Revision #: 2 Date: October 14th, 2019

SW Engineering CSC 648 / 848 Fall 2019

Milestone 1:

Use cases, High Level Requirements and Architecture

GatorDater

Team 204:

Tejas Vajjhala: Team Lead, Github Master

Sanil Rijal: Front-end Lead

Thomas Zhang: Front-end assist

Aaron Li: Back-End Lead, Database Master

Milestone	Version	Date Submitted For Review
Milestone 1	2	10/14/2019
Milestone 1	1	10/03/2019

Table of Contents

Executive Summary

GatorDater is an online application that facilitates the scheduling of office hours for students and professors. Previously, students would need to schedule with professors via a paper list located outside the professor's office room, or send an email to the professor which makes it hectic to synchronize with each other. This causes students to fall behind in class, and be unable to even ask for help as they are not able to get in touch with the professor face to face outside of class.

By using GatorDater, office hours are easily scheduled through the interface and professors do not need to wait in their office for a student that may or may not show up during an unscheduled period. By scheduling online, both students and professors have a clear calendar of their appointments and can ensure that they have an allocated time to discuss with each other. With the addition of the calendar, both students and professors are able to view and update their current status regarding time slots and availability. This allows for more fluidity and ensures less stress on both students and professors alike.

Main Use Cases

Use Cases : Student

1) A San Francisco State University student, Wesley, wants to schedule an appointment with his Math professor, Valerie for follow up questions. After reviewing his schedule, he comes to the conclusion that he is available after 3pm. He finds his professor's name and an associated time slot after 3pm. He then makes the reservation under Wesley.
2) A San Francisco State University Student Wesley, has reserved an appointment with programming Professor Shawn for tomorrow from 3:30 pm to 4:30 pm. But, Wesley realizes that he may not be able to attend the meeting because of his grad assignment that is due tomorrow. So, he logs in and cancel the appointment; making the professor office slot available for other students.

Use Cases: Professor

1) Professor Jake wants to view his upcoming office hours. He checks his calendar and sees his upcoming scheduled appointments. He realizes that he will not be available during one of the booked hours, and cancels the scheduled meeting. He then realizes this will be a recurrent issue and sets a block which makes students unable to book office hours during the specified hours/days.
2) A San Francisco State University professor, Jeff, just started teaching and wanted to host office hours. However, he did not want to attend school on his off days at his office, if no students were to show up. Therefore, he posted his time slot online for students to reserve. Once, a time slot is occupied, Jeff will arrive to school and host his office hours accordingly.
3) A San Francisco State University professor, Valerie, logs in and shall see the reservations for the upcoming office hours. She sees that one of her office hour's time slots at 5pm is reserved by Wesley. On the day of, Valerie waits for Wesley at her office to do her office hours.
4) A San Francisco State University Professor Jacob, logs in and see that a student has booked an appointment with him from 3:00 pm to 4:00 pm. But, the professor has an emergency and realizes that he will not be able to wait for the student. So, he cancels the appointment and sends the possible alternative office hours to the student.

Use Cases: Unregistered User

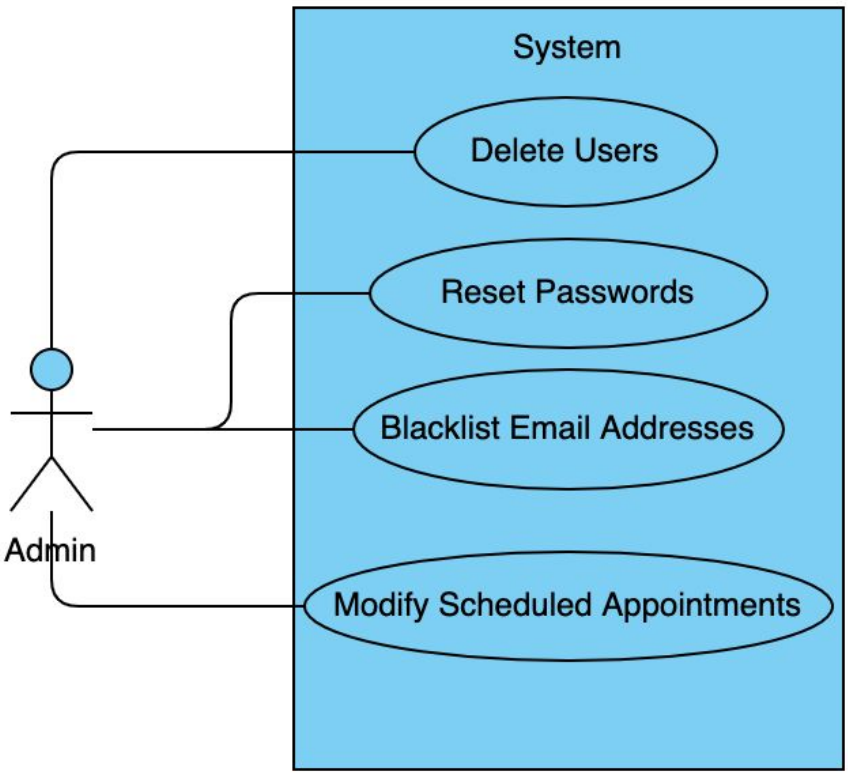
1) Jenny wants to schedule an office hour appointment with her professor. She has never used GatorDater before. She loads the website, signs up for the service, then logs in. She searches for her professor, and views the free slots for office

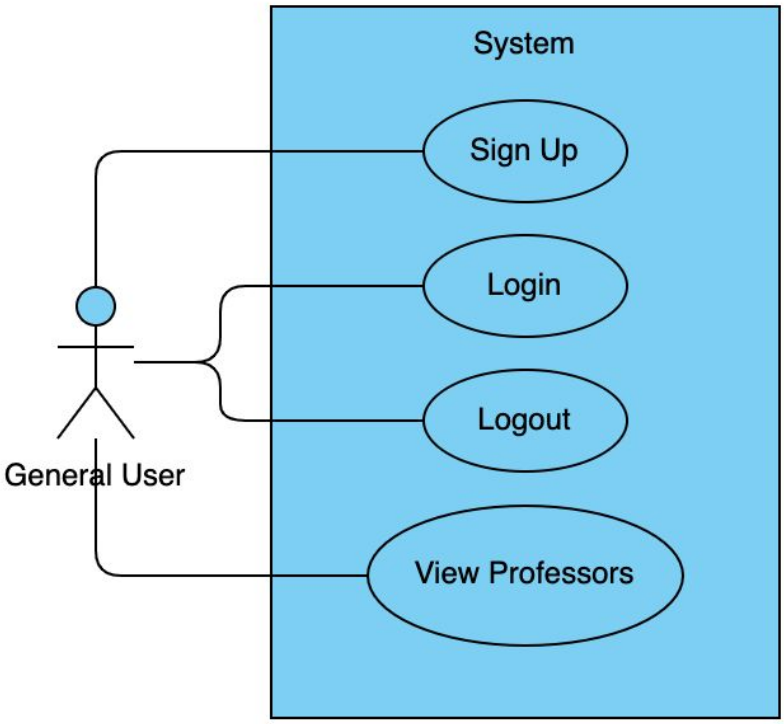
hours. She then books a time slot with the professor and has it saved on her personal calendar.

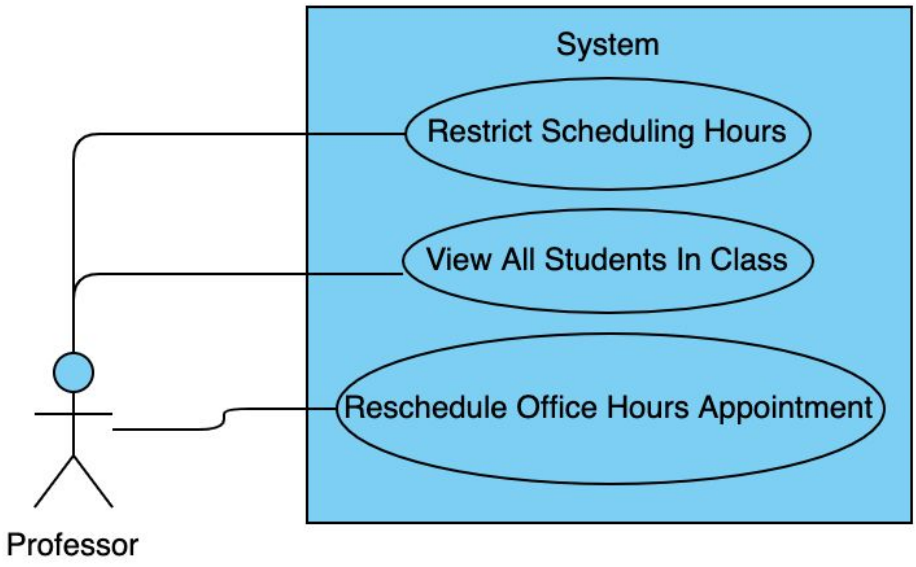
Use Cases: Admin

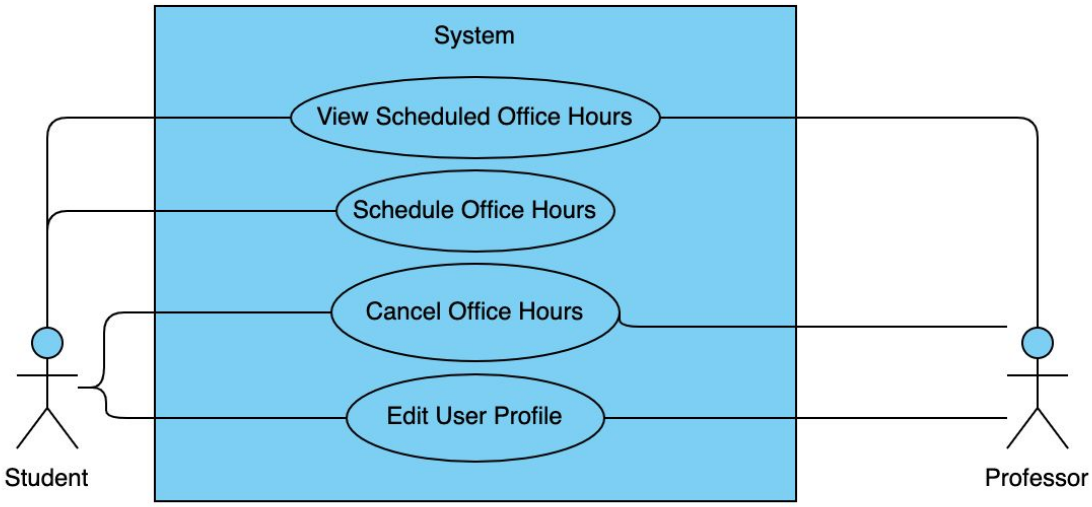
1) A user has been spamming professors and blocking all of their office hours. In order to free up these sections for other users, and prevent this user from blocking off hours, admin Joe deletes this user's account and blacklists his email address.

Diagrams:


System
Delete Users
Reset Passwords
Blacklist Email Addresses
Modify Scheduled Appointments
Admin


System
Sign Up
Login
Logout
View Professors
General User


System
Restrict Scheduling Hours
View All Students In Class
Reschedule Office Hours Appointment
Professor


System
View Scheduled Office Hours
Schedule Office Hours
Cancel Office Hours
Edit User Profile
Student
Professor

List of main data items and entities

Users:

1) Registered User who has an account
2) Unregistered User who doesn't have an account

Student:

1) Registered User who can schedule office hours with a professor
2) Registered User who can cancel their appointment
3) Registered User who can view the calendar with their respected hours

Professor:

1) Registered User who can post and accept/decline hours
2) Registered User who can view the calendar with all the respected students hours

Admin:

1) Registered User who can access all data and content and modify the database
2) Registered User who can schedule office hours with professor
3) Registered User who can post and accept/decline hours

Application:

1) Calendar to display scheduled office hours associated with students and professors
2) User profiles for students and professors to provide a bio
3) Search bar to display multiple professors

Initial list of functional requirements

For Registered Users:

1) Users shall be able to make an account
2) Users shall be able to login
3) Users shall be able to access the office hour page
4) Users shall be able to cancel their scheduled office hours appointment
5) Users should be able to update their profile
6) Users shall be able to specify which class they belong to
7) Users shall be able to specify whether they are professor or student
8) Users shall be able to specify which major they are in
9) User shall be able to contact the creators for help

For Registered Users That Are Students:

1) Users shall be able to search for their professors
2) Users shall be able to reserve a time slot
3) Users shall be able to cancel their appointment
4) Users shall be able to schedule multiple hours
5) Users shall be able to monitor respected hours on the calendar
6) Users shall be able to have multiple appointments

For Registered Users That Are Professors:

1) Professor shall be able to post their available time slots.
2) Professor shall be able to restrict hours from being scheduled
3) Professor shall be able to specify which class is allowed to schedule office hours
4) Professor shall be able to block a whole day from hosting office hours
5) Professor shall be able to monitor schedules hours on the calendar

System Tasks:

1) System shall make reserved time slots unavailable
2) System shall send an email when a user signs up
3) System shall add scheduled office hours to personal calendar
4) System shall check professor's calendar for free slots
5) System shall add selected office hours to professor's calendar
6) System shall send email to student and professor when office hours scheduled
7) System shall send email to student/professor when meeting cancelled.

List of non-functional requirements

Availability:

1) Scheduling Office Hours System shall be available for use 24/7
2) Register/Login System shall be available for use 24/7
3) In the event that a system is non-operational, there will be a notification to inform the users about its unavailability

Documentation:

1) All documents of the system must be embedded in source code

Capacity:

1) All system should be able to handle at least 10 users at once
2) All system should be able to handle at least 20 scheduling at once

Interoperability:

1) The system shall be able to work on any browser (Chrome, IE, Firefox, etc)
2) Should be compatible with mobile usage

Maintainability:

1) Any system shall not be down for maintenance for more than 24 hours

Performance:

1) Load time shall not take more than 3 seconds

Security:

1) Access will be granted to only legitimate users
2) Access to only registered users
3) The system will store passwords as hashed
4) Every unsuccessful attempt by user to access the system will be recorded

Competitive analysis

	Outlook	iLearn	Google Calendar	GaterDater
Calendar	+	+	+	++
Messages	+	+	-	+
Reservation	-	-	-	++
Search/Filter	+	-	-	+
Reminder	-	-	+	+

In terms of scheduling, all the applications listed above can perform some type of scheduling. Like the analysis above, all competitors contain a calendar like Google Calendar to notify the users of the current month and day. However, GaterDater brought it up to another level. Where the users are able to not only view but also reserve a time slot from the calendar. Which solves iLearn and Outlooks problems of both students and professors being clueless on the status of office hours. Referring back to reserving time slots, the competitors doesn't provide a reservation function. Where as GaterDater allows users to reserve or unreserve their desired time slot with their professor. In addition, users are able to search for their respective professor to schedule hours for more simplicity. Unlike most of the other applications, it will also become color coordinated to indicate the current status regarding hours. Therefore, GaterDater will be the next big thing.

High-level system architecture and technologies used

Server Host: Google Compute Engine 1vCPU 614MB RAM (Free tier for now)
Operating System: Ubuntu 18.04 Server
Database: MySQL 8.0.17
Web Server: Apache 2.4.41
Server-Side Language: Node.js, PHP
Additional Technologies: PHPMyAdmin, maven, bootstrap
IDE: IntelliJ
Web Analytics: Google Analytics
SSL Cert: Lets Encrypt (Cert Bot)
SASS: 3.5.5

Team

Tejas Vajjhala: Team Lead, Github Master
Sanil Rijal: Front-end Lead
Thomas Zhang: Front-end assist
Aaron Li: Back-End Lead, Database Master

Checklist

Team found a time slot to meet outside of the classroom: DONE
Github master chosen: DONE
Team decided and agreed together on using the listed SW tools and deployment server: DONE
Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing: DONE
Team lead ensured that all team members read the final M1 and agree/ understand it before submission: DONE
Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.) DONE