused on scheduling, while also mapping the optimal routes to classes, all done with one login. |

| Why is the problem important? |
| Helps students get a schedule under control, assisting them in performing to the best of their ability. |

| How is the problem solved today? (If it is.) |
| Google Maps does limited work in map optimization. |

| Identify the outcome of the project. |
| The primary outcome is a web app where users can view upcoming assignments and find the location of the optimal route to their classes. **Reverse engineering and figuring out a way to replicate how these programs do it is the goal.** Each member should be able to learn how to use API’s, the languages required to use them, and implementing and testing them on Travis CI by the end of the project, all while working as a group in industry-style workflow. |

| Specification |
| Use a student’s schedule to create a personalized calendar with a text file upload on a server that stores their information. Also, use the schedule and Google maps, combined with a database that would help in determining the optimal route from class to class. |

| Design |
| Design should not be graphically intensive. There should be (at least) two buttons on the front end: one to add events on a calendar, and one to access the map generation feature. |

| Prototyping |
| - 1st Prototype: web based schedule to Google Calendar functionality |
| - 2nd Prototype: web design with Google Calendar and Maps functionality |

| Testing: |
We will use local hosting and Travis-CI to test our data structures and functionality.

**How do you plan to articulate, design, and implement a solution?**

Using publicly available APIs to work with the data and a server.

**List the implementation platform and technologies you will use to develop the solution:**

- Web App (JavaScript in React)
- Front End and Loading APIs (Using JavaScript)
- Java for Algorithms (Spring)
- Google Calendar and Map API

**List initial milestones (w/Sprint/dates) & how you will achieve them:**

**Sprint 1: April 15 - 26, Planning April 10-12**

- Get all the APIs authenticated and working with web app, and understanding how to efficiently use them. (Krishna + Daniel)
- Setting up a basic version of the web app. (K + D)
- Learn how to translate information from APIs, and build off of API authentication and use. (Juan)
- Setting up a way to translate the UCSB Interactive map into something we can use. (Arthur)
- Learn how to input translated information into Google Calendar (Iris)

**Sprint 2: April 29 - May 10, Planning April 26**

- Initial test of design, implement revisions as needed.
- Start making sure privacy is ensured.
- Develop previous application further.

**Sprint 3: May 13 - May 24, Planning May 10**

- Testing of map routing, using our own schedules.
- Ensuring privacy.
- Test scheduler and accuracy.
- Optimizing client side webapp
Burn Down Chart
System Architecture Overview

Requirements (Functional and Non-Functional)

(Tests in Trello)

Use Case #1: User Can Login

Actors: Web-App, User, React

Precondition: User is a UCSB Student with a valid UCSB gmail account (and is a student in the text file contained on a server meant to simulate the UCSB GOLD API), and inputs their information correctly into the Gmail login using their UCSB email.

Flow of Events:

1. User opens Web-App
2. User sees Gmail login interface on web-app, and enters log-in information
3. This information is used to access their course information contained in a text file on a server and this course information it placed into the Google Calendar.
Postcondition: Web-app has access to information and can use that to build schedule and map routes for classes.

Acceptance Test:

Scenario: Login is successful

- Given the user is a UCSB student (and their info is in the textfile)
- And their login information is inputted correctly
- When the Web-App attempts to access and view user’s schedule
- Then GauchoSpace will allow it to do so
- And the Web-App will be able to use the information to input.

Exception Case:

Scenario: Login is unsuccessful

- The User is not a UCSB Student with valid credentials (is not a student in the textfile)
- The User incorrectly inputted their information
- The web-app fails to pull information from the text server using the user’s information.

Trello Card Link:

https://trello.com/c/g3zAUHRH/56-get-google-oauth-working-2-days

Use Case #2: Add Node

Actors: User, Website Interface

Precondition: User is logged into the website

Flow of Events:

Basic:

1. User clicks an “Add Node” button
2. The website allows the user to type the building name in
3. The website allows the user to type in whether it is done biking or walking (default: walking)
4. The user accepts the input, if it is valid
5. Node is added at the bottom of the list

Alternative:

1. User clicks an “Add Node” button
2. The website allows the user to type the building name in
3. The website allows the user to type in whether it is done biking or walking (default: walking)
4. User has exceeded their 10 node limit, do not add new node and warn user
5. User’s input is invalid, inform user

Postcondition: User has a new node on their list of nodes, or there is no new node added

Acceptance Test:

Scenario: Node successfully added

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User types in “Anacapa Hall” and their preferred method of transport
- The user confirms the input
- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps, Anacapa Hall

Exception Case:

Scenario: Cannot add node (exceeds 10 limit)

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps, Anacapa Hall, HSSB, Phelps, HFH, Santa Cruz Hall
- User types in “Anacapa Hall” and their preferred method of transport
- The user confirms the input
- The webapp gives an error: “The list cannot exceed 10 locations”

Trello Card Link:

| Use Case #3: Move Node position |

Actors: User, Website Interface

Precondition: User is logged into the website, and at least two nodes are in the list

Flow of Events:

Basic:

1. User selects a node
2. It can be moved up one slot or down one slot, being replaced with the node in the slot above it or below it (as in, Node N in a list of length N cannot move down).
Postcondition: The node is moved up or down, depending on whether it is not in slot 1, or at the last possible slot in the list.

Acceptance Test:

Scenario: Node is moved up/down

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User selects “Girvetz”
- The user presses to move it up/down
- If it was pressed up: the list is as follows: Girvetz, Santa Cruz Hall, Library, Gym, Phelps
- If it was pressed down: the list is as follows: Santa Cruz Hall, Library, Girvetz, Gym, Phelps

Exception Case:

Scenario: Node cannot be moved

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User selects “Santa Cruz Hall”
- The user presses to move up
- No error message occurs, it cannot be moved up from first in list
- User selects “Phelps”
- The user presses to move down
- No error message occurs, it cannot be moved down from last in list

Trello Card Link:

| Use Case #4: Remove Node |

Actors: User, Website Interface

Precondition: User is logged into the website, user has at least 1 node.

Flow of Events:

Basic:

1. User clicks the node they want to delete
2. The user confirms deletion
3. Node is removed from list
4. All nodes under the deleted node are moved up one slot

Postcondition: User has one less node on their list
Acceptance Test:

Scenario: Node is removed

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User selects “Girvetz”
- The user confirms the input
- The list is as follows: Santa Cruz Hall, Library, Gym, Phelps

Trello Card Link:

| Use Case #5: Generate map |

Actors: User, Interface, Google Maps API Function

Preconditions: There are no more than 10 Nodes, and at least 2 Nodes, in the user’s data table.

Flow of Events:

1. For each node, our code will screenshot the ICM map
2. The maps will then be overlaid and then put into one image
3. A print button will show

Postcondition: The map is generated and is displayed on the website, showing the optimal paths to getting to class.

Acceptance Test:

Scenario: Map is generated correctly

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User selects “Generate Map”
- A map showcasing the optimal paths (taking into account the methods of transportation requested) to getting from Santa Cruz Hall, to Girvetz, to Library, to Gym, to Phelps is made

Exception Case:

Scenario: For whatever reason, the map cannot be generated

- The list is as follows: Santa Cruz Hall, Girvetz, Library, Gym, Phelps
- User selects “Generate Map”
- Error message shows: “Sorry, we couldn’t generate the map right now. Try again in a few hours.”
Use Case #6: Print map

Actors: User, Website Interface

Precondition: User is logged into the website and there is a map already generated

Flow of Events:

Basic:

1. User clicks a “Print Map” button
2. The website opens the map in a print window

Postcondition: User has a downloaded version of their map.

Acceptance Test:

Scenario: Map image is turned into PDF

- User selects “Print map”
- The web browser opens up the print window for the map
- OR the web browser downloads a pdf of the map to be printed
- Message is shown: “If a window did not open or the pdf does not download, try again”

Exception Case:

Scenario: Map does not bring up print window

- User selects “Print map”
- Print window does not open
- OR web browser does not download
- Message is shown: “If a window did not open or the pdf does not download, try again”

Use Case #7: Edit Calendar Entry before putting into Calendar

Preconditions: User has already generated at least one entry.
Flow of Events:

Basic path:

1. The entry generated has some mistakes
2. The user edits the entry to improve it
3. The user confirms to add the individual entry to calendar

Alternative Path:

1. The user does not want to edit the entry
2. The user confirms to add the individual entry to calendar

Postcondition: Entries are edited or kept the same, and then are manually approved to be added to the calendar. The entries that are confirmed are removed from the list.

Trello Card Link:

**Use Case #8: Add All Entries to Calendar**

Preconditions: User has already generated at least one entry, and we have information to input into Calendar.

Flow of Events:

Basic path:

1. The user clicks on “add all”
2. The website asks for confirmation
3. The user confirms and all entries are added to calendar and removed from the list

Alternative Path: The user cancels adding all entries onto calendar

Postcondition: Every entry is added to the calendar, and there are no more entries on the list to account for, OR no entry is added.

Trello Card Link:

| Use Case #9: Import Map List |

Actors: User, Website Interface, Client storage

Precondition: User is logged into the website and has a text file of their list they want to import

Flow of Events:

Basic:

1. User clicks a “Import List” button
2. The website imports the list, reads it, and generates the list on the website interface
3. The list can be generated if the user wants

Alternative:

1. User clicks a “Import List” button
2. The website imports the list, attempts to read it, but the text file is not in proper format
3. The website writes an error message

Postcondition: User has imported their map from their computer.

Acceptance Test:

Scenario: Text file is uploaded, read from, and not stored in server side

- User selects “Import List”
- The website imports the list, attempts to read it, and checks if it is in proper format
- The list is properly imported
- “The list file could not be read, please try again”

| Use Case #10: Export Map List |

Actors: User, Website Interface, Client storage
Precondition: User is logged into the website and has a text file of their list they want to download to their machine (to save)

Flow of Events:

Basic:

1. User clicks a “Export List” button
2. The website write the list to a temporary text file
3. The website prompts for the user to download
4. The text file is deleted on website exit

Postcondition: User has downloaded their map to their computer.

Acceptance Test:

Scenario: Text file is downloaded from user list

- User creates their own list
- User clicks a “Export List” button
- The website write the list to a temporary text file
- The website prompts for the user to download
- The text file is deleted on website exit
Appendix

- Front End and Loading APIs (Using JavaScript)

- Google Calendar API
- Java for data structure
- Web App (JavaScript)