Student Academic Calendar Software

Course: CS/SE/CE 3354 Software Engineering

Instructor: *Barbara Maweu*

Team: Gate 8

Members: Zaid Hilal, Zuhaib Buchh, Nicholas Hudson, Joshua Feng, Noah Spain, Omer Tariq

GitHub: omert-dev/3354-gate8

Introduction

Project Overview (What is group 8 doing?)

Purpose:
To create a unified academic calendar platform for students to manage their schedules, classes, assignments, and events in one place.

Key Objectives:

- Simplify academic planning and event organization
- Support multiple calendar views (daily, weekly, monthly, agenda)
- Offer customizable categories and easy navigation

Calendar

Manage your schedule

New Event

November 2024

Today

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8 Team Meeting	9
10	11	12 Project Review	13	14	15 Client Call, Lunch	16
17	18	19	20 Workshop	21	22	23
24	25 Birthday Party	26	27	28	29	30

Upcoming Events

- **Team Meeting**
  November 8 • 10:00 AM
- **Project Review**
  November 12 • 2:00 PM
- **Client Call**
  November 15 • 11:00 AM
- **Lunch**
  November 15 • 12:30 PM

Task Delegation (Who does what)

Feedback Received from Professor:

Delegated tasks focused only on implementation. Include roles for requirements, design, and testing.

Actions Taken:

- Added clear roles for all phases:
  - **Requirements Gathering:** Omer, Nicholas, Zaid
  - **Design:** Joshua, Omer, Zuhaib
  - **Implementation:** Joshua, Nicholas, Zuhaib
  - **Testing:** Noah, Omer, Zaid

Software Process Model (The model we chose and why?)

Chosen Model: *Incremental Process Model*

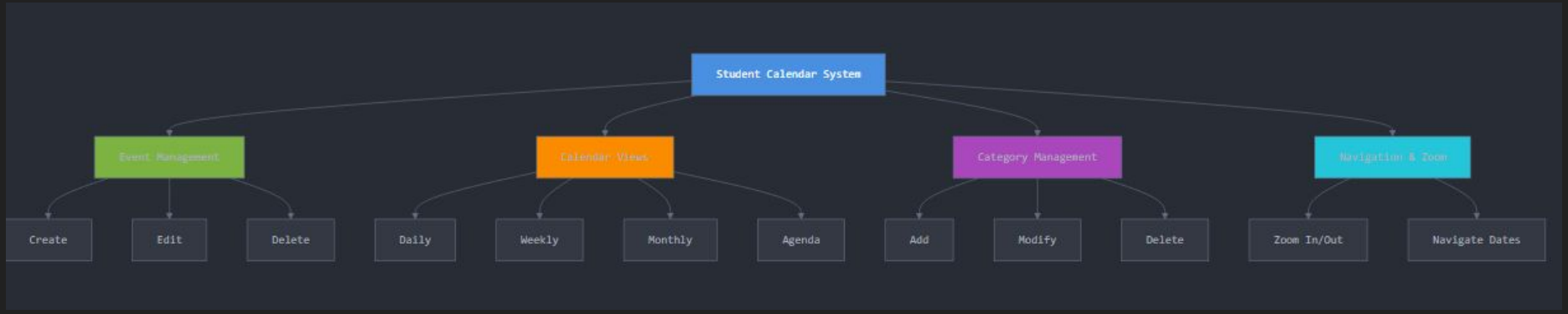
Why:

- Divides work into independent modules (daily, weekly, monthly, event management, etc.)
- Allows parallel development and testing
- Supports flexibility and user feedback integration

Software Requirements (Functional)

Functional Requirements:

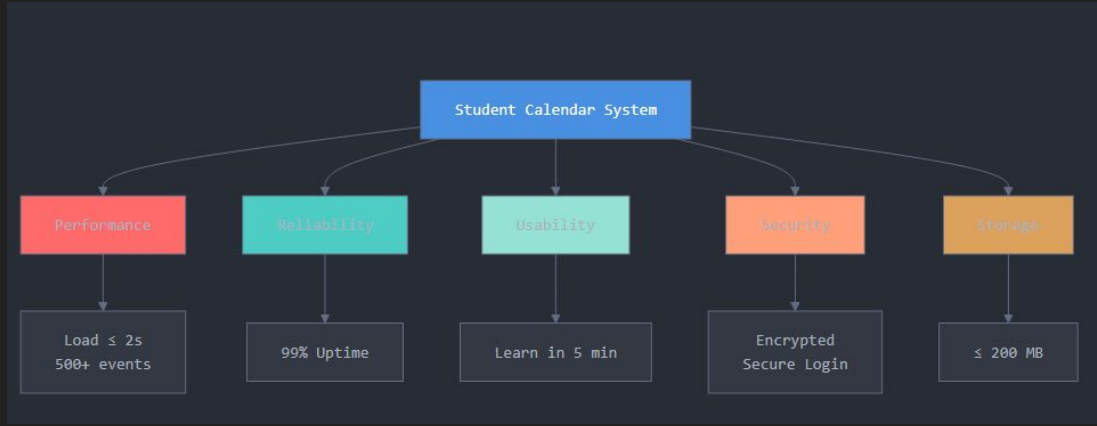
- Event Creation & Management
- Multiple Calendar Views (daily/weekly/monthly)
- Agenda Board for upcoming events
- Category Management (add/modify/delete)
- Zoom & Navigation Controls



Software Requirements (Non-Functional)

Product Requirements:

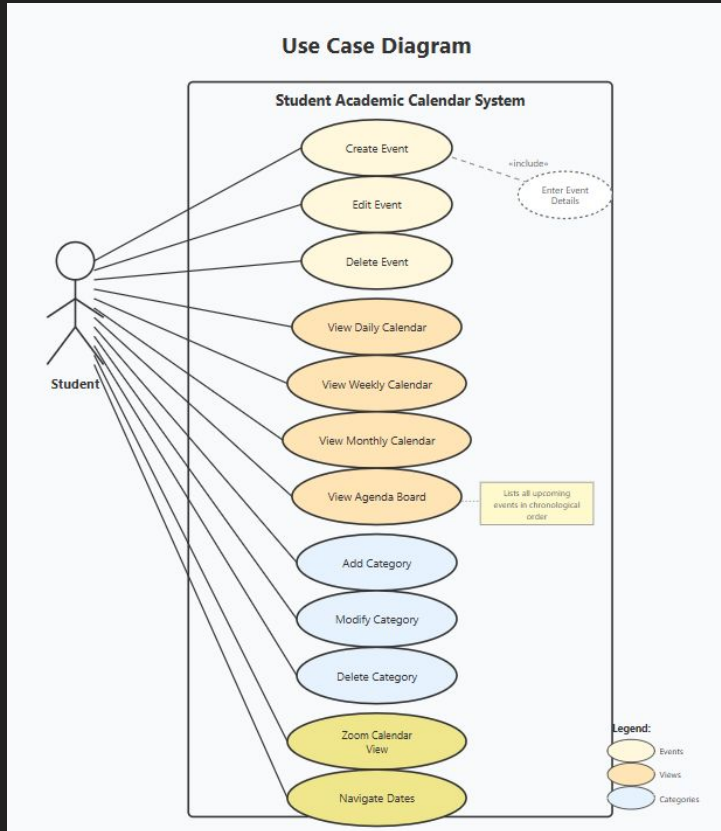
- Efficiency: Load within 2 seconds
- Reliability: 99% uptime
- Usability: Learn within 5 minutes
- Performance: Handle 500+ events per user
- Space: ≤ 200 MB
- Security: Encrypted storage, secure login



System Design: Use Case Diagram

Actor: Student (primary user)
Key Use Cases:

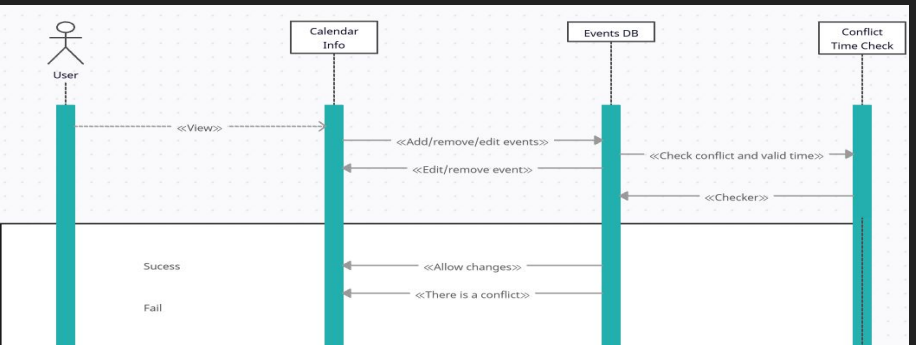
- Create, Edit, Delete Events
- View Calendar (daily/weekly/monthly/agenda)
- Add/Modify/Delete Categories
- Zoom and Navigate Views

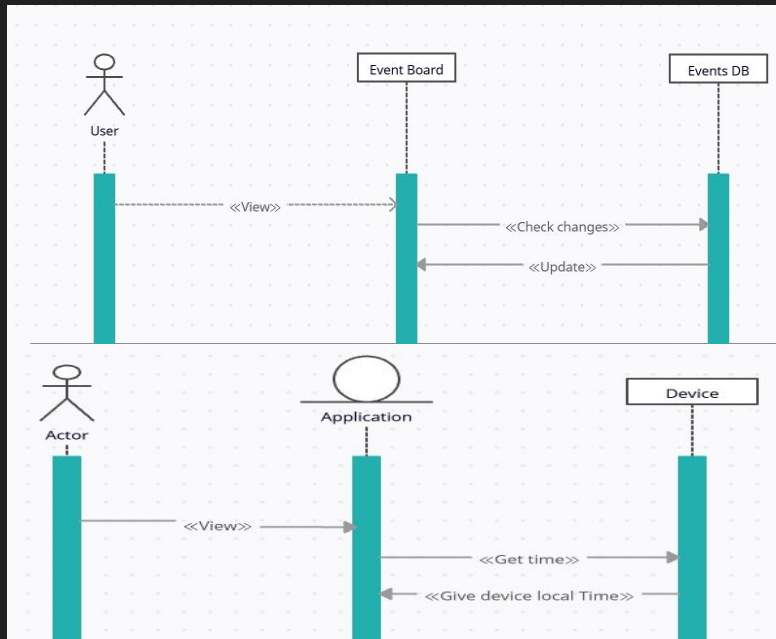


Sequence Diagram

Scenarios Modeled:

- Calendar Views (monthly, weekly, daily)
- Event Board interactions
- Local time synchronization
- Conflict checking

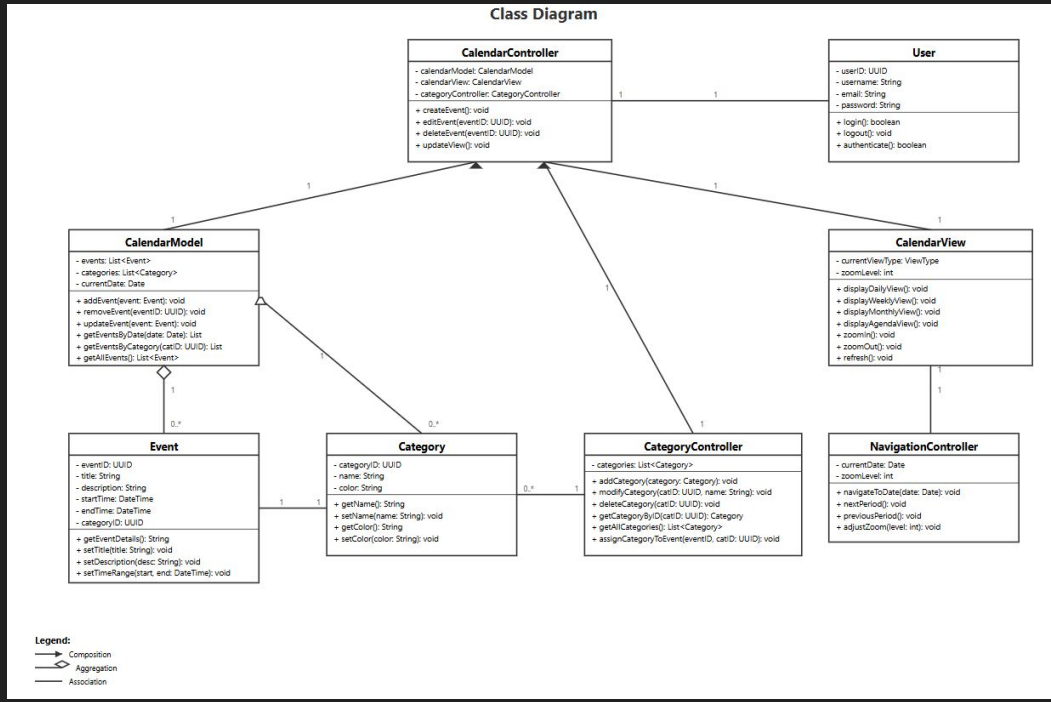




Class Diagram

Core Classes:

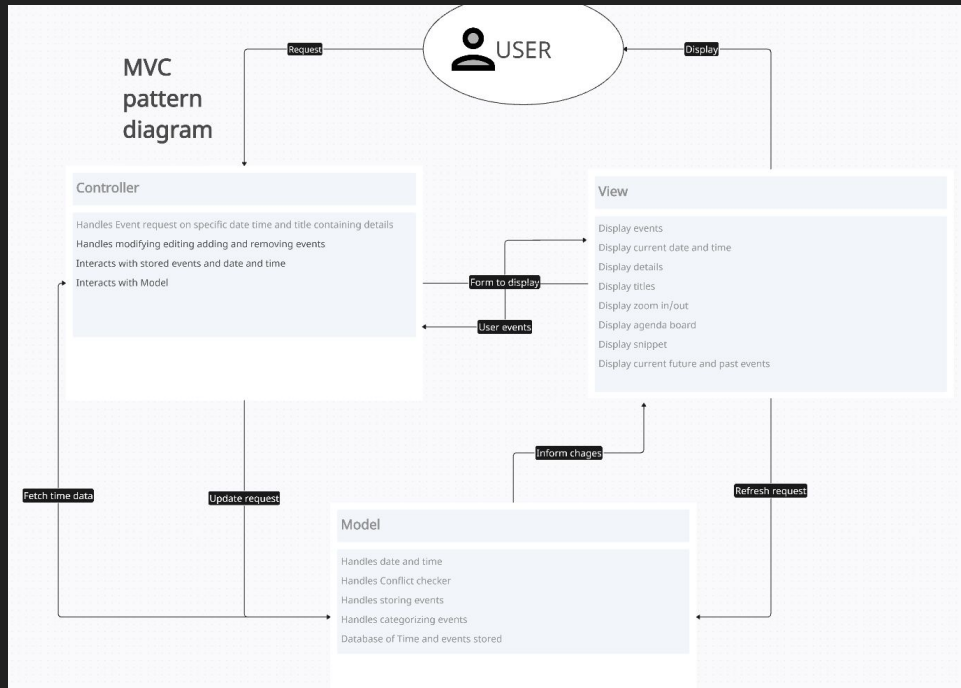
- `CalendarController` – manages all operations
- `CalendarModel` – stores data (events, categories)
- `CalendarView` – displays calendar
- `Event`, `Category`, `User`
- `NavigationController`, `CategoryController`



Architectural Design

Pattern: Model–View–Controller (MVC)

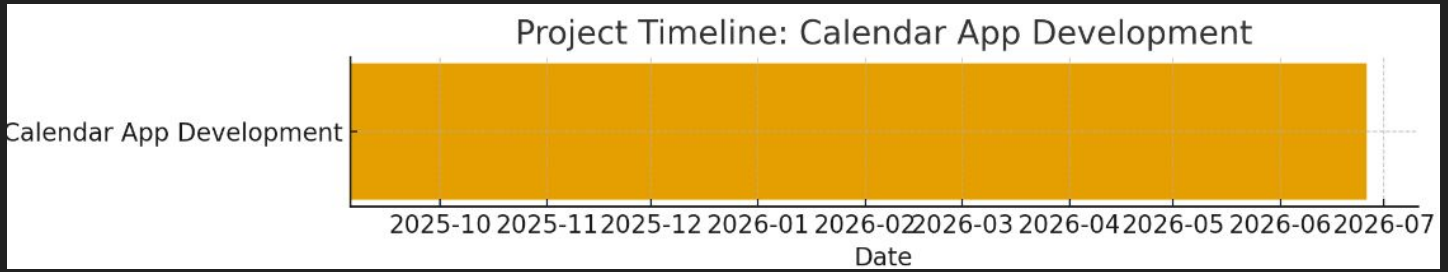
Flow:
User → View → Controller → Model → Controller → View → User



Project Scheduling

Timeline:

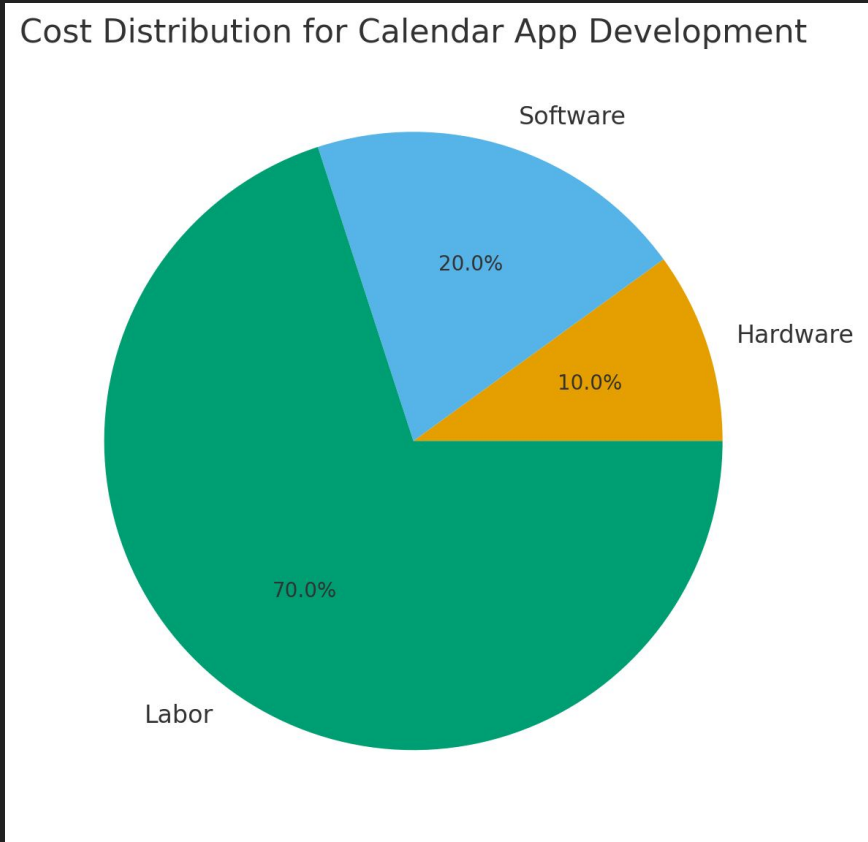
- **Start:** Sept 5, 2025
- **End:** June 26, 2026
- Duration: ~6 months
- Workdays: Mon–Fri, 2 hrs/day
- Weekends excluded for balanced workload



Cost, Effort & Pricing Estimation

Estimation Method: Function Point (FP)

- Identified ~135 FP
- 1 FP ≈ 5 dev hours → **675 total hours**
- Used for cost, effort, and pricing estimation



Testing Plan

Unit to test: `checkTimeConflicts(startTime, endTime, excludeEventId)`

Purpose: Ensure the method correctly detects overlapping events and ignores an event with the `excludeEventId`.

Test Strategy:

Provide an array of mock events.

Checks:

1. **No Conflict:** Verify that the method returns an empty array if the proposed event does not overlap with any existing events.
2. **Conflict Detected:** Verify that overlapping events are returned correctly.
3. **Exclude Event ID:** Verify that a specific event can be excluded from conflict checking.

Test Type:
Unit Test — testing a single logical method independently.

Test Environment:

- Module Programming Language: JavaScript
- Testing Tool:Jest
- IDE: VS Code

Method & Test Code

CheckTimeConflicts Method Code:

```
checkTimeConflicts(startTime, endTime, excludeEventId = null) {
    return this.events.filter(event => {
        if (event.id === excludeEventId) return false;

        const eventStart = new Date(event.startTime);
        const eventEnd = new Date(event.endTime);

        return (startTime < eventEnd && endTime > eventStart);
    });
}
```

Jest Test Code

```
import { CalendarApp } from './CalendarApp.js';

describe('CalendarApp checkTimeConflicts', () => {
  let app;

  beforeEach(() => {
    app = new CalendarApp();
    app.events = [
      { id: '1', startTime: '2025-11-13T10:00:00', endTime: '2025-11-13T11:00:00' },
      { id: '2', startTime: '2025-11-13T12:00:00', endTime: '2025-11-13T13:00:00' }
    ];
  });

  test('no conflict returns empty array', () => {
    const conflicts = app.checkTimeConflicts(
      new Date('2025-11-13T11:00:00'),
      new Date('2025-11-13T12:00:00')
    );
    expect(conflicts).toEqual([]);
  });

  test('overlapping event returns conflict', () => {
    const conflicts = app.checkTimeConflicts(
      new Date('2025-11-13T10:30:00'),
      new Date('2025-11-13T11:30:00')
    );
    expect(conflicts.length).toBe(1);
    expect(conflicts[0].id).toBe('1');
  });

  test('excludeEventId ignores that event', () => {
    const conflicts = app.checkTimeConflicts(
      new Date('2025-11-13T10:30:00'),
      new Date('2025-11-13T11:30:00'),
      '1'
    );
    expect(conflicts).toEqual([]);
  });
});
```

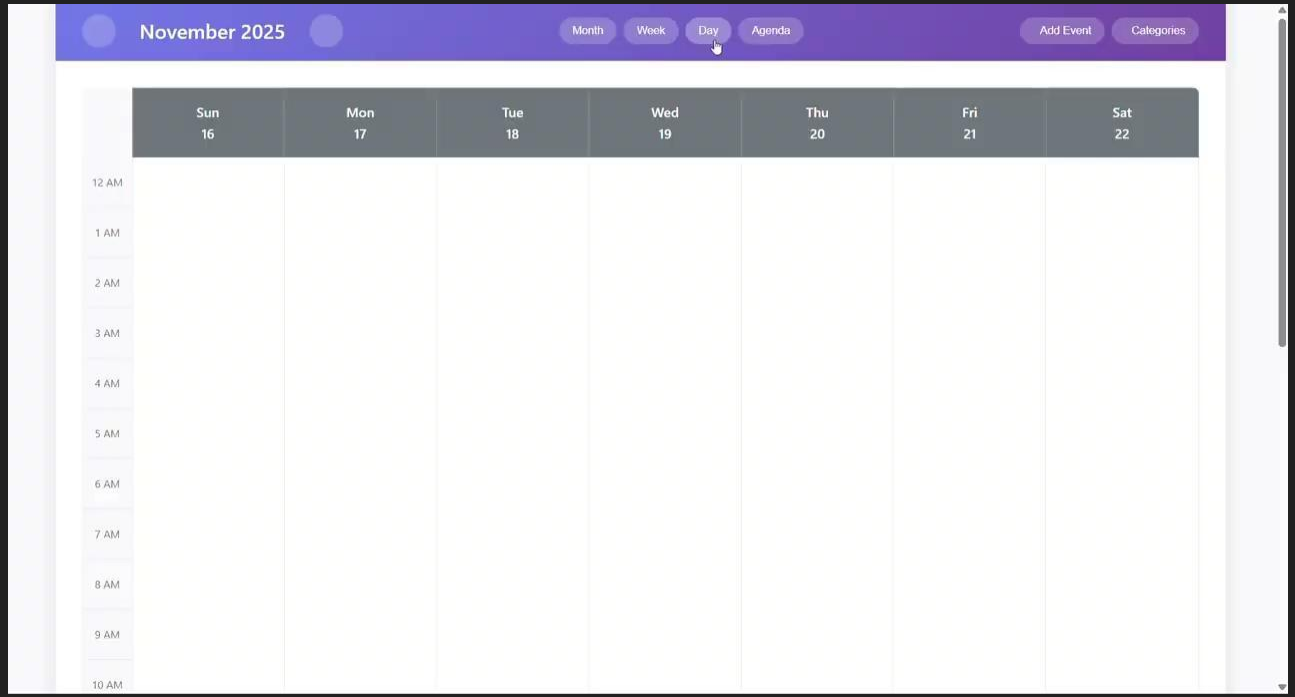
Test Results

Test Case	Input	Expected Result	Actual Result
No conflict	11:00–12:00	[]	[] ✅
Overlap with Event 1	10:30–11:30	[Event 1]	[Event 1] ✅
Exclude Event 1	10:30–11:30, exclude '1'	[]	[] ✅

Comparison with Similar Solutions

Feature	SaaS (Google Calendar)	Open Source	Our System
Custom Academic Workflow	✗	✓	✓
Offline Use	✗	✓	✓
Free to Use	✗	✓	✓
Data Privacy	Limited	High	✓

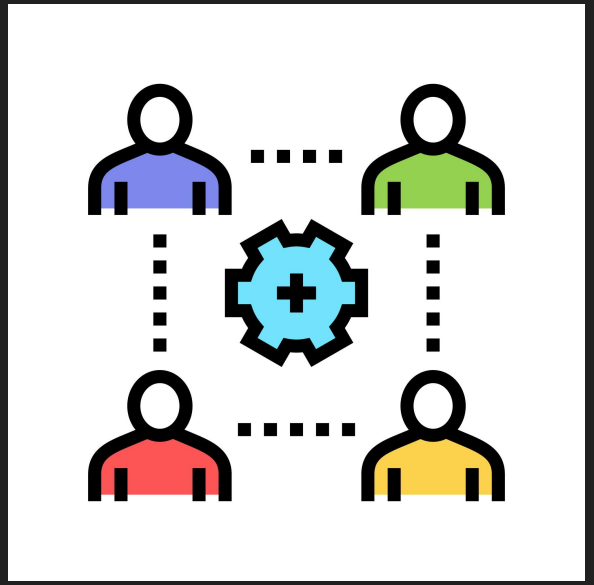
Demo


November 2025
Month
Week
Day
Agenda
Add Event
Categories
Sun 16
Mon 17
Tue 18
Wed 19
Thu 20
Fri 21
Sat 22
12 AM
1 AM
2 AM
3 AM
4 AM
5 AM
6 AM
7 AM
8 AM
9 AM
10 AM

Evaluations & Changes

Main Changes Made:

- Expanded team roles beyond coding
- Updated UML & architecture diagrams
- Switched to Function Point analysis
- Added automated testing



Why:
To meet professor feedback, improve realism, and ensure professional project completeness.

Conclusion

Summary:

- Functional, secure, and efficient calendar for students
- Built with MVC and Incremental model
- Realistic schedule, cost, and testing approach

Next Steps:

- Expand event sync features
- Integrate notifications and shared calendars

References

IEEE Citation Style

[1] M. Watson, “Custom Software vs. Off-the-Shelf Solutions: A Complete Cost-Benefit Analysis for Growing Businesses,” *Full Scale*, Jun. 19, 2025. [Online]. Available: https://fullscale.io/blog/custom-software-vs-off-the-shelf-cost-analysis/

[2] A. Park, “Open-Source Software vs. Proprietary Software: What to Know,” *Heavybit*, Apr. 13, 2023. [Online]. Available: https://www.heavybit.com/library/article/open-source-vs-proprietary

[3] J. Nielsen, “10 Heuristics for User Interface Design,” *Nielsen Norman Group*, Apr. 24, 1994. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/