CS/SE/CE 3354.004 Software Engineering

Deliverables #2 Report

Orbit

The Calendar/Lifestyle App for College Students

Github Repository:

https://github.com/LuisCGutierrez/3354-orbit/tree/main

Delegated Tasks by Member for Deliverables #2:

- **Luis Gutierrez**
  - Create outline for presentation
  - Write conclusion section of report
- **Kevin Manjit**
  - Create refined and finished presentation
  - Committed unit test code, presentation slides, and report to GitHub
  - Organized recording of presentation
  - Verified report for submission
- **Cooper Brice**
  - Assisted effort and cost calculations
  - Assisted project scheduling
  - Formatted IEEE citations
- **Joseph Park**
  - Assisted in comparison with Outlook Calendar and Google Calendar
  - Calculated Estimated Effort and Cost with FP and Person Weeks
  - Video Splicing
- **Donald Berthelsen**
  - Performed comparison with Outlook Calendar and Google Calendar
- **Carlos Vazquez**
  - Calculated Project Scheduling and Work hours
  - Calculated Cost, Effort, Pricing estimation
- **Skyler Lee**

- Wrote and verified test code for event unit

“Everything Req. and Submitted in Deliverable #1”

Included in the bottom of the report.

Calculated items based on our designed Project if implemented:

a. **Project Scheduling:** Make an estimation of the schedule of your project.
   i. Please provide a start date, end date by giving justifications about your estimation.

1. Requirements Gathering & Planning (1-2 days)
   - Defining core features, identifying compatible platforms
2. UI/UX Design (1-2 days)
   - Mock designs for main screens (i.e. calendar, event, and setting views)
   - User experience flow and accessibility considerations
   - Design approval and iterations
3. Backend Development (2-4 days)
   - Database Schema (events, users, notifications)
   - API (Application Programming Interface) development for CRUD operations (Create, Read, Update, Delete)
4. Frontend Development (2-4 days)
   - Implement calendar view (monthly, weekly, daily)
   - Event creation and editing forms
   - Notification and reminder UI
5. Testing & QA (1-2 days)
   - Unit and integration tests
   - Usability testing
   - Bug fixing and performance optimization
6. Deployment & Documentation (1 day)
   - Package and deploy application
   - Create user guide and technical documentation for usage

   ii. Also provide the details for:
      1. Whether weekends will be counted in your schedule or not

- No, the team's main business days are Monday-Friday.

2. What is the number of working hours per day for the project?
    - 8 hours per day for the project, following agile programming, with 6 hours focused on development and 2 hours for meetings, reviews, and planning
    - This assumes that our team does not accept overtime

b. **Cost, Effort, and Pricing Estimation**
    i. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project.
    ii. Please choose one of the two alternative cost modeling techniques and apply that only:
        1. Function Point (FP)
        2. Application composition

Our team will be using the **Function Point (FP)** cost estimation method to estimate the price of our project. The FP method estimates effort based on the number of function points, representing user-visible functionality in the following steps:

1. Determine function category count
    - External Inputs (EI): Event creation, reminders, recurring events
    - External Outputs (EO): Notifications, calendar views
    - External Inqueries (EQ): Search events, filter by date
    - Internal Logical Files (ILF): Event database, user profiles.
    - External Interface Files (EIF): Integration with Google Calendar.

2. Determine Complexity

Component	Count	Low	Average	High
EI	10	3	4	6
EO	6	4	5	7

EQ	4	3	4	6
ILF	2	7	10	15
EIF	1	5	7	10

An average complexity is assumed for all component categories.

3. Compute gross function point **(GFP)**:
    - EI: 10 inputs (event creation, edit, reminders) → Average (4) →
        - $10 \times 4 = 40$
    - EO: 6 outputs (notifications, calendar views) → Average (5) →
        - $6 \times 5 = 30$
    - EQ: 4 inquiries (search/filter) → Average (4) →
        - $4 \times 4 = 16$
    - ILF: 2 internal files (events, users) → Average (10) →
        - $2 \times 10 = 20$
    - EIF: 1 external interface (Google Calendar) → Average (7) →
        - $1 \times 7 = 7$

    GFP = 40 + 30 + 16 + 20 + 7 = **113 FP**

4. Determine processing complexity **(PC)**

    Data Communications (1)
    - The app is standalone, minimal external communication except optional calendar sync.

    Distributed Processing (0)
    - No distributed architecture; runs locally or on a single backend.

    Performance (4)
    - Needs fast response for calendar rendering and event operations, especially with large schedules.

    Heavily Used Configuration (2)
    - Moderate usage expected by students, but not enterprise-level heavy load.

    Transaction Rate (2)

- Event creation and updates occur regularly but not at high transaction volumes.

Online Data Entry (5)
- Core functionality involves entering events, reminders, and recurring schedules.

End-User Efficiency (4)
- Must be intuitive and quick for students to manage schedules efficiently.

Complex Processing (2)
- Some complexity in handling recurring events and reminders, but not heavy computation.

Reusability (3)
- Code should be modular for future enhancements or reuse in other apps.

Installation Ease (5)
- Needs simple installation for standalone deployment (one-click or minimal setup).

Operational Ease (5)
- Students expect easy operation without technical knowledge.

Multiple Sites (0)
- Single deployment scenario, not multiple geographic sites.

Facilitate Change (4)
- Should allow easy updates for academic calendar changes or new features.

Security (5)
- High importance for protecting student data and event details.

Sum of Scores = 42

5. Compute processing complexity adjustment **(PCA)**

   PCA = 0.65 + 0.01(42) = **1.07**

6. Compute function point **(FP)**:

Adjusted FP = GFP * PCA
Adjusted FP = 113 * 1.07 ≈ **121 FP**

(Can maybe consider rounding up to 140 FP for contingency and extra features)

7. Compute estimated effort:

   Effort (person-weeks) = FP / Productivity

   Assuming average Productivity of 0.25 FP / hr

   121 FP / (0.25 FP/hr) = **484 hrs**

   Effort (person-weeks) = 484 / 40 ≈ 12 person weeks

   12 person-weeks / 6 people = **2 weeks**

8. Convert Effort to Cost:
   Cost = Effort × Hourly Rate ($/hour for developers).

   <Calculated in section e. for designated personnel>

c. Estimated cost of hardware products (such as servers, etc.)

   No cost in hardware. We are deploying a software application.

d. Estimated cost of software products (such as licensed software, etc.)

   No cost in licensing, since the Python Software Foundation License (PSFL) allows developers to use, modify, and distribute software without restrictive obligations

e. Estimated cost of personnel (number of people to code the final product, training cost after installation)

Allocating hours across roles…

- lead developer ($60/hr)
  - 20% of work → 96 hours
  - 96 * 60 = $5,760
- 2 developers (hours combined) ($50/hr)
  - 40% of work → 192 hours
  - 192 * 50 = $9,600
- code researcher ($45/hr)
  - 10% of work → 48 hours
  - 48 * 45 = $2,160
- designer ($45/hr)
  - 10% of work → 48 hours
  - 48 * 45 = $2,160
- product manager ($55/hr)
  - 20% of work → 96 hours
  - 96 * 55 = $5,280

Total Personnel Cost = $5,760 + $9,600 + $2,160 + $2,160 + 5,280
= **$24,960**

Since personnel is the only cost, the project cost estimate is: **$24,960**

Test Plan for Software:

a. Describe the test plan for testing a minimum of one unit of your software.

The method getEventDuration() in Event returns the total number of minutes between two LocalDateTime values locally stored within an instance of an Event. The goal is to ensure that it correctly handles normal, zero-length, and invalid-length inputs of an Event's start date time and end date time.

b. As 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.

The sample is provided in the GitHub Repository.

c. Clearly define what test case(s) are provided for testing purposes and what results are obtained. (Ch 8)

Test Case 1: testNormalDuration()
This test ensures the method correctly calculates the duration of an event in minutes under normal conditions.

Input values: start = 1:00, end = 3:00
Test response: Valid input
Expected output: 120 minutes
Result: 120 minutes

Test Case 2: testZeroDuration()
This boundary test ensures the method correctly calculates the duration of an event in minutes under the smallest valid time interval.

Input values: start = 1:00, end = 1:00
Test response: Boundary case (start and end time are equal)
Expected output: 0 minutes
Result: 0 minutes

Test Case 3: testEndBeforeStart()
This test ensures the method correctly identifies invalid inputs to prevent system crashes, unwanted interactions, or data corruption.

Input values: start = 7:00, end = 2:00
Test response: Invalid input, constructor throws IllegalArgumentException
Expected output: IllegalArgumentException thrown
Result: IllegalArgumentException thrown

Test Case 4: testNullDuration()
This test ensures the method correctly identifies invalid inputs to prevent system crashes, unwanted interactions, or data corruption.

Input values: start = null, end = null
Test response: Invalid input (null check by constructor), constructor throws IllegalArgumentException
Expected output: IllegalArgumentException thrown
Result: IllegalArgumentException thrown

Comparison of your work with similar designs:

a. This step requires a thorough search in the field of your project domain. Please cite any references you make.

Feature	Orbit	Outlook Calendar [1]	Google Calendar [2]
Daily/weekly/montly view	✓	✓	✓
Create/modify/delete events	✓	✓	✓
View events	✓	✓	✓
Set event categories	✓	Some	X
Set event colors	✓	X	✓
Notify user of events	✓	✓	✓
Display holidays/weekends	✓	✓	✓
Zoom in/out	✓	X	X
App size	< 10 MB	428 MB [3]	254.6 MB [4]
Self contained	✓	X	X
Backups	Local	Online	Online

Conclusion:

a. Please make an evaluation of your work.
    i. Describe any changes that you needed to make (if any), if things have deviated from what you had originally planned.

    Overall, we have mostly stayed true to our original vision of the app. Our software idea's simplicity has led us to avoiding potential scope creep and maintaining a tight focus on select features. The only significant change in our design was the non-functional language requirement of creating the program in C. We decided to build the program with Java.

    ii. Give justification for such changes.
    Our test code highlighted the powerful testing in Java using Junit. This gave us a more industry-standard method of testing with a more secure language as well. Our team also will be able to design more efficiently with our higher knowledge of Java over C.

References:

a. Please include properly cited references in IEEE paper referencing format. (You may see a referencing example in the sample IEEE paper in URL: https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf
b. It means that your references should be numbered, and these numbers be properly cited in your project report.

1.5. List if any

[1] Microsoft, "Introduction to the Outlook Calendar," *Microsoft Support*. [Online]. Available: https://support.microsoft.com/en-us/office/introduction-to-the-outlook-calendar-d94c5203-77c7-48ec-90a5-2e2bc10bd6f8. [Accessed: Nov. 22, 2025].

[2] Google, “Google Calendar: Online Calendars for Business | Google Workspace,” *Google Workspace*. [Online]. Available: https://workspace.google.com/products/calendar/. [Accessed: Nov. 22, 2025].

[3] Microsoft, “Microsoft Outlook,” *App Store*. [Online]. Available: https://apps.apple.com/us/app/microsoft-outlook/id951937596. [Accessed: Nov. 22, 2025].

[4] Google, “Google Calendar: Get Organized,” *App Store*. [Online]. Available: https://apps.apple.com/us/app/google-calendar-get-organized/id909319292. [Accessed: Nov. 22, 2025].

CS/SE/CE 3354.004 Software Engineering

Deliverables #1 Report

Orbit

The Calendar/Lifestyle App for College Students

Github Repository:

https://github.com/LuisCGutierrez/3354-orbit/tree/main

Delegated Tasks by Member for Deliverables #1:

- **Luis Gutierrez**
  - I will create a GitHub Repository named "3354-orbit".
  - I will add every member and the TA to this repository.
  - I will add a link to the Github Repository in the report.
  - I will action the feedback from our proposal (see end of this report).
- **Kevin Manjit**
  - I will formulate the content for the README.md file and commit it.
  - I will write out delegated tasks for Deliverables #1.
  - I will write out the assumptions we are making for this app (government-wise and user-wise).
  - I will organize the final report and repository for this milestone submission.
- **Cooper Brice**
  - I will commit the project scope file identical to the requirements in the "Project Topic Ideas" document provided as the calendar app was a given idea.
  - I will make and add the Use Case Diagram.
- **Joseph Park**
  - I will write out our plan to action the feedback from our proposal.
  - I will pick out the most appropriate software process model (V Model) and write out why it works for this project.

- **Donald Berthelsen**
    - I will write out all functional and non-functional requirements.
- **Carlos Vazquez**
    - I will make and add the Sequence Diagram.
- **Skyler Lee**
    - I will make and add the Class Diagram.
    - I will make and add the Architectural Design and Diagram (MVC).

Assumptions:

- We are assuming that a downloadable local app (single-user) is more beneficial for students for the sake of productivity and isolation.
    - In other words, Orbit operates entirely offline.
- In the US, user data has to be protected since students may enter sensitive information.
    - This is protected as the app is not connected to any servers, cloud system, network, etc.
    - All data is local.
- While students will benefit from a mobile app, we are assuming that a desktop app will be much more beneficial for first launch. In the business context, mobile support will become available once the desktop app is established. However, it is out of scope for this semester project and we will only focus on the launch of the desktop app.
- We are assuming our ICP are 18-25 year olds, so we will focus on minimalism and structure as the main pillars of our app.

Feedback Response:

Feedback:

“Good Start. Clearer details of the proposed implementation are expected either in the proposed implementation section or in the task delegation.

Include tasks with implementation specifics in the report for deliverable #1.

The task section could be more detailed. Include tasks with respect to requirements gathering, design and testing. (for eg- who will create use case diagrams, sequence diagrams etc)”

Response:
We will clear up the details of the implementation through the task delegation section in our Deliverables #1 Report. To be more specific, we will create tasks that specify the requirements that need to be created and followed and the diagrams that must be modeled. We have identified more specific tasks using the functional requirements, creating a clearer vision for what roles need to be fulfilled and what action steps can be taken to reach those goals. This feedback is actioned at the end of this document.

Software Process Model:
The Software Process Model employed in this project is the **V Model**. This model was chosen for its simplicity and efficiency. As the calendar app isn’t a large scale project, the software process model doesn’t have to be as complex. Also the inflexible nature of the V Model/Waterfall Model is not a concern in this case, as the requirements are fairly stable and unlikely to change in the semester that we are developing the software. The V Model was chosen over the Waterfall Model because it is more specific on the testing that happens after code is generated. To ensure the program meets all of the requirements, it will have to go through a series of tests at the end to make sure that it is suitable.

Software Requirements:
Functional Requirements -

1. The user shall be able to view events. Events shall be viewable individually, or grouped by day, week, or month.

Function	View events
Description	Displays all events to the user that fall within the specified time frame. Possible timeframes are 1 day, 1 week, or 1 month. Can alternatively view events individually
Inputs	Day the requested time window beings StartTime, the size of the requested time window Window (Day, Week, Month) or the individual event RequestedEvent, day the week starts on WeekStart
Source	Day and Window from CalendarController, WeekStart from Regional Settings Service
Outputs	Displayed event(s) in the specified window
Destination	View
Action	If RequestedEvent is specified then display that event in an individual event view. Otherwise display a single day view if Window is Day, a week view if Window is Week, or a month view if Window is Month, and StartTime must be a calendar day. Days begin at midnight, weeks begin on WeekStart, and months begin on the 1st. WeekStart defaults to Sunday if it can not be found by Regional Settings Service.
Requires	StartTime is a valid value
Precondition	None
Postcondition	The events are displayed to the user
Side effects	None

2. The user shall be able to create events. Created events shall not overlap with existing events.

Function	Create events
Description	The user shall be able to create events to be stored on the calender

Inputs	Event name Title, event start time StartTime, event end time EndTime, event location Location, if the event is recurring IsRecurring, event category Category
Source	User
Outputs	Created event Event, return code Result, event notice EventNotice
Destination	Event to Calendar, Result to User, EventNotice to NotificationController
Action	Create a new event and store it on the calendar. All events must at least have a Tile, StartTime, and EndTime. If IsRecurring is not specified, it defaults to false. If Category is not specified, it defaults to null. Events can not be created to overlap in time with an existing event. Result is OK if the operation succeeded, or ERROR followed by a description of what went wrong if the operation failed. If event creation succeeds and has a start time in the future, create a notification using Event and send it to NotificationController as EventNotice.
Requires	Title, StartTime, and EndTime are all not null
Precondition	Calendar has enough memory to store an Event, NotificationController has enough memory to store a Notification
Postcondition	Event is created and stored if valid
Side effects	None

3. The user shall be able to edit events. All event parameters available to the user during event creation shall be accessible during event editing. Edited events shall not be made to overlap with existing events.

Function	Edit events
Description	The user shall be able to edit events on the calendar. Events must remain valid after being edited

Inputs	New event name Title, new event start time StartTime, new event end time EndTime, new event location Location, if the new event is recurring IsRecurring, new event category Category, event to be edited OldEvent
Source	New event from User, OldEvent from Calendar
Outputs	Edited event NewEvent, return code Result, event notice EventNotice
Destination	Event to Calendar, Result to User, EventNotice to NotificationController
Action	Edit an existing event from the calendar and store it with updated information. If any of the parameters are not specified, they default to the values of OldEvent. Title can not be empty. Events can not be edited to overlap in time with another existing event. Result is OK if the operation succeeded, or ERROR followed by a description of what went wrong if the operation failed. If event editing succeeds and has a start time in the future, create a notification using Event as EventNotice. Remove the Notification for OldEvent if it exists from NotificationController, and send EventNotice to NotificationController.
Requires	There is at least 1 event in the calendar
Precondition	NotificationController has enough memory to store a Notification
Postcondition	Event is edited and stored if valid, and the related notification is updated
Side effects	None

4. The user shall be able to delete events.

Function	Delete events
Description	The user shall be able to delete events on the calendar

Inputs	Event to be deleted Event, list of notifications Notifications
Source	Event from User, Notifications from NotificationController
Outputs	Return code Result
Destination	User
Action	Delete the specified Event from the calendar. Fails if Event does not exist on the calendar. If the delete succeeds, search Notifications for the event's related notification and delete it if it exists.
Requires	There is at least 1 event in the calendar
Precondition	Event exists on the calendar
Postcondition	Event is deleted from the calendar
Side effects	None

5. The user shall receive a notification when the start time for an event is near.

Function	Notify user of events
Description	The user shall be notified when an event has a start time that will occur within 10 minutes.
Inputs	List of notifications Notifications
Source	NotificationController
Outputs	Notification for the event that will occur soon Notification
Destination	Notification to View
Action	Periodically scan Notifications once per minute to find any notifications with a send time in the past. For each notification this applies to, send the notification to view as Notification and remove it from Notifications. This process shall operate as background activity.

Requires	There is at least 1 event with a start time in the future
Precondition	Notifications is not empty
Postcondition	Notifications decreases in size when a notification is sent
Side effects	None

6. Holidays and weekends shall be indicated in the calendar view. Holidays shall differ depending on the region of the user.

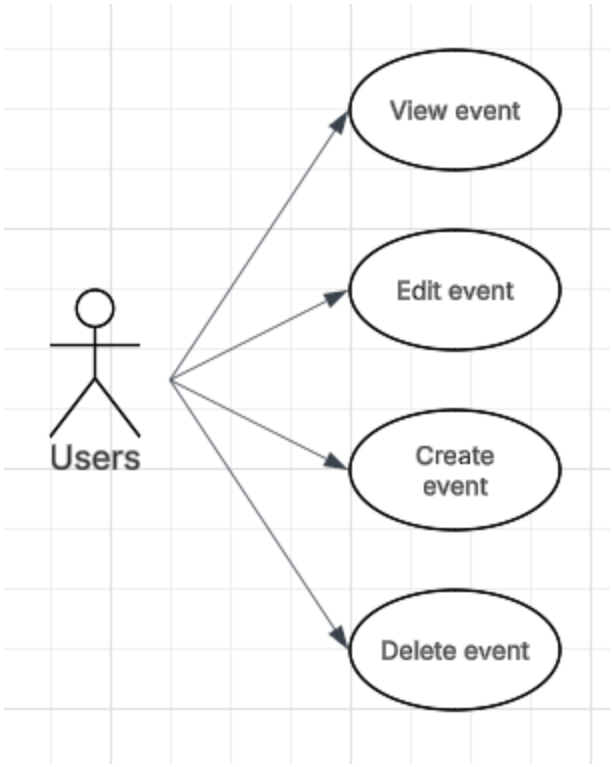
Function	Display holidays and weekends
Description	Holidays and weekends shall be included on the calendar. Holidays differ depending on the region of the user
Inputs	Calendar OldCalendar, regional holidays LocalHolidays
Source	OldCalendar from Calendar, LocalHolidays from Regional Settings Service
Outputs	Updated calendar NewCalendar
Destination	NewCalendar to Calendar
Action	Add weekends and local holidays from LocalHolidays to OldCalendar. Holidays are similar to events but can not be edited and last for the duration of the day they occur. Holidays can overlap in time with Events. Holidays are only visible when viewing the Calendar by Week or Month. Return the OldCalendar with holidays added to Calendar as NewCalendar.
Requires	Regional Settings Service can determine the location of the user
Precondition	Calendar does not have holidays and weekends listed
Postcondition	Calendar has holidays and weekends listed
Side effects	None

Non-Functional Requirements -

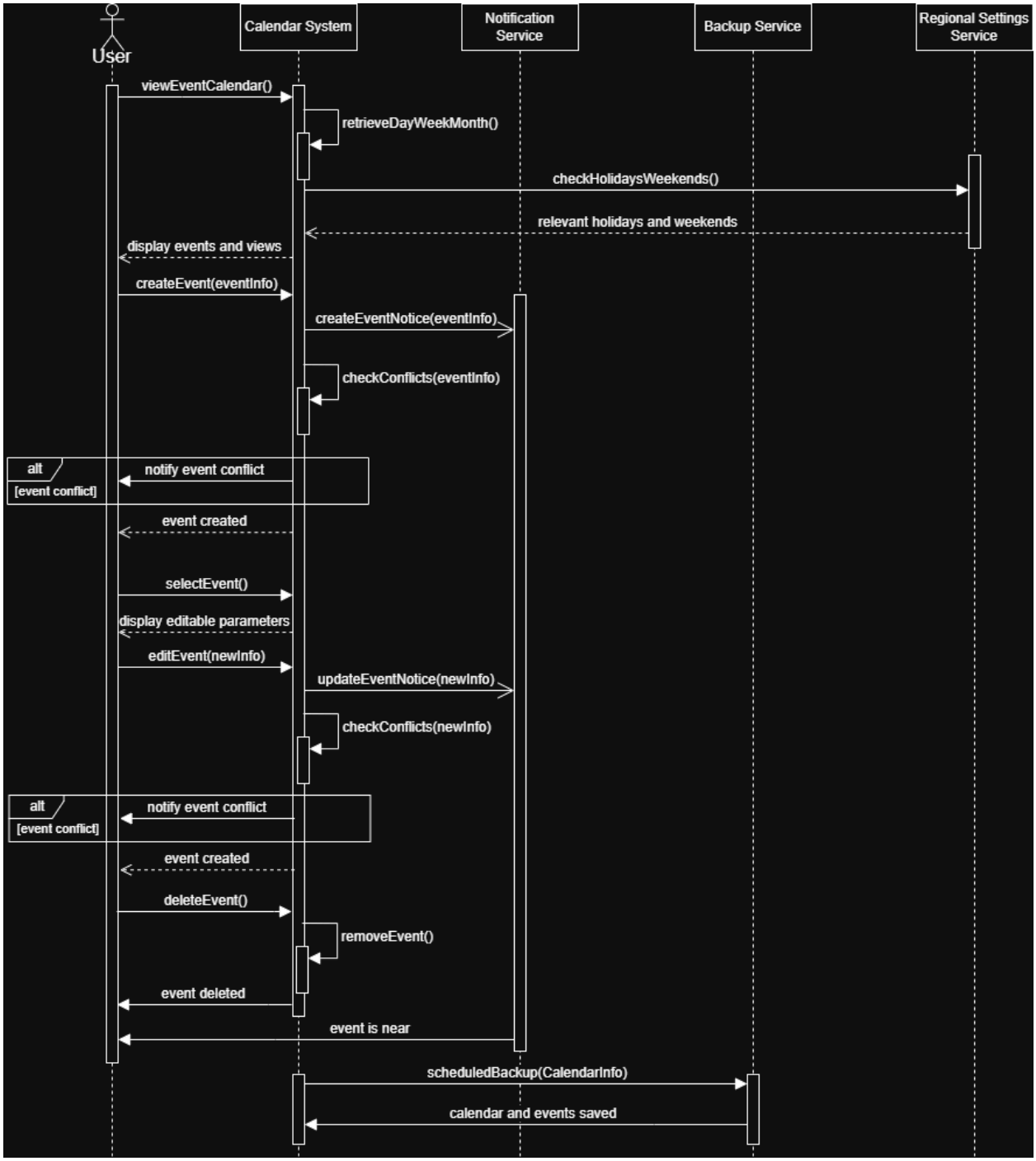
1. (Usability Requirements) Users shall be able to effectively use the application without training.
2. (Performance Requirements) The application shall respond to user inputs in under 0.2 seconds.
3. (Space Requirements) The application shall be under 10 megabytes when containing no events.
4. (Dependability Requirements) The application shall have a mean time to failure of at least 8 hours.
5. (Security Requirements) Event and category names shall be in plain text.
6. (Environmental Requirements) The application shall be available on Windows and Linux systems.
7. (Operational Requirements) The system shall have a failure rate of at most 0.1% when processing user inputs.
8. (Development Requirements) The application shall be written in C.
9. (Regulatory Requirements) The application shall not interact with any other application running in the user's environment.
10. (Ethical Requirements) Displayed holidays shall be appropriate for the user's culture.
11. (Accounting Requirements) Events shall be saved automatically during recurring daily, weekly, and monthly backups.
12. (Safety/Security Requirements) Event notifications shall be the only background activity performed by the application.

Use Case Diagram:

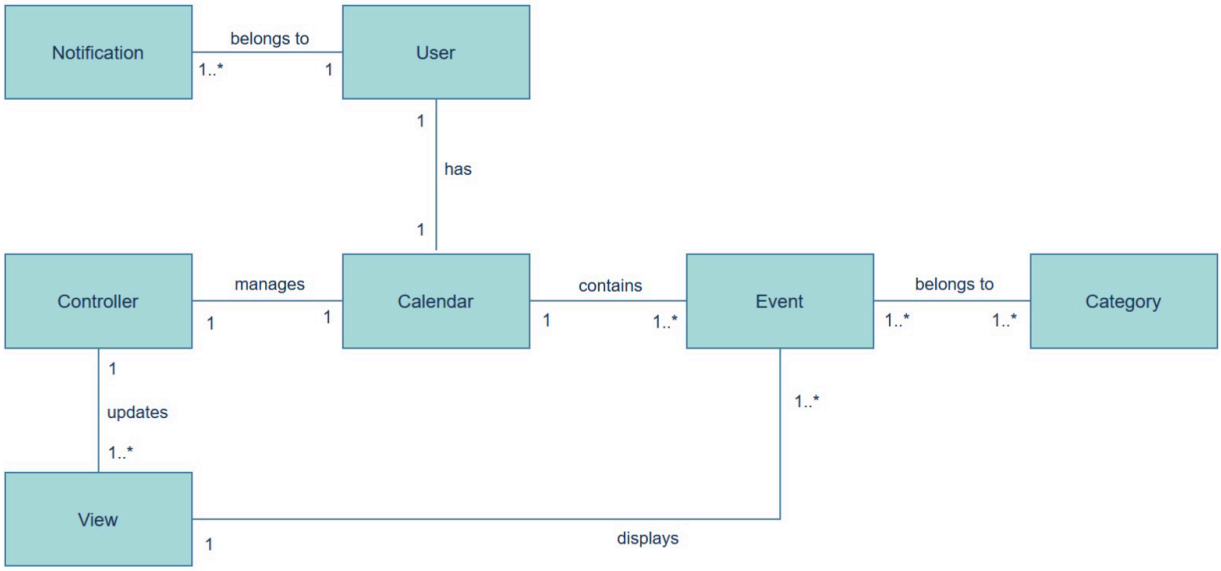
https://lucid.app/lucidchart/1902533d-c823-45a2-ba76-27b13332b21a/edit?viewport_loc=-796%2C-25%2C2217%2C1087%2C0_0&invitationId=inv_54f30cf9-1e65-4eff-ab4d-71e41503068d

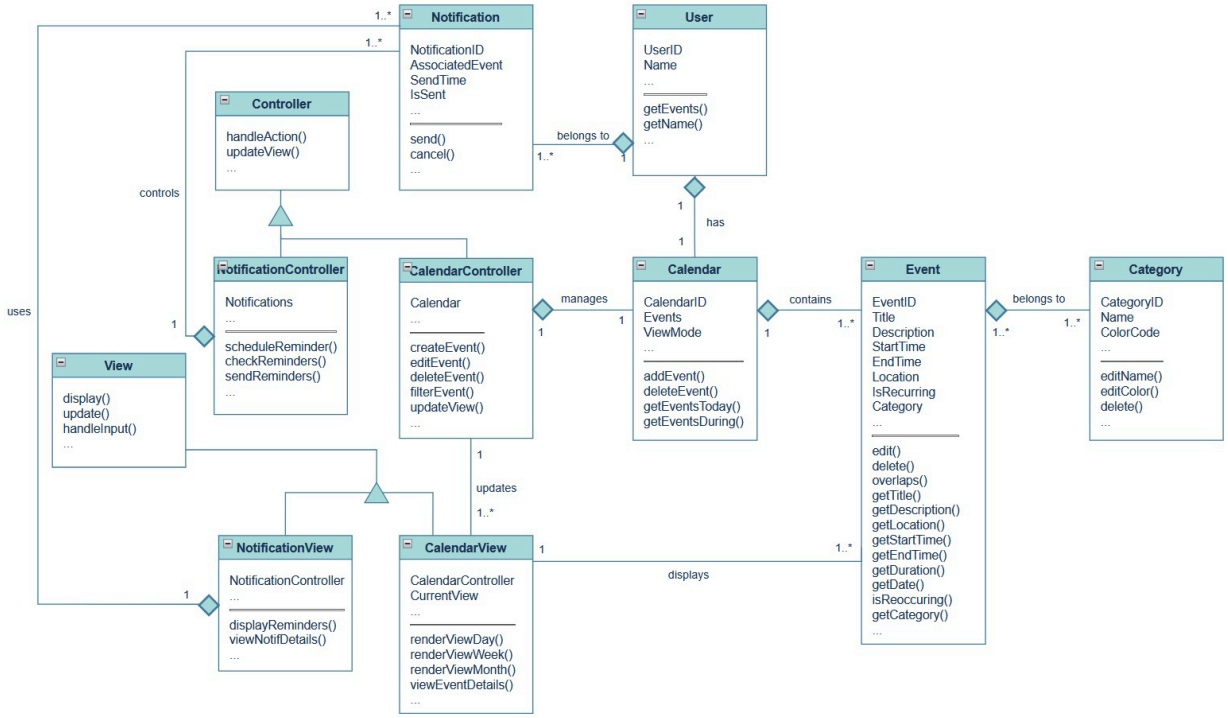


Sequence Diagram:
https://drive.google.com/file/d/18_s47WzeNB2MggM57I1wn8R5CIOFZLkj/view?usp=drive_link

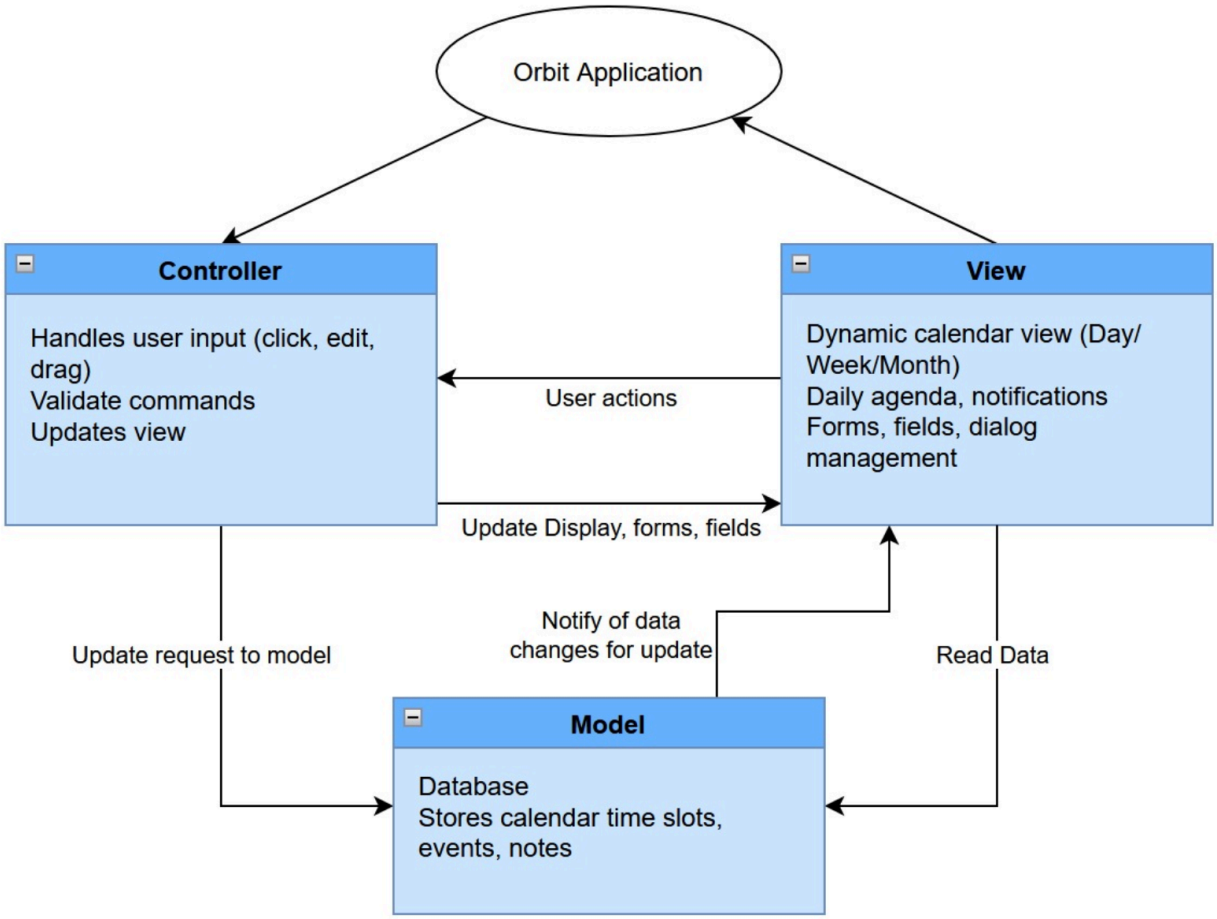


Class Diagram:





Architectural Design (MVC Diagram):



Revised Total Tasks based off Feedback:

1. Coordinate within a meeting to re-iterate our roles according to the project deliverable and implementation itself. (Everyone)
2. Decide upon an appropriate software process model to begin planning out the project more thoroughly. (Joseph)

REQUIREMENTS MODELING

3. Conduct voluntary surveys across the UT Dallas campus, asking and identifying from a free-response form of life-organizational tools used by students with some provided examples (ie. physical calendars, alarms, school resources, other software such as Excel). (Donald, Joseph)
4. Review popular articles and student forums like university blogs or on Reddit for common tools and features used for organization and planning. (Donald)
5. Research reliable libraries, language-specific restrictions, and example applications built within the Java interface to assess its capabilities and limits.

Another part of the requirements modeling phase before further higher-detail phases. (Louie)

ARCHITECTURAL DESIGN

6. Based off the research, we will collaborate to decide upon an architectural pattern according to our project implementation. (Everyone)
7. According to our chosen option, model the pattern with respect to the example figures provided. (Skyler)

COMPONENT DESIGN

8. Develop use-case diagram to plan for multiple scenarios. (Cooper)
9. Plan out sequencing in coordination with the base calendar system along with additional services such as mobile notifications, general default regional settings, and a backup service. (Carlos)
10. Create the class diagram that will let us modularize into Java and move onto code generation. (Skyler)

CODE GENERATION

11. Based off of component design, design all of the UI features and screens using Figma. Communicate ideas represented in case of confusion before handing off to developers. (Kevin)
12. Discuss classes and functions to separate development workload based on every component design diagram to plan out module cohesion. This will lead to the creation of an executable software. (Skyler, Cooper, Carlos)

UPWARDS V MODEL

13. Before even starting on the unit testing phase, we will discuss and coordinate our roles for the “right-side” of the V Model to plan out an optimal review division of our software. This will be documented when the software is finished.