CS3354 Software Engineering

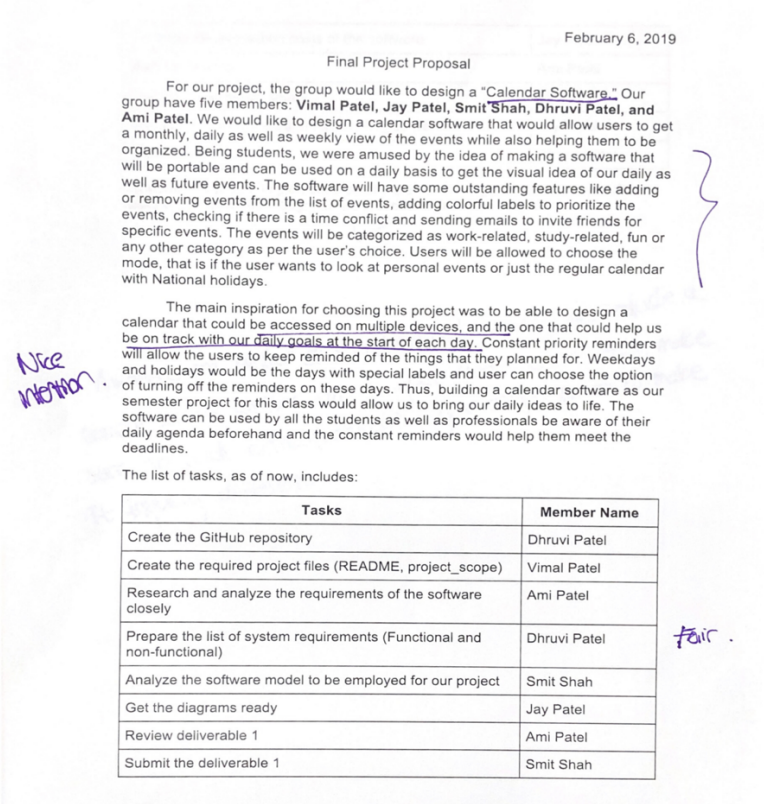
Final Project Deliverable 1

Calendar Software

Ami Patel
Dhruvi Patel
Jay Patel
Vimal Patel
Smit Shah

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 are did to comply with the proposed changes/requests for additions to your project.

- Feedback provided by the instructor on our proposal is that "As there are similar software implemented already, please include a comparison with similar software in your proposal report. Also, make sure to add extra feature(s) to your design so as to make it uniquely different."
- As per the feedback provided by the instructor, we choose a very re-known application to compare with our final project calendar software, Google Calendar. While comparing with google calendar, we came to a conclusion that although the google calendar is so advanced and complex with versatile features, it still lacks addition of event labels or notes in natural languages by users. Thus, to make our calendar software distinctly competitive and unique, we chose to add a peculiar feature for every event called "Language Picker".
- We are adding multiple languages to the calendar so that user can use the calendar in their own native languages such as Spanish, German, French, Chinese, Gujarati and Hindi.
- Hence, we made successful changes and added "Language Picker" feature to our final project.



February 6, 2019

Final Project Proposal

For our project, the group would like to design a "Calendar Software." Our group have five members: **Vimal Patel, Jay Patel, Smit Shah, Dhruvi Patel, and Ami Patel**. We would like to design a calendar software that would allow users to get a monthly, daily as well as weekly view of the events while also helping them to be organized. Being students, we were amused by the idea of making a software that will be portable and can be used on a daily basis to get the visual idea of our daily as well as future events. The software will have some outstanding features like adding or removing events from the list of events, adding colorful labels to prioritize the events, checking if there is a time conflict and sending emails to invite friends for specific events. The events will be categorized as work-related, study-related, fun or any other category as per the user's choice. Users will be allowed to choose the mode, that is if the user wants to look at personal events or just the regular calendar with National holidays.

The main inspiration for choosing this project was to be able to design a calendar that could be accessed on multiple devices, and the one that could help us be on track with our daily goals at the start of each day. Constant priority reminders will allow the users to keep reminded of the things that they planned for. Weekdays and holidays would be the days with special labels and user can choose the option of turning off the reminders on these days. Thus, building a calendar software as our semester project for this class would allow us to bring our daily ideas to life. The software can be used by all the students as well as professionals be aware of their daily agenda beforehand and the constant reminders would help them meet the deadlines.

Nice motivation.

The list of tasks, as of now, includes:

Tasks	Member Name
Create the GitHub repository	Dhruvi Patel
Create the required project files (README, project_scope)	Vimal Patel
Research and analyze the requirements of the software closely	Ami Patel
Prepare the list of system requirements (Functional and non-functional)	Dhruvi Patel
Analyze the software model to be employed for our project	Smit Shah
Get the diagrams ready	Jay Patel
Review deliverable 1	Ami Patel
Submit the deliverable 1	Smit Shah

Fair.

Compute the estimation costs of the software	Jay Patel
Start the testing	Ami Patel
Edit the presentation slides	Vimal Patel
Evaluate the software before submitting deliverable 2	Jay Patel
Submit deliverable 2	Dhruvi Patel

However, the proper implementation of the software might require some additional tasks which would be thought as we proceed towards the project.

✓

As there are similar sfw implemented already, pls include a comparison w/ similar sfw in your final report. Also, make sure to add extra feature(s) to your design so as to make it uniquely different.

1. [10 POINTS] Setting up a GitHub repository:

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.
The TA's GitHub account info is as follows:
TA GitHub id: OmeedUTD
TA email: oea170001@utdallas.edu
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.

Important Note:
· Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities.
· Do not include credentials (e.g., UTD ID) in the repository.
· Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work!

URL: https://github.com/patelami3431/3354-SoftwareGeeks

2. [5 POINTS] Delegation of tasks: Who is doing what

Tasks	Member Name
Creating the GitHub repository	Ami Patel
Creating the required project files (README.md)	Dhruvi Patel, Ami Patel
Creating the required project files (project_scope.md)	Jay Patel
Researching and analyzing the requirements of the software closely, and preparing the list of system requirements (Functional and non-functional)	Smit Shah (Functional) Vimal Patel (Non Functional)
Analyzing the software model to be employed for our project	Dhruvi Patel
Getting the Use-case diagram ready	Jay Patel
Getting the Sequence diagrams ready	Ami Patel
Getting the Class diagram ready	Dhruvi Patel
Analyzing the software and choosing the architectural design	Vimal Patel
Reviewing project deliverable 1	Ami Patel
Submitting project deliverable 1	Smit Shah

3.[5 POINTS] Which software process model is employed in the project and why. (Ch 2)

After analyzing all software process models, our group has finalized **incremental process model** for our project. This is mainly because this specific model involves making of “increments” as time goes by. The calendar software can require addition of new and trendy features as time increases and choosing this model helps us perform this task better. It would be easier to add all new features to the software and deploy it faster for the time being [1].

4. [15 POINTS] Software Requirements including

4.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)

1. The user should be able to look for the events for any specific date or month.
2. The user should be able to pick any language from the available ones to add any event to the calendar.
3. The user should be able to delete any event.
4. The user should be able to send the invites for an event to different people if needed.
5. The user should be able to make modifications in any existing event.
6. The software should be able to generate and display the list of daily or monthly events in chronological order.
7. Each user should be identified using a unique username and login password.

4.b.) [10 POINTS] Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 - Ch 4)

Product Requirements:

Space Requirements:

1. The software would require 200 MB of space on any device it is being installed.

Performance Requirements:

1. The software would operate online 100% of the time.
2. Downtime of the software should not exceed 10 seconds.
3. Software should take no longer than one minute to update the calendar view.

Usability Requirements:

1. The software would require the user to make the contacts accessible in order to send the invites for any event.
2. All the events before one year shall be deleted automatically.

Organizational Requirements:

Operational Requirements:

1. Users of this software shall create an account using their gmail accounts only.
2. Users shall authenticate themselves using their unique user ID.

Developmental Requirements:

1. The software can be downloaded/updated on any platform, Android or IOS.

External Requirements:

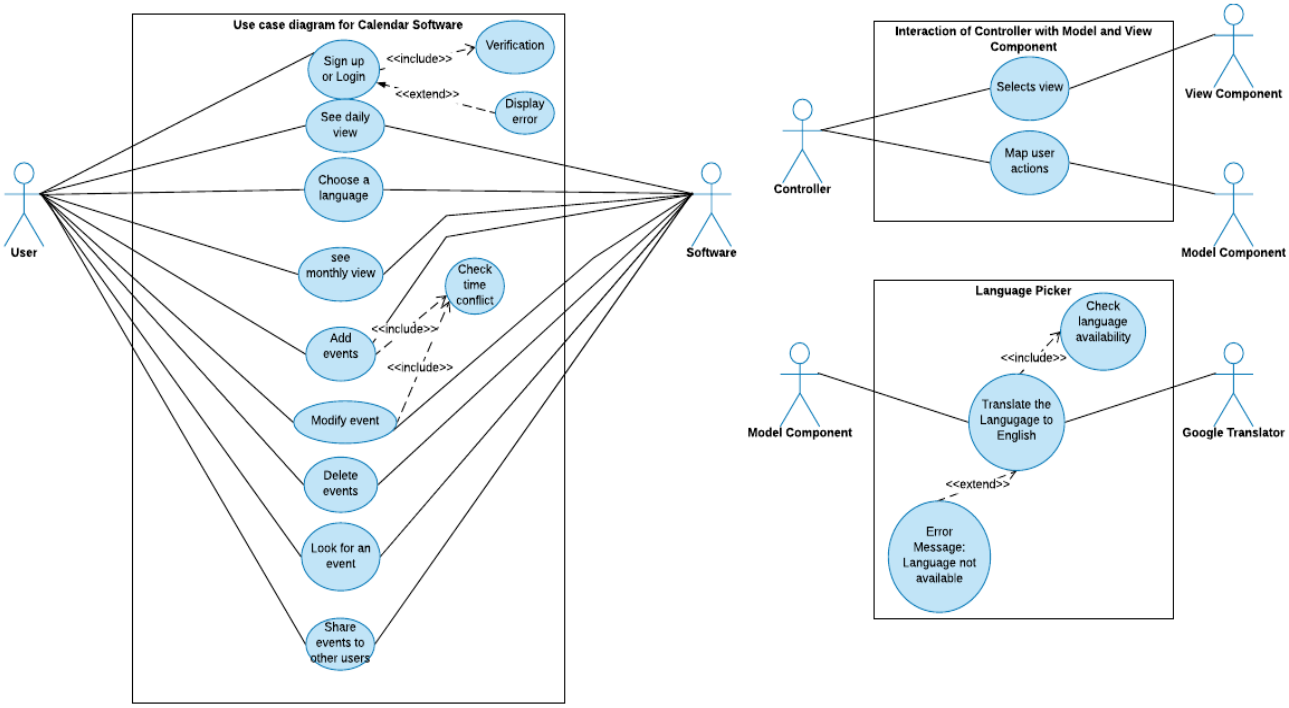
Safety/Security Requirements:

1. The software shall perform user authentication to support the privacy of users.

Regulatory Requirements:

1. Users are responsible for the contents/events they add or delete.
2. The software would not constrain or restrict any language used by the user.

5. [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)



Use case tabulations:

System	Calendar Software (View component)
Use case	**Sign up or Login**
Actors	User, Calendar Software, View component
Data	Through the View component, the user accesses the user account, either by logging into the existing account or creating a new user account. Once the user login information is verified, the user is given access to the account. If the user chooses to create a new account, user identity will be verified as well. If the user does not exist in the component, an error is thrown.
Stimulus	User clicks to log in/create the user account.
Response	Calendar Software accesses the controller component to verify the user account, and enable user to access the account
Comments	None

System	Calendar Software (View component)
Use case	**Look for an event**
Actors	User, Calendar Software, View component, Controller, Model component
Data	The user can access the event details by clicking Look Event in view component. The view component passes the information to controller, which then access the Model to return event details to user. The controller will determine through UID that if user is authorized to access the event detail or not. If not, an error gets displayed.
Stimulus	User clicks on look for an event
Response	View Component accesses the Model component to display the event details.
Comments	None

System	Calendar Software (view component)
Use case	**Add an event**
Actors	User, Calendar Software, View component, Controller, Model Component
Data	The user can add an event by clicking Add Event in view component. the view component passes information to controller component, which then determines the user authorization to add an event. If user is authorized, the controller will pass the information to Model. The Model will check if the new created event has any time conflict with existing event. If there is no time conflict, the model will create new event, and notify View component about new event details. An error message will be displayed if user is unauthorized or if there is time conflict.
Stimulus	User clicks to add an event.
Response	View Component accesses the Controller and Model component to add a new event.
Comments	None

System	Calendar Software (view component)
Use case	**Modify an event**
Actors	User, Calendar Software, View component, Controller, Model Component
Data	The user can modify an event by clicking Modify Event in view component. The view component passes information to controller component. If user is authorized to modify the event, the controller will pass the information to Model. If there is no time conflict, the model will replace the new modified event with existing event and notify View component about the modification. An error message will be displayed if user is unauthorized or if there is time conflict.
Stimulus	User clicks to modify an event.
Response	View Component accesses the Controller and Model component to modify an event.
Comments	None

System	Calendar Software (view component)
Use case	**Delete an event**
Actors	User, Calendar Software, View component, Controller, Model Component
Data	The user can delete an event by clicking Modify Event in view component. The view component passes information to controller component. If user is authorized to delete the event, the controller will pass the information to Model. The Model will delete an event and notify the View component. An error message will be displayed if user is unauthorized to delete an event.
Stimulus	User clicks to delete an event.
Response	View Component accesses the Controller and Model component to delete an event.
Comments	None

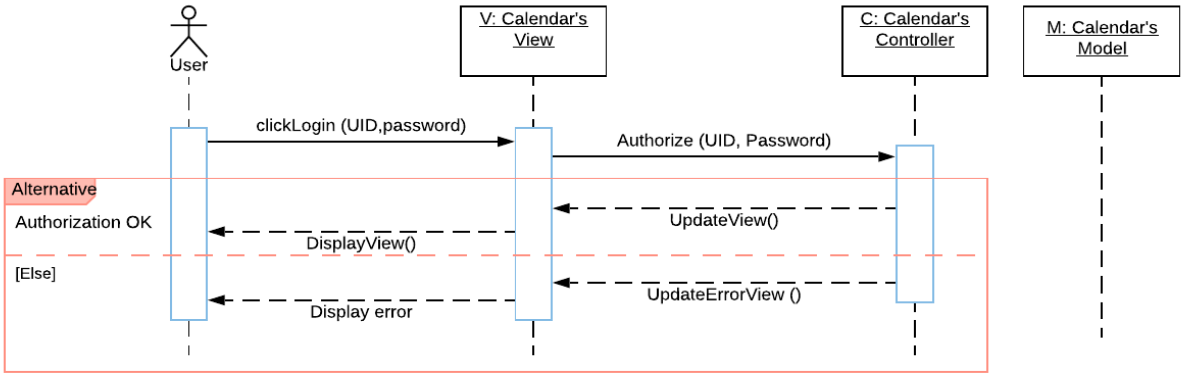
System	Calendar Software (view component)
Use case	**Share an event**
Actors	User, Calendar Software, View component, Controller, Model Component
Data	The user can share an event by clicking “Send Invites” and providing sender’s UID or email in view component. The view component passes information to controller component. If user is authorized to share the event, the controller will pass the information to Model. The Model will send the event invites to sender’s UID or email and notify the user about shared event through view component. An error message will be displayed if user is unauthorized.
Stimulus	User clicks to share an event with another user
Response	View Component accesses the Controller and Model component to share an event.
Comments	None

System	Calendar Software (view component)
Use case	**Get monthly/daily view**
Actors	User, Calendar Software, View component, Controller, Model Component
Data	The user can get monthly/daily view of events by clicking “Daily view/ Monthly view” in view component. The view component passes information to controller component. If user is authorized to get daily and monthly view, the controller will pass the information to Model. The Model will get the list of events and display the event list to user through view component. An error message will be displayed if user is not authorized to view event list.
Stimulus	User clicks to get the monthly/daily view.
Response	View Component accesses the Controller and Model component to get monthly/daily view of calendar.
Comments	None

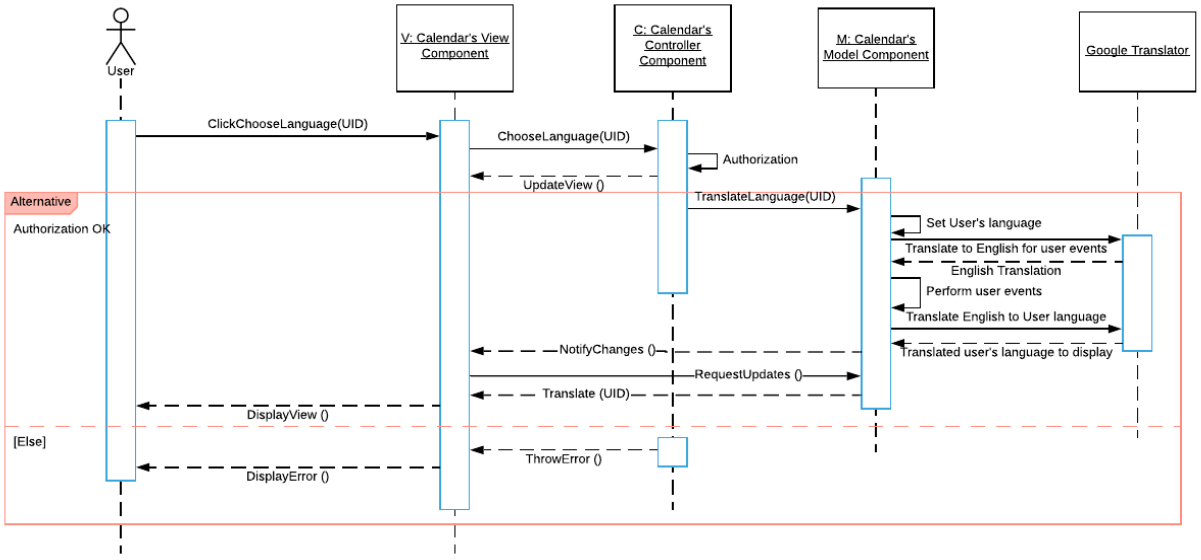
System	Calendar Software (view component)
Use case	**Choose Language**
Actors	User, Calendar Software, View component, Controller, Model Component, Google Translator
Data	The user can choose the language by clicking Choose Language in view component. The view component passes information to controller component. If user is authorized to change language, the controller will pass the information to Model. The Model will set the chosen language as user language and access the Google Translator to change calendar view from English to user language. The model component will notify the view component to change calendar view to user chosen language. An error message will be displayed if user is not authorized to change the language.
Stimulus	User clicks to change language.
Response	View Component accesses the Controller, Model component, and Google Translator to translate the language.
Comments	None

6. [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)

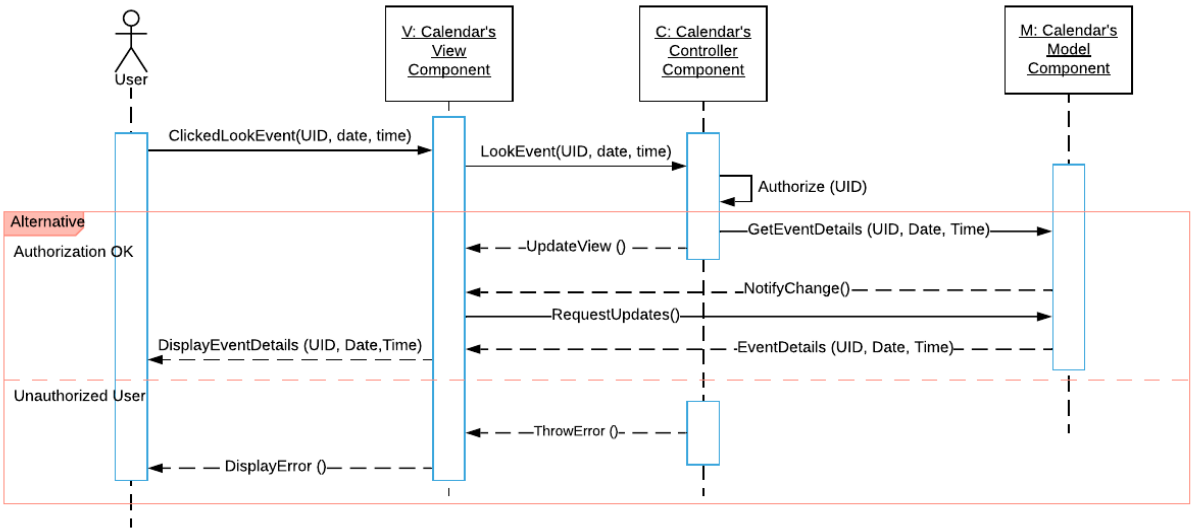
Sign Up/Login



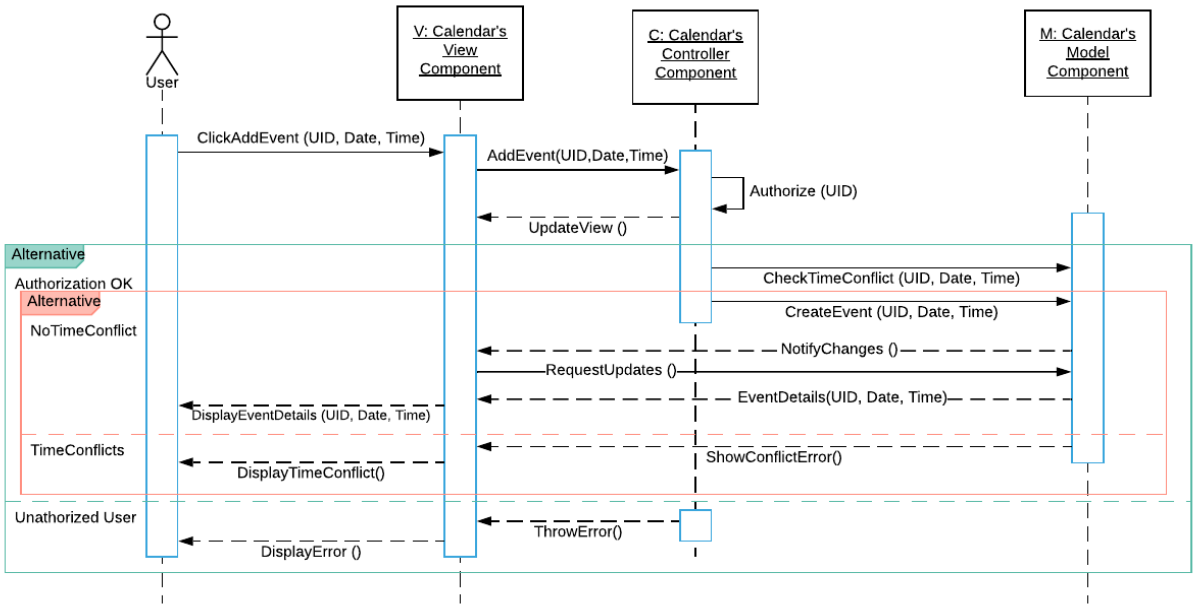
Choose a Language



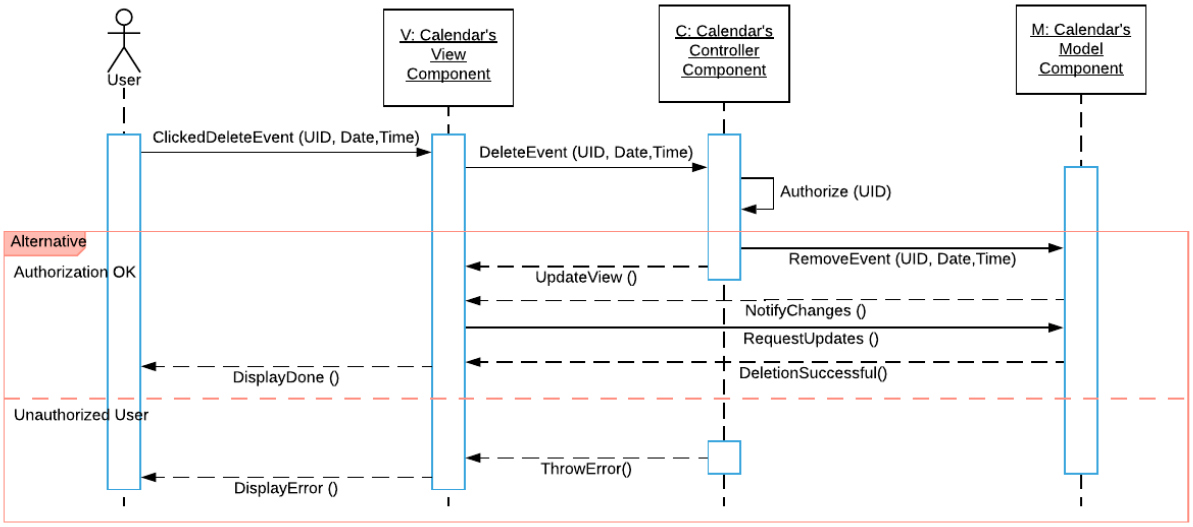
Look for an Event



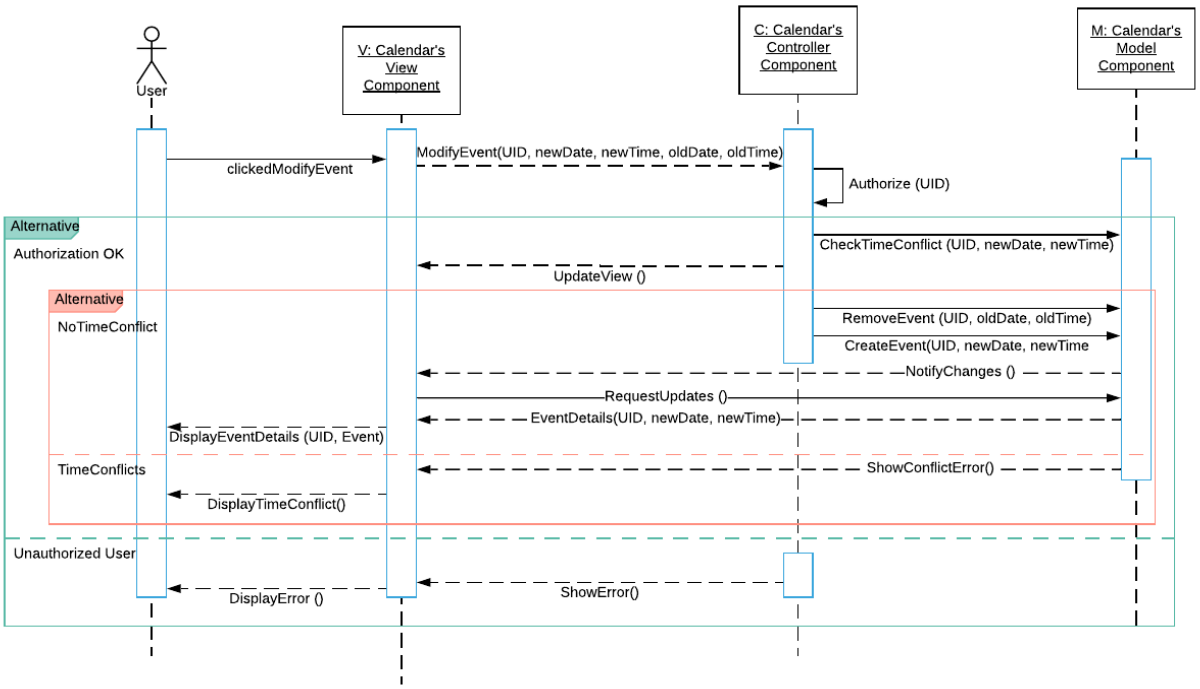
Add an Event



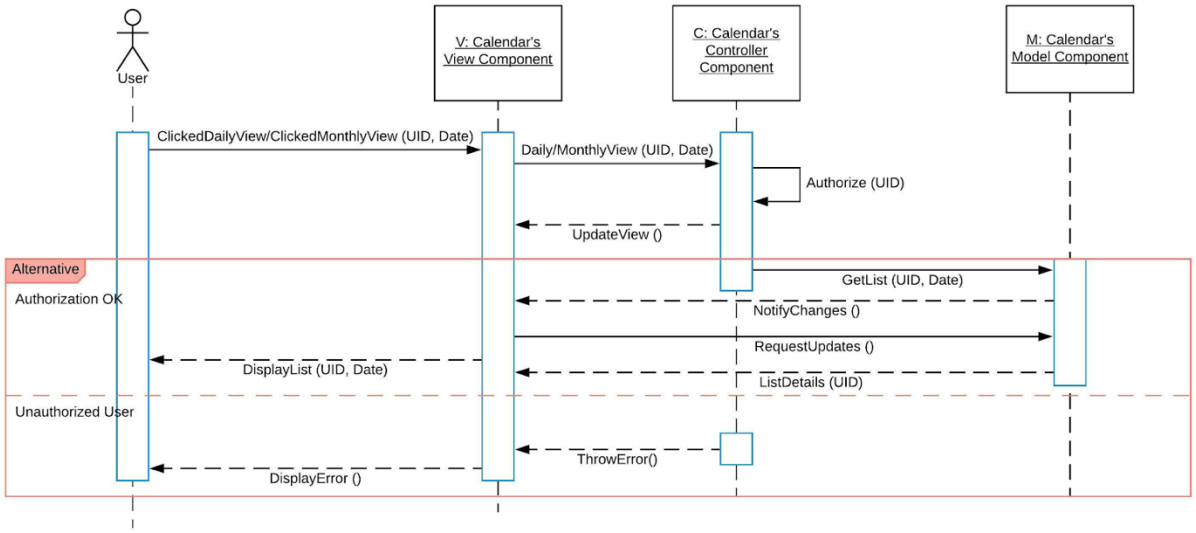
Delete an Event



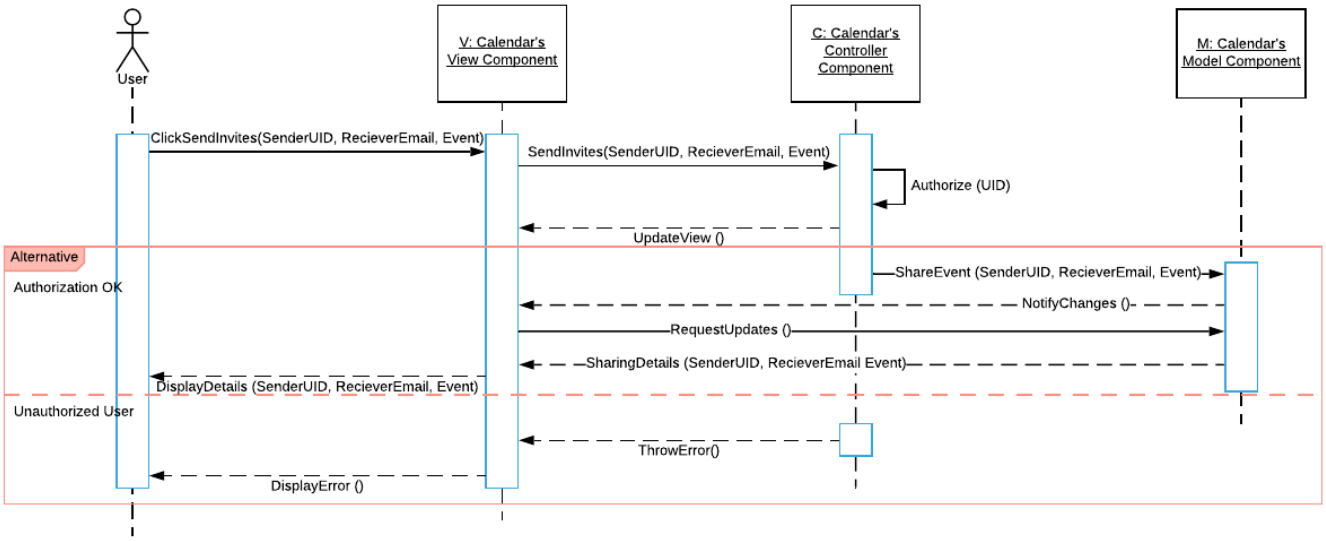
Modify Event



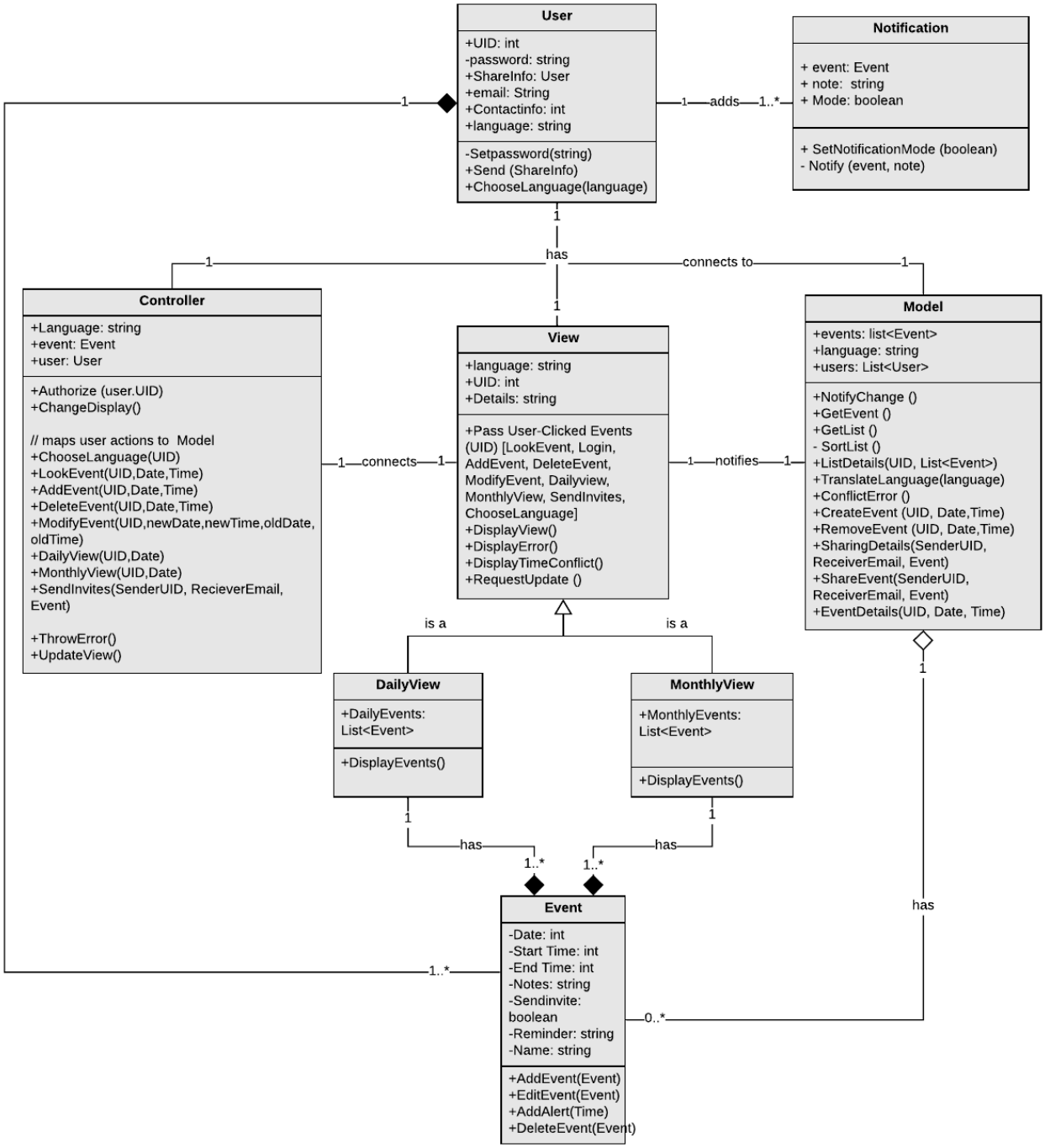
Get Daily/Monthly View



Share an Event/Send Invites



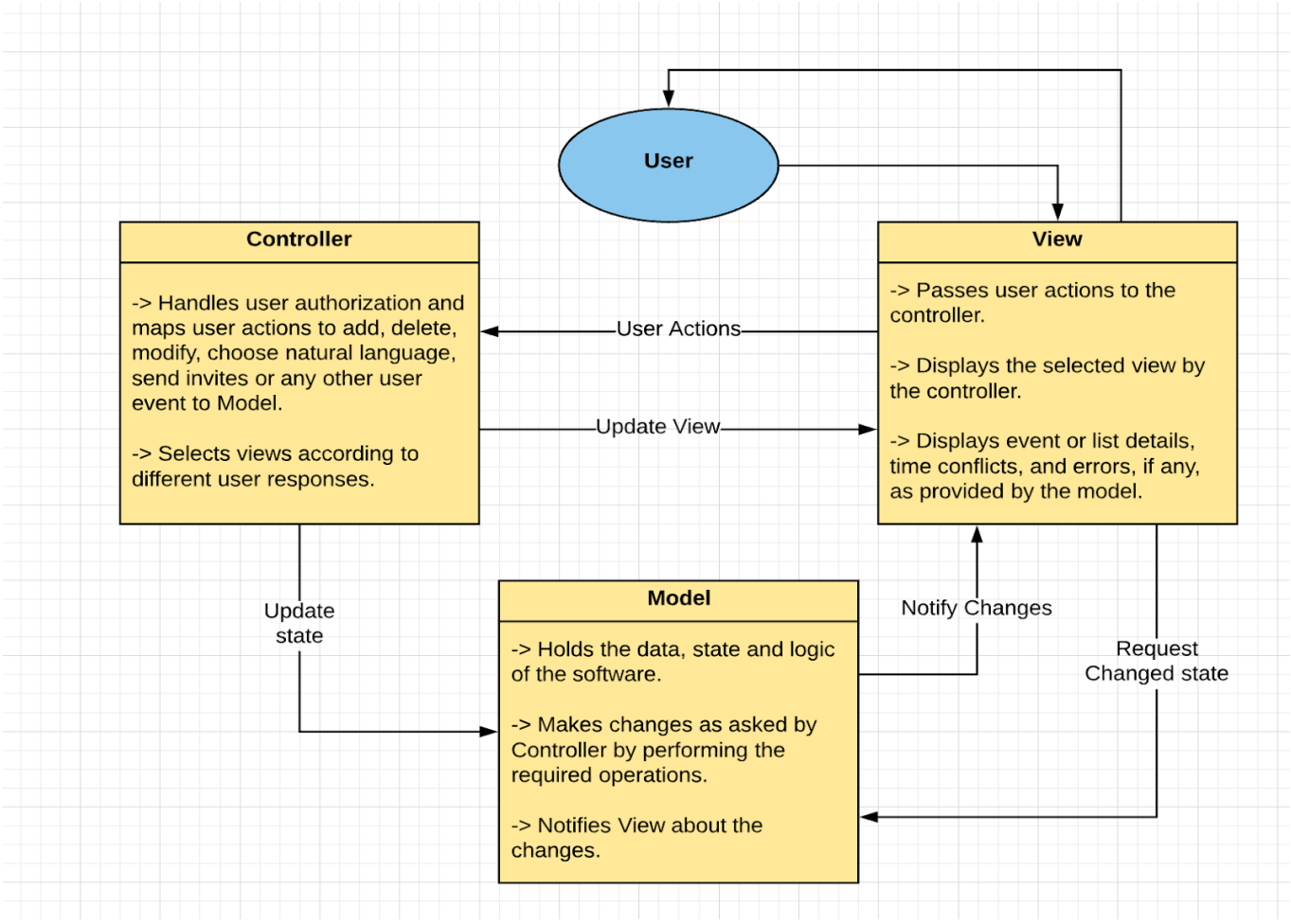
7. [15 POINTS] Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



8. [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)

a. Model-View-Controller (MVC) pattern (similar to Figure 6.6)
b. Layered architecture pattern (similar to Figure 6.9)
c. Repository architecture pattern (similar to Figure 6.11)
d. Client-server architecture pattern (similar to Figure 6.13) e. Pipe and filter architecture pattern (similar to Figure 6.15)

After analyzing all patterns, our group decided to use MVC architecture pattern for calendar software. MVC architecture is typically used when there are multiple ways to view and interact with data. This is also used when future requirements for interaction and presentation of data are unknown. Therefore, considering these facts, we decided MVC would be right for the calendar software [1].



Model component:
This component contains the system data and logic associated with that data. The data, logic, and rules of the calendar software will be handled by this component. It will contain all of the methods related to calendar software including get monthly/daily view, add/delete an event, view event, modify/edit event and check time conflict when editing events as could be seen in the above class diagram [1].

View component:
View component of MVC manages how the data is presented to the user. The user will directly interface with the view. After interacting, View would pass these user events to the controller so that it could further map it to the model. Once being handled by the model, the Model component notifies the View component of the changes been made and thus, the view component displays directly the details [1].

Controller component:
The controller handles user interactions and passes these interactions to the Model components and makes necessary changes as requested by the user. Controller is a mediator, which will interact with both model and view components based on user's actions. The controller will request the model component to change, modify, delete, add an event or perform any other operation. Basically, the controller takes user input through view and figures out what it means to the model. Also, the controller will select the view to be displayed according to the user request and passes the selected view to the View component [1].

References:

[1] Ian Sommerville, *Software engineering*, 10th ed. Harlow: Pearson Education, 2016.