CS 3354 Software Engineering

Final Project Deliverable 2

Homework Calendar Application

Payson Eberz
Van Luong
Abuzar Naqvi
Josh Basker
Mattia Moren
Rishikeshan Bharathi
Andrew Pham
Ali Bidhendi

1. [5 POINTS] Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that in the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student fairly.

Payson Eberz: GitHub project_scope, Deliverable 1 Q1 “Our Response,” Deliverable 1 Document Formatting, and help with app comparison

Van Luong: Functional Requirements, Draft Description Feedback, GitHub URL

Abuzar Naqvi: Project Proposal Document Formatting, Sequence diagrams, conclusion

Josh Basker: Non-Functional Requirements, Application UI Design

Mattia Moren: Create GitHub Repository, Class Diagram, Price and timeline estimations

Rishikeshan Bharathi: Use Case Diagrams, Application UI Design, Documents and Slides Formatting, GitHub commits

Andrew Pham: GitHub README file commit, Software Process Model, Deliverable 2 Doc and Slides setup, Test Plan scenario, Test Plan code setup

Ali Bidhendi: Architectural Design - MVC Diagrams, Rewrite and provide a revised version of the Test code, discussed the UI design environment.

2. [10 POINTS] Everything required and already submitted in Final Project Deliverable 1. Please specify this part as "Project Deliverable 1 content".

Project Deliverable 1 Content

1. [5 POINTS] Please attach here the Final Project draft description (that contains the instructor feedback). It is ok to include a picture of the original document. Address the feedback provided for your proposal by listing what you did / plan to do to comply with those proposed changes and or requests for additions to your project.

Our Response:

We will be making a calendar app that allows:
● Timekeeping: displays the current time and date
● Event Tracking: allows the user to add events and record holidays
● Viewing: Different views yearly, monthly, weekly, daily
● Transferring: Maybe be able to take information from other calendar apps and integrate it here: Notion, Google Calendar, etc…
● Customization: Multiple different cosmetic or functional layouts to help tailor it to the user
This doubles as a homework planner which does the following:
● Course management: Add, edit, and delete courses with details such as course name, professor, location, and meeting times
● Assignment tracking: Keep track of upcoming assignments, deadlines, and submission details
● Reminders and notifications: Set reminders for classes, assignments, and exams
● Integration with university systems: Import course schedules and assignment details directly from university platforms if possible
● Task prioritization: Ability to prioritize assignments and tasks based on deadlines and importance
● Progress tracking: Monitor your progress on assignments and track completed tasks
● Collaboration: Share course schedules and assignment details with classmates/friends for group projects and study sessions

Our Motivation:

As a group, we've collectively felt the strain of managing our university coursework, especially when it comes to staying on top of assignments and deadlines across various courses. It's a challenge we've all faced—juggling multiple responsibilities while trying to maintain a balanced academic life. Recognizing the need for a solution, we've come together to develop a calendar and homework planning app tailored specifically for university students.

Our shared goal is to alleviate the stress and confusion that often accompanies academic planning by providing a user-friendly tool that simplifies the process of organizing courses, tracking assignments, and managing deadlines. With this app, we aim to empower students like ourselves to take control of their academic schedules, enabling them to focus more on learning and less on worrying about missed deadlines or overlooked assignments.

Task Delegation:

As a group, we will design the User Interface of the Calendar Application, by dividing different parts of the software to each of us. This can later be changed based on the complexity of each UX while we design the app throughout the project timeline. For Project Deliverable 1, we split each of the requirements as follows in question 3. As a group, we will discuss which software process model will be employed in our project.

Instructor Feedback:
A practical idea of a tool that promises a lot of potential use.
In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.
Fair delegation of tasks.
Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

Our Response:
Comparison with similar applications: In the final report, we will mention how our calendar app differs from a regular one because of its ability to cater to school related events and assignments. Its unique feature to be able to import deadlines and events from a university calendar (i.e. eLearning calendar) sets it apart from regular calendar apps and makes it easier for students to manage their time. In addition, being able to link their calendar with friends to see how their schedules match up is a feature that few calendar apps possess.
We changed our delegation of tasks to make it more fair, as specified in **#3**.

2. [10 POINTS] Setting up a Github repository. Please use your utdallas email accounts only for each group member.
1.1. Each team member should create a GitHub account if you don't already have one.
1.2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).
1.3. Add all team members, and the TA as collaborators. Our TA has already posted her GitHub info in EL: TA GitHub id: TA email:
1.4. Make the first commit to the repository (i.e., a README file with [team name] as its content).
1.5. Make another commit including a pdf/txt/doc file named "project_scope". If you choose a predefined topic (one of the 4 topics described in the "Project Topic Ideas" section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.

1.6. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.
https://github.com/Tank52/3354-effectiveUmbrella

3. [5 POINTS] Delegation of tasks: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.
Mattia Moren - 2. (1.1, 1.2, 1.3) Create GitHub Repository & 8. Class Diagram
Andrew Pham - 2. (1.4) GitHub README file commit & 4. Software Process Model
Payson Eberz - 2. (1.5) GitHub project_scope & 1. “Our Response” & Document Formatting
Van Luong - 5a. Functional Requirements & 1. Draft Description Feedback & 2. (1.6) add URL
Josh Basker - 5b. Non-Functional Requirements
Rishi Bharathi - 6. Use Case Diagrams
Abuzar Naqvi - 7. Sequence Diagrams
Ali Bidhendi - 9. Architectural Design - MVC Diagrams

4. [5 POINTS] Which software process model is employed in the project and why. (Ch 2)
The software process model chosen for this application is the Prototyping Model. The reason for this is that we deemed it necessary to develop a working prototype in order to better realize and evaluate what the necessary detailed features/functions are desired for the application. We also wanted the flexibility of not adhering to the waterfall aspect of the spiral model since some aspects do not need to be released in stages, and can be continuously worked upon by various team members. Elements of the incremental process may be used as some features may be worked upon without the need of the completion of another.

5. [15 POINTS] Software Requirements including

5.a.) [5 POINTS] Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

- Users can add events/assignments/deadlines to their calendar (specifies title, course name, type (test/hw/event/deadline etc.), submission deadline/date, time, description, location). Some fields listed can be null/not applicable.
- Users can delete events/assignments/deadlines.
- Users should be able to scroll through their calendar to any date they wish, with options to view in monthly, daily, and yearly modes.
- Users should be able receive notifications for events/assignments/deadlines they’ve created.
- Users should be able to search for events/assignments/deadlines on their calendars
- Users can import events/assignments/deadlines from their school’s platform calendar (i.e., eLearning calendar) if applicable.
- Users can link their calendars with friends to see how their schedules match up.

5.b.) [10 POINTS] Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one non- 4 functional requirement for each

of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements to be able to provide one for each. Please explicitly specify if you are considering such assumptions.)

Efficiency Requirements: The app should load any view (yearly, monthly, weekly, daily) within 2 seconds on standard mobile devices to ensure a smooth user experience.

Dependability Requirements: The app must have an uptime of 99.9%, ensuring that users can access their schedules and assignment information almost anytime.

Security Requirements: User data, including personal schedules, course details, and assignment information, must be encrypted both in transit and at rest, using industry-standard encryption protocols.

Regulatory Requirements: The app must comply with the General Data Protection Regulation (GDPR) for users in the European Union, ensuring user data privacy and security.

Ethical Requirements: The app should not use user data for advertising purposes without explicit consent, respecting user privacy and ethical standards in data usage.

Usability Requirements: The app should be accessible to users with disabilities, complying with the Web Content Accessibility Guidelines (WCAG) 2.1 Level AA standards.

Performance Requirements: The app should support simultaneous access by up to 10,000 users without significant degradation in response times.

Space Requirements: The app should require no more than 150MB of space on the device after installation, keeping storage use efficient for users.

Environmental Requirements: The development and operation of the app should prioritize energy-efficient processes to minimize its carbon footprint, including optimizing server usage and encouraging digital over physical task management.

Operational Requirements: The app must be compatible with both iOS and Android platforms, ensuring a wide range of device support for users.

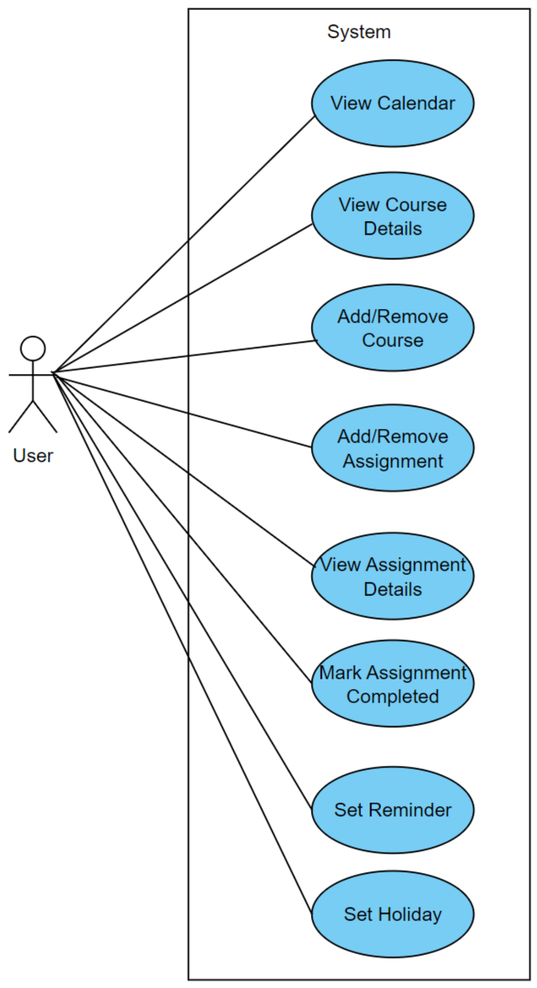
Development Requirements: The app's development process should incorporate continuous integration and continuous deployment (CI/CD) practices to streamline updates and bug fixes.

Legislative Requirements: The app must adhere to the Children's Online Privacy Protection Act (COPPA) if it is to be used by children under the age of 13, ensuring the safety and privacy of young users.

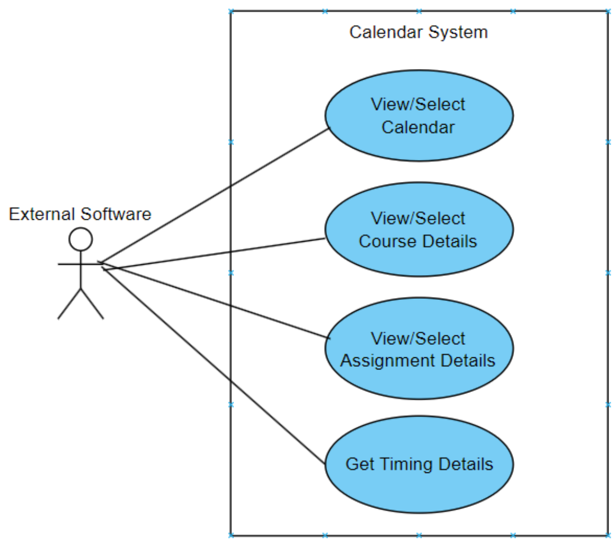
Accounting Requirements: The app should provide detailed logging of user transactions and interactions for auditing purposes, ensuring transparency and accountability in its operation.

Safety/Security Requirements: The app must include robust authentication mechanisms, such as two-factor authentication, to prevent unauthorized access to user information.

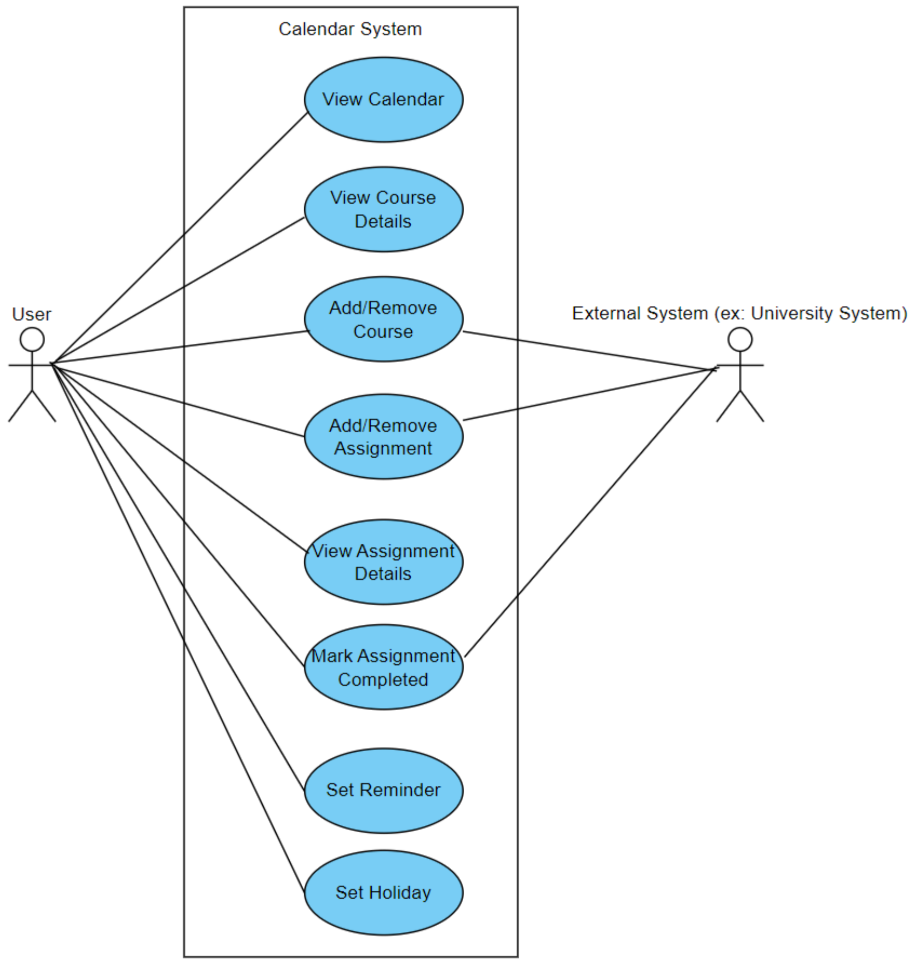
6. [15 POINTS] Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note than there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)



First Use Case Diagram: The user is the only person interacting with the Calendar System. This will be our main functionality design of our project.

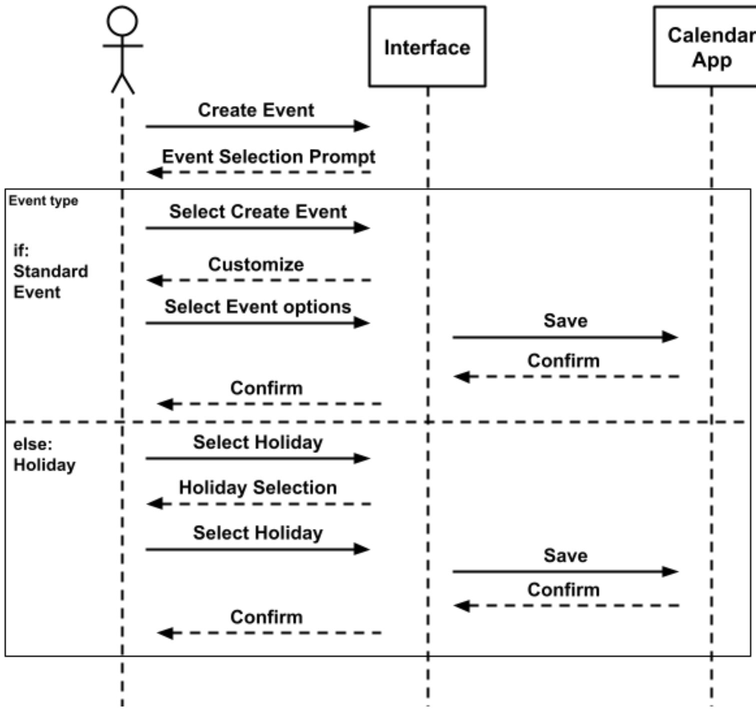


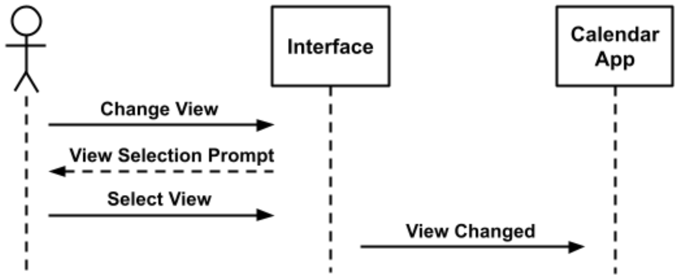
Second Use Case Diagram: Another use case scenario for this application is the ability for external software to view different calendars and it's details (courses, assignments, etc).

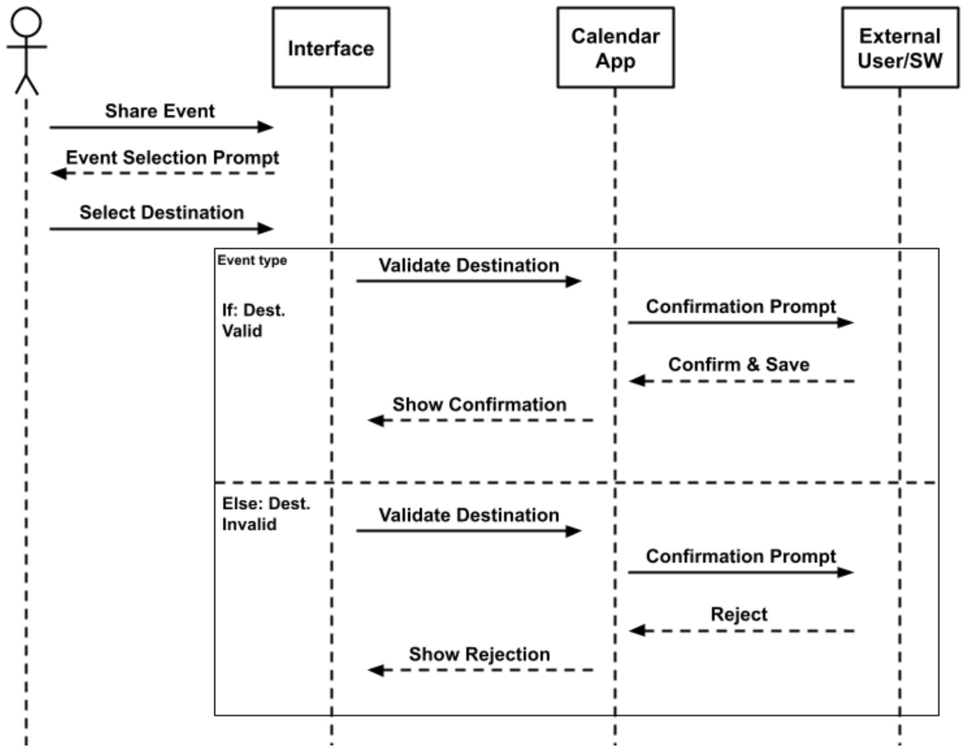


Third Use Case Diagram: We can also implement this application to have interaction with the user as well as external systems such as the user's university system. Where the university system can automatically add courses and assignments to the calendar system.

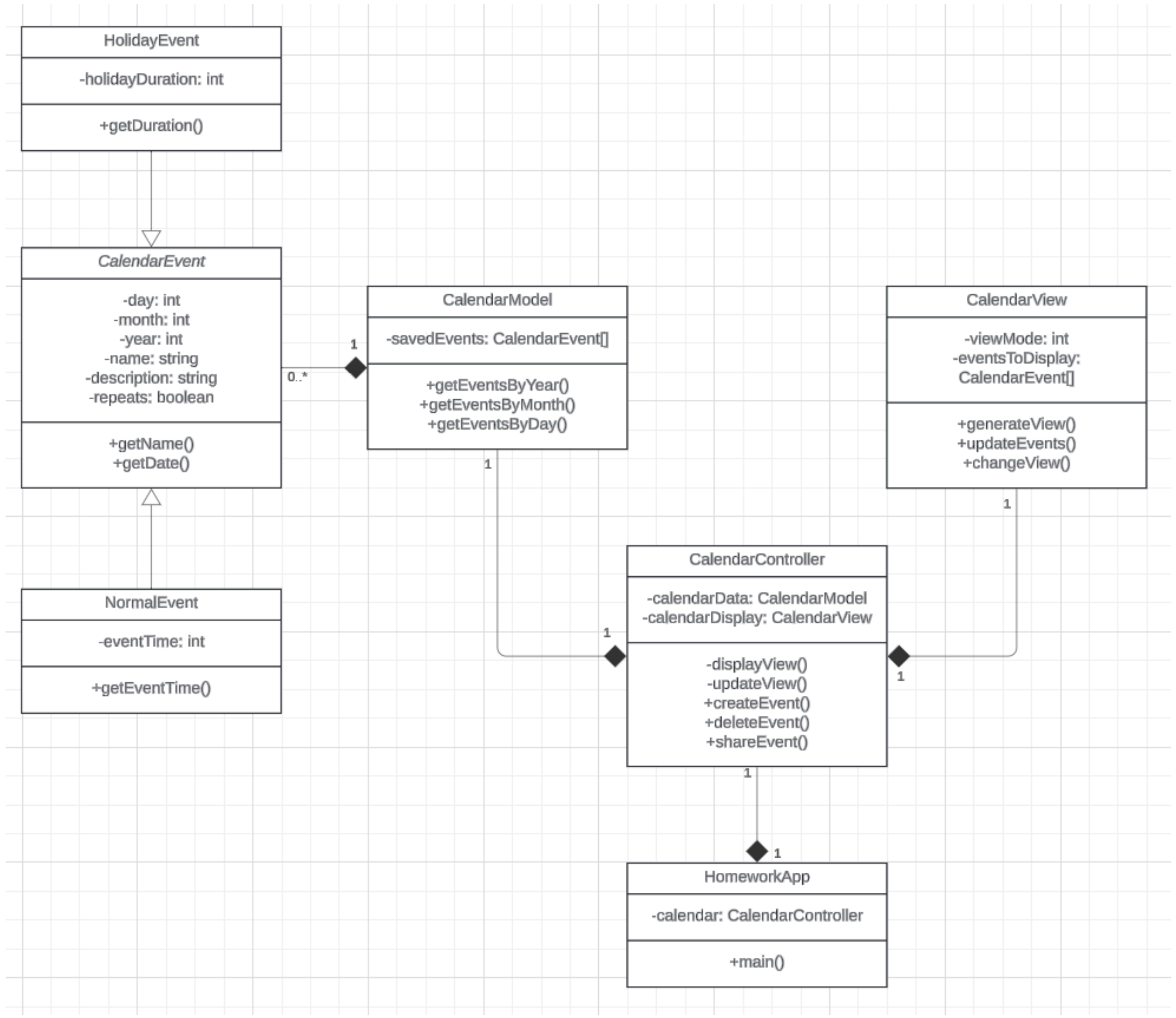
7. [15 POINTS] Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)



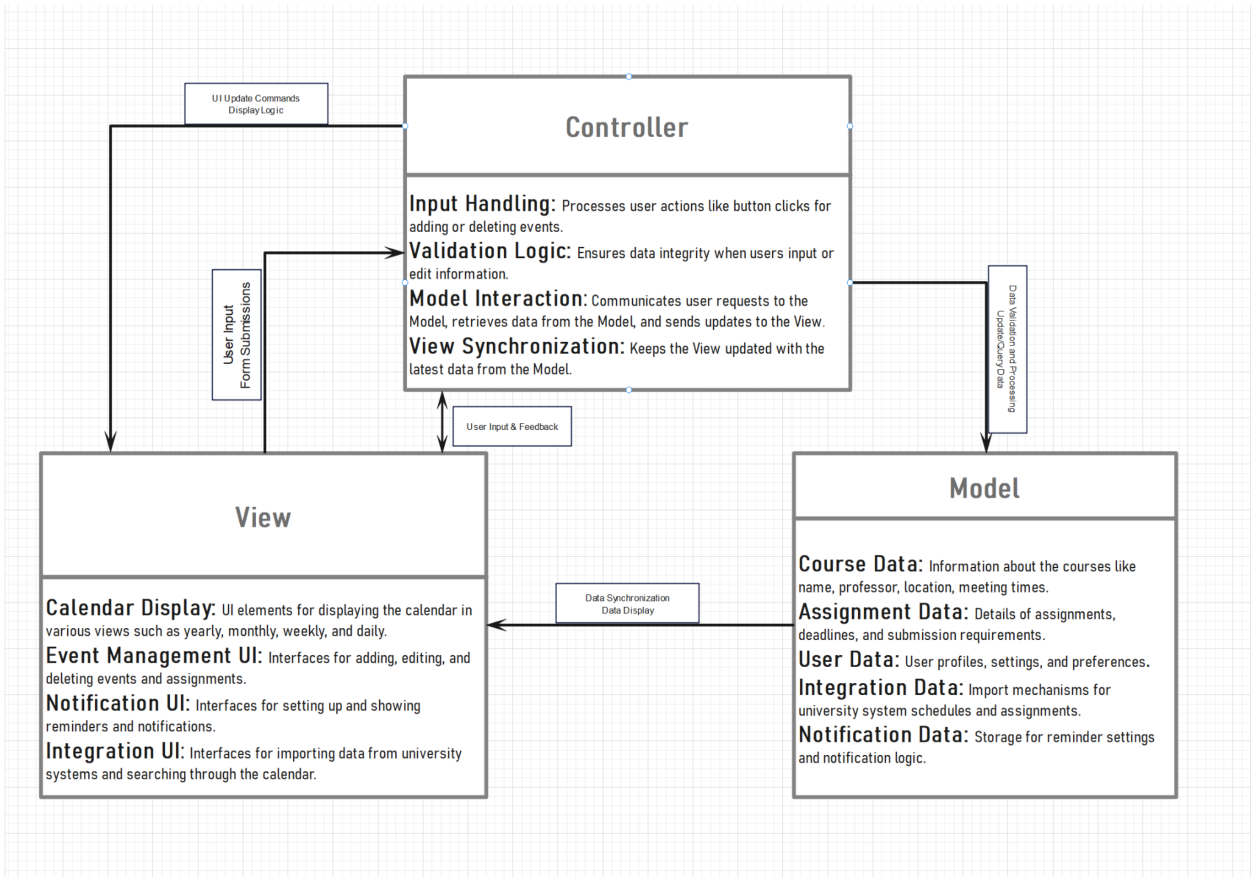



Interface
Calendar App
External User/SW
Share Event
Event Selection Prompt
Select Destination
Event type
If: Dest. Valid
Validate Destination
Confirmation Prompt
Confirm & Save
Show Confirmation
Else: Dest. Invalid
Validate Destination
Confirmation Prompt
Reject
Show Rejection

8. [15 POINTS] Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



9. [15 POINTS] Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3) 9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6) 9.2. Layered architecture pattern (similar to Figure 6.9) 9.3. Repository architecture pattern (similar to Figure 6.11) 9.4. Client-server architecture pattern (similar to Figure 6.13) 9.5. Pipe and filter architecture pattern (similar to Figure 6.15)



Controller

Input Handling: Processes user actions like button clicks for adding or deleting events.
Validation Logic: Ensures data integrity when users input or edit information.
Model Interaction: Communicates user requests to the Model, retrieves data from the Model, and sends updates to the View.
View Synchronization: Keeps the View updated with the latest data from the Model.

View

Calendar Display: UI elements for displaying the calendar in various views such as yearly, monthly, weekly, and daily.
Event Management UI: Interfaces for adding, editing, and deleting events and assignments.
Notification UI: Interfaces for setting up and showing reminders and notifications.
Integration UI: Interfaces for importing data from university systems and searching through the calendar.

Model

Course Data: Information about the courses like name, professor, location, meeting times.
Assignment Data: Details of assignments, deadlines, and submission requirements.
User Data: User profiles, settings, and preferences.
Integration Data: Import mechanisms for university system schedules and assignments.
Notification Data: Storage for reminder settings and notification logic.

Project Deliverable 2 Content cont.

IMPORTANT NOTE: The following items will all need to be calculated / worked on based on the project you are designing. As an example, if a team of 7 students in CS3354 class is working on the development of a hospital information system, this group will prepare the project scheduling, cost, effort and pricing estimation calculations based on the hospital information system design, NOT based on their 7 student team. Think of the analogy to the “Inception” movie: What you will be working on is the dream in a dream, i.e. the dream in the second level, NOT in the first level.

3. [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:

3.1. [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

Our project should take roughly one week to complete. We would begin development on May 13th and should be able to finish around May 20th. Justifications for our estimate is provided via the function point analysis below. Given the small nature of our project, we decided working on the project full time would be manageable since the duration is rather short. Our estimates were made using 8 hour work days, not including the weekends.

3.2. [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)
- Application composition

We used the function point model to estimate the cost of development for our project.

	Function Category	Count	Complexity			Count x Complexity
			Simple	Average	Complex	
1	User input	12	3	4	6	36
2	User output	9	4	5	7	36
3	User queries	6	3	4	6	18
4	Files and tables	30	7	10	15	210

5	External interfaces	3	[✓ 5]	7	10	15

GFP: 315

Processing complexity:

3, 3, 0, 3, 3, 0, 0, 0, 3, 3, 2, 3, 2, 4
PCA = 0.65 * 0.01(29) = 0.94
FP = 315 * 0.94 = 296.1

296.1 / 50 work per week / 5 people = 6 days

$35 * 5 people * 6 days = $8,400

3.3. [5 POINTS] Estimated cost of hardware products (such as servers, etc.)

Our app will be completely local and offline. There are no additional costs imposed due to hardware requirements.

3.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)

Our project does not require any kind of closed-source software, nor does it need any kind of existing software to fulfill a special task. All of it will be designed from the ground up, meaning our software will be designed using free, open-source software licenses such as GPL.

3.5. [5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation)

The product will be released as a stand-alone, offline application. Minimal to no maintenance should be required after release. Bug fixes will be completed during the final stages of development, so additional costs will not be incurred afterwards.

4. [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.

A test plan for a unit of this software is checking if the submitted date has a valid month in it. The app would have multiple ways of inputting a date, such as through text or a drop-down menu. Through text, a month can be in the following formats:

Full Name: January, january, February, March, etc.

Abbreviated to 3 letters: Jan, jan, Feb, Mar, etc.

As Numbers: 01, 1, 2, 03, etc.

Through the text method, it must be valid through the appropriate test cases.

For one unit, a combined getter/setter function receives the simulated string input from the user, and checks it with valid month formats. Essentially, the test receives the getter/setter return value and compares it with the strings that are the correct response.

Test Code available on GitHub Repository.

5. [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.

One popular calendar app is the built-in calendar on Apple devices. While many of the functions between our app and the Apple device calendars are similar, such as the daily/monthly/yearly view and being able to add events with a specific date and time, the fact that our app caters toward students with school-related activities is what sets us apart. More specifically, the Apple calendar app doesn't have features that incorporate a user's course schedule. In addition, Apple has options to sync their calendar with other calendar services using wifi, but users can also use the app locally, similar to our app.

Another more similar application would be MyStudyLife [1], a free online student planner that has a calendar view, a task list, exam list, and class schedule view. One significant difference between our apps is that MyStudyLife is web based, with the data being held online, as opposed to our app being completely local. There are pros and cons with this, the pros of online is that it allows sync between devices, but the con is that there must always be connectivity to the internet. Another difference lies in the functionality requirements of the two applications. For instance, there is a progress bar for tasks on MyStudyLife. This function allows for users to see what portion of their tasks they have completed. This feature allowed the team to open up to possibilities of how to further improve our application in the future. In addition, one way that our app stands out against MyStudyLife is that all notable deadlines, including exams and assignments, can all be viewed in one section labeled "Upcoming Assignments", while in MyStudyLife you must view tasks and exams separately. Furthermore, another notable difference between our app and MyStudyLife is the color scheme. While white and green, which is what MyStudyLife's colors focus on, can be a great choice for this kind of application, our team wanted to choose colors that represent positivity and growth [2] to subtly encourage our users. That is why we decided to implement a green and yellow color scheme.

Sources cited in #7.

6. [10 POINTS] Conclusion -

The development journey of our homework/planner app has been marked by a steadfast commitment to delivering a solution tailored to the needs of university students. Our primary objective centered on creating a robust tool that simplifies academic planning and enhances the overall student experience.

Throughout the development process, we remained receptive to feedback, both from our instructor and those on our team, ensuring that our app stood out and tackled problems faced by university students. By strategically highlighting its unique features, such as seamless integration with university systems and collaborative scheduling options, we aimed to carve a niche solution to the problems faced by our constituents.

While our initial plan outlined a specific software process model, we opted for the Prototyping Model to afford us the flexibility necessary for iterative refinement. This decision proved instrumental in adapting to evolving requirements and delivering a solution that resonates with our target audience. Despite encountering minor deviations from our initial roadmap, such as adjustments to feature prioritization and user interface enhancements, each decision was guided by our desire to meet the user's needs effectively.

The collaborative effort among team members was integral to our success, with each member contributing their expertise and dedication to different facets of the project. Task delegation was managed thoroughly, and this shows in the work we have done. Looking ahead, our focus remains on continuous improvement and user-centric innovation. By leveraging user feedback and embracing emerging technologies, we are poised to further enhance the app's functionality and user experience, solidifying its position as a future indispensable tool for university students.

In summary, our journey in developing the homework/planner app has been characterized by professionalism, diligence, and a dedication to delivering excellence. We are confident that our app will play a significant role in empowering students to manage their academic responsibilities effectively, ultimately contributing to their overall success.

7. [5 POINTS] References: Please include properly cited references in IEEE paper referencing format. Please review the IEEE referencing format document at the URL: https://ieee-dataport.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.

[1] My Study Life, Ltd, *My Study Life: Online Student Planner*, March 27, 2021 [Online]. Available: https://mystudylife.com/. [Accessed: 19, April, 2024].

[2] B. M, "Principles of color in UI Design," Aug. 31, 2023 [Online]. Available: https://uxplanet.org/principles-of-color-in-ui-design-43708d8512d8. [Accessed: 19, April, 2024].

8. [10 POINTS] Presentation slides. No min/max number of slides enforced. Please make sure that you can complete presentation within 20 (twenty) 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)
- 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.

You may use any style in your presentations. A slide show is recommended as it helps display highlights of your talk to the audience and to yourself. The following is a list of suggested content for your presentations:

A brief introduction to your project topic
List of requirements
Use case diagram that contains use cases
Sequence diagram – if too many, one of them only
Design Class Diagram (DCD)
User Interface Design
Comparison with similar work (if any), or emphasizing its significance and uniqueness (if there is not any)
Conclusion and Future Work

Link to slides:

https://docs.google.com/presentation/d/15Y5Ecgj9jBaOCe818zBX3Rf2NmK3-FZBFjzd6RGYH6U/edit?usp=sharing

10. [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)
- Presentation slides

Link to GitHub (Already Added All The Members and TA):

https://github.com/Tank52/3354-effectiveUmbrella