**Project Abstract**

This application, **StudyScapes**, will allow users to **login** to their account and view the Simon Fraser University campus as a virtual map. Login **views** will differ for faculty and students. This uses the **Web API for Google Maps** for location and time updates. As a **real-time feature**, this application will use **Socket.io** to update requests made to professors or students for meetings and study groups in real time. This application enables quick meetups and easy navigation for users. The application is not just practical, but fun! There are minigames incorporated to help the user kill time while waiting between meetups (they can even battle their professors!).

**Customer**

**Faculty:** Teachers and TAs at Simon Fraser University that would like to interact in a convenient way with students, plan meetings, coordinate meeting locations, schedule or find events on campus, and enjoy minigames.

**Students:** Students at Simon Fraser University that would like to communicate with all their professors or TAs in one convenient application. They would like to meet with other students or faculty at SFU, get assistance in finding their way around the school, find specific buildings or rooms, schedule or find events on campus, and meet other students or just have fun through minigames.

**Competitive Analysis**

What makes StudyScapes so unique is that it applies specifically to the Simon Fraser University Burnaby Campus. It incorporates ideas from many other successful programs into one main application for ease of access. Although Google Maps is useful, it does not provide a detailed view of the Burnaby Campus. StudyScapes will elaborate on the map with this campus in mind, allowing for much more extensive interaction and location finding around the University. SFU Snap was an inspiration for this, as it helped students to search for and travel between rooms and buildings across campus. StudyScapes expands this to allow for scheduling events and meetings around campus with students and faculty alike. It also provides ice-breakers for students in the form of minigames and different views for faculty and students, so that each user sees the information that directly applies to their own needs.

**Main Features - Epics**

**Map:** Provide an interactive map that updates to user location so they can find other users and locations on the Simon Fraser University Campus.

**Meetings:** Enable faculty and students to easily interact and meet up for studying or course/career discussion.

**Events:** Have events held by clubs, faculty or student societies show on the map as icons in real-time, so users can quickly learn about the activity on campus.

**Minigames:** Allow students and faculty to interact in mini-games with those nearby based on location. This would allow users to play small various games that can act as ice-breakers between their fellow peers.

**User Stories**

**Actors**:

**Faculty** (Professors and Teaching Assistants) that can view and cancel meetings. They can also interact with students, view the map, view and schedule events, and play minigames. Students can view, request, reschedule, and cancel meetings with Faculty and other students. They can view and schedule events, view the map for specific rooms and buildings, and play minigames.

**Students** can view, request, reschedule, and cancel meetings with Faculty and other students. They can view and schedule events, view the map for specific rooms and buildings, and play minigames.

**Current Iteration Stories**

**Name/Description:** A new student is lost on the Burnaby campus

**Actors:** New student named ‘Sally’

**Triggers/Preconditions:** Sally logs in to StudyScapes and opens the map page to find her current location and destination

**Actions/Postconditions:** Display a campus map showing her current location and chosen destination (potentially with a path)

**Acceptance Tests:**

* Location services are enabled
* Current location can be found and display
* Destination is valid and can be displayed

**Iteration:** Started in *001*

*---*

**Name/Description:** A student wants to meet with another student

**Actors:** Multiple students named ‘Maddie’ and ‘Bud’

**Triggers/Preconditions:** Maddie accesses the meetup page and schedules a meeting with Bud

**Actions/Postconditions:** Displays a form for Maddie to specify the meeting location and time, then sends a request to Bud

**Acceptance Tests:**

* Meetup form is displayed properly
* Bud is an existing student and a request can be sent to him
* Meeting location is valid and can be chosen
* Form can be submitted
* Meeting data is stored in the meetup table

**Iteration:** Started in *001*

**---**

**Name/Description:** A professor wants to review their scheduled meetups

**Actors:** A faculty member, Professor Abe

**Triggers/Preconditions:** Professor Abe views the meetups page on their account

**Actions/Postconditions:** Displays a table that only lists the meetings including Professor Abe. Each row shows the corresponding student, the requested room, and the requested time.

**Acceptance Tests:**

* Meetup table is displayed with just Professor Abe’s meetings
* Valid rooms and times are displayed
* The corresponding student exists
* Finished/cancelled meetups are not displayed

**Iteration:** Started in *001*

**---**

**Name/Description:** A professor wants to cancel a scheduled meetup

**Actors:** A faculty member (Professor Bee) and a student (Erik)

**Triggers/Preconditions:** Professor Bee chooses to cancel her meeting with Erik from her meetups page, by clicking the cancel button next to the listed meeting

**Actions/Postconditions:** The meeting is listed as cancelled on the table (greyed out) and Erik is notified of this cancellation on his meetups page.

**Acceptance Tests:**

* The isCancelled attribute for that meeting is set to true
* The row is greyed out and the cancel button is no longer clickable
* Erik is able to view the meetup as cancelled through his meetups page
* The scheduled meeting remains in the database

**Iteration:** Started in *001*

**---**

**Name/Description:** A student responds to a cancelled meeting

**Actors:** A faculty member (Professor Bee) and a student (Erik)

**Triggers/Preconditions:** Erik wishes to view the meeting that was cancelled by Professor Bee, and confirm the cancellation

**Actions/Postconditions:** Erik clicks ‘confirm cancellation’ from the meetups page and the meeting is removed from both his and Professor Bee’s meetups

**Acceptance Tests:**

* The meetup (row) is removed from the *meetup* database
* Erik’s page is refreshed to no longer show the meeting
* Professor Bee can no longer see the meeting on her meetup page

**Iteration:** Started in *001*

**---**

**Name/Description:** A new student wants to meet other students with similar interest on the Burnaby campus

**Actors:** New student named ‘Patric’

**Triggers/Preconditions:** Patric logs in to StudyScapes and opens the events page

**Actions/Postconditions:** After finding a suitable event he can now meet students who share same interests as him

**Acceptance Tests:**

* Different events are displayed to the users
* Students can drop in to their desirable event

**Iteration:** Started in *001*

*---*

**Name/Description:** A student has a 5 hour break between two classes and he/she wants to pass some time while being on the campus

**Actors:** Stacey a fourth year student

**Triggers/Preconditions:** Stacey logs in to StudyScapes and opens the mini games

**Actions/Postconditions:** Display different options for mini games

**Acceptance Tests:**

* Game services are enabled
* Scores can be found and display

**Iteration:** Started in *001*

**Future Iterations**

**---**

**Name/Description:** A new student is lost and looking for their classroom.

**Actors:** A new student, Justin

**Triggers/Preconditions:** Student Justin clicks the Room Finder Tab on their page.

**Actions/Postconditions:** Shows a map allowing Justin to search for his classroom or view the entire university as a whole.

**Acceptance Tests:**

* View every class room
* Rooms are searchable

**Iteration:** Implement in Iteration 2

**---**

**Name/Description:** A new student is looking to make new friends at SFU, wants to play a game.

**Actors:** A student, Sophia, wants to play a game with someone.

**Triggers/Preconditions:** Student Justin clicks Minigames tab on their page.

**Actions/Postconditions:** Shows different types of minigames and allows them to play with people nearby.

**Acceptance Tests:**

* Able to connect with other players
* Game logic is correct
* Able to choose what type of game to play
* Able to correctly interact with game buttons.

**Iteration:** Implement in Iteration 2

**---**

**Name/Description:** Viewing upcoming events on campus.

**Actors:** A user (student/faculty)

**Triggers/Preconditions:** User clicks the “View Upcoming Events” on a page.

**Actions/Postconditions:** Shows a page with a table where the rows are Events showing its name, host group, website link, location, time, and date in chronological order.

**Acceptance Tests:**

* An Events table is displayed with information described in postconditions.
* Events are listed in chronological order (closest to farthest from current time).
* Events prior to current time are not shown.
* Cancelled events have colored text (red) and shows “CANCELLED” under “time” (“CANCELLED” should be reflected in the database under “time” also).

**Iteration:** Implement in Iteration 2

**---**

**Name/Description:** Viewing events happening today on the map.

**Actors:** A user (student/faculty)

**Triggers/Preconditions:** The user goes to the campus map page (not viewing room finder)

**Actions/Postconditions:** The map loads showing SFU campus. Events listed to happen today are shown with event markers at their respective locations.

**Acceptance Tests:**

* The map is rendered correctly, showing SFU campus.
* Each event marker represents an event happening *today*.
* Each event marker is shown at their correct locations.
* Clicking on a marker displays detailed information regarding the event.

**Iteration:** Implement in Iteration 2

---

**Name/Description:** Hiding a type of marker on the map.

**Actors:** A user (student/faculty)

**Triggers/Preconditions:** The user clicks on any of the box from a group of checkbox options to for hiding a type of marker on the map (i.e. hide location markers, hide event markers, etc.).

**Actions/Postconditions:** The map is re-rendered with the desired markers to be hidden gone.

**Acceptance Tests:**

* The map is re-rendered with the same position and zoom level as prior to triggering the event.
* The desired markers to be hidden are not seen while the others are still shown.
* The hidden markers cannot be interacted with.

**Iteration:** Implement in Iteration 2

**User Interface Requirements**

![A close up of text on a white background

Description automatically generated]()

![A close up of text on a white background

Description automatically generated]()

![A close up of a sign

Description automatically generated]()

![A close up of a sign

Description automatically generated]()

![A close up of a sign

Description automatically generated]()

**Meeting 2 Overview**

Our team discussed separating SFU’s Room Finder from our Google map. More research is needed regarding Room Finder and Esri. Our map should be for seeing building locations, nearby students, and current events while Room Finder should be solely used for finding classrooms. We also decided to add an Admin user who can view all database tables for ease of debugging and monitoring.

**Plan for iteration #2**

The plan in iteration 2 will have Celina implement the Meeting Scheduling System, and Josh implement the Events database and integrate Events with the map. Parth plans to start designing minigames, and Mandeepa is looking into sockets for a messaging system.

**URLs**

**GitHub Repository:** *https://github.com/Guojiaxi/sfu-cmpt276proj.git*

**Heroku link:** *https://cmpt276proj-jlguo.herokuapp.com/*

**Heroku Git Link:** *https://git.heroku.com/cmpt276proj-jlguo.git*

**Features Board**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| COMPLETED | IN PROGRESS | TO BE IMPLEMENTED | PREREQUSITE(S) NEEDED | TO BE DESIGNED | BROKEN  /REDESIGN |
| map |  | Logged-in tracking | Plotted meetings | Meeting notifying system |  |
|  |  | Sign-up system | Plotted participating events | Event notifying system |  |
|  |  | Dashboard schedule timetable | Event markers | Meeting creation UI |  |
|  |  | Room finder | meeting invitation/response | Event posting UI |  |
|  |  | Place map markers | minigames | Event submission approval system |  |
|  |  | User location tracking |  | User profile page |  |
|  |  | Server-side storage for meetings |  |  |  |
|  |  | Server-side storage for events |  |  |  |
|  |  | Events timetable |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

GREEN = FEATURE COMPLETED

BLUE = IN PROGRESS (\*Put down who is working on thing\*)

RED = FEATURE TO IMPLEMENT (Can be worked on at current moment)

YELLOW = PREREQUSITE NEEDED

PINK = TO BE DESIGNED

GRAY = BROKEN/NEEDS TO BE REDESIGNED

Study Scapes

-Parameters

-guest homepage = ?

-user homepage = ?

-Security

-login

-login system

-logged-in tracking (needs knowledge on cookies and stuff idk)

-sign-up

-sign-up system

-Dashboard

-schedule

-schedule timetable

-plotted meetings (needs schedule timetable)

-plotted participating events (needs schedule timetable)

-user-defined profile page

-Map

-room finder

-geolocation

-map

-place markers (locationArray acquired!)

-user location tracking (needs knowledge of sockets)

-event markers (needs server-side storage of events)

-Meeting scheduling

-meeting creation

-UI for ‘building’ a meeting

-server-side storage of meetings

-meeting communication

-meeting notifying maybe through email?

-invitation/response (needs server-side storage and meeting notifying system)

-Campus Event tracking

-event listing

-timetable

-event notifying system (optional)

-event posting

-UI for posting an event

-server-side storage of events

-event submission approval

-Minigames (needs real-time user location tracking)