
Orbit
The Calendar App for College Students
Donald Berthelsen, Cooper Brice, Luis Gutierrez,
Skyler Lee, Kevin Manjit, Joseph Park, Carlos Vazquez

Table of Contents

Objective

College life can be extremely hectic. Here's our idea in solving that problem.

A truth of life is that college students are largely **unorganized**. Even when trying to use traditional calendar apps, many still struggle with managing plans, deadlines, job applications, and more.

Even events and clubs across campus can become **hard to keep track of** as they rely on word-of-mouth or physical pamphlets and posters.

We, the Orbit team, have utilized our software engineering knowledge to design **Orbit**, the calendar app for college students that utilizes next-gen features focused on classes, internships, and more. Instead of using 10-20 different tools to manage their classes and work life, students would now be able to do it all in just **one place**.

Cost Estimation

Function Point (FP) Cost Estimation

1. **Function category count**
   - EI: 10 (event create/edit/delete, login, reminders)
   - EO: 6 (notifications, calendar views)
   - EQ: 4 (search/filter)
   - ILF: 2 (events, users)
   - EIF: 1 (Google Calendar etc.)

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

3. **Compute gross function point (GFP)**

GFP = 10*4 + 6*5 + 4*4 + 2*10 + 1*7

= 113 FP

4. **Determine Processing Complexity (PC)**
Data Communications(1), Distributed Processing(0), Performance(4), Heavily Used Configuration(2), Transaction Rate(2), Online Data Entry(5), End-User Efficiency(4), Complex Processing(2), Reusability(3), Installation Ease(5), Operational Ease(5), Multiple Sites(0), Facilitate Change(4), Security(5)

Sum of scores = 42

Function Point (FP) Cost Estimation (cont.)

5. **Compute Processing Complexity Adjustment**

PCA = 0.65 + 0.01(42)

= 1.07

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

Adjusted FP = GFP * PCA

= 113 * 1.07

≈ 121 FP

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

Final Personnel Cost

Lead developer **($60/hr)**
- 20% of work → 96 hours

Developers (2) **($50/hr)**
- 40% of work → 192 hours

Code Researcher **($45/hr)**
- 10% of work → 48 hours

Designer **($45/hr)**
- 10% of work → 48 hours

Product Managers (2) **($55/hr)**
- 20% of work → 96 hours

Total Personnel Cost =

(96*60) + (192*50) + (48 *45) + (48*45) + (96*55)

= $24,960

Timeline

Project Length

480 hours

480 work hours divided by a 8 hour work day, 5 business days in a week, and 6 workers.

The project can take 2 weeks at the very least.

Considering if the timeline is completed sooner, the project could reduce costs by reducing complexity to achieve a faster timeline and bring a functional product to users sooner.

1. Research and planning

4 day

2. UI/UX Design

6 days

3. Backend Development (DB, API, etc.)

8 days

4. Frontend Development (Views, user events)

6 days

5. Testing & QA

2 days

6. Deploy & Document

2 days

Requirements

Functional Requirements

These are just some of the most representative functional requirements that highlight our app.

We defined high-level statements on what our system services should do before describing our non-functional requirements.

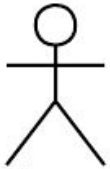
The user shall be able to **create** events. Created events shall **not overlap** with existing events.	UTD clubs' and organizations' owner users' events created shall **automatically appear** for students that are members of these clubs and organizations.	**Holidays and weekends** shall be indicated in the calendar view. Holidays shall differ depending on the region of the user.	**eLearning homework deadlines** shall be automatically added as events in the calendar.

Non-Functional Requirements

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

System Modeling

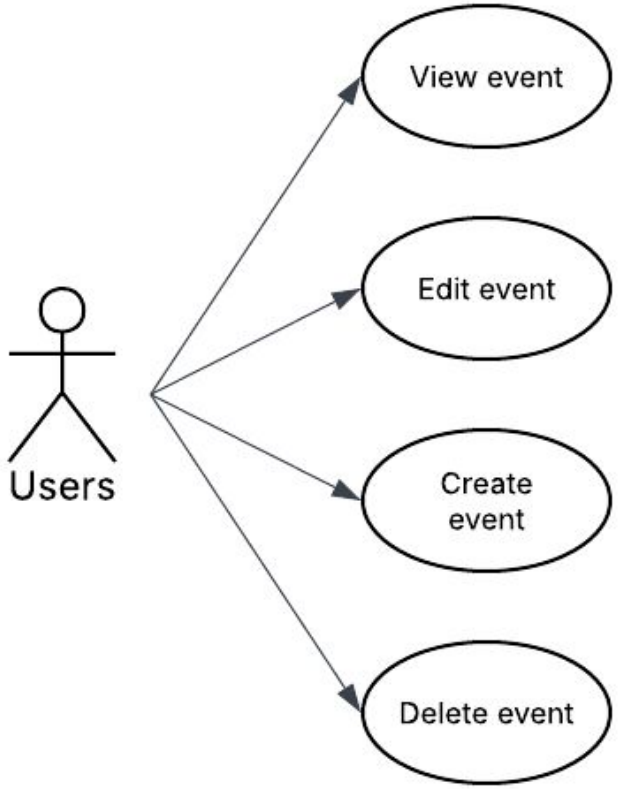
Use Case Diagram



- **“Simple and Intuitive”**

Personal event planning has never been easier.

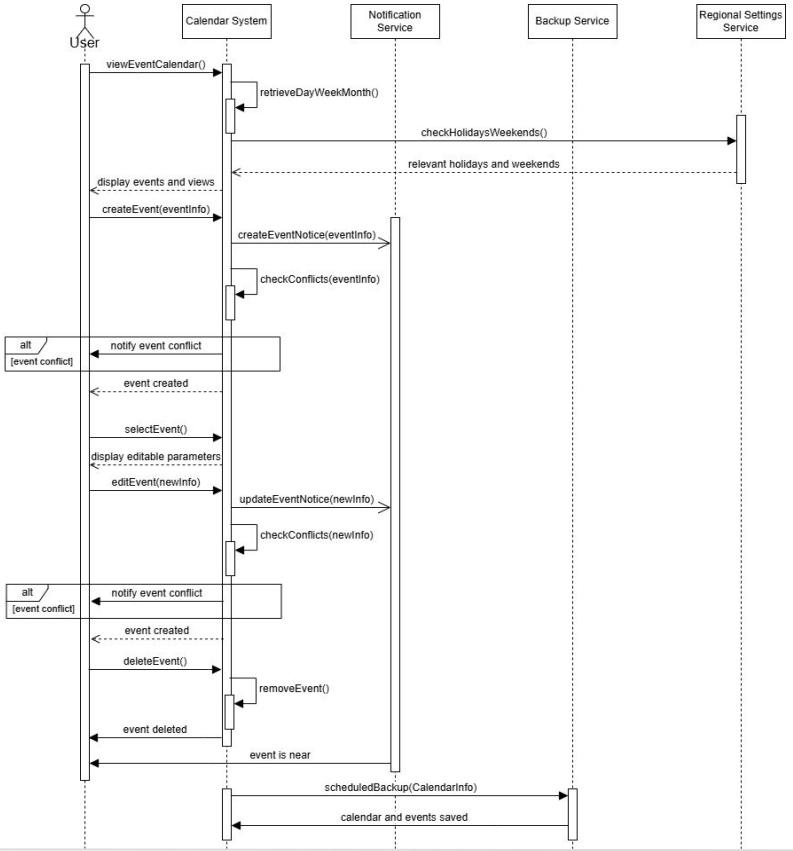
We kept what makes calendar and life organization so reliable and easy to use. We’re just building on top of that!



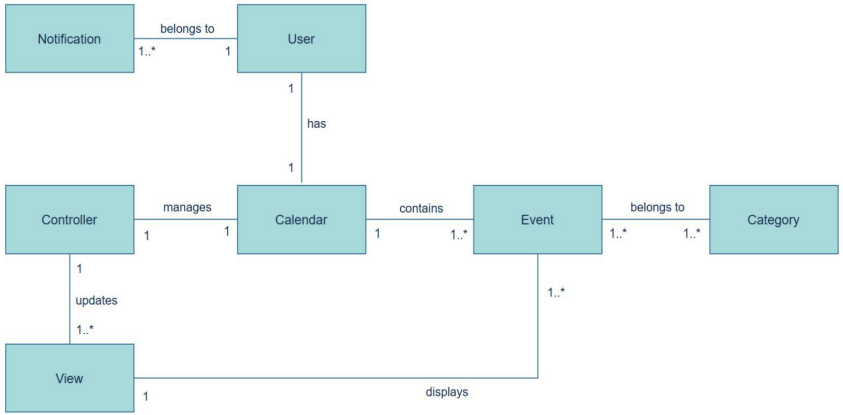
Sequence Diagram

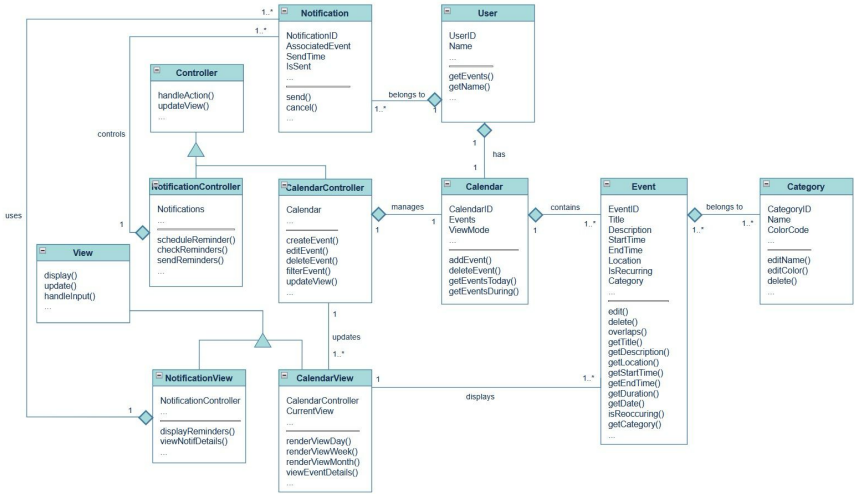
We have outlined all potential events and subsequent alternative interactions for event creation, notification, and deletion between various objects.

We have also provided how our simple yet secure backup service will be available immediately upon request.



Class Diagram



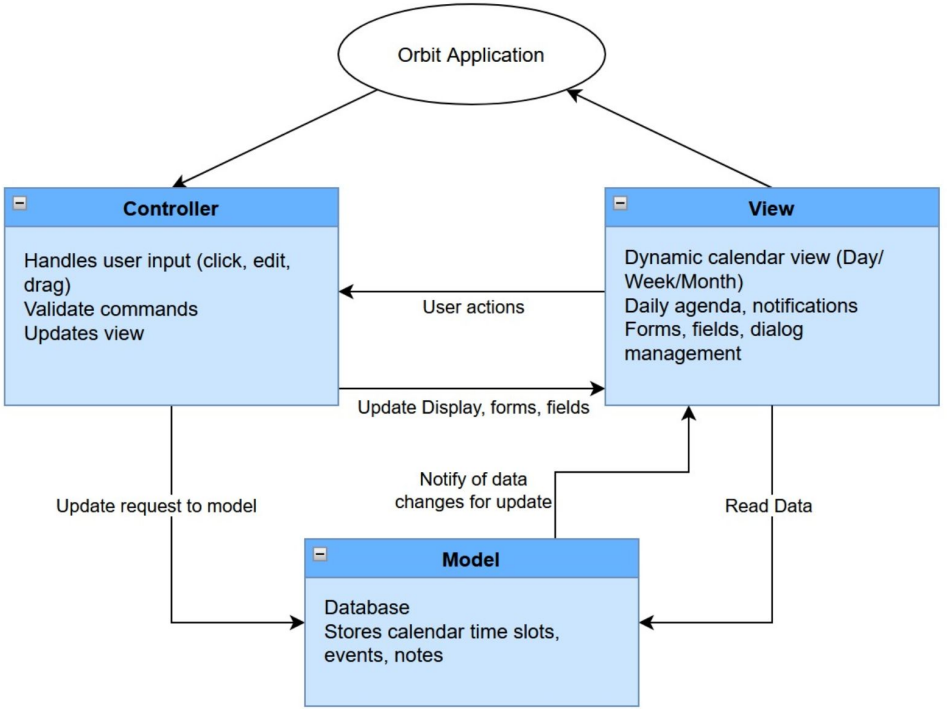


We first started with defining the cardinalities and objects to be implemented.
We then went **above-and-beyond** to construct and present every objects' attributes and methods with more to be potentially defined.

Architectural Design: Model-View Controller

As a team, we decided on this pattern based on our group size and project scale where we can easily separate work between these three different components, allowing us to test and re-iterate more independently and to our strengths.

This model applies well to our project as we can handle changes to the database depending on specific colleges' wishes and our own updates.



Orbit

Got Questions?

Contact us
Donald.Berthelsen@UTDallas.edu
Cooper.Brice@UTDallas.edu
Luis.Gutierrez@UTDallas.edu
Skyler.Lee@UTDallas.edu
Kevin.Manjit@UTDallas.edu
Joseph.Park@UTDallas.edu
Carolos.Vazquez@UTDallas.edu