CS3354 Software Engineering
Final Project Deliverable 2

PlanZenith: Reaching Your Goals

Priyanka Amalkar
Alonso Toji
Karina Batra
Anjana Bharadwaj
Alexis Kaufman
Safa Mohammed
Zac Hays

Task Delegation:

- Priyanka Amalkar
  - Proposal: Daily view, color marking
  - Deliverable 1: Listing non-functional requirements, committing the project scope to the Github repository
  - Deliverable 2: Add deliverable 1 content, project scheduling, submission to eLearning
- Alonso Toji
  - Proposal: Agenda view, blocks of no scheduling
  - Deliverable 1: Explaining our chosen software process model, creating the Github
  - Deliverable 2: Cost/effort/pricing estimation, cost of personnel
- Karina Batra
  - Proposal: Weekly view, holidays/weekends color
  - Deliverable 1: Listing functional software requirements, making + committing the README file
  - Deliverable 2: Conclusion
- Anjana Bharadwaj
  - Proposal - Recurring events, scroll/zoom
  - Deliverable 1: Explaining our chosen architectural design and diagram
  - Deliverable 2: Cost/effort/pricing estimation, cost of hardware
- Alexis Kaufman
  - Proposal: Edit/delete events, event alert
  - Deliverable 1: Creating the sequence diagram
  - Deliverable 2: Comparison with similar designs
- Safa Mohammed
  - Proposal: Monthly view, add event
  - Deliverable 1: Creating the class diagram
  - Deliverable 2: Cost of software, commit everything to GitHub
- Zac Hays
  - Proposal: Check time conflicts, add/delete event categories, task completion
  - Deliverable 1: Creating the use case diagram
  - Deliverable 2: Test plan
- Everyone
  - Proposal: ~
  - Deliverable 1: ~
  - Deliverable 2: References, presentation slides

Project Deliverable 1 Content

Final Project Draft Description

A Calendar Software

1.1 Views

- 1.1.1 Monthly view: show all days in a month, and event snippet for each day
- 1.1.2 Weekly view: show all days in a week, and event snippet for each day
- 1.1.3 Daily view: show all events in a day, sorted by their starting time
- 1.1.4 Agenda view: show all events in future as a list

1.2 Events

- 1.2.1 Add an event with starting and ending time
- 1.2.2 Check time conflicts when adding events
- 1.2.3 Add recurring events
- 1.2.4 Edit & delete events
- 1.2.5 Event alert
- 1.2.6 Add/delete event categories
- 1.2.7 Color marking for different category of events
- 1.2.8 Task completion for events

1.3 Other

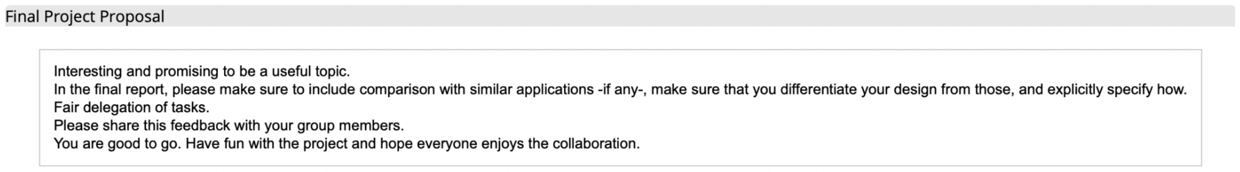
- 1.3.1 Holidays & weekends should be in special colors
- 1.3.2 Zoom in/out, and scroll support when necessary
- 1.3.3 Create blocks where nothing can be scheduled

Detailed Description of Motivation:

We decided to choose this particular project because of the many professional groups it can easily be applied to. A calendar software would be beneficial for students, from high school throughout college, for class scheduling, homework and assignment due dates, and exam times. It can also be used in the professional world: for hospitals that need to schedule appointment times, tech industries that need to schedule meetings and project timelines, as well as educators that need to prepare lesson plans.

Addressing Final Project Proposal Feedback:

In the final project proposal feedback, we were told to "include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how". To go about this, we will first conduct some research on similar applications, then find features that they may be lacking, and integrate those changes into our project.



Final Project Proposal

Interesting and promising to be a useful topic.
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.

Github Repository Link:

https://github.com/AlonsoToji/3354-PlanZenith

Software Process

In the case of our project, the spiral model seemed to make the most sense. The biggest benefit the spiral model offers is its iterative developmental nature and strong customer involvement. Since not all features are needed in the beginning, and there will always be user feedback and recommendations, the requirements will be changing along the way. This will lead to a need for regular updates which either new features or improving the performance of current features. Furthermore, the cyclic nature allows for flexibility to adapt to these changing requirements. The calendar app will also benefit from the spiral model's emphasis on risk management. Being able to handle issues early in the pipeline will make later development much easier.

Software Requirements

Functional Requirements

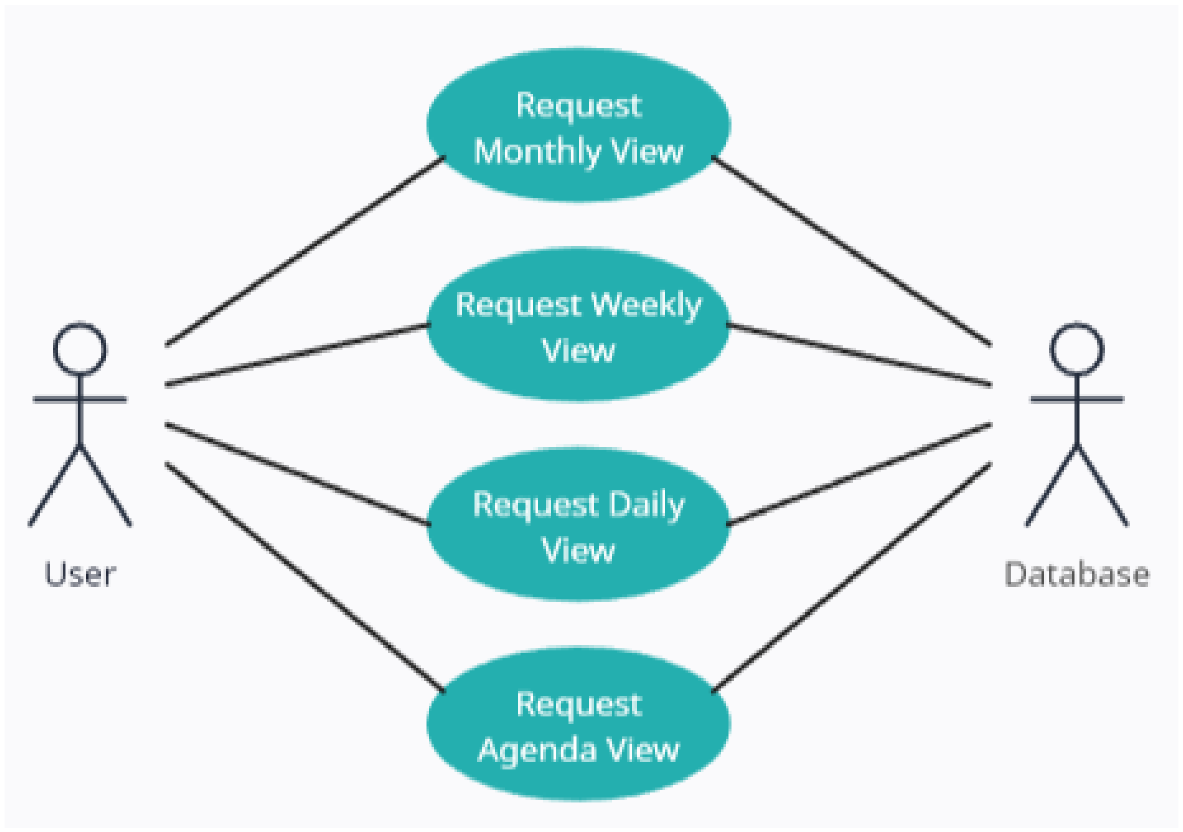
- The calendar software shall display to the user a Monthly view that shows all the days in a month, a Weekly view that displays all the days in a week, a Daily view that displays all events scheduled for a specific day, and an Agenda view that displays all future events as a list.
  - For each day in the Monthly view, the software shall display a snippet of events scheduled for that day.
  - For each day in the Weekly view, the software shall display a snippet of events scheduled for that day.
  - The events in the Daily view shall be sorted by their starting time.
  - The events in the Agenda view shall be listed in chronological order.
- A user of the calendar software should be uniquely identified by a user ID, and events should be associated with the user's profile, thus allowing for individualized event management for the respective user.
- A user should be able to add events to the calendar software, specifying a starting and ending time for each event, edit event details such as title and time, and delete events from the calendar.
- A user should have the ability to create recurring events with different recurrence patterns, including daily, weekly, or custom recurrences.
- The software should check for time conflicts when users attempt to add and/or edit events or blocks and provide an error notification to the user if conflicts are detected between the new event/block and the user's existing calendar.
- A user should have the ability to set event alerts and reminders for scheduled events, specifying the timing and type of notification.
- A user should have the ability to define time intervals during which scheduling of events is restricted (blocked).

Non-Functional Requirements

- Product Requirements - The PlanZenith system will be available to its users at all times due to the calendar system being 24/7.
    - Usability Requirements - This system will be organized simplistically in order for the user to navigate to different features without multiple steps.
    - Efficiency Requirements - This system should be able to store data from multiple events, as many as the user decides, without crashing.
        - Performance Requirements - This system will take 1-2 seconds for response time with strong internet and no more than 10 seconds with weak internet. It will sync automatically every 10-15 minutes.
        - Space Requirements - This system will take up approximately 230MB. A similar software, Google Calendar, takes up about 215MB so we increased our system's space to add more functionalities.
    - Dependability Requirements - This system will successfully alert the user of an event a specified amount of time beforehand, and must not fail to update the calendar with any changes made to events.
    - Security Requirements - This system will send alerts to the user's email if multiple incorrect login attempts take place to hack into the user's calendar.
- Organizational Requirements - The PlanZenith system users will need to log in to their account in order to access their individual calendar.
    - Environmental Requirements - The user can access this system from any environment as long as they have access to the internet.
    - Operational Requirements - This system will be used by users that want to organize their personal or professional lives with a calendar system. They can add/remove different events and set recurring events as well.
    - Development Requirements - This system must be developed in an efficient way, with validation testing at each step in the development process for every feature to ensure that the user does not deal with bugs.
- External Requirements - The PlanZenith system will maintain adherence to laws regarding ethical software development.
    - Regulatory Requirements - This system will comply with regulatory standards that maintain the user's privacy.
    - Ethical Requirements - This system will ensure that all standards and laws are met so that it will be accepted by the user and general public.
    - Legislative Requirements - This system will operate within software development laws such as the US Safe Web Act which protects the user's confidential information.
- Accounting Requirements - This system will be free of charge to all users.
- Safety/Security Requirements - This system will require the user to enter their username and password in order to gain access to their calendar.

Use Case Diagram

View Operations Use Case Diagram



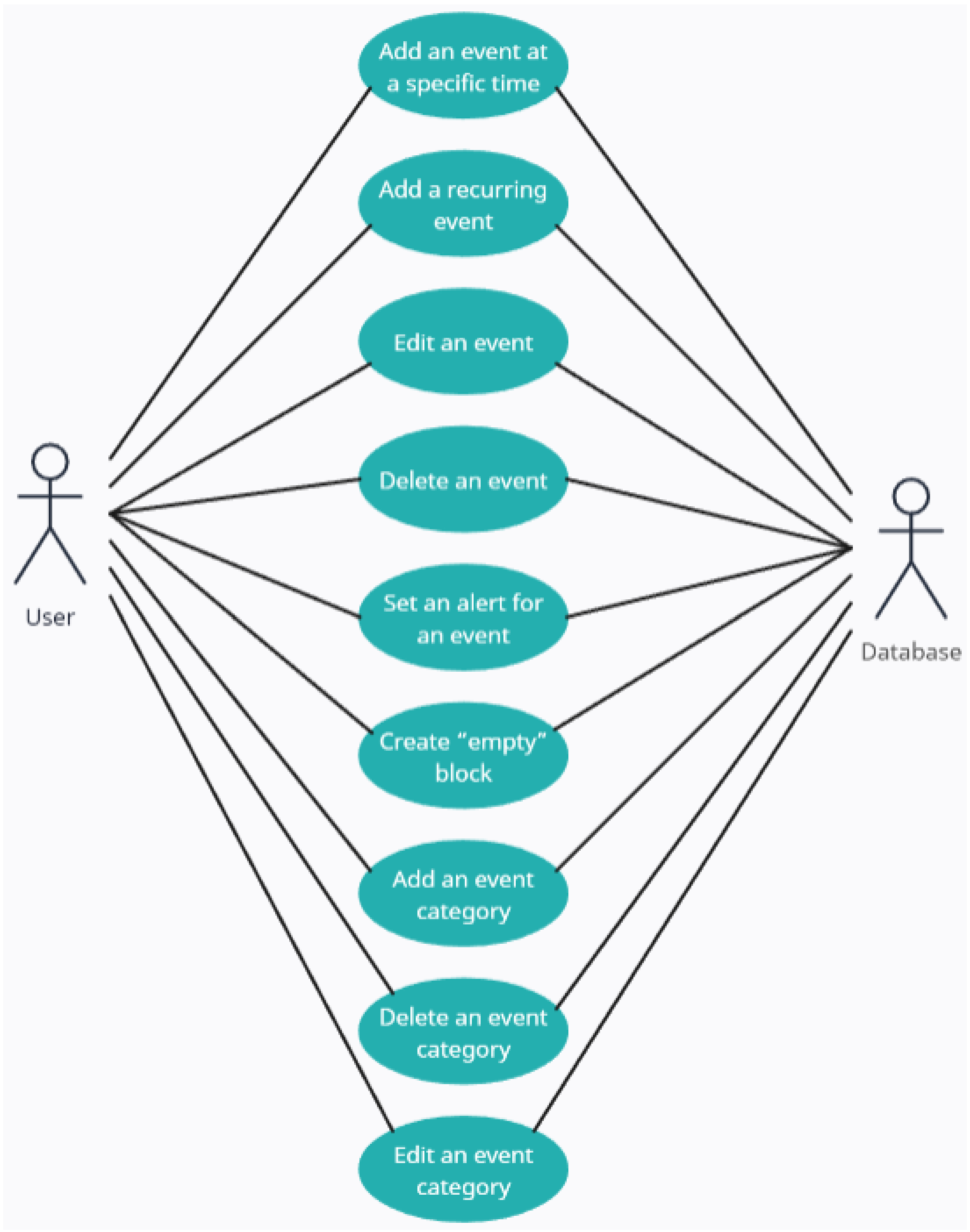
Views: Request Monthly View	
Actors	User, Database
Description	This view shows all the days in the month, with a snippet of the events scheduled for each day.
Data	Events on each day of the month.
Stimulus	Monthly view selected by User.
Response	Monthly view displayed on screen.
Comments	

Views: Request Weekly View	
Actors	User, Database
Description	This view shows all the days in the week, with a snippet of the events scheduled for each day.
Data	Events on each day of the week.
Stimulus	Weekly view selected by User.
Response	Weekly view displayed on screen.
Comments	

Views: Request Daily View	
Actors	User, Database
Description	This view shows all the events scheduled in a day, sorted by starting time.
Data	Events on the day, their starting times.
Stimulus	Daily view selected by User.
Response	Daily view displayed on screen.
Comments	

Views: Request Agenda View	
Actors	User, Database
Description	This view shows all future events, sorted in chronological order.
Data	All future events, their starting dates and times.
Stimulus	Agenda view selected by User.
Response	Agenda view displayed on screen.
Comments	

Event Operations Use Case Diagram



Events: Add event at specific time	
Actors	User, Database
Description	Adds an event to the user’s calendar with a start and end time.
Data	Information about the event (including start time and end time), list of events on the same day and their start and end times.
Stimulus	User issues command.
Response	Confirmation that the event was added or an error message.
Comments	Function checks if there are any time conflicts with the new event and returns an error message if so, otherwise operation succeeds.

Events: Add recurring event	
Actors	User, Database
Description	Adds an event to the user’s calendar that occurs multiple times.
Data	Information about the event (including start time and end time), list of events on the same day(s) and their start and end times, when/how often the event recurs.
Stimulus	User issues command.
Response	Confirmation that the event was added or an error message.
Comments	Function checks if there are any time conflicts with the new event and returns an error message if so, otherwise operation succeeds.

Events: Edit an event	
Actors	User, Database
Description	Changes information about an existing event in the user’s calendar.
Data	Event to be edited, field(s) to be edited, new values for those fields.
Stimulus	User issues command.
Response	Confirmation that the edits were made or an error message.
Comments	If the user changed the start or end time of the event, a function checks for conflicts and responds appropriately.

Events: Delete an event	
Actors	User, Database
Description	Removes an event from the user’s calendar.
Data	The event to be removed.
Stimulus	User issues command.
Response	Confirmation that the event was deleted.
Comments	

Events: Set an alert for an event	
Actors	User, Database
Description	Sets an alert that will notify the user about an event at the specified time.
Data	Event to notify about, timing and type of alert.
Stimulus	User issues command.
Response	Confirmation that the alert has been set.
Comments	

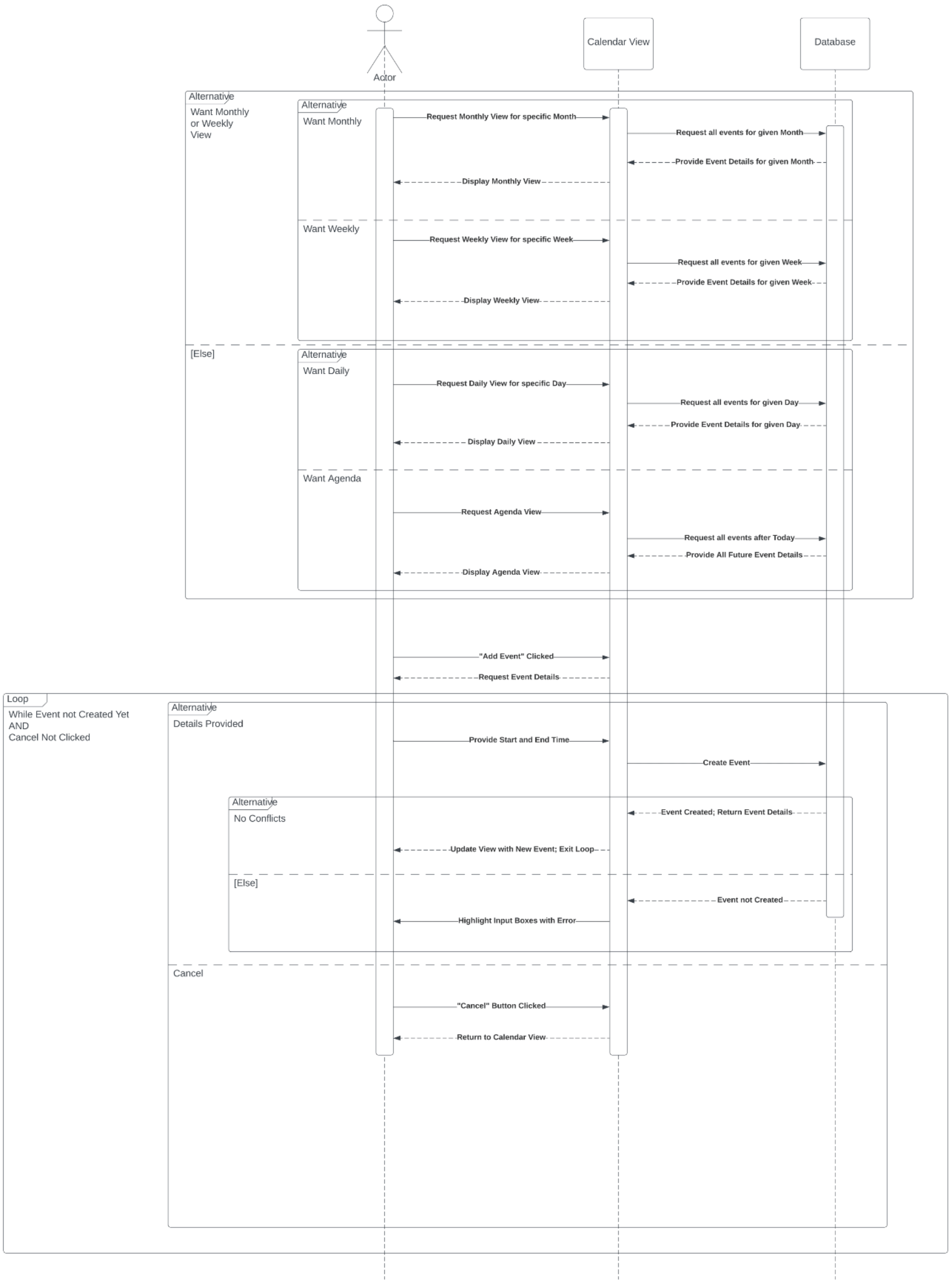
Events: Create “empty” block	
Actors	User, Database
Description	Adds a block to the user’s calendar where nothing can be scheduled.
Data	Start time, end time.
Stimulus	User issues command.
Response	Confirmation that the block was added or an error message.
Comments	Function checks if there are already events scheduled in time period and returns an error message if so, otherwise operation succeeds.

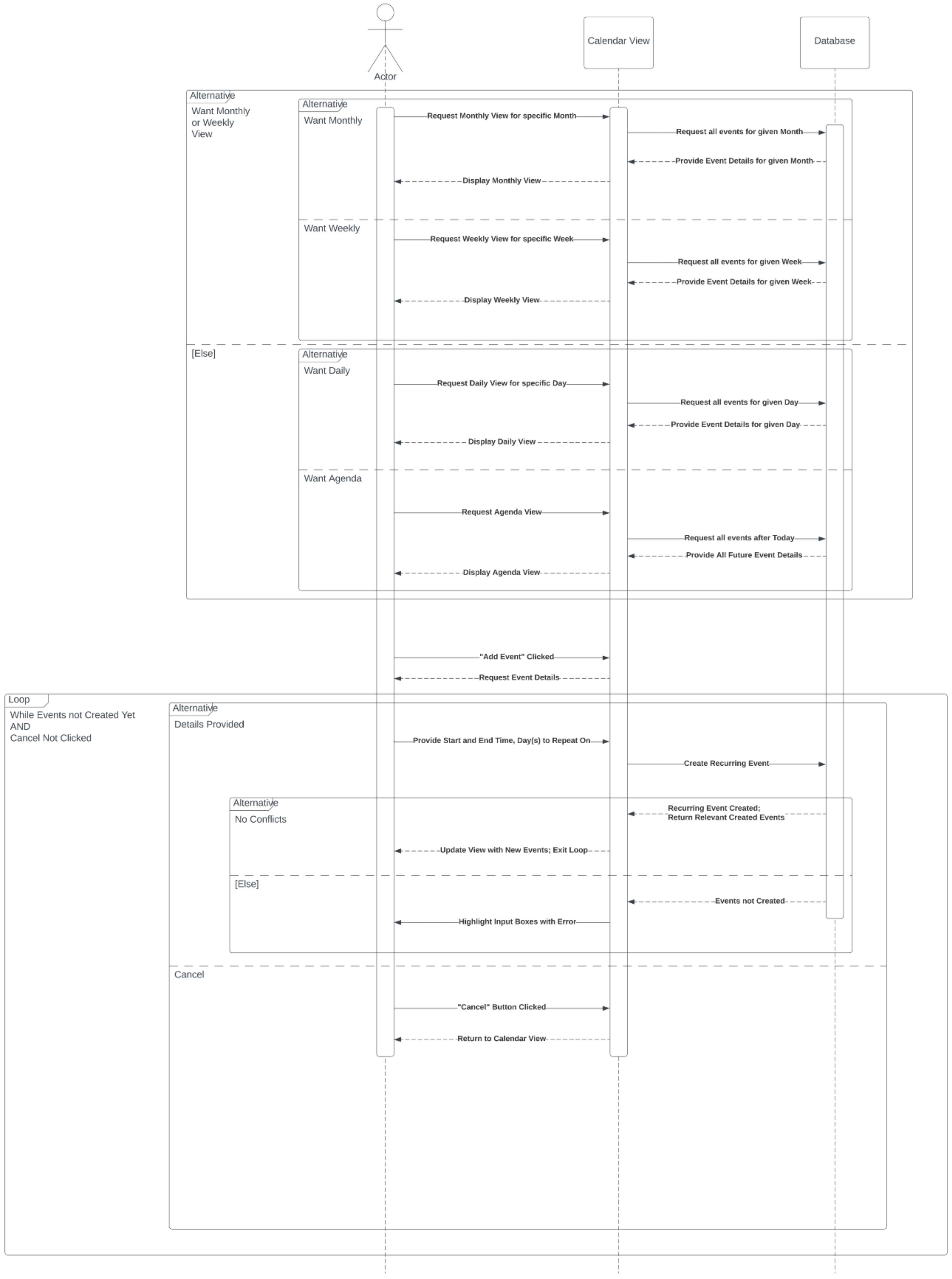
Events: Add an event category	
Actors	User, Database
Description	Add a category into which events may be sorted.
Data	Name, color, events to include.
Stimulus	User issues command.
Response	Confirmation that the category was created.
Comments	

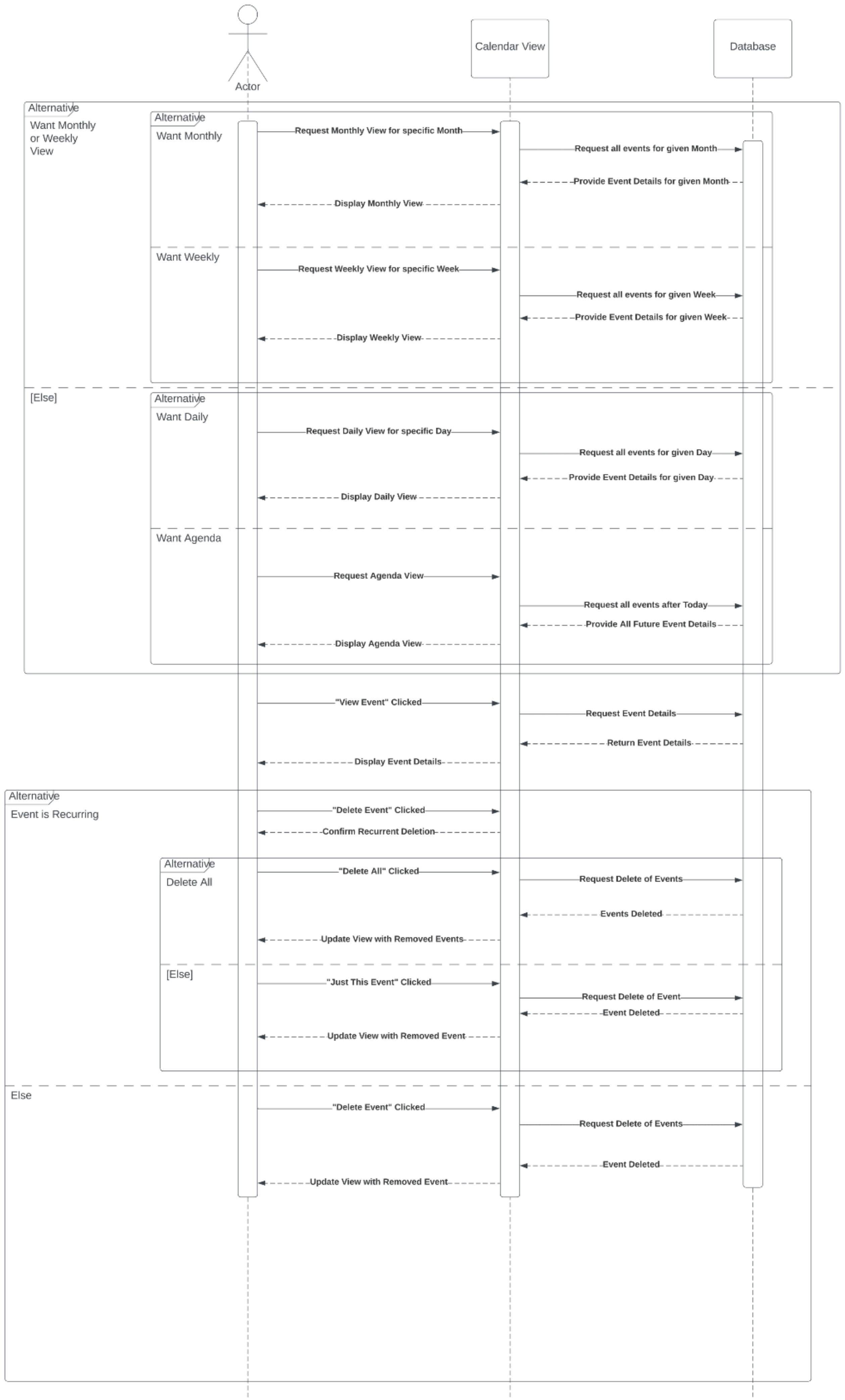
Events: Delete an event category	
Actors	User, Database
Description	Removes a category and sets the events it contained to have no category.
Data	Category to remove.
Stimulus	User issues command.
Response	Confirmation that the category was deleted.
Comments	

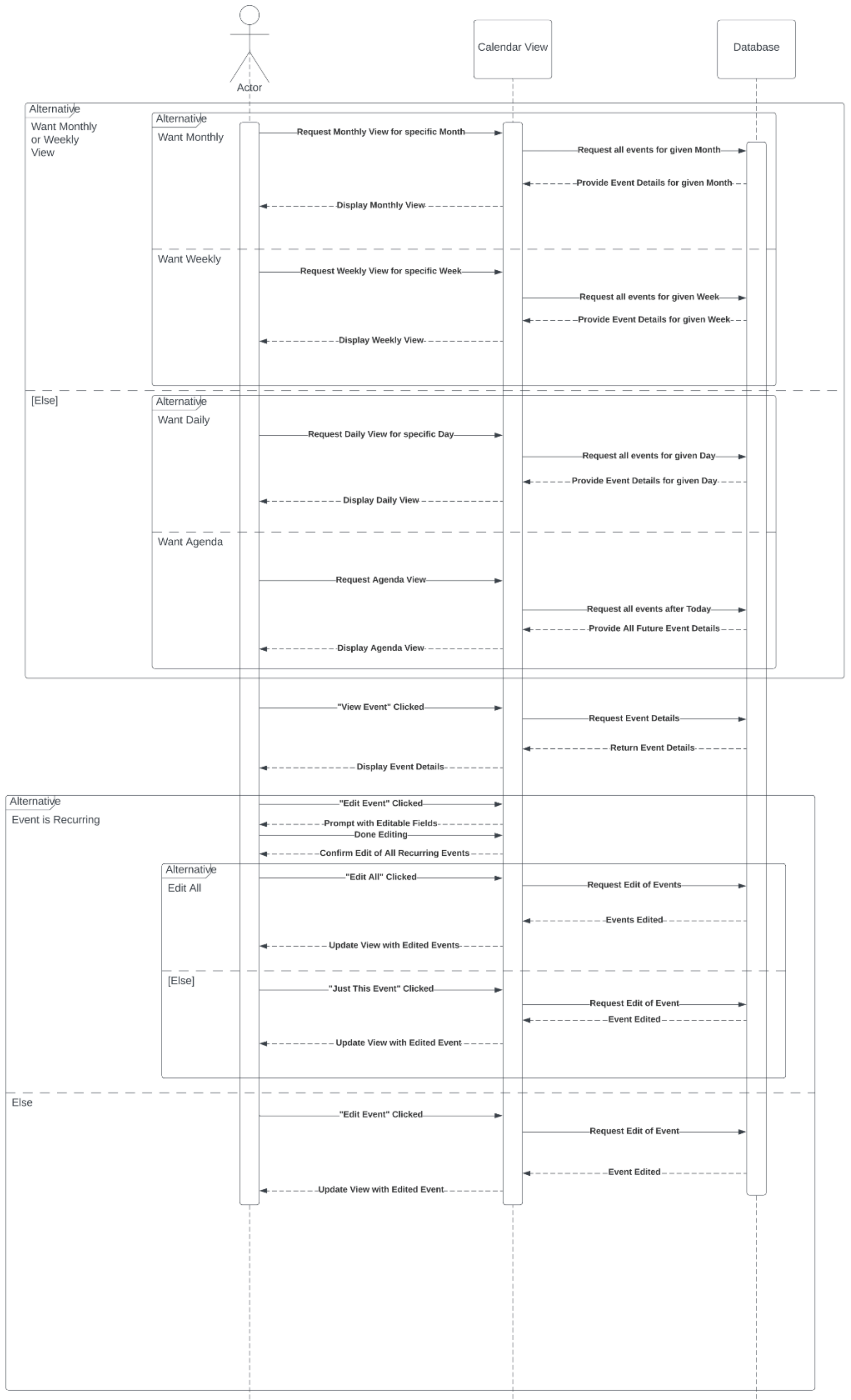
Events: Edit an event category	
Actors	User, Database
Description	Changes information regarding a category. Used for changing name, color, or adding/removing events from the category.
Data	Category to change, field(s) to change, new values for those fields.
Stimulus	User issues command.
Response	Confirmation that the edit was made.
Comments	

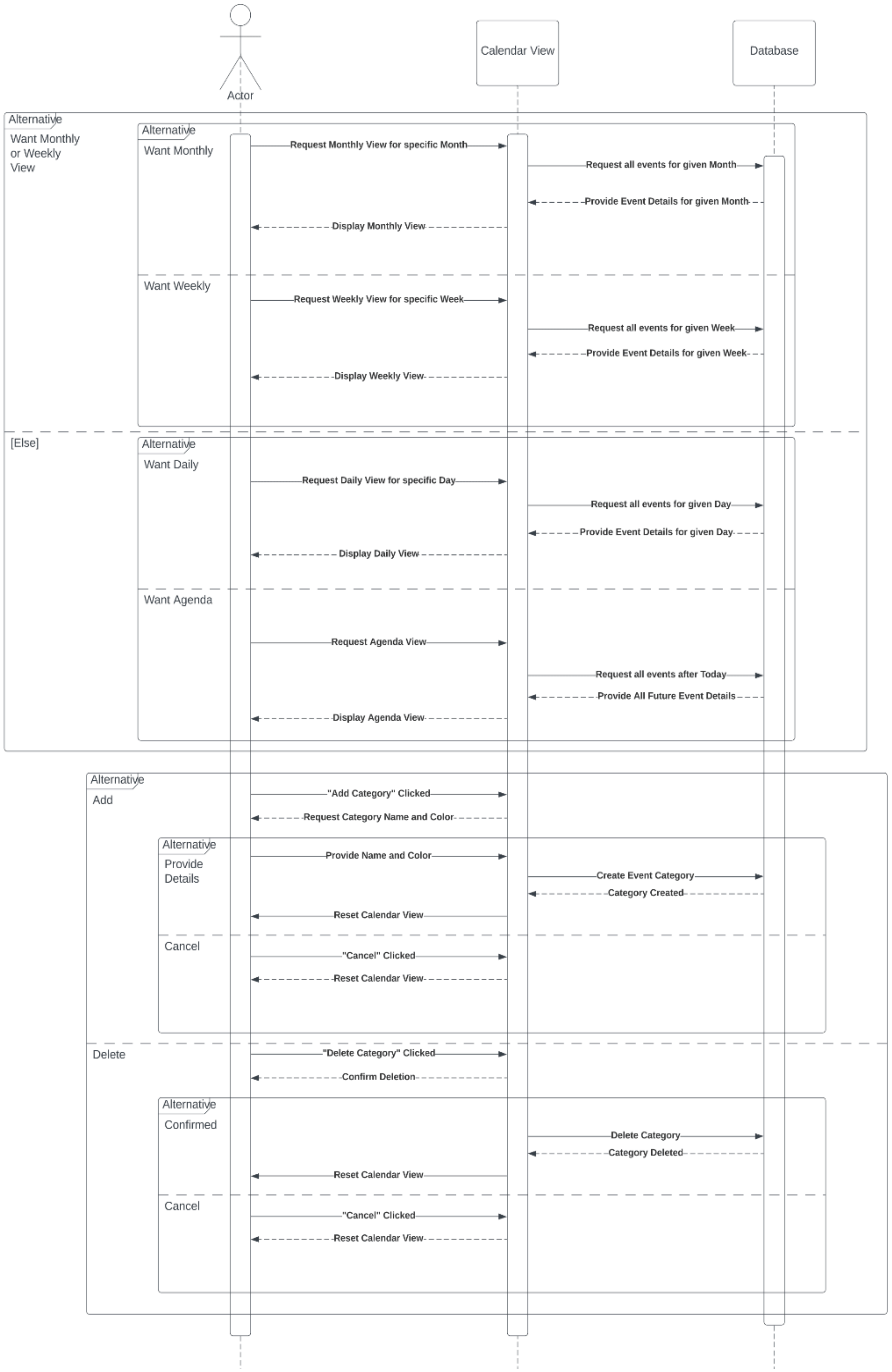
Sequence Diagrams

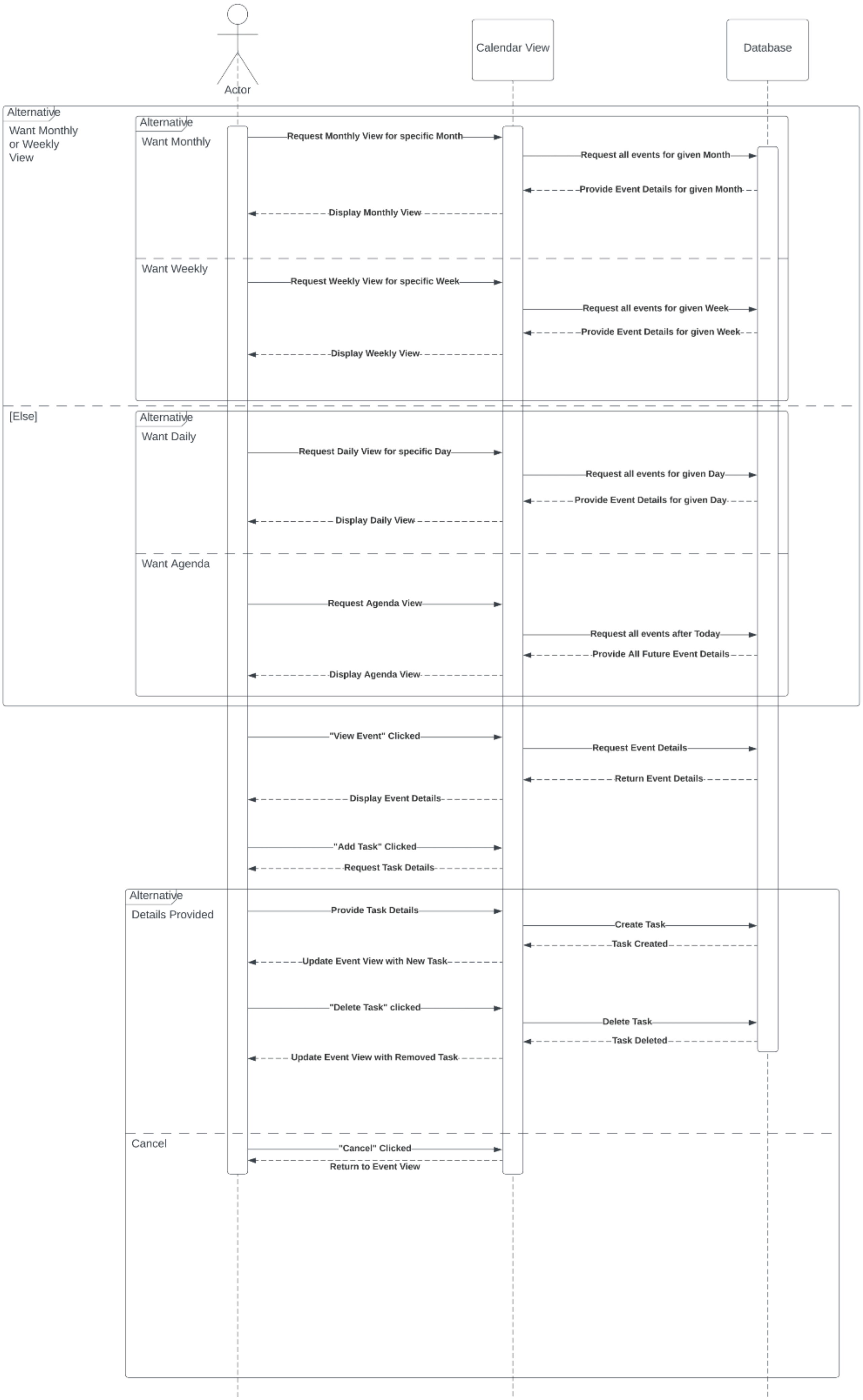

Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"Add Event" Clicked
Request Event Details
Loop
While Event not Created Yet AND Cancel Not Clicked
Alternative
Details Provided
Provide Start and End Time
Create Event
Alternative
No Conflicts
Event Created; Return Event Details
Update View with New Event; Exit Loop
[Else]
Event not Created
Highlight Input Boxes with Error
Cancel
"Cancel" Button Clicked
Return to Calendar View

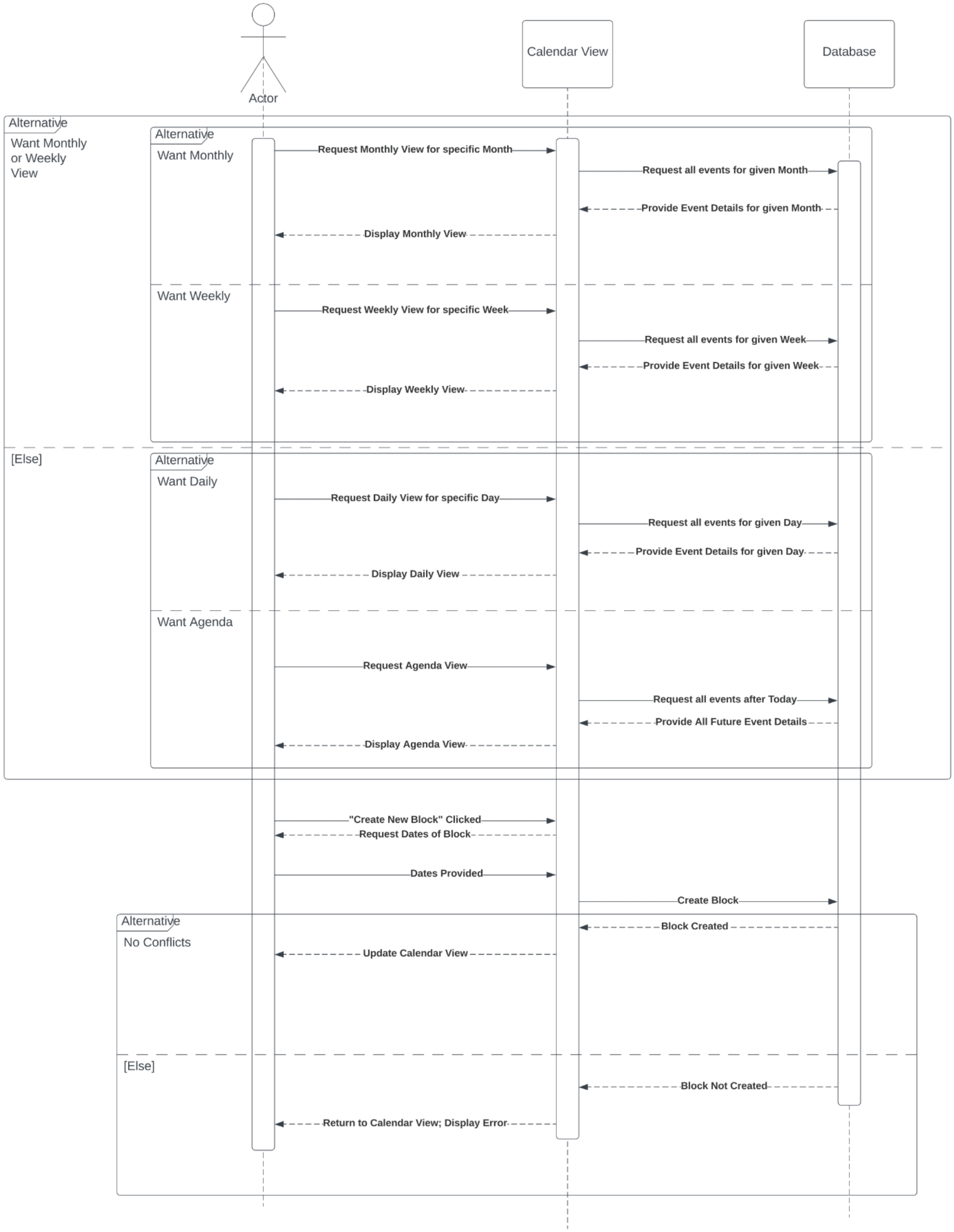

Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"Add Event" Clicked
Request Event Details
Loop
While Events not Created Yet AND Cancel Not Clicked
Alternative
Details Provided
Provide Start and End Time, Day(s) to Repeat On
Create Recurring Event
Alternative
No Conflicts
Recurring Event Created; Return Relevant Created Events
Update View with New Events; Exit Loop
[Else]
Events not Created
Highlight Input Boxes with Error
Cancel
"Cancel" Button Clicked
Return to Calendar View


Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"View Event" Clicked
Request Event Details
Return Event Details
Display Event Details
Alternative
Event is Recurring
"Delete Event" Clicked
Confirm Recurrent Deletion
Alternative
Delete All
"Delete All" Clicked
Request Delete of Events
Events Deleted
Update View with Removed Events
[Else]
"Just This Event" Clicked
Request Delete of Event
Event Deleted
Update View with Removed Event
Else
"Delete Event" Clicked
Request Delete of Events
Event Deleted
Update View with Removed Event

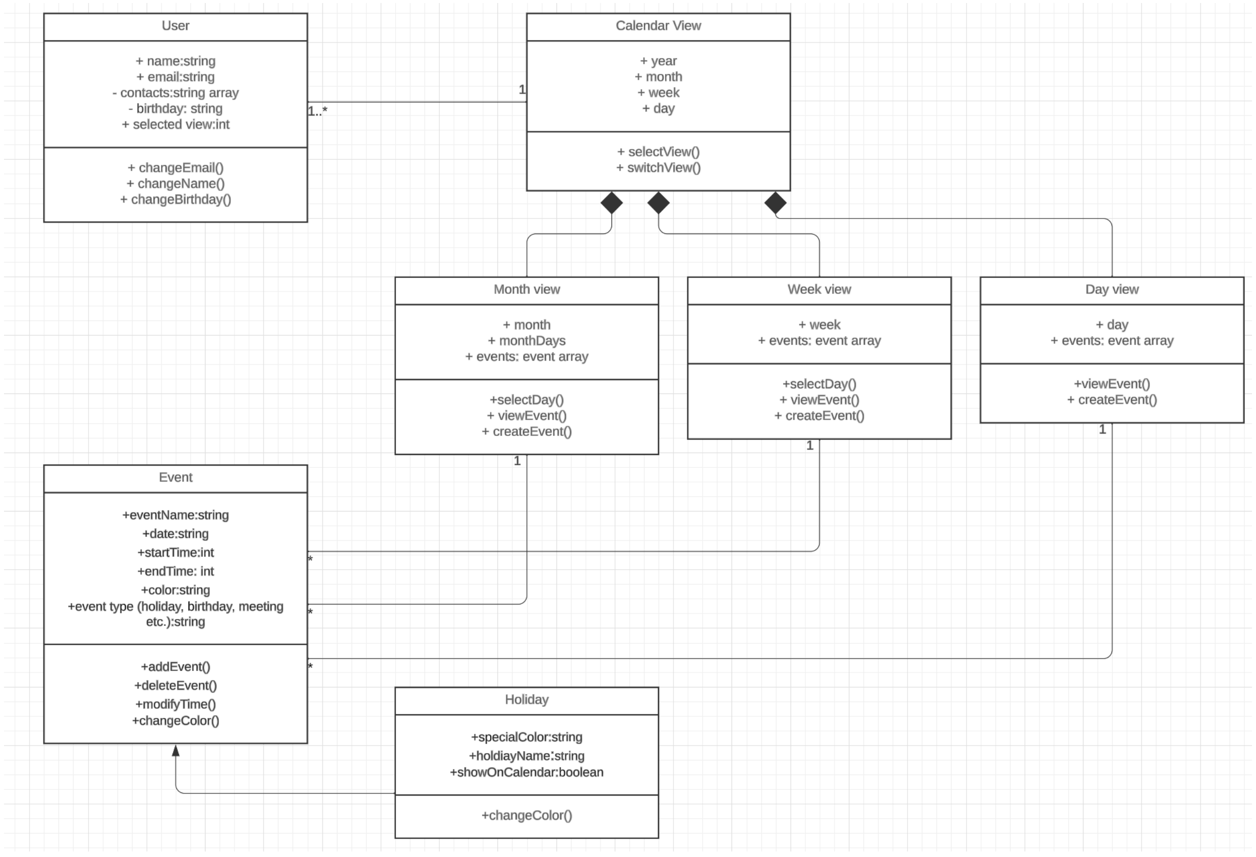

Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"View Event" Clicked
Request Event Details
Return Event Details
Display Event Details
Alternative
Event is Recurring
"Edit Event" Clicked
Prompt with Editable Fields
Done Editing
Confirm Edit of All Recurring Events
Alternative
Edit All
"Edit All" Clicked
Request Edit of Events
Events Edited
Update View with Edited Events
[Else]
"Just This Event" Clicked
Request Edit of Event
Event Edited
Update View with Edited Event
Else
"Edit Event" Clicked
Request Edit of Event
Event Edited
Update View with Edited Event


Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
Alternative
Add
"Add Category" Clicked
Request Category Name and Color
Alternative
Provide Details
Provide Name and Color
Create Event Category
Category Created
Reset Calendar View
Cancel
"Cancel" Clicked
Reset Calendar View
Delete
"Delete Category" Clicked
Confirm Deletion
Alternative
Confirmed
Delete Category
Category Deleted
Reset Calendar View
Cancel
"Cancel" Clicked
Reset Calendar View


Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"View Event" Clicked
Request Event Details
Return Event Details
Display Event Details
"Add Task" Clicked
Request Task Details
Alternative
Details Provided
Provide Task Details
Create Task
Task Created
Update Event View with New Task
"Delete Task" clicked
Delete Task
Task Deleted
Update Event View with Removed Task
Cancel
"Cancel" Clicked
Return to Event View

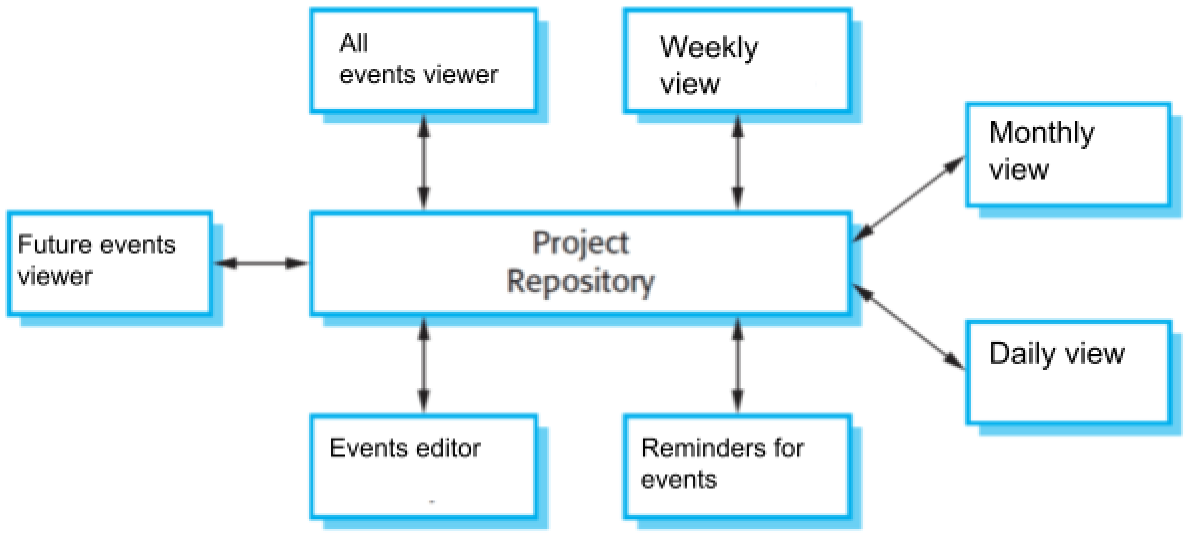

Actor
Calendar View
Database
Alternative
Want Monthly or Weekly View
Alternative
Want Monthly
Request Monthly View for specific Month
Request all events for given Month
Provide Event Details for given Month
Display Monthly View
Want Weekly
Request Weekly View for specific Week
Request all events for given Week
Provide Event Details for given Week
Display Weekly View
[Else]
Alternative
Want Daily
Request Daily View for specific Day
Request all events for given Day
Provide Event Details for given Day
Display Daily View
Want Agenda
Request Agenda View
Request all events after Today
Provide All Future Event Details
Display Agenda View
"Create New Block" Clicked
Request Dates of Block
Dates Provided
Create Block
Alternative
No Conflicts
Block Created
Update Calendar View
[Else]
Block Not Created
Return to Calendar View; Display Error

Class Diagram



Architectural Design

The project's architectural design is the repository style. This architecture is well suited for applications that depend heavily on data, like a calendar or planner app. In this case, the planner will have multiple views, such as day view, week view, and month view, as well as agendas and events. Additionally, the user can add, delete, or edit the events in these views as they like. Since all of the planner's features are dependent on the stored data, having a layer of abstraction between the application and the data source allows simplicity in data retrieval and storage. Additionally, since the data access logic is encapsulated within the repository, that code can be reused multiple times, which is necessary, as the application uses the same in multiple different ways. Additionally, with the repository acting as a buffer, changing the type of database is simplified. If a NoSQL database needs to be switched to a SQL database, then having a repository makes this process much more simple. Overall, the repository architecture fits PlanZenith very well.



Project Deliverable 1 Content

Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and Staffing

Project Scheduling

The scheduling of the project is based on the function point technique used below. If we started our project on Monday January 8th 2024, then it would take us approximately 1 week, or about 5 days, and we would be expected to complete our project on Monday January 15 2024. We will not be counting weekends when scheduling, and we have allotted about 6 hours per day towards working on the project. Since we have seven members working on this project, we do not anticipate it taking more than 1.5 weeks to complete this project. The best way to further detail the schedule would be to use a Gantt chart, which will provide more information into the breakdown of steps, duration for each step, predecessor, and assigned contributor(s).

Cost, Effort and Pricing Estimation.

Assumptions: 7-member team with a productivity of 60 function points per person-week

	Functional Category	Count	Complexity			Count * Complexity
			Simple	Average	Complex	
1	User Input	10	3	4	6	30
2	User Output	6	4	5	7	24
3	User Queries	7	3	4	6	21
4	Data Files & Relational Tables	11	7	10	15	77
5	External Interfaces	2	5	7	10	10
					GFP	**162**

For Determining Processing Complexity: (questions from the slides**)**

1.	5	8.	3
2.	3	9.	2
3.	3	10.	2
4.	4	11.	4
5.	2	12.	2
6.	4	13.	2
7.	2	14.	5

PCA: 0.65 + 0.01[0(0) + 0(1) + 6(2) + 3(3) + 3(4) + 2(5)] = **1.08**

Function Point = GFP * PCA

= 162 * 1.08 = **174.96**

Effort = FP/(productivity in function points per week)   (*round up)
        = 174.96/60 = **2.916 person-weeks rounds to about 3 person-weeks**
Duration = (# of person-weeks)/(# of team members)   (*round up)
        = 3/7 = **0.429 weeks rounds to about 1 week**

Since our application is being built from the ground up, we felt as though FP would make the most sense for making our estimations. As stated in the powerpoint, Application Composition works best for applications that are built on top of other products.

Estimated cost of hardware products (such as servers, etc.)
Many factors need to be considered to determine the cost of the hardware products for PlanZenith. Since this is a data based application, cost effective servers will be needed. For our data, we will be using databases instead of virtual machines or other services. For this, we can use something like AWS or MongoDB. On average, a moderately sized database costs $150 a month. This goes up or down depending on the number of users and scalability questions. We would also need services such as content delivery networks and load balancers which would cost another $50 a month. To ensure there's no data leakage, some sort of security certificate and licensing should be used. This can be expensive, around $100 a month. While this is not an exhaustive list, this brings the total to $300 a month for hardware costs.

Estimated cost of software products (such as licensed software, etc.)
Software licenses are important because they establish the rights and legal framework for interaction for everyone involved. The cost of a software license varies based on the model chosen. For our project, a pay-per-use license model would be the best option because it provides us and users with flexibility. After all, you would pay only for your usage. We would probably host our application on the cloud such as AWS or Google Cloud which employ the same pay-per-use model. To host our application on Google Cloud, we receive around $300 in free credits to run, test, and deploy workloads. After these credits are used up, the cost comes to around $0.204 per GB/month for SSD storage capacity in Cloud SQL.

Estimated cost of personnel (number of people to code the end product, training cost after installation): For development, we are planning on a team of 7 members. For the maintenance of the app, we plan to have 2 developers. Assuming each is paid $50 an hour and 1 week of training, we estimate approximately $4,000 for training. In addition, to maintain the software for a month, assuming these developers only need about 10 hours a week total for maintenance, monthly maintenance should be close to $2,000, given they are both fully trained on the software and the 10 hours per week are distributed between both developers.

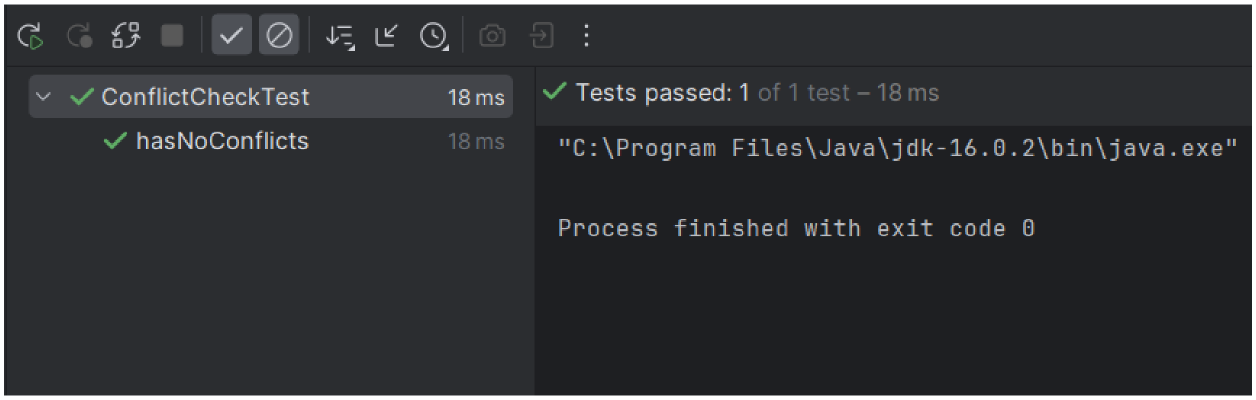
Software Test Plan

The development team for PlanZenith aims to create a well-crafted, polished product. In order for this to happen, the program must undergo rigorous development testing. As an example of this testing process, we will take a deeper look at the 'unit testing' stage for module 1.2.2, 'Check for time conflicts when adding events.' The goal is to ensure that the specified method is functional, by using a variety of different test cases.

The purpose of this method is to ensure that a new event added does not occur at the same time as any of the pre-existing events in the user's calendar. For the purpose of testing the module, we will have hard-coded arrays of events, and their respective start and end times. In the real product, these values would be retrieved from the repository. The method will receive a new event, with its start and end time, and determine whether there are any conflicts.

In order to accurately test this module, three unique test cases were used. In Test Case 1, there were no conflicts with the event being added, and so the method should return an array with only null values. In Test Case 2, there was a single conflict, so the array should contain a single valid index. In Test Case 3, there were multiple conflicts, so there should be a corresponding amount of indices in the returned array. By having a variety of test cases, we ensure that any issue that arises in the module can be found and addressed. This module passed all three test cases.

Here is the result of running the test class on the example code:



```
✓ Tests passed: 1 of 1 test – 18 ms
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe"

Process finished with exit code 0
```

The module's code and the test class are included as separate files in the project zip.

Comparison with Similar Designs

I - Introduction of Similar Designs

The similar designs analyzed in this part include “Google Calendar”, and “Samsung Calendar”. These were chosen because they are widely used by many; More specifically, Samsung’s Calendar is used most frequently simply because it is natively installed on many mobile devices. Google Calendar provides a very nice web interface for calendars, and all of these software applications allow for synchronization with each other. For example, events, reminders, and accounts can be synchronized on Samsung and Google, and vice-versa.

II - Analysis of “Views”

As observed in Cheryl Vaugn’s article [1] entitled “Samsung Calendar vs. Google Calendar vs. Outlook: Which Android Calendar App Should You Use?”, Google Calendar offers the display of events in the following options: “Day”, “Week”, “Month”, “Year”, “Schedule”, and “4 days”. Their “Schedule” view is very similar to our “Agenda” view in that it shows all events as a list; it defaults to starting at the current day, showing a line where the current time is, but also displaying earlier events of the current day. Our “Agenda” view is different in that it only shows FUTURE events in this manner. Additionally, Google Calendar offers a Yearly view, which PlanZenith does not. The other views, “Day”, “Week”, and “Month” share the same functionality as PlanZenith. There is also an extra button at the top left, which allows a shortcut to select the current day; it simply says “Today”.

Also observed in Vaugn’s article [1], Samsung’s Calendar, on the mobile edition, has the following 5 options: “Day”, “Week”, “Month”, “Year”, and “Reminder”. The “Reminder” option can be considered an external view or external resource because it redirects to Samsung’s Reminder application. Samsung Calendar does not offer an “Agenda” view like PlanZenith, but it does offer the “Year” view, which PlanZenith does not offer. The other views, “Day”, “Week”, and “Month” share the same functionality as PlanZenith. It is also worth mentioning that Samsung Calendar defaults to a monthly view, and the “Day” view can be accessed by clicking on a day in the currently displayed month. There is also an extra button, at the top right, which allows a shortcut to select the current day; it is a square with the digit of the current date inside of it, such as “13” to represent November 13th, if that is the current date.

III - Analysis of “Events”

Within Samsung Calendar, since it is a mobile application, users have the ability to get alerts on their phones. These alerts have options of priority, namely “Low”, “Medium”, and “High” which can be chosen from a GUI. Our application does not have these options when getting alerts, as it does not provide support for an external interface, nor does it use the system’s alert functions. As of now, the alerts for PlanZenith are designed simply to pop up within the main view of the app.

For the design of start and end times, it can be seen in “A Complete Guide to Samsung Calendar.” [2] from *Calendar* that Samsung Calendar offers a toggle between “Time” and “All day”, such that when “Time” is selected, users can enter the start/end times, while if they switch the mode to “All day”, those inputs disappear.

This is different from our flow, as PlanZenith has “All Day” represented as a single on/off switch, which when toggled hides the start/end time fields. Samsung Calendar also allows for more specific, pre-determined interfaces to be added to events such as “Location”, “Time zone”, and “Attach file”, while PlanZenith offers a “Description” field, event category selection, and task completion information on an event. Another thing worth noting is that Samsung Calendar does not create any warnings when adding overlapping events to a calendar, nor does it stop the user from doing so. This may be good behavior for a mobile app, where user input should be minimal, compared to accessing a calendar from a webpage via a laptop or desktop, since these offer more room on the screen for such warnings. PlanZenith does have a feature to check time conflicts when adding events, in the form of a pop-up displaying the conflicting event, along with the two options “Edit Event” and “Ignore”.

Within Google Calendar, “priority alerts” are not offered, similar to PlanZenith; all alerts are treated equally. Within the interface of new event creation, there are many fields that PlanZenith does not yet offer, such as “Add attachment”, “Add location”, “Add video conferencing”, and “Add people”. It shares the functionality with PlanZenith to add a description to an event, as well as add alerts; However, Google Calendar also allows multiple alerts to be added, with specific times such as “30 minutes before”. Like Samsung Calendar, overlapping events can be created without any warnings from the application or website, and the event is just added as normal.

IV - Analysis of Extra Features

PlanZenith's design includes a feature where holidays and weekends are in special colors. Google Calendar does not change the color of events or views on the weekend blocks, but it does create a separate calendar for holidays, and all events in this calendar share the unique color of that calendar. This means that the “Holidays” calendar can be turned on/off as well, to hide all holidays from the calendar. PlanZenith does not yet have this feature. Samsung Calendar does not natively display holidays on its calendar, though these can be added manually or imported through other calendar applications like Google Calendar. It also does not color weekends differently from any other block.

PlanZenith has a feature to zoom in and out and to scroll where applicable. When a pinching, click, or zoom interaction is made on PlanZenith, such as in the Agenda View, the event in which the cursor is hovering will be opened in a closer view. The opposite will occur if a zoom-out interaction is made, such that the application will return to the most recently seen overall view. PlanZenith also has scroll support for the Agenda view, which would occur when there are more events than can fit on the page at the time. Google Calendar does not appear to have scroll support within events, but a scroll motion up will go to the previous month, while a scroll motion down will go to the next month. This may be a desired behavior in the future for PlanZenith but is not currently implemented. Samsung Calendar has scroll support within their “Day” view, such that when an overflow of events happens like PlanZenith, the user can scroll to see more. It does not have zoom support in the form of pinch motions, but clicking on a block has the same effect as zoom in, which is that it brings up a more detailed view.

Finally, PlanZenith allows users to create blocks where nothing can be scheduled. It allows a start and end time, start and end date, and notes to be added when creating a block. Google Calendar does not offer this feature, nor does Samsung Calendar.

Conclusion

PlanZenith is a calendar application that includes features such as multiple calendar views, different event interactions, task management, and more. PlanZenith has incorporated use case diagrams, sequence diagrams, a class diagram, cost estimations, a software test plan, etc. Many of the decisions for PlanZenith including the software architecture pattern and project scheduling were collaborative efforts. PlanZenith also has a Figma UI which demonstrates all of the features offered by this application.

A change this project faced occurred when deciding upon the software architecture pattern being used. Initially, the MVC software architecture made the most sense because PlanZenith is web-based. However, the Repository architecture pattern was ultimately chosen because of PlanZenith's strong dependency on data. This transition ensured more efficient data handling, aligning seamlessly with PlanZenith's reliance on data integrity and accessibility.

Another change required was differentiating PlanZenith from other calendar software competitors. This required PlanZenith to emphasize features such as time blocking, weekend-specific coloring, and the zoom-in/zoom-out features to be unique. These distinctive features were needed to create a unique identity for PlanZenith in a crowded market, enhancing its appeal for users seeking other calendar solutions.

Despite these minor changes, the overarching vision for PlanZenith remained intact, ensuring a consistent trajectory toward a comprehensive calendar application.

In summary, PlanZenith is a quality calendar application. An improvement that could have been made to PlanZenith in this stage would be implementation through code. Future plans for PlanZenith include more fields such as location and attachments during new event creation, machine learning for insights, and collaborative features so users can share calendars.

References

[1] C. Vaughn, “Samsung Calendar vs. Google Calendar vs. Outlook: Which Android Calendar App should you use?,” MUO, http://www.makeuseof.com/google-calendar-vs-samsung-calendar-vs-outlook-android (accessed Nov. 16, 2023).

[2] “A Complete Guide to Samsung Calendar,” Calendar, http://www.calendar.com/samsung-calendar/ (accessed Nov. 16, 2023).

Presentation

SE Project Presentation

https://docs.google.com/presentation/d/1BDSz0mgluOW0hcrwltoMfpfGW5-xKil972nJSV14ylU/edit?usp=sharing