3354 Course Team Project Specifications (Part IV)

PROJECT DELIVERABLE # 2 Report

1. Title of your project
2. Delegation of tasks including each team member (first name and last name)
   - Timur Esenaliev: (6. Comparison of your work with similar designs)
   - Daniel Hirsh (4.Project Scheduling + 9. c,d,e)
   - Wayne Le (9.f Use case and 9.g sequence diagrams)
   - Divya Narayan (7-8)
   - Baylor Resnick (9.h.i-vi)
   - Meenakshi Sundar Rajan: (4. Project scheduling)
   - Lilian Zeng: (5. A test plan for your software)
3. Everything required and already submitted in Course Team Project
4. 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: [Meenakshi Sundar Rajan]
5. A test plan for your software: [Lilian Zeng]
6. Comparison of your work with similar designs [Timur Esenaliev]:
7. Conclusion: [Divya Narayan]
8. References: [Divya Narayan]
9. Presentation Slides must have sections:
   a. Title of your project together with participants
   b. Objective of the project designed
   c. Cost estimation
   d. Project timeline (timeline of the project designed, NOT the time you've spent on it)
   e. Functional and non-functional requirements. If it is too long, select representative items.
   f. Use case diagram [Wayne Le]
   g. Sequence diagram for a selected representative operation of the project. [Wayne Le]
   h. Class diagram
      i. Architectural design
      ii. Model-View-Controller (MVC) pattern (similar to Figure 6.6)
      iii. Layered architecture pattern (similar to Figure 6.9)
      iv. Repository architecture pattern (similar to Figure 6.11)
      v. Client-server architecture pattern (similar to Figure 6.13)

vi. Pipe and filter architecture pattern (similar to Figure 6.15)

REPORT:

1) Title: Calendar Software

2) Delegation of tasks including each team member (first name and last name)

1. Timur Esenaliev: (6. Comparison of your work with similar designs, 9.f)
2. Daniel Hirsh (4.Project Scheduling + 9. c,d,e)
3. Wayne Le (9.f Use case and 9.g sequence diagrams)
4. Divya Narayan (7-8)
5. Baylor Resnick (9.h.i-vi)
6. Meenakshi Sundar Rajan: (4. Project scheduling)
7. Lilian Zeng: (5. A test plan for your software)

3) Everything required and already submitted in Course Team Project Deliverable #1

Everything required and is already submitted in deliverable #1.

For reference here is deliverable #1,
Part_III_Deliverable_I_Course_Team_Project_Specification(1) - Copy.docx

For reference here is the project proposal,
Project Proposal 3354 Team 4 - Copy.docx

Deliverable 1 - Assumptions: Lilian Zeng

This app assumes that there will be multiple users, and each user will have different events saved in a database using the ical format. App assumes users have access to internet and a stable connection.

Users will be able to make and delete different tasks alongside setting reminders, sharing and editing events. Events cannot happen at the same time. Users will also be able to see their events from 3 different views. The one-day view, week view and month view.

Deliverable 1: Feedback: Divya Narayan

The feedback was regarding the software use case and real-world applications. The application is meant to be used for anyone who wants or needs to schedule or plan their life. This application is useful for anyone as this is an organizational tool.

Deliverable 1: Software process model: Divya Narayan

The software process model fitting for this project would be the incremental process model. This is because each feature in the calendar app can be developed in small increments and tested independently. This also helps gather feedback from users for each feature added.

Deliverable 1 - Software Requirements: Meenakshi Sundar Rajan

Functional Requirements (6.1):

The calendar software system is designed to help users manage their schedules efficiently and intuitively. The following functional requirements outline the main features of the system:

1. Event Management: The system shall allow users to create, edit, and delete events within the calendar.
2. Reminders: The system shall allow users to set, update, and remove reminders for upcoming events.
3. Multiple Views: The system shall allow users to view their calendar in three modes – day view, week view, and month view.
4. Conflict Prevention: The system shall prevent users from scheduling overlapping events at the same time.
5. Sharing and Collaboration: The system shall allow users to share specific events with other registered users.
6. Synchronization: The system shall synchronize user data with an online database to ensure events are saved and retrievable across devices.
7. Import/Export: The system shall allow users to import and export calendar data in .ics (iCalendar) format for interoperability with other calendar tools.

Non-functional Requirements (6.1):

The following non-functional requirements specify the system's quality attributes and external constraints based on the categories described in *Figure 4.3 (Sommerville, Ch. 4)*.

A. Product Requirements:

- Performance: The system shall display daily, weekly, or monthly calendar views within 3 seconds under normal network conditions.
- Reliability: The system shall maintain a minimum uptime of 99% for calendar synchronization and event retrieval operations.
- Usability: The user interface shall be intuitive, visually clear, and accessible to users with varying technical backgrounds.
- Efficiency: The system shall minimize storage usage by optimizing how recurring events and reminders are stored.

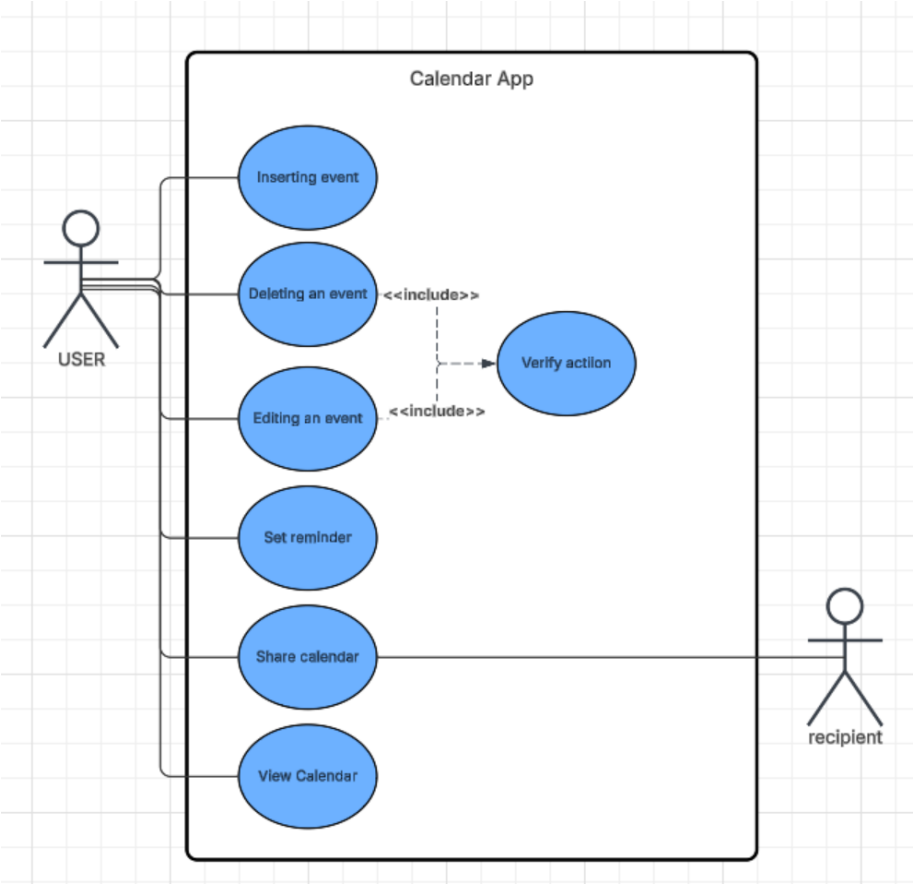
B. Organizational Requirements:

- Security: User credentials and event data shall be stored securely using encryption and hashed passwords.
- Maintainability: The system's architecture shall be modular to allow easy updates and integration of new features in the future.
- Portability: The application shall be deployable on multiple platforms, including web browsers and mobile devices.

C. External Requirements:

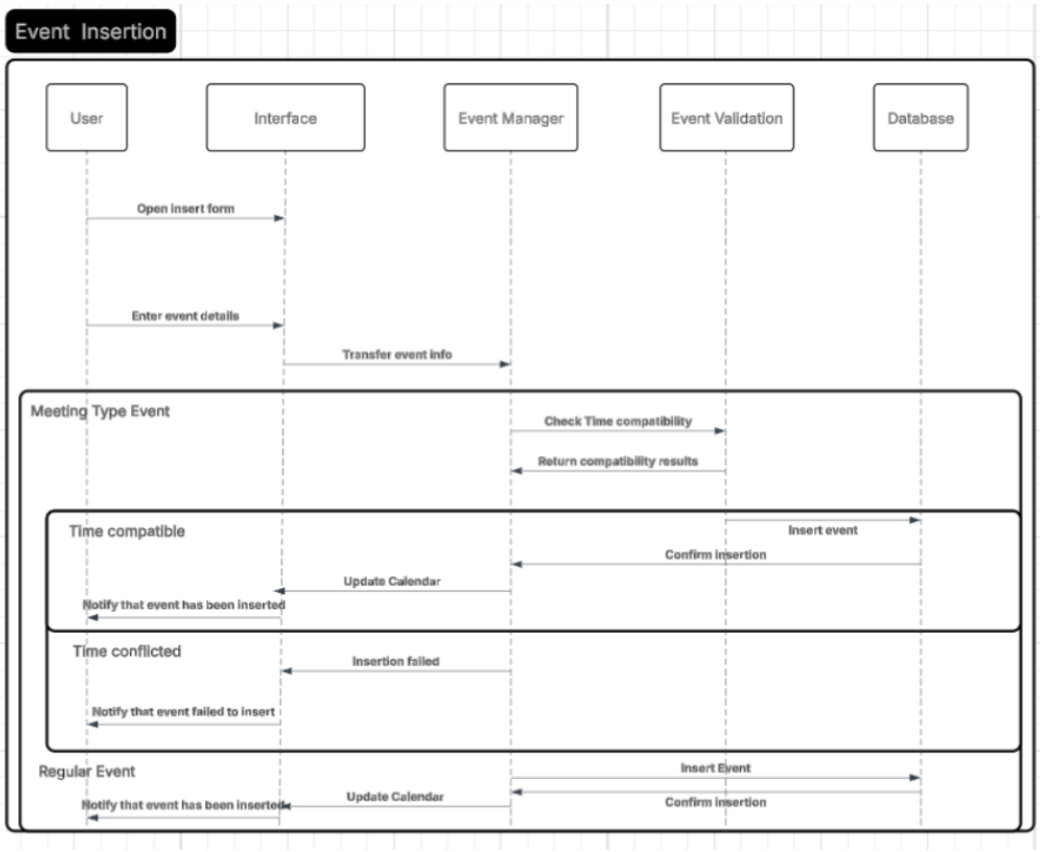
- Legal: The system shall comply with relevant data protection and privacy laws, such as GDPR, to safeguard user information.
- Ethical: The system shall not collect, share, or sell personal user data without explicit consent.

Deliverable 1 - Use Case Diagram: Wayne Le

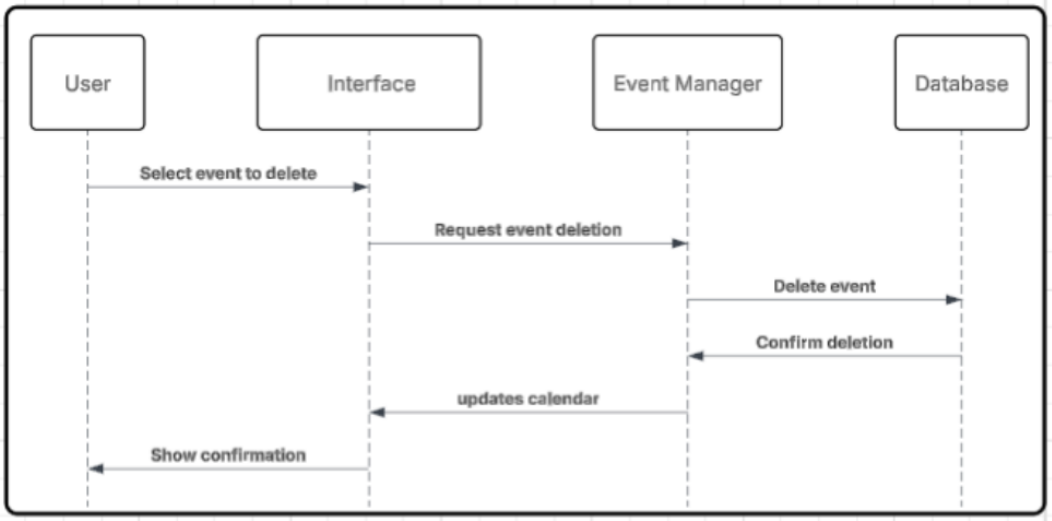


Deliverable 1 - Sequence Diagram: Wayne Le

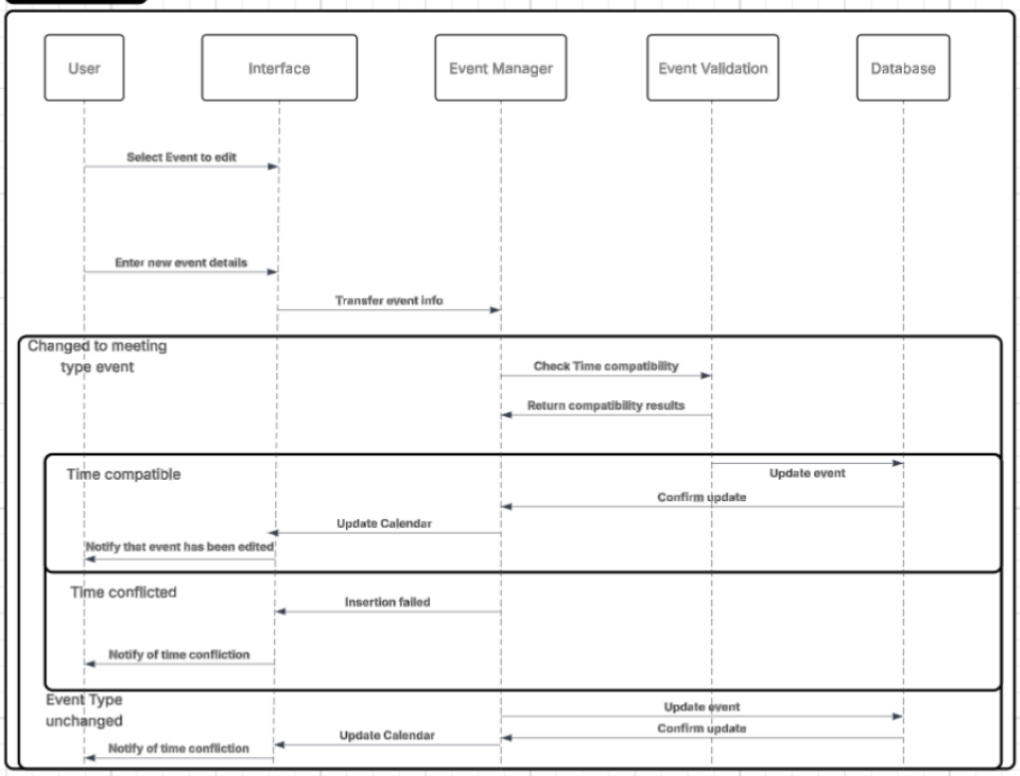
Event Insertion



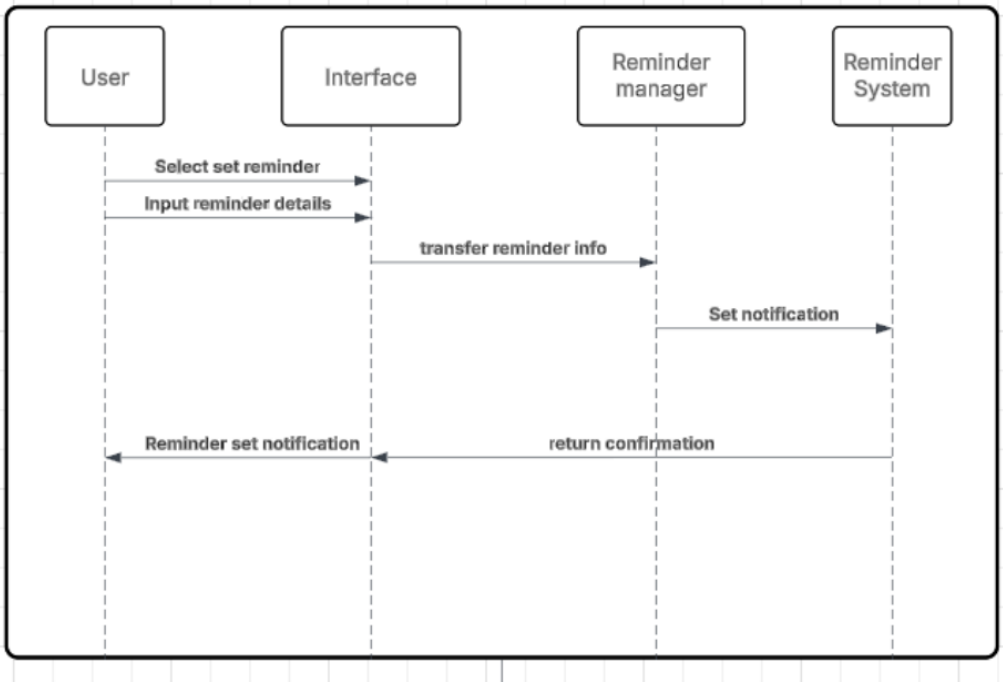
Event Deletion

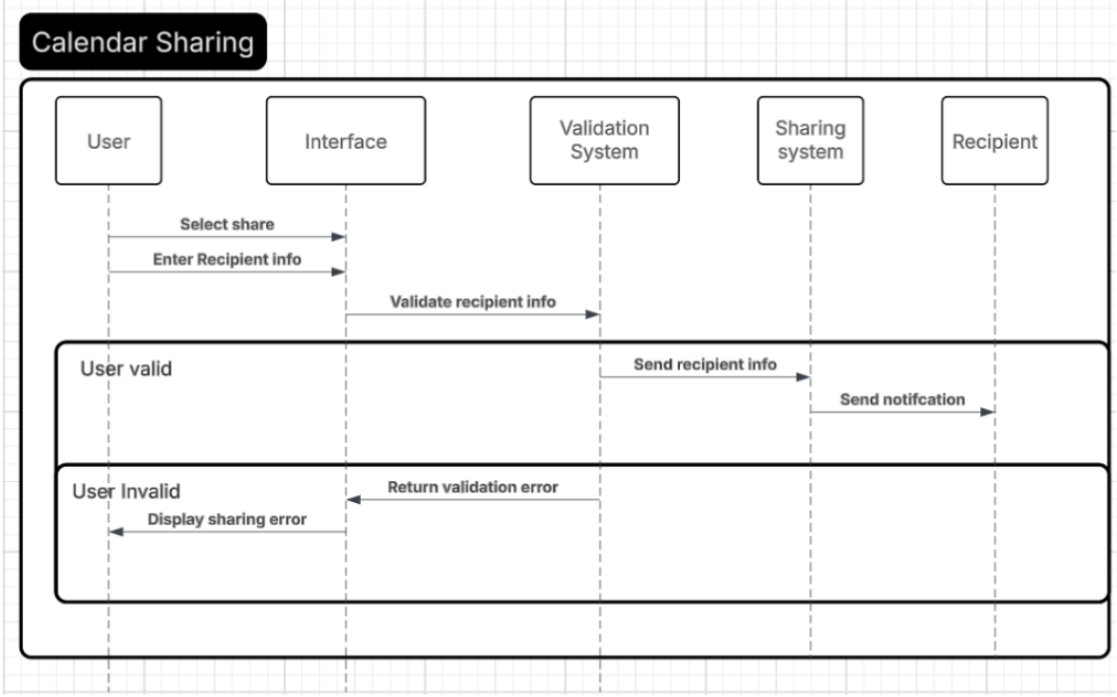


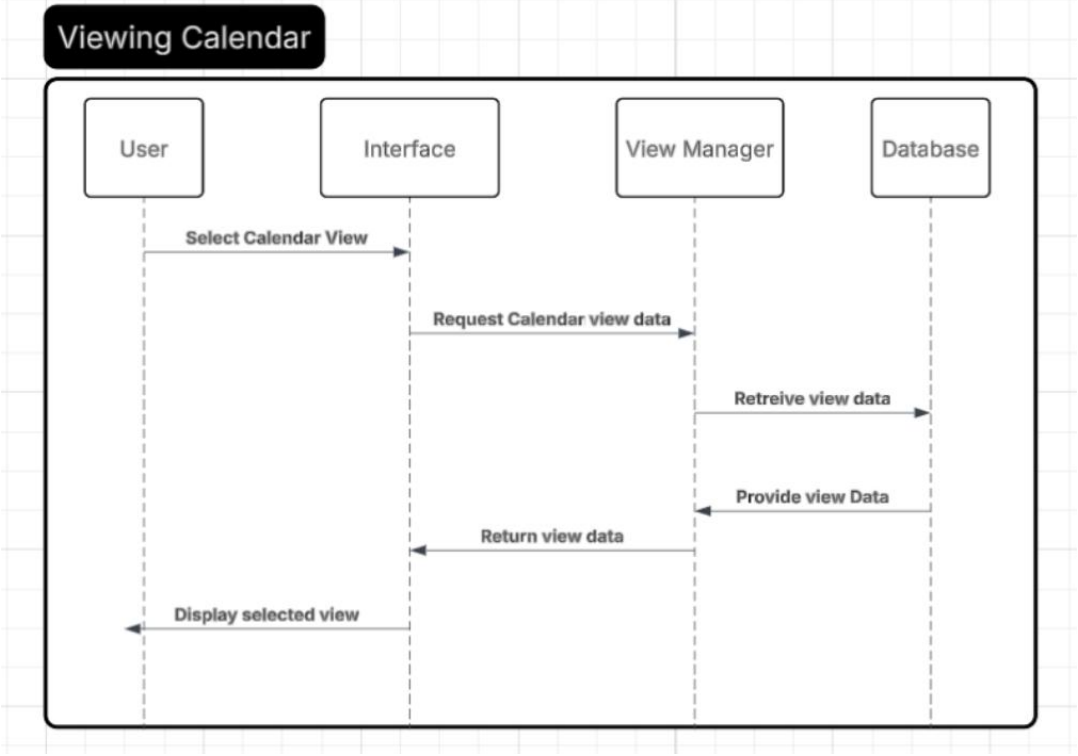
Event Editing



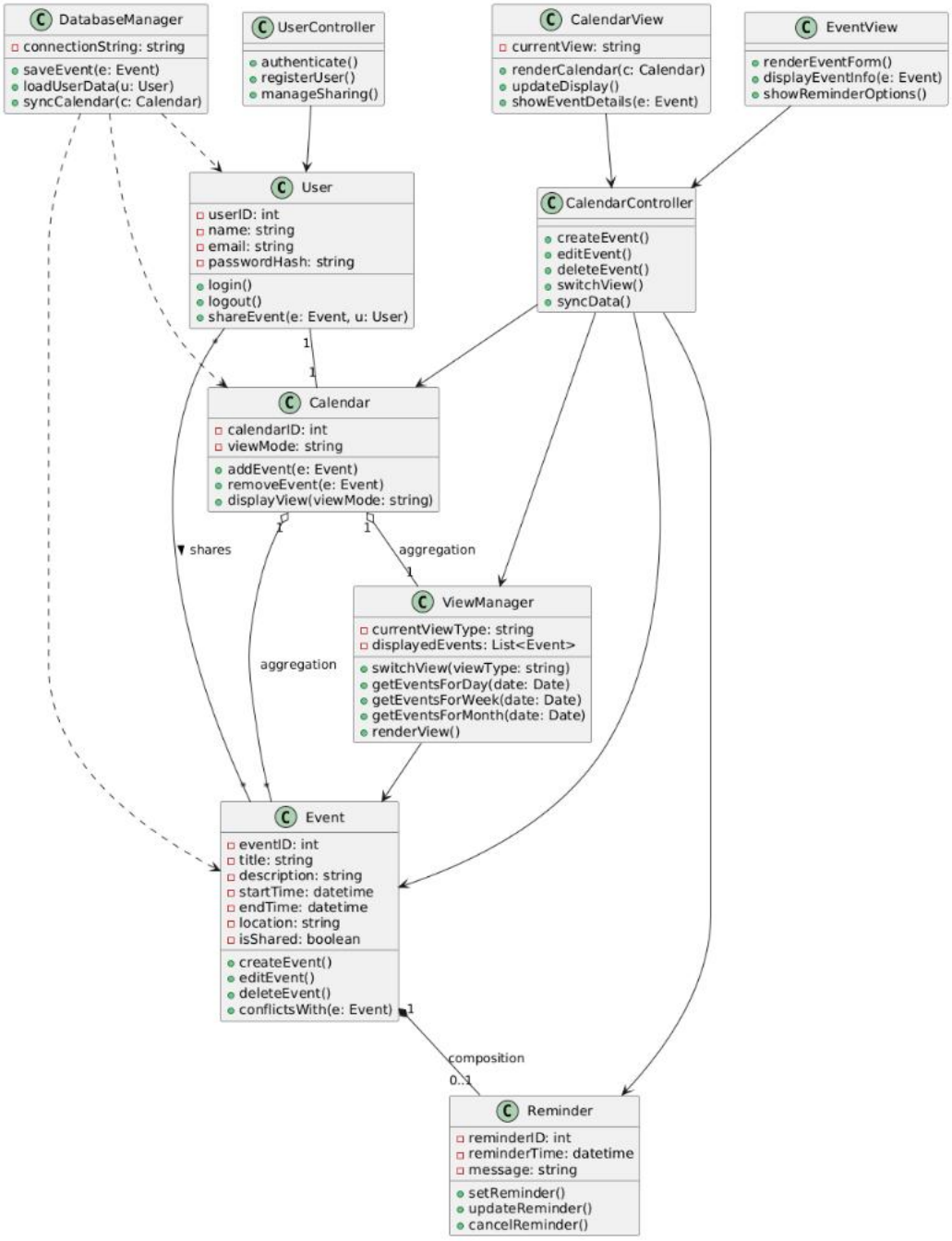
Setting a reminder




Calendar Sharing
User
Interface
Validation System
Sharing system
Recipient
Select share
Enter Recipient info
Validate recipient info
User valid
Send recipient info
Send notifcation
User Invalid
Return validation error
Display sharing error

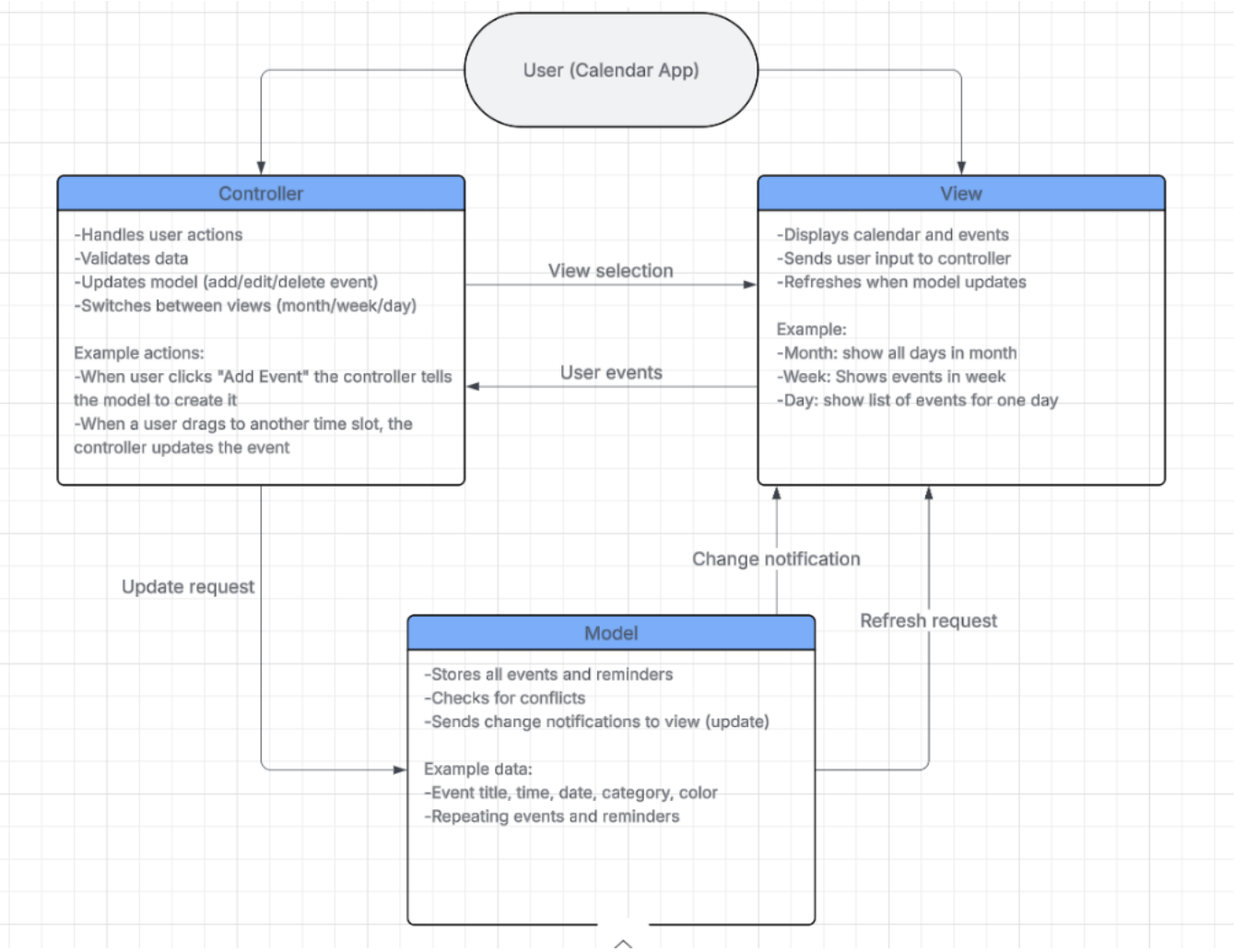

Viewing Calendar
User
Interface
View Manager
Database
Select Calendar View
Request Calendar view data
Retreive view data
Provide view Data
Return view data
Display selected view

Deliverable 1- Class Diagram : Daniel Hirsh



Deliverable 1 - Architectural design: MVC Pattern : Timur Esenaliev

Pattern: Model-View-Controller (MVC)



4) Project Scheduling, Cost, Effort, and Pricing Estimation, Project duration & Staffing

This section provides a detailed analysis of the project scheduling, effort estimation, cost estimation, pricing strategy, and staffing assumptions for establishing the commercial-grade **Calendar Software** project (multi-view interface, cloud sync, sharing features, reminder system, scalable backend). All calculations and schedules are performed **from the perspective of a real software company developing this product**, not from the viewpoint of the student group. The estimations follow industry-standard project management practices and use the **Function Point (FP)** cost modeling technique as required.

Project Scheduling:

- Start Date: January 5, 2026

- End Date: January 23, 2026
- Total Duration: 3 weeks

This duration is justified based on industry norms for medium-complexity cloud-based productivity applications that include scheduling, reminders, cloud synchronization, calendar sharing, and multi-view UI components.

A commercial-grade Calendar Software (multi-view interface, cloud sync, sharing features, reminder system, scalable backend) requires:

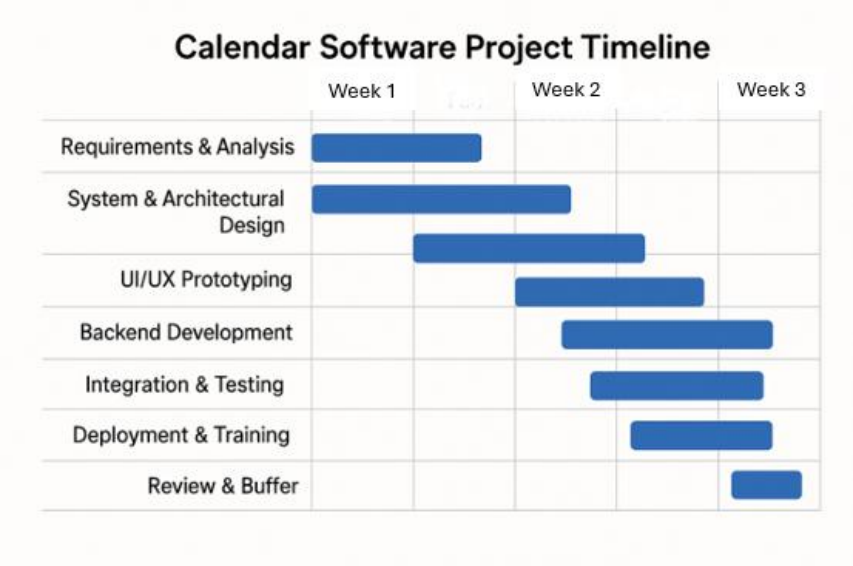
- Requirements gathering
- UI/UX design
- Backend implementation
- Database setup
- API creation
- Testing & integration
- Deployment & training

Work-week Assumptions:

- **Working Days:** Monday–Friday
- **Weekly Work Hours:** 40 hours

Phase	**Description**
Requirements & Analysis	user stories, FP estimation & analysis
System & Architectural Design	UML diagrams, MVC architecture
UI/UX & Front-end Design	Wireframes, usability design
Backend Development	Event logic, reminders, sync
Integration & Testing	Unit tests, performance testing
Deployment & Training	Rollout + training
Buffer & Final Review	Final adjustments

Project Timeline (Gantt Chart):



Cost, Effort, and Pricing Estimation:

- The **Function Point (FP)** method was used to estimate system size and development effort. FP analysis evaluates system functionality based on user-visible features such as inputs, outputs, data files, and system interactions.

Function Point Calculation:

Unadjusted Function Points (UFP)

Component	Count	Complexity	FP Weight	GFP
External Inputs	6	Medium	4	24
External Outputs	5	Medium	5	25
External Inquiries	4	Medium	4	16
Internal Logical Files	3	High	10	30
External Interface Files	2	High	7	14

Total GFP – 109

Technical Complexity Adjustment (TCA):

- Average factor influence = 3
- TCA = 0.65 + (0.01 * 42) = 1.07

Adjusted Function Points (AFP):

- AFP = 109 * 1.07 = approx. 117 FP

Effort Estimation:

- Industry average: 1 FP = 5 hours
- Effort = 117 * 5 = 585 hours

Cost Estimation:

Assuming an average developer rate of $60/hour:

Total Development Cost:

585 hours * $60/hr ≈ $35000

Hardware Cost Estimation:

Hardware Component	**Cost**
Cloud Server (AWS EC2)	$200
Database Server (AWS RDS)	$300
CI/CD Build Server	$300
Backup & storage	$400

Total Hardware Cost = $1,200

Software Cost Estimation:

Software	**Cost**

JetBrains License (team)	$150
Figma Pro (team)	$250
Email Notification API (SendGrid)	$100
Monitoring Tools (Datadog)	$100

Total Software Cost = $600

Personnel Cost Estimation (Team Structure):

Role	**Weekly Salary**	**Weeks**	**Cost**
1 Project Manager	$2,500	3	$7,500
2 Full-stack Developers	$2,125 each	3	$12,750
1 UI/UX Designer	$1,750	3	$5,250
1 QA Engineer	$1,625	3	$4,875

Subtotal Personnel Cost = $30,375
Training & Installation = $3,000

Total Personnel Cost = $33,375

Final Summary Table

Category	**Cost**
Development Cost (FP-Based)	$35,000
Hardware Cost	$1,200
Software Cost	$600
Personnel Cost	$33,375
Total Cost	**$70,175**

All in all, the calendar software system project requires an estimated 585 **hours** of development effort and spans approximately 3 weeks. The overall estimated cost of developing the system is **$70,175.** Hardware, software, and staffing requirements were estimated using realistic assumptions for a medium-scale cloud-based productivity application.

5. A test plan for your software: [Lilian Zeng]:
https://github.com/TimurEsen/3354.006_Team4

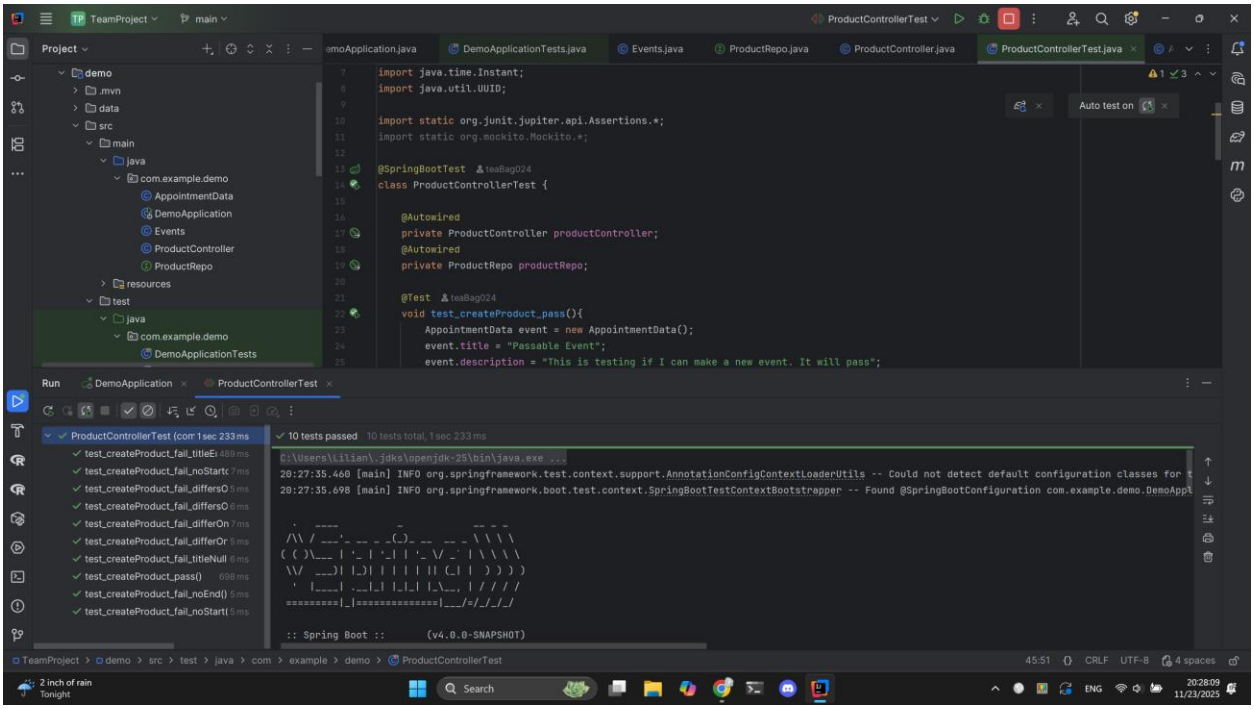
The calendar software will be tested based on the different components and functionality. The Backend is written in Java so we will use JUnit to test functionality. The following is code that tests adding events to the database [1].

The following function must make sure the event has a name/title, and that start date/time must come before end date/time. The Unit test inserts new events into the database and makes sure that invalid events will not be made. The test passes if either the correct error is thrown or if the values in the database match what was originally imputed into the test.

Unit test code:
https://github.com/TimurEsen/3354.006_Team4/blob/main/demo/src/test/java/com/example/demo/ProductControllerTest.java

Function it tests:
https://github.com/TimurEsen/3354.006_Team4/blob/main/demo/src/main/java/com/example/demo/ProductController.java

After running JUnit Tests to see if it passes.



	Name	Description	Expected	Output
1	test_createProduct_pass	Inputs all correct arguments	passed	passed
2	test_createProduct_fail_titleNull	Inputs an event with a null Title	IllegalArgumentException	IllegalArgumentException

3	test_createProduct_fail_titleEmpty	Inputs an event with an empty string as title	IllegalArgumentException	IllegalArgumentException
4	test_createProduct_fail_noStart	Inputs an event with no start time	IllegalArgumentException	IllegalArgumentException
5	test_createProduct_fail_noEnd	Inputs an event with no end time	IllegalArgumentException	IllegalArgumentException
6	test_createProduct_fail_noStartorEnd	Inputs an event with neither start or end time	IllegalArgumentException	IllegalArgumentException
7	test_createProduct_fail_differOnSameDay_mins	Inputs an event with start time that is after end time by minutes	IllegalArgumentException	IllegalArgumentException
8	test_createProduct_fail_differsOnSameDay_hours	Inputs an event with start time that is after end time by hours	IllegalArgumentException	IllegalArgumentException
9	test_createProduct_fail_differsOnDifferentDay_validTime	Inputs an event with start time that is after end time only for the days, the hours match	IllegalArgumentException	IllegalArgumentException

10	test_createProduct_fail_differOnDifferentDay_invalidTime	Inputs an event with start time that is after end time by day and by time.	IllegalArgumentException	IllegalArgumentException

6) Comparison of your work with similar designs [Timur Esenaliev]:

Our proposed calendar application shares many similarities with existing systems such as Google Calendar, Apple Calendar, and Microsoft Outlook Calendar, but has design decisions that differentiate the project in structure, requirements, and scope.

Our Calendar software has similarities in core event management. Like Google Calendar and Apple Calendar, our software has the ability to create, edit, delete, and view events. These functions also support recurring events, reminders, and time settings, which are features listed in our functional requirements. Another similarity is the multiple viewing modes. Most online calendar tools allow users to switch between day, week, and month views. Our design adopts the same three-view structure, which is a standard since it supports short-term and long-term planning. Apps such as Google Calendar [2] store data on the cloud to support cross-device access. Our requirement, functional requirement 6, states that we will sync data via an online database to ensure events are saved and retrievable across devices. Finally, the iCalendar (.ics) format is standard and widely used in industry to import/export events. Our design aligns with this convention to maintain compatibility with other calendar systems (Referenced from functional requirement 7).

However, there are some differences and improvements from our calendar software compared to popular industry versions. Commercial calendars warn users of overlapping events, but our design prevents the creation of overlapping events entirely. This enforcement supports cleaner schedule management. Another difference is that instead of building full team-based calendars like Google Workspace [4] or Outlook [3], our project focuses only on event-level sharing. This keeps the design lightweight, easier to code, and simpler for everyday users who only need to share specific events rather than entire schedules. While the type of model used for Google Calendar is unknown, our project uses an incremental model, allowing each feature to be built and tested in controlled stages. Finally, real-world calendars often support offline access, multiple accounts, and heavy customization; our design keeps these elements minimal. The goal is to create a clean, intuitive scheduling tool without the complexity that can overwhelm new users.

7. Conclusion [Divya Narayan]:

One of the major changes we made to the project was to hone our requirements. Since one of the main feedback items was to make the requirements actionable, we implemented that by specifically addressing the functional and non-functional and utilizing unit tests to test the requirements. Further, we specified our use cases especially in terms of who can use our software in the real world. This software can be used by anyone as it is software with simple features. Overall, we believe our work encompasses all that is required and has been implemented well.

8. References [Divya Narayan]:

[1] "Helped with debugging and learning how to use Springboot and H2" how do I start a spring boot application with h2?. Gemini, Flash 2.5, Google, https://gemini.google.com/app.

[2] Google, “Google Calendar,” *Google.com*, 2019. [Online]. Available: https://calendar.google.com/. [Accessed 23 Nov. 2025].

[3] Outlook, “Outlook,” *Live.com*, 2025. [Online]. Available: outlook.live.com/calendar/0/. [Accessed 23 Nov. 2025].

[4] Google, “Google Calendar: Online Calendars for Business | Google Workspace,” *workspace.google.com*, 2025. [Online]. Available: workspace.google.com/products/calendar/. [Accessed 23 Nov. 2025].