**CS3354 Software Engineering Final Project Deliverable 2**

# Procrastify

Group 6

Alexander Ho, Megan Ingram, Michael Kasman, Samah Khan, Alan Liao,
Damian Ozuna, Sarah Tempelmeyer

1. Delegation of tasks:
   Deliverable 1:
   Software Model - Sarah
   Functional Requirements - Megan
   Non-functional Requirements - Damian
   Use Case Diagram - Samah
   Sequence Diagrams - Alexander, Samah, and Sarah
   Class Diagram - Sarah and Michael
   Architectural Design - Alan

   Deliverable 2:
   Project Scheduling - Sarah and Samah
   Algorithmic Estimation - Everyone
   Cost(Hardware, Software, Personnel) - Samah and Damian
   Testing - Michael, Alex, and Alan
   Comparison - Sarah and Samah
   Conclusion - Megan

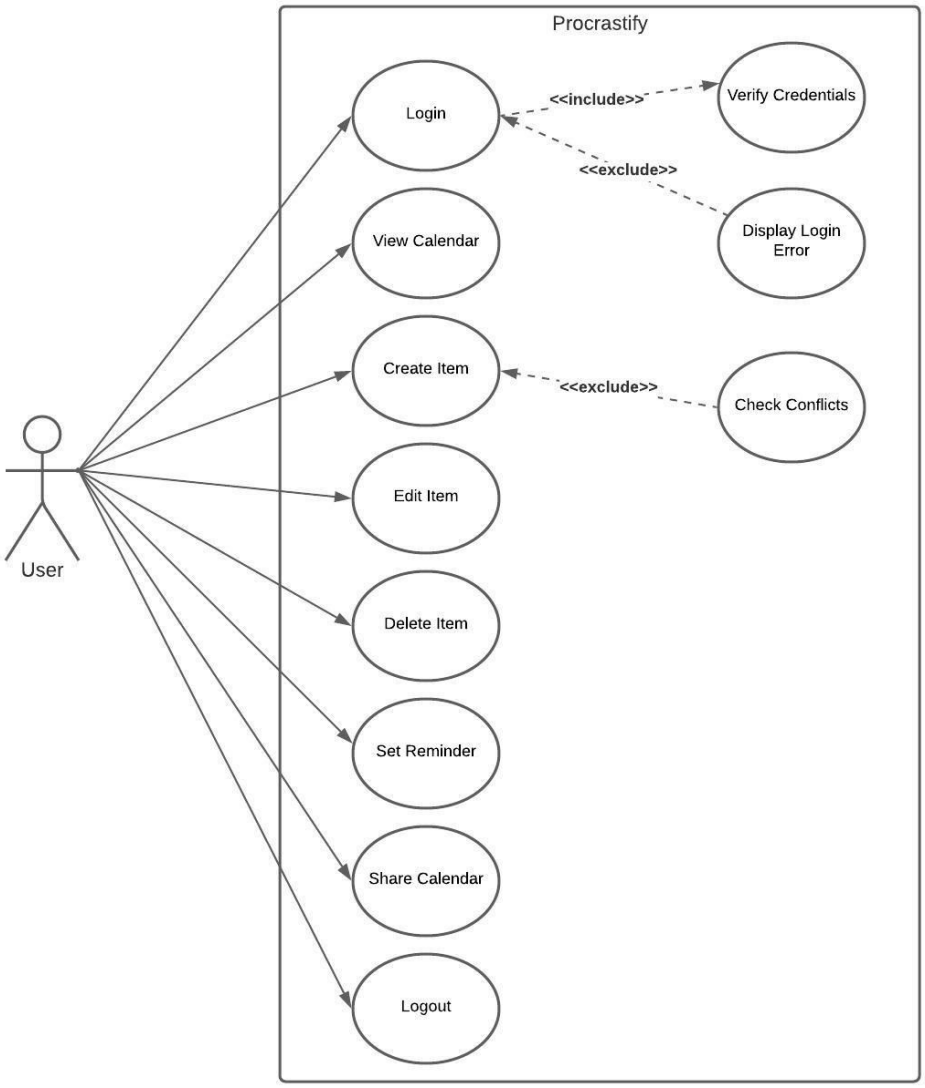
2. Project Deliverable 1 Content:
   a. GitHub URL: https://github.com/MCKasman/3354-Group-Six.git
   b. Delegation of tasks:
      Software Model - Sarah
      Functional Requirements - Megan
      Non-functional Requirements - Damian
      Use Case Diagram - Samah
      Sequence Diagrams - Alexander, Samah, and Sarah
      Class Diagram - Sarah and Michael
      Architectural Design - Alan
   c. Software Process Model
      i. The incremental process model will be utilized when developing Procrastify. This model is a software process model which entails that iterations repeat of the five framework activities (communication, planning, modeling, construction, deployment) which follow one another consecutively. Each of the project deliverables moves the team towards final written code and, theoretically, towards implementation and validation. Although this code will not be released to actual users, it can be assumed that the group would perform additional code adjustments addressing customer feedback. This can be represented through future iterations of communication, planning, modeling, construction, and development once more.

d. Software Requirements
    i. Functional requirements:
        1. A user should have a unique email and password to both identify themselves and login with
        2. A user should have a weekly/monthly/yearly view, agenda view, to-do list view, and a notes view in a calendar display with all items entered by said user
        3. A user should be able to enter an item with a title, content, category, and date details and have it stored on their calendar
        4. A user should be able to search their calendar for all items entered by date, title, category, and course(if provided by user)
        5. A user should be able to edit or delete any chosen item from their calendar
        6. The user should receive notifications from the system for upcoming items at intervals selected by the user
        7. A user should be able to share their calendar through the email provided during login
    ii. Non-functional requirements:
        1. Product Requirements
            a. Usability Requirements: The application must be intuitive enough to navigate through the calendar as well as its functions with ease. The user should be able to search, create, delete, or edit objects from the calendar display.
            b. Efficiency Requirements
                i. Performance Requirements: Application must be efficient enough to operate smoothly on mobile devices by maintaining a relatively small footprint in terms of CPU and memory usage.
                ii. Space Requirements: Application must be compact enough to be stored on limited space of new and older mobile devices. Ideally the application will be smaller than 500 MB.
            c. Dependability Requirements: The system should reliably provide its core functionality of creating objects and organizing the data within them even when offline.
            d. Security Requirements: The application must authenticate the user's account before granting access to data stored on the app. This can be implemented in different ways, such as touch or face ID as well as through a password.
        2. Organizational Requirements

a. Environmental Requirements: The operating environment of the system will be on mobile devices.
b. Operational Requirements: The system will be used to automatically organize objects by date and type (Homework, quizzes, notes). These objects are initialized with data given by the user and are stored by the application on the user's device.
c. Development Requirements: This application will be written using Java as an Android app.

3. External Requirements
    a. Regulatory Requirements: This application must fall in line with FTC regulations such as the Federal Trade Commission Act where developers and operators must adopt reasonable privacy practices.
    b. Ethical Requirements: The application must ensure the security of the user's data. The app will adhere to the user's granted level of access (such as to saved pictures or other files) when gathering information from the user's device. This data will not be used in ways other than the functionality of the application.
    c. Legislative Requirements
        i. Accounting Requirements: This application will fall in line with 26 CFR § 1.861-18, which outlines tax based on any income and transactions involving computer programs.
        ii. Safety/Security Requirements: The application must be in compliance with General Data Protection Regulation (EU GDPR) which ensures the maintenance of records of the consent and usage of data from users.

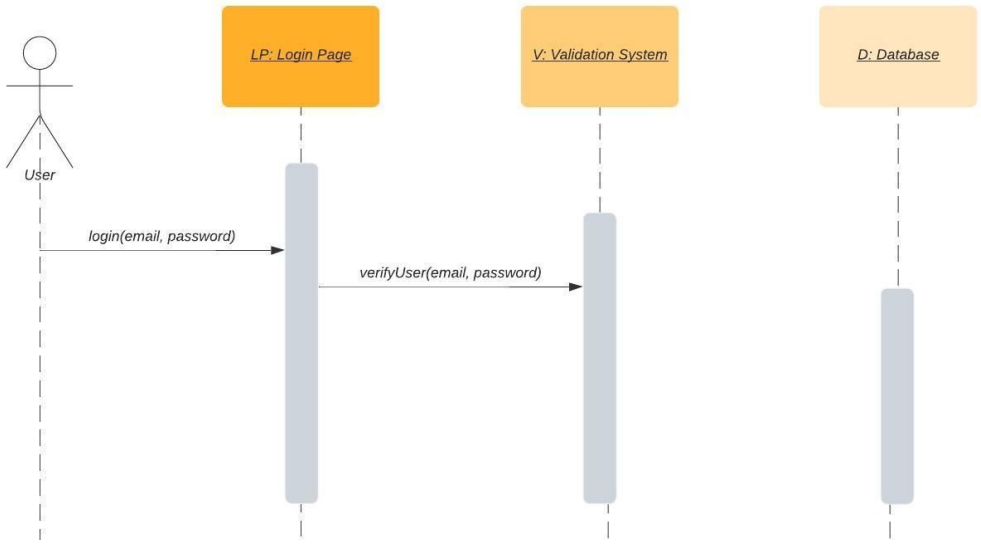
e. Use Case Diagram



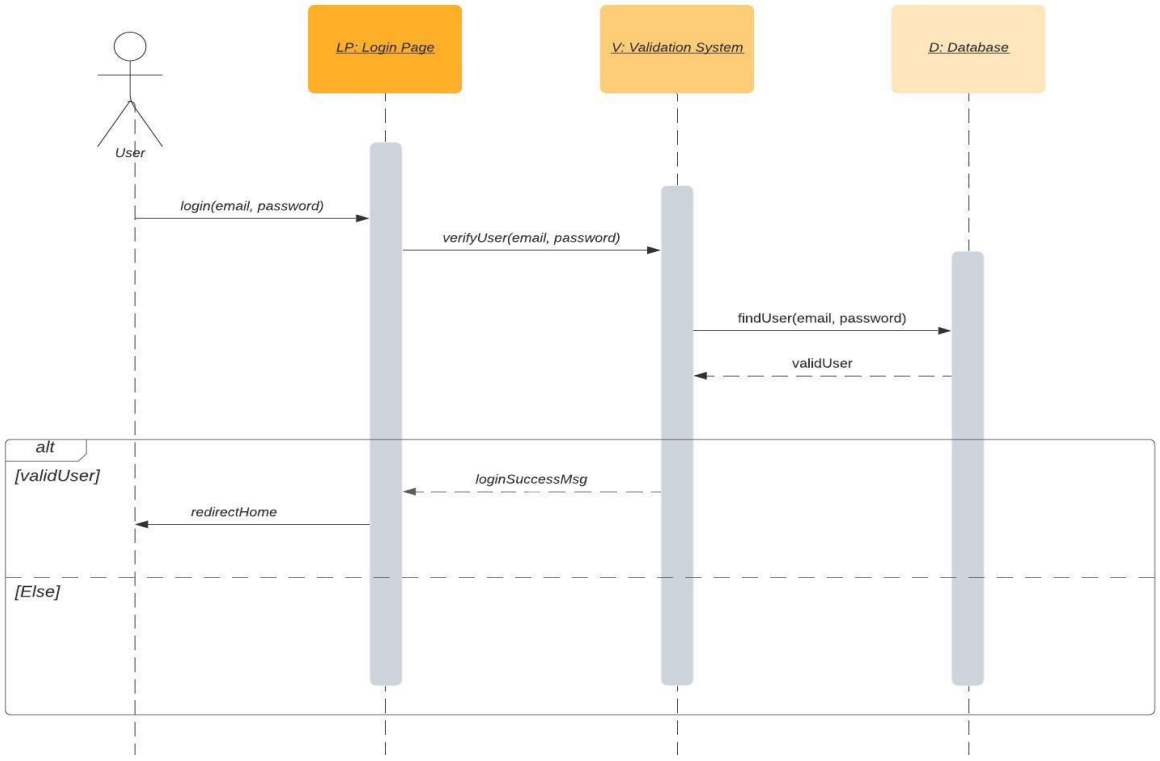

Item includes an event, a to-do list, a to-do list task, or a note. “Check Conflicts” is executed if item is an event.

f. Sequence Diagrams

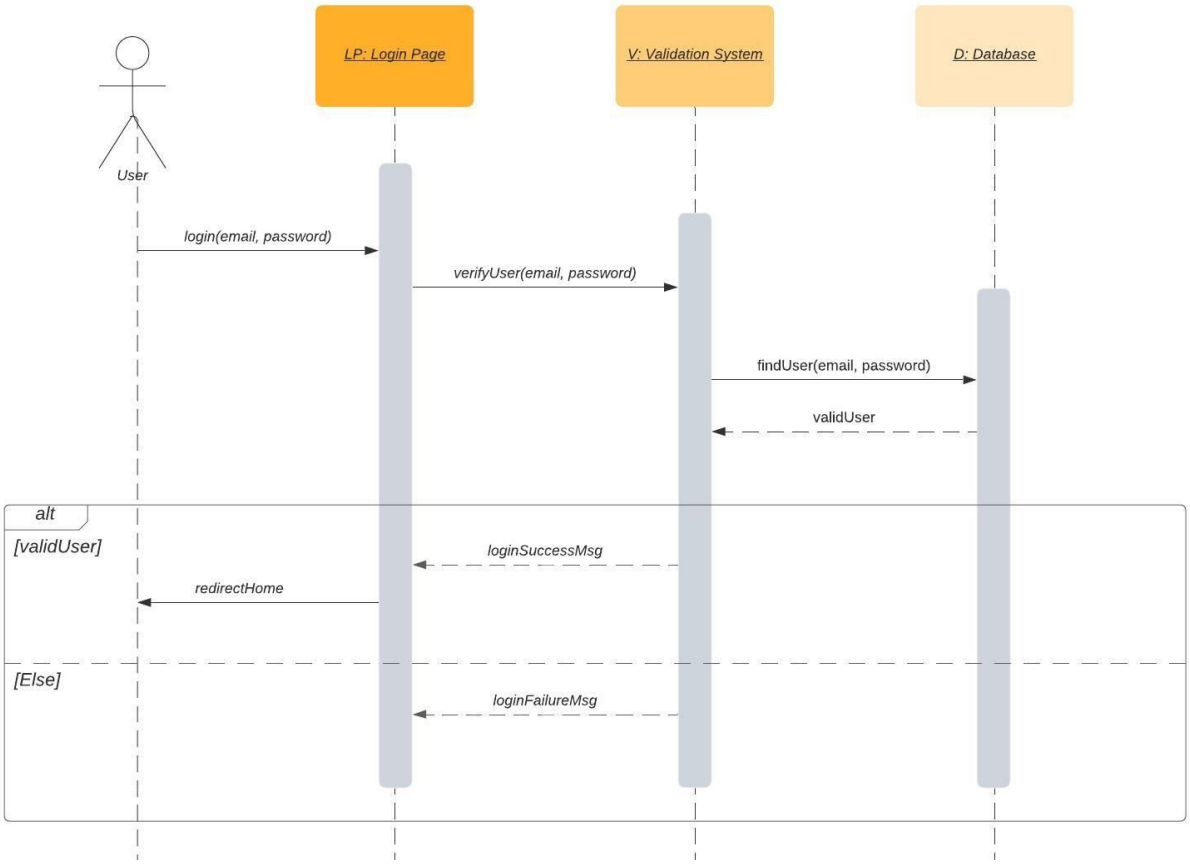
Login



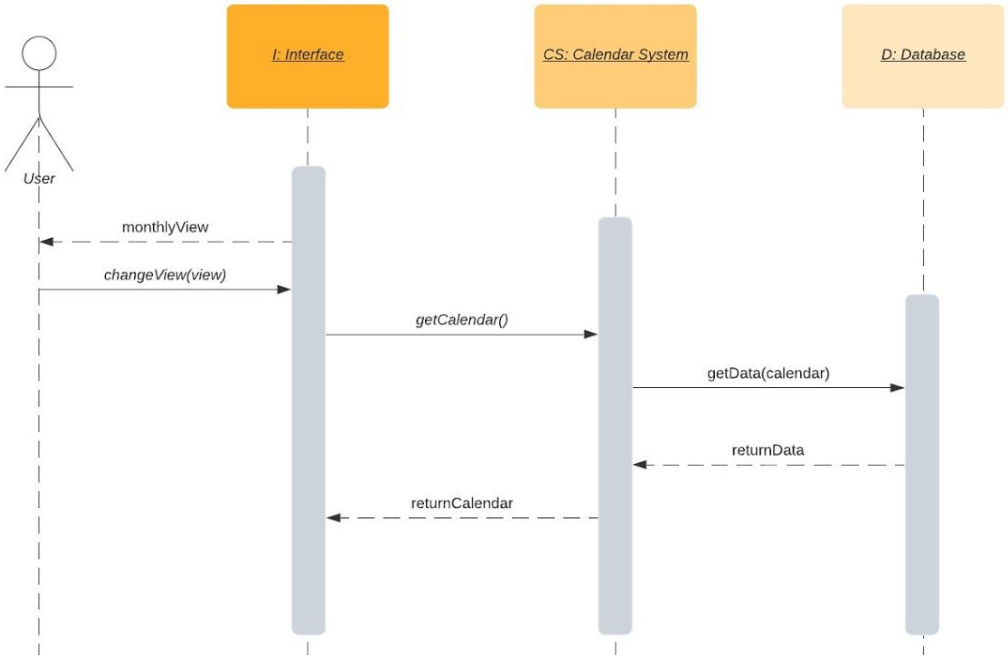

Verify Credentials



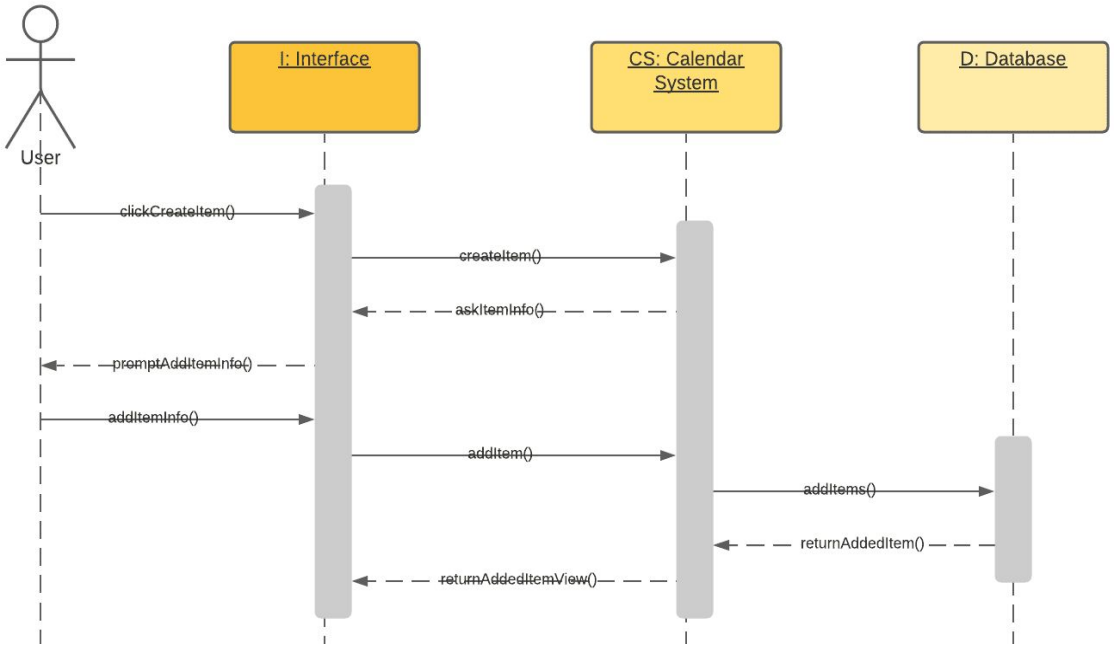
Display Login Error



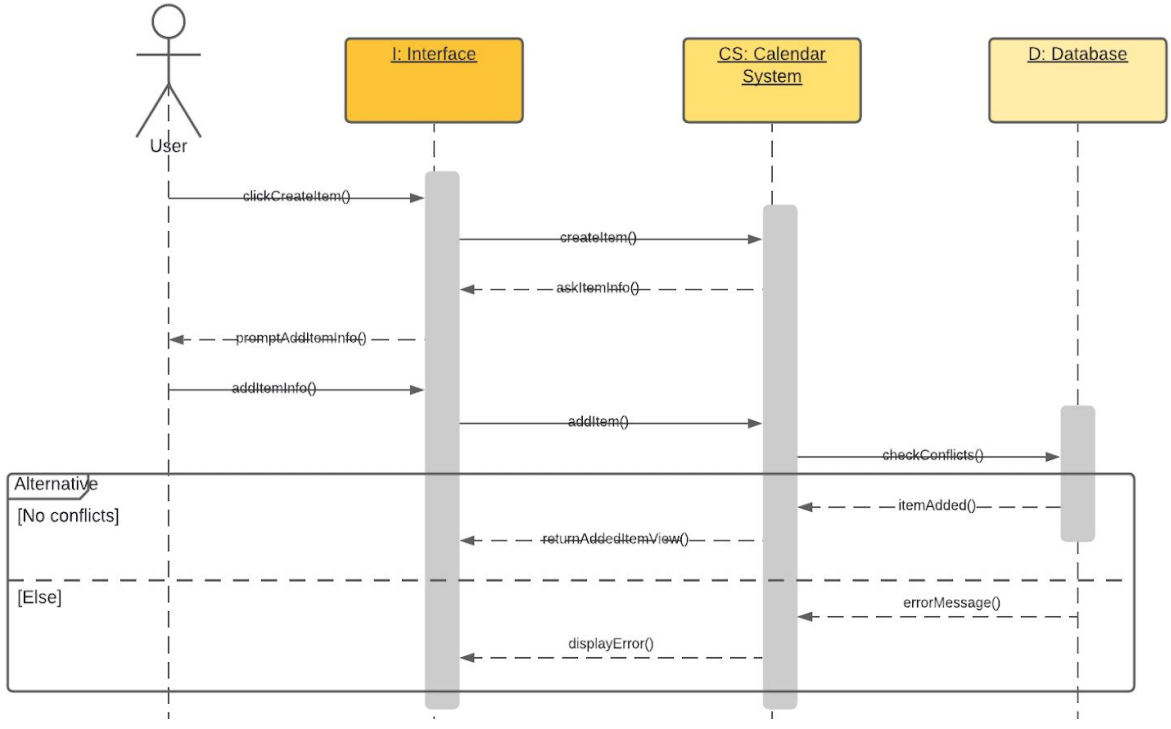

View Calendar



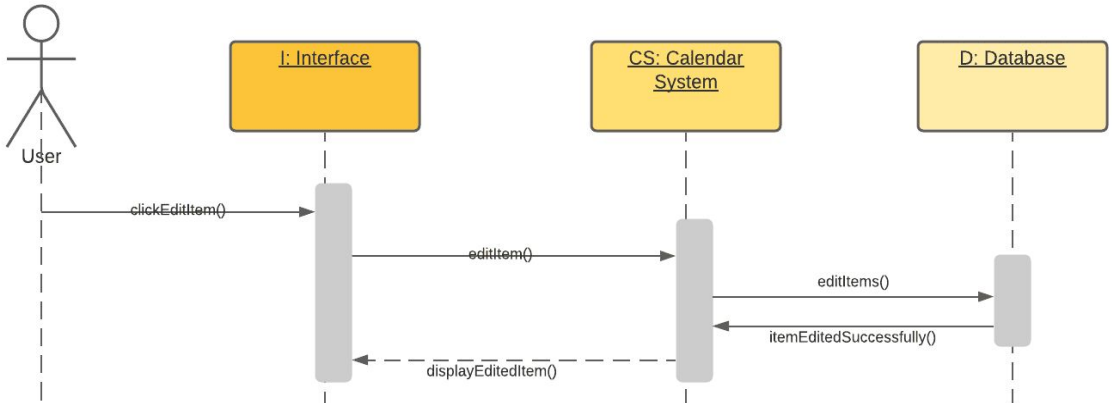
Create Item



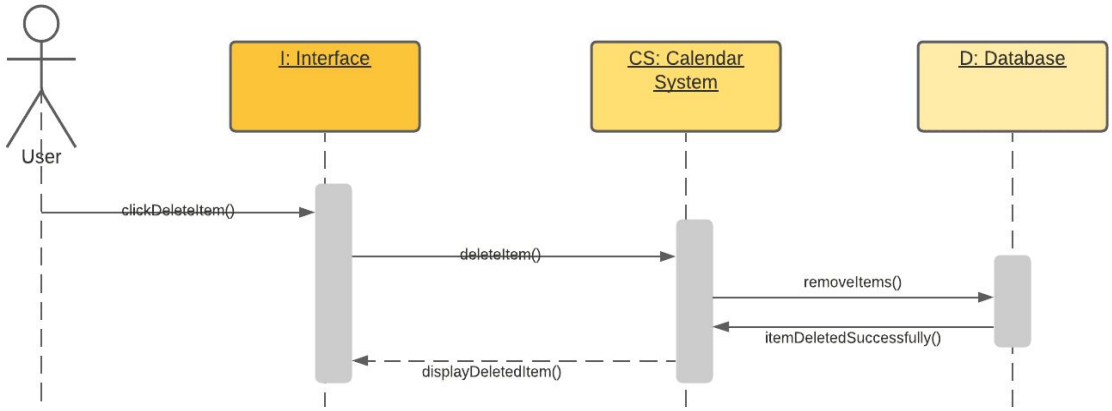

Check Conflicts



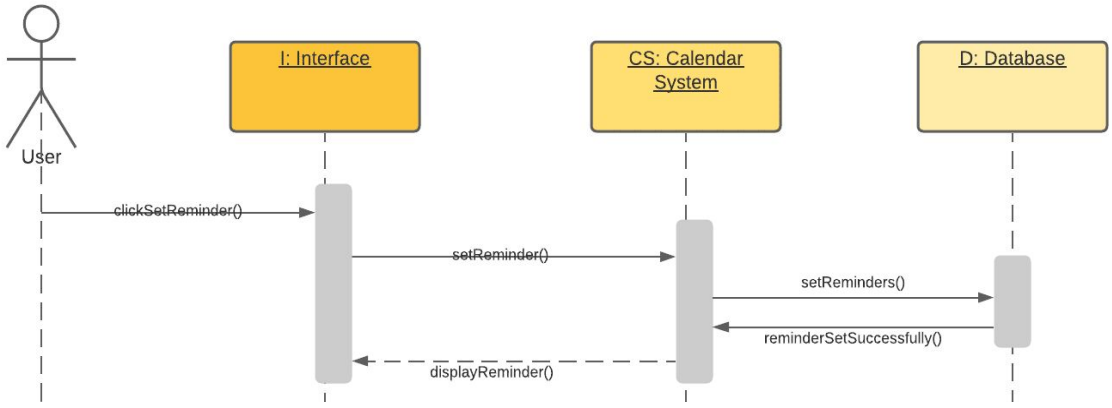
Edit Item



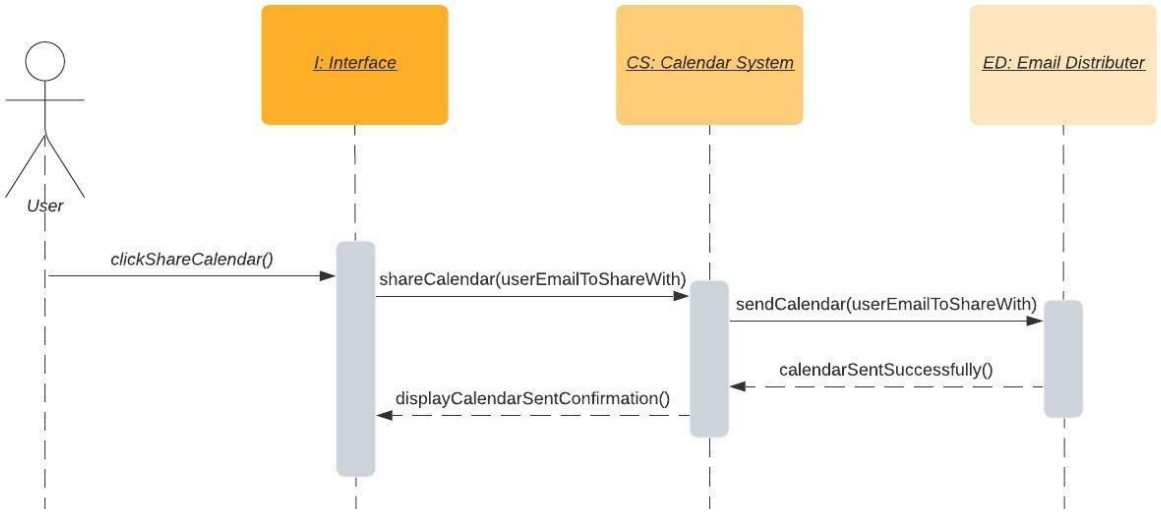

Delete Item



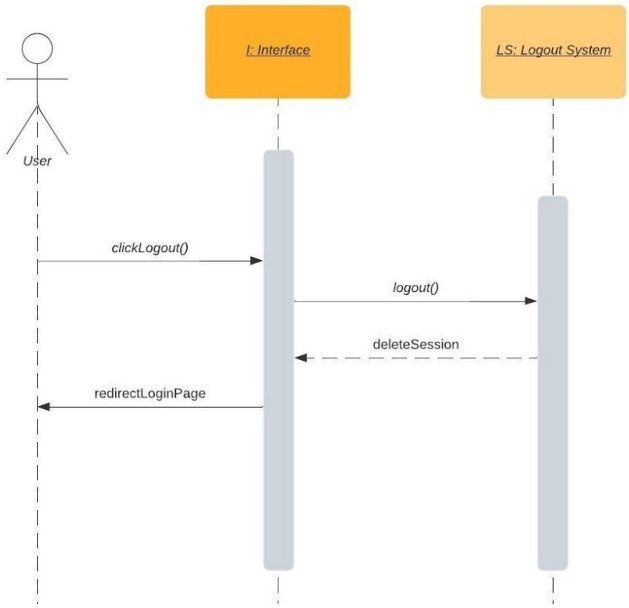

Set Reminder



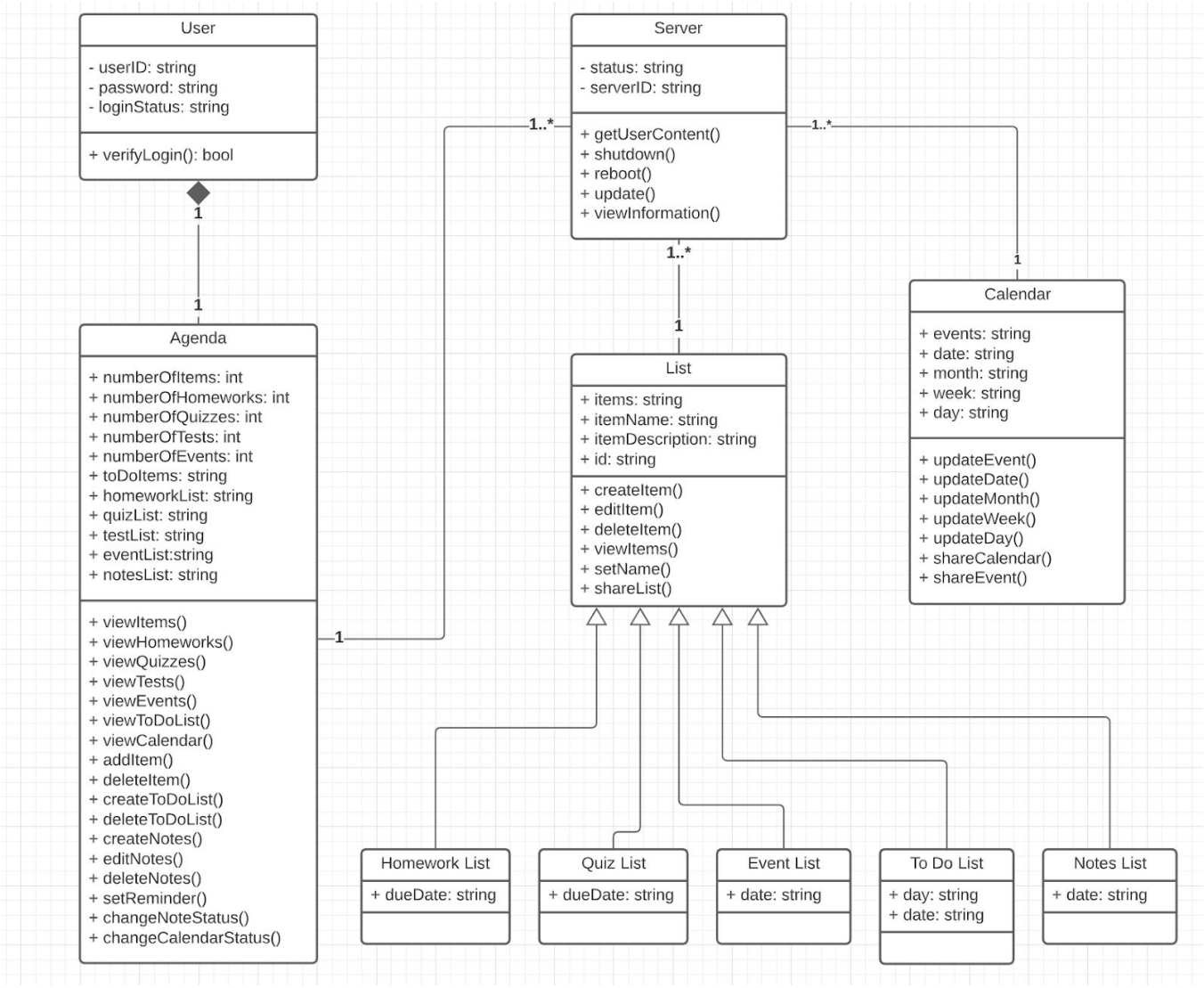
Share Calendar




Logout

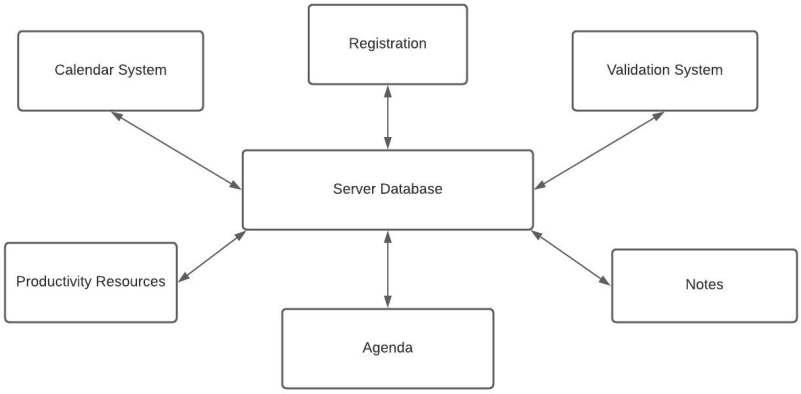


g. Class Diagram




h. Architectural Design

Repository architecture pattern: A repository architecture pattern will be utilized for the development of Procrastify.



3. [5 POINTS] Choose only one of the following two options and specify clearly which is your choice. This will help us post a live presentation schedule. Please understand that we wouldn't know which group will present live, and which group will prerecord instead before the due date of final project deliverable2. So, thank you very much for your patience. We will post a presentation schedule once we have each group's submission.
   **- Option 1: Present live during scheduled class time via BBC (Blackboard collaborate). Each member of the group has to be present and participate in this option. No need to prerecord your presentations if you choose option 1 as your live presentations will be recorded. Still, you should include your (non-recorded) presentation slides into your project deliverable2 submission bundle.**

4. [35 POINTS] Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:
   4.1. [5 POINTS] Project Scheduling. Make an estimation on the schedule of your project. Please provide start date, end date by giving justifications about your estimation. Also provide the details for:
   - Whether weekends will be counted in your schedule or not
   - What is the number of working hours per day for the project

   Start Date- Jan 4, 2021

| Task | Duration (weeks) | Deadline | Endpoint |
|---|---|---|---|
| Design Interface | 2 | Jan 15, 2021 | Application design and UI/UX |
| Design Software | 2 | Jan 15, 2021 | Processes, classes, and tools |
| Design Database | 2 | Jan 15, 2021 | Processes, tools, information security |
| Create Design Specifications | 1/2 | Jan 20, 2021 | Report on specifications |
| Complete Design | 3 | Jan 22, 2021 | Design |
| Complete Prototype | 2 | Feb 5, 2021 | Prototype |

| | | | |
|---|---|---|---|
| Develop System Modules | 6 | Mar 19, 2021 | Interface, software, and database developed |
| Create Development Specifications | 1/2 | Mar 19, 2021 | Report on specifications |
| Complete Development | 6 | Mar 19, 2021 | Developed application |
| Perform System Testing | 2 | Apr 2, 2021 | Testing successes and failures |
| Correct System Errors | 2 | Apr 16, 2021 | Tested application |
| Deploy | 1 | Apr 23, 2021 | Public application on server |

End Date: Apr 23, 2021

- Weekends will not be worked
- 8 hour workdays

Assuming the productivity of the development team is 30 function points per person-week, the following calculations would lead to the conclusion that E = FP/productivity = 187.2 FP / 30 FP per person-weeks = 6.24 weeks or 44 working days. Approximately 44 days will be allocated for each component of the project: design/prototyping, development, and testing/deployment. The project will begin Jan 4, 2021. Weekends will not be counted within the project scheduling, nor will holidays. The estimated project end date will be Apr 23, 2021.

4.2. [15 POINTS] Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Please choose one of the two alternative cost modeling techniques and apply that only:

- Function Point (FP)

The algorithmic estimation selected was the function point method. In this method the steps include:

a. Determine function category count
   - User input: 7, Username, password, event, note, course, to-do, date
   - User output: 7, Calendar, events, notes, to-do, reminders, error notification, success notification
   - User queries: 4, Search, delete, edit, add
   - Data files and relational tables: 6, Events file, notes file, to-do, login, settings, calendar table
   - External interfaces: 3, web application, iOS, and android

b. Determine complexity
   - User Input: Average
   - User Output: Average
   - User Queries: Average
   - Data Files: Average
   - External Interfaces: Average

c. Compute gross function point (GFP)

| | **Function Category** | **Count** | **Complexity** | | | **Count x Complexity** |
|---|---|---|---|---|---|---|
| | | | **Simple** | **Average** | **Complex** | |
| **1** | **Number of user input** | 7 | 3 | 4 | 6 | 28 |
| **2** | **Number of user output** | 7 | 4 | 5 | 7 | 35 |
| **3** | **Number of user queries** | 4 | 3 | 4 | 6 | 16 |
| **4** | **Number of data files and relational tables** | 6 | 7 | 10 | 15 | 60 |
| **5** | **Number of external interfaces** | 3 | 5 | 7 | 10 | 21 |
| | | | | | **GFP** | 160 |

GFP = 160

d. Determine processing complexity (PC)

(1) Does the system require reliable backup and recovery? **3**
(2) Are data communications required? **5**
(3) Are there distributed processing functions? **3**
(4) Is performance critical? **4**
(5) Will the system run in an existing, heavily utilized operational environment? **3**
(6) Does the system require online data entry? **5**
(7) Does the online data entry require the input transaction to be built over multiple screens or operations? **2**
(8) Are the master files updated online? **3**
(9) Are the inputs, outputs, files, or inquiries complex? **3**
(10) Is the internal processing complex? **3**
(11) Is the code designed to be reusable? **4**
(12) Are conversion and installation included in the design? **4**
(13) Is the system designed for multiple installations in different organizations? **5**
(14) Is the application designed to facilitate change and ease of use by the user? **5**

PC = 2x1 + 3x6 + 4x3 + 5x4 = 2+18+12+20 = 52

e. compute processing complexity adjustment (PCA)
PCA = .65 + .01(52) = 1.17
f. compute function point (FP) = GFP x PCA
FP = 160 x 1.17 = 187.2 FP

4.3. [5 POINTS] Estimated cost of hardware products (such as servers, etc.)
Work Machine Budget: $800/per machine
Work Machine Total: $800 x 6 person team = **$4,800**

Amazon Web Services[1] Server Cost:
Number Users = 100,000 users
Average Number of User Activities per day = 5 activities
Total Activities/Log Entries per day = 100,000 x 5 = 500,000
Approximate Log Entry Size = 500 bytes
500 bytes x 500,000 activities = 250 Million bytes of data = 2.5 GB of log data per day(scaled by a factor of 10 for worst case)
Log Data per month: 75 GB
Amazon S3 Data Storage Cost = $0.23/per GB
Total Cloud Storage Cost = $0.23 x 75 GB = **$17.25 per month**

4.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)
Auth0 login authentication - For 10,000 active users: $228 per month
Jira Software Standard Package - $70 per month
Amazon Simple Notification Service - $0.50 per million notifications

4.5. [5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation)
6 Person Development Team

- 3 Software Engineers/Backend Developers
- 2 Front-End Developers
- 1 Database Administrator

4 Month Contract
Salary = $50/hr = $2,000/week = $8,000/month
Total Personnel Cost = $8,000 x 4 months = **$32,000**

5. [10 POINTS] A test plan for your software: Describe the test plan for testing minimum one unit of your software. As an evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as additional document in your zip file submitted.

The software unit that we decided to test is the verification/ login function and we will do this by utilizing JUnit. This unit test will determine if the user attempting to login has an account in the database. If the user is found, then the unit test accepts. If the user is not found, then the unit test fails. For testing purposes, we have created a hashmap that houses current working accounts with the username and password in parallel with one another. The test will then accept the username input and password input and compare them to the data in the hashmap. We created a class named LoginTest.java that verifies the login credentials with JUnit testing and the instance of User class - account object. We have designed seven test cases for login:

1. Correct Username & Password - True
2. Wrong Username Only - False
3. Wrong Password Only - False
4. Wrong Username & Password - False
5. Empty Username Only - False
6. Empty Password Only - False
7. Empty Username & Password - False

6. [10 POINTS] Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make.
Top 3 Competitors-
    1. Google Calendar[2]

- Google Calendar Features:
    - Day/Week/Month/Year/Schedule View
    - Create an event, reminder, or task
    - Give an event a title, time, guests, location, and description
    - Automatically lists holidays
- Procrastify Extra Features:
    - Procrastify will be able to perform the above features as well as-
    - Agenda view to view by upcoming homeworks, quizzes, exams, etc.
    - Organize by event type (homeworks, quizzes, exams)
    - Give category color to each calendar event
    - Can create a to-do list
    - A user will be able to search for all items by date, title, category, and course
    - Note-taking capability to hold notes for courses within folders

2. Any.do calendar[3]
    - Any.do Features:
        - Create an event, reminder, or to-do list
        - Day/Week/Month/Year/Task View
        - Create a to-do list with tasks. To-do list uploads to calendar
        - Add a one-time, recurring, or location based reminder
        - Add tags/color category to events
    - Procrastify Extra Features:
        - Procrastify will be able to perform the above features as well as-
        - Procrastify users will have the ability to share a calendar with others
        - Reminder settings that don't disrupt workflow
        - Agenda view will have scrolling abilities to compare days side by side
        - Note-taking capability to hold notes for courses within folders
3. Outlook Calendar[4]
    - Outlook Calendar Features:
        - Daily/Weekly/Monthly View
        - Categorize events by course
        - Give an event a title, time, guests, location, and description
        - Set a reminder before an event
    - Procrastify Extra Features:
        - Procrastify will be able to perform the above features as well as-
        - Yearly View added
        - Procrastify will be able to organize by event types (homeworks, quizzes, exams), enabling the user to view events by importance levels
        - Allow calendar to remain public or private within share network
        - Note-taking capability to hold notes for courses within folders

- Can give category color to each calendar event
- Can create a to-do list
- Can create reminders. Multi-platform reminder settings
- Automatically lists holidays

7. [10 POINTS] Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

    We have designed a well thought out and useful app to organize and display events such as homework, quizzes and exams. The focus on school events sets our design apart from similar products. Through deliverables 1 and 2 we have provided our design criteria and requirements and have brought this idea to fruition. Besides actually creating the app, we have satisfied all expected requirements for this project.

    Our main changes stem from the feedback we received on our Final Project Proposal. The feedback stated to determine how our product differed from already existing products. While our focus on school related events already sets our product apart, we decided to add more flexibility around creation, organization, and displaying events to further set our product apart. We added the ability to organize by event type (as mentioned in the question above), the ability to courses as an identifier for sorting and planning, and we added the ability to share schedules. The ability to create and compile notes utilizes our application as a single source for schoolwork in terms of class material and class scheduling. Note-taking is a significant feature not present in scheduling apps. Specifically, the ability to share schedules and agendas was added to provide more flexibility and options for our users as they might wish to have a copy of their generated agenda outside of our app. Users have complete control over the privacy status of events and calendars within their share network. Deviation was considered for the share feature in response to privacy concerns. However, it was included due to the market value and presence of vast user options. Customization is key in this application, therefore allowing the users to have many options in creating an event, providing custom color categories, and sharing options were greatly considered.

    While small concepts were added, there was very little deviation from the app's original core idea. Each change enhanced the original idea and worked with the original capabilities to provide a better product.

8. [5 POINTS] References: Please include properly cited references in IEEE paper referencing format. Please review the IEEE referencing format document at the URL: https://ieeedataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf).
    It means that your references should be numbered, and these numbers properly cited

in your project report.
Also include:

9. [10 POINTS] Non-recorded (no-voice) presentation slides. No min/max number of slides enforced. Please make sure that you can complete presentation within 15 (fifteen) minutes.
   Following template could be a good start to prepare your presentations. As each project topic is different, a variety in presentation style is expected and welcome.
   - Title of your project together with participants
   - Objective of the project designed
   - Cost estimation
   - Project timeline (timeline of the project designed, NOT the time you've spent on it)
   - Functional and non-functional requirements. If too long, select representative items.
   - Use case diagram
   - Sequence diagram for a selected representative operation of the project.
      - Class diagram
      - Architectural design
      - Model-View-Controller (MVC) pattern (similar to Figure 6.6)
      - Layered architecture pattern (similar to Figure 6.9)
      - Repository architecture pattern (similar to Figure 6.11)
      - Client-server architecture pattern (similar to Figure 6.13)
      - Pipe and filter architecture pattern (similar to Figure 6.15)
   - Preferably a demo of user interface design that shows screen to screen transitions though no full functionality is required.
   - OPTIONAL: IF implemented the project, a demo of your implementation.
   **https://docs.google.com/presentation/d/1MgHai75Qdiv_e5uJBjGFYxi0NgOSuKisrOQtU-QJhg0/edit?usp=sharing**

10. ~~OPTIONAL PART. Your program code (if fully implemented the project, not required otherwise). Please note that implementation is not required for the final project. Groups are welcome to implement their work, if they choose to do so. [This part may qualify for extra credit, if you implement and submit the implementation code together with your project. The extra credit will be determined based on the quality of your implementation]~~
11. [5 POINTS] GitHub requirement:
    Make sure at least one member of your group commits everything for project deliverable 2 to your GitHub repository, i.e.
    - Your final project deliverable2 report

- Unit test code for a sample unit of your project
- Implementation code (if you have implemented your project)
- Non-recorded (no voice) presentation slides

References

[1] “Amazon S3,” aws.amazon.com. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 07-Nov-2020].

[2] “Google Calendar training and help,” *Google Workspace Learning Center*. [Online]. Available: https://support.google.com/a/users/answer/9247501?hl=en. [Accessed: 06-Nov-2020].

[3] S. S. Moritz, “The Calendar View,” *Any.do Help Center*. [Online]. Available: https://support.any.do/calendar-view/. [Accessed: 06-Nov-2020].

[4] “Introduction to the Outlook Calendar,” *Outlook*. [Online]. Available: https://support.microsoft.com/en-us/office/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8. [Accessed: 06-Nov-2020].