

Project Vision:

We will be making a Planner application that can support event interactions, recognize event proposals, recognize resources, allow booking event times, and disallow two groups from booking to the same resources. The Planner should be able to create/delete events on the Planner, archive events, delete events that have already passed, and edit pre-existing events.

Requirements

Functional

1. **Events** should interact with services
2. The **planner** should support sponsored search results
3. The **planner** should enforce logical scheduling
4. The **planner** should have a user hierarchy
5. The **planner** should support searching
6. The **planner** should be able to share information
7. The **user** can create an account
8. The **user** can view resources
9. The **user** can interact with services
10. The **user** can interact with planner
11. The **user** can interact with events
12. The **user** can interact with friends
13. The **user** can update settings
14. The **user** can filter search results
15. The **user** can conduct business

Non-Functional

1. The planner, and its features, will be available in the English language.
2. Documentation for the planner will be written in English
3. All event dates are posted in the format of the Gregorian calendar
4. The planner assumes Central Standard Time
5. The planner should have a GUI

[illegible]

ID UC Create New User

Created By Bryce Clark

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Precondition

- User does not have an account

Postcondition

- A User is created

Main success scenario

1. User provides Credentials
2. User is created

Extensions

a.* Anytime User hits cancel

1. User sent to login

ID UC Login

Created By Bryce Clark

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Precondition

- User wants to log in

Postcondition

- User is logged in

Main success scenario

1. User provides credentials
2. User is logged in

Extensions

a.* anytime User exits login

1. User sent back to desktop

ID UC Add Event

Created By Bryce Clark

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Precondition

- User wants to add an event to their planner

Postcondition

- Event is added

Main success scenario

1. User selects Add Event
2. User picks option
3. Option is executed

Extensions

a.* anytime User exits add event

1. Event is restored before User made any edits

1a. User selects event from data base

1. User selects an event

2. Event is added to planner

2a. User selects create an event

1. User enters event details
2. User selects save
3. Event is added to data base
4. Event is added to planner

ID UC Manage Friends

Created By Benjamin Kilpatrick

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User wants to manage friends

Postcondition

- Friends are updated

Main success scenario

1. User selects other user
2. Other user given access to user information
3. Other user's access to user is changed

Extensions

a.* anytime User exits manage friends

1. No more updates are preformed
- 2.a User wishes to restrict access from a user with permission
1. Other user's permission to user is removed

ID UC Filter search results

Created By

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Precondition

- User wants to filter search results

Postcondition

- Search results are updated

Main success scenario

1. User selects filter search results
2. User picks option
3. Option is executed

Extensions

a.* anytime user wants to exit

1. User sent to desktop

b.* anytime user wants to clear filter

1. User clears filter

2. Filter is cleared

2a. Filter by event

1. Results are filtered by event

2. Users get results

2b. Filter by location

1. Results are filtered by location

2. Users get results

2c. Filter by service

1. Results are filtered by service

2. Users get results

ID UC Set Notifications

Created By

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- Notifications are not set to the User/Business User wants

Postcondition

- Notifications are

Main success scenario

1. User chooses to set notifications
2. User chooses an option
3. Notifications are set

Extensions

a.* anytime User no longer wants to set notifications

1. Return to Planner

2a. User sets event notifications

1. Event notifications are set

2. Return to planner

2b. User sets service notifications

1. Service notifications are set

2. Return to planner

ID UC View Profiles

Created By

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Precondition

- User wants to view profiles of another User

Postcondition

- User views or cannot view other user's profiles

Main success scenario

1. User wants to view profile
2. User selects profile to view
3. User views profile

Extensions

a.* anytime user can exit out of view profile

1. the user is no longer presented with the profile

2a. Profile could be set to private

1. User will get a warning message

ID UC Search Planner

Created By Yutai Xue

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User want to search for resources in Planner

Postcondition

- User gets result

Main success scenario

1. User inserts keyword and apply filters to search
2. User gets results

Extensions

a.* anytime User stops searching

1. Characters left in the search will not be remembered between instances

b.* anytime User wants to clear search

1. User clears search
2. Search is cleared

2a. User does not receive any results

1. System prompts user to reconsider their search term or filter

ID UC Set Setting

Created By Yi Ding

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User wants to set settings

Postcondition

- Settings are updated

Main success scenario

1. User wants to set setting
2. User updates the settings

Extensions

a.* anytime User wants to exit settings

1. User ends action

2a. User forgets to save settings and exits

1. Settings is reset back before User edited it

ID UC Sponsored Search Results

Created By Yi Ding

Scope Travel planner

Stakeholders and interests

Business User

- Service Providers

Precondition

- Business User want to advertise

Postcondition

- Top search results are sponsored by Business User

Main success scenario

1. Business Users want to advertise
2. Business Users select advertise and inserts budget
3. Sponsor ranking is updated

Extensions

a.* anytime Business Users wants to exit advertise

1. Business Users ends action

2a. Business Users decide to not invest and exits advertise

1. Business Users ends action

ID UC Edit Event

Created By Yutai Xue

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User want to view Planner

Postcondition

- User view Planner

Main success scenario

1. User selects Planner to view
2. User views Planner

Extensions

a.* anytime User wants to exit select Planner

1. User sent back to the desktop

2a. User denied from viewing Planner

1. User receives a warning message

2. User remains on select Planner

ID UC Manage Services

Created By

Scope Travel planner

Stakeholders and interests

Business owner

- Wants to make profit by advertising services

Business employee

- Wants to advertise services that they provide

Precondition

- User with business access wants to manage service

Postcondition

- Service is updated

Main success scenario

1. Business User picks service
2. Business edits the service attributes
3. Service is edited

Extensions

a.* anytime Business User exits manage service

1. Business User ends action
2. Service is restored before Business User made any edits

1a. The user does not select a service

1. New service is created
2. Business User inputs information
3. Business User ends action and service is updated

3a. Business User selects delete service

1. Business User selects service to delete
2. Business User confirms deletion
3. Business User ends action and service is updated

ID UC Manage Booking

Created By Benjamin Kilpatrick

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User want to manage booking

Postcondition

- Booking has been updated

Main success scenario

1. User selects booking
2. User edits the booking
3. Booking is saved

Extensions

a.* anytime User exits manage booking

1. User ends action

2. Booking is restored before User made any edits

1a. User selects creates new booking

1. User inputs information

2. User ends action and booking is updated

3c. User deletes booking

1. User confirms deletion

2. User ends action and booking is updated

ID UC Manage Planner

Created By Yutai Xue

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- User want to manage planner

Postcondition

- Planner is updated

Main success scenario

1. User wants to manage a planner
2. User picks option
3. Option is executed

Extensions

a.* anytime User exits manage planner

1. User ends action

2. Planner is restored before User made any edits

2a. User selects add planner

1. User inputs information

2. User ends action and planner is updated

2b. User selects edit planner

1. User selects planner to edit

2. User edits information

3. User ends action and planner is updated

2c. User selects delete planner

1. User selects planner to delete

2. User confirms deletion
3. User ends action and planner is updated

ID UC Payment

Created By Yi Ding

Scope Travel planner

Stakeholders and interests

User

- The person/persons who will be traveling on the trip.

Business User

- Service Providers

Precondition

- Payment is needed

Postcondition

- Payment has been paid

Main success scenario

1. User selects checkout
2. User gives payment information
3. User payment is archived

Extensions

a.* anytime User exits payment

1. User ends action

2a. User cannot afford the total

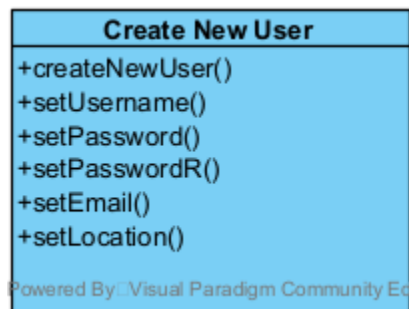
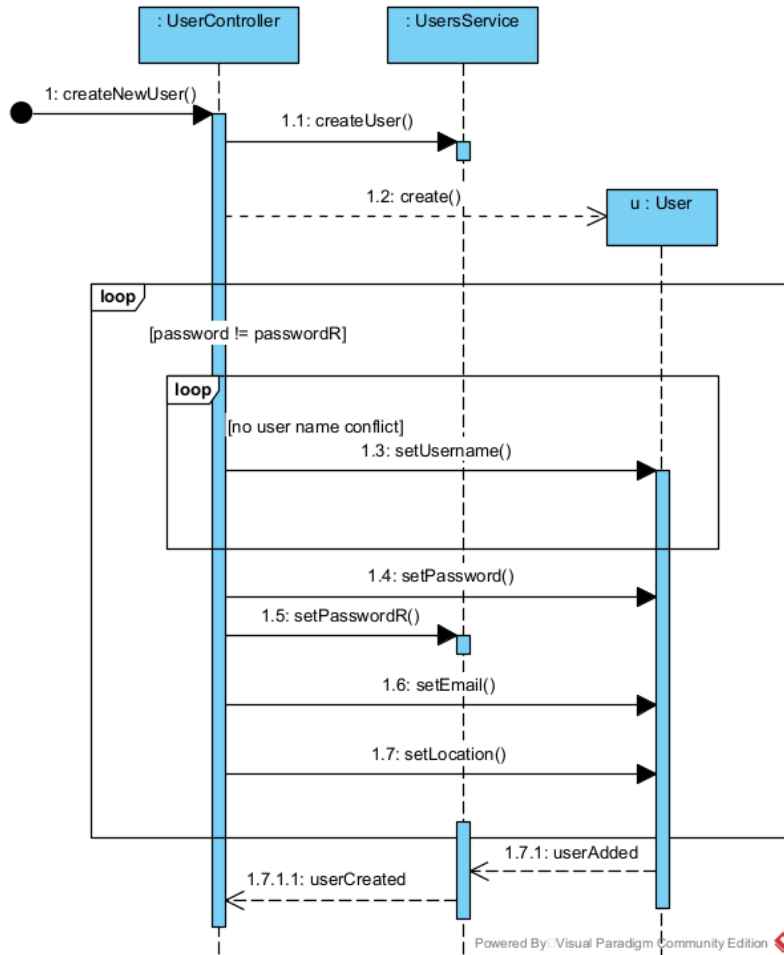
1. User receives a warning message

2. User ends action

SD and Operation Contracts

UC Create New User

Create new user – Bryce



Contract: createNewUser()

Operation: begins the create new user process

Cross Reference: user login

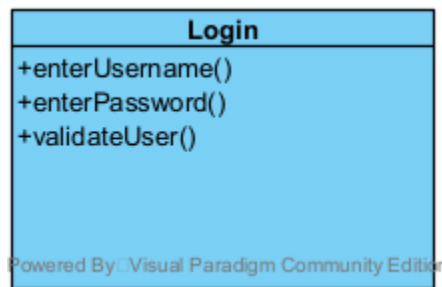
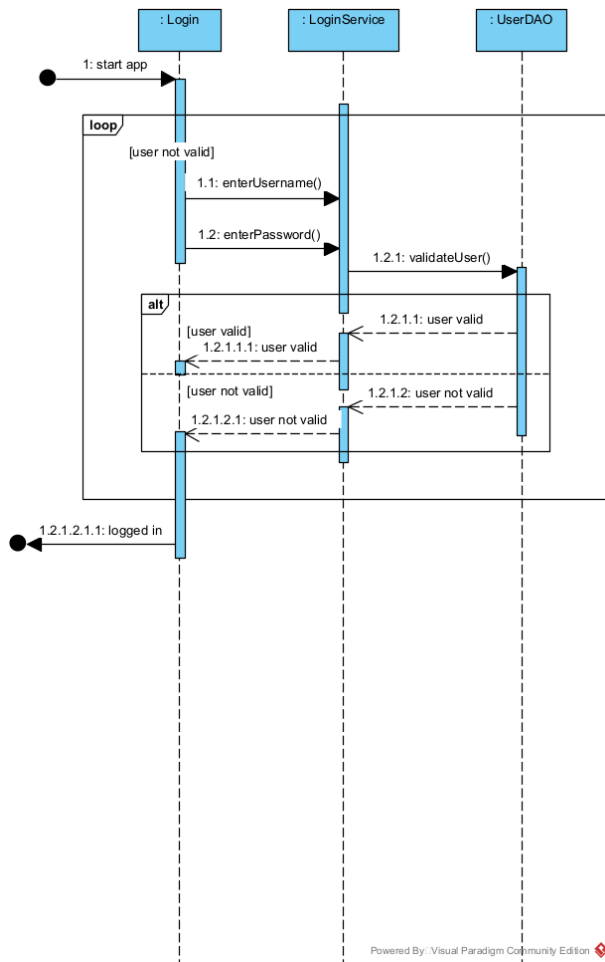
Pre-conditions: A user does not have an account.

Post-condition: A user is created.

Contract: setUsername()
Operation: sets the username credential of the new user Cross Reference: user login Pre-conditions: A new user has no username set. Post-condition: A new user has a username
Contract: setPassword()
Operation: sets the password credential of a new user Cross Reference: user login Pre-conditions: A new user does not have a password. Post-condition: A new user password is created.
Contract: setPasswordR()
Operation: sets the passwordR credential of a new user Cross Reference: user login Pre-conditions: A new user has not reentered their password. Post-condition: Password and PasswordR match.
Contract: setEmail()
Operation: sets the email of the user Cross Reference: user login Pre-conditions: A new user account does not have an email. Post-condition: A user email has been set.
Contract: setLocation()
Operation: sets the location of the user Cross Reference: user login Pre-conditions: A new user account does not have a location set Post-condition: A new user has a location set

UC Login

Login – Bryce

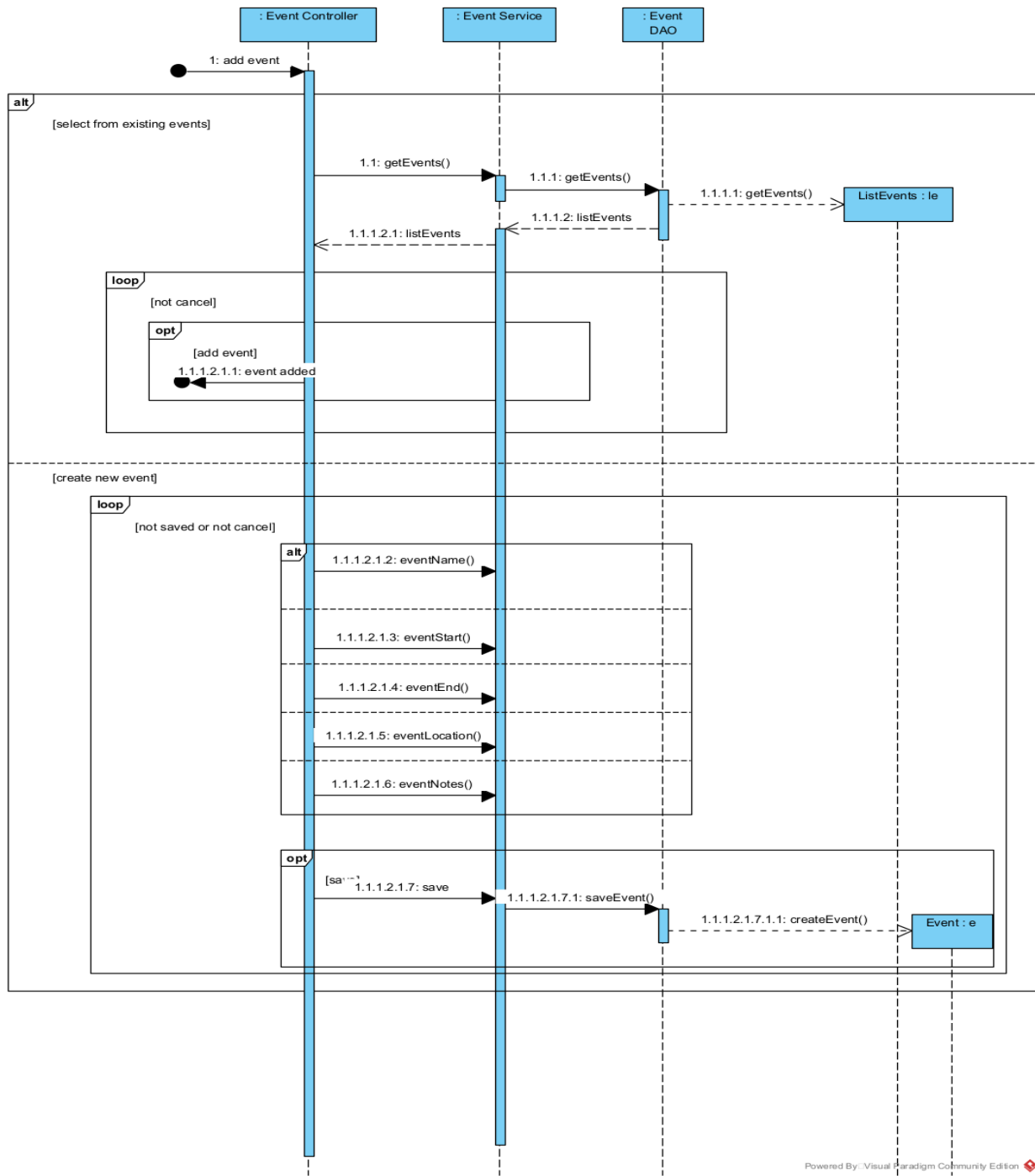


Contract: enterUsername()
Operation: user enters their username Cross Reference: Login Pre-conditions: User has not entered their username. Post-condition: User waits for validation.
Contract: enterPassword()
Operation: user enters their password Cross Reference: Login Pre-conditions: User has not entered their password. Post-condition: User waits for validation.

Contract: validateUser()
Operation: confirms that the username and password is valid Cross Reference: Login Pre-conditions: User has entered their username and password. Post-condition: User is now logged in.

UC Add Event

Add Events – Bryce

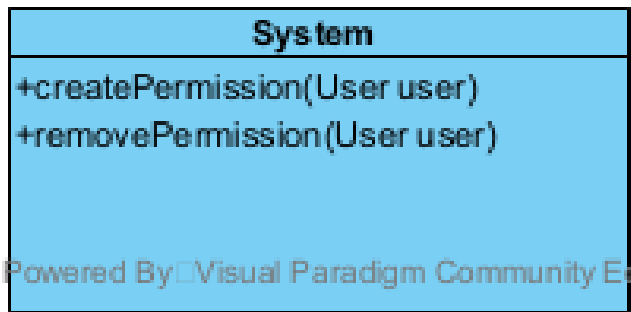
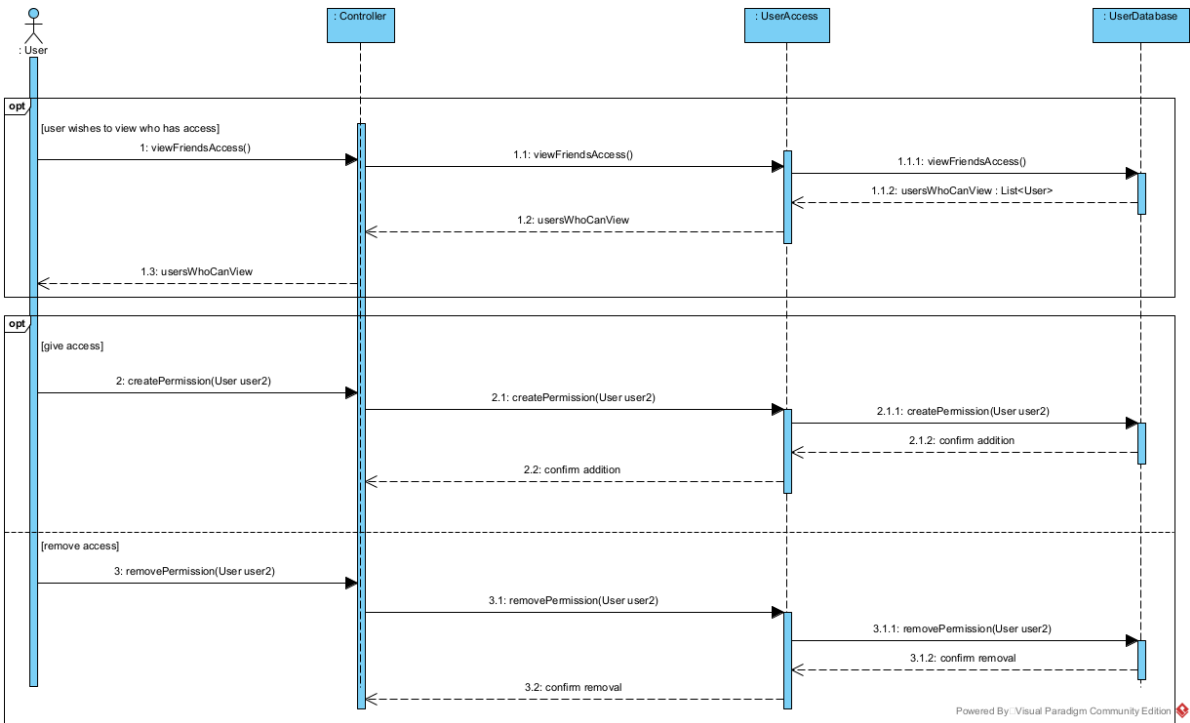


Add Event
+getEvents() +eventName() +eventStart() +eventEnd() +eventLocation() +eventNotes() +saveEvents()

Contract: getEvents()
Operation: gets all events from the database Cross Reference: add event Pre-conditions: User wants to add event from the database. Post-condition: The selected event is added to planner.
Contract: eventName()
Operation: user enters the name of the event Cross Reference: add event Pre-conditions: User wants to create a new event. Post-condition: A new event has been created and added to database.
Contract: eventStart()
Operation: user enters the start time of the event Cross Reference: add event Pre-conditions: User wants to create a new event. Post-condition: A new event has been created and added to database.
Contract: eventEnd()
Operation: user enters the end time of the event Cross Reference: add event Pre-conditions: User wants to create a new event. Post-condition: A new event has been created and added to database.
Contract: eventLocation()
Operation: user enters the location of the event Cross Reference: add event Pre-conditions: User wants to create a new event. Post-condition: A new event has been created and added to database.
Contract: addNotes()
Operation: user enters any notes about the event Cross Reference: add event Pre-conditions: User wants to create a new event. Post-condition: A new event has been created and added to database.
Contract: saveEvent()
Operation: user has created a new event Cross Reference: add event Pre-conditions: User has created a new event. Post-condition: The event has been added to the database.

UC Manage Friends

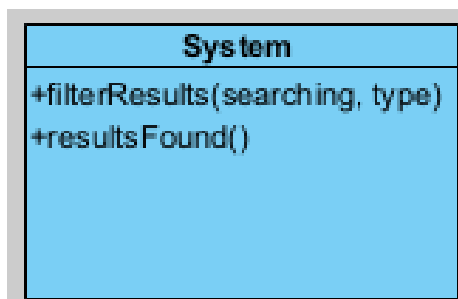
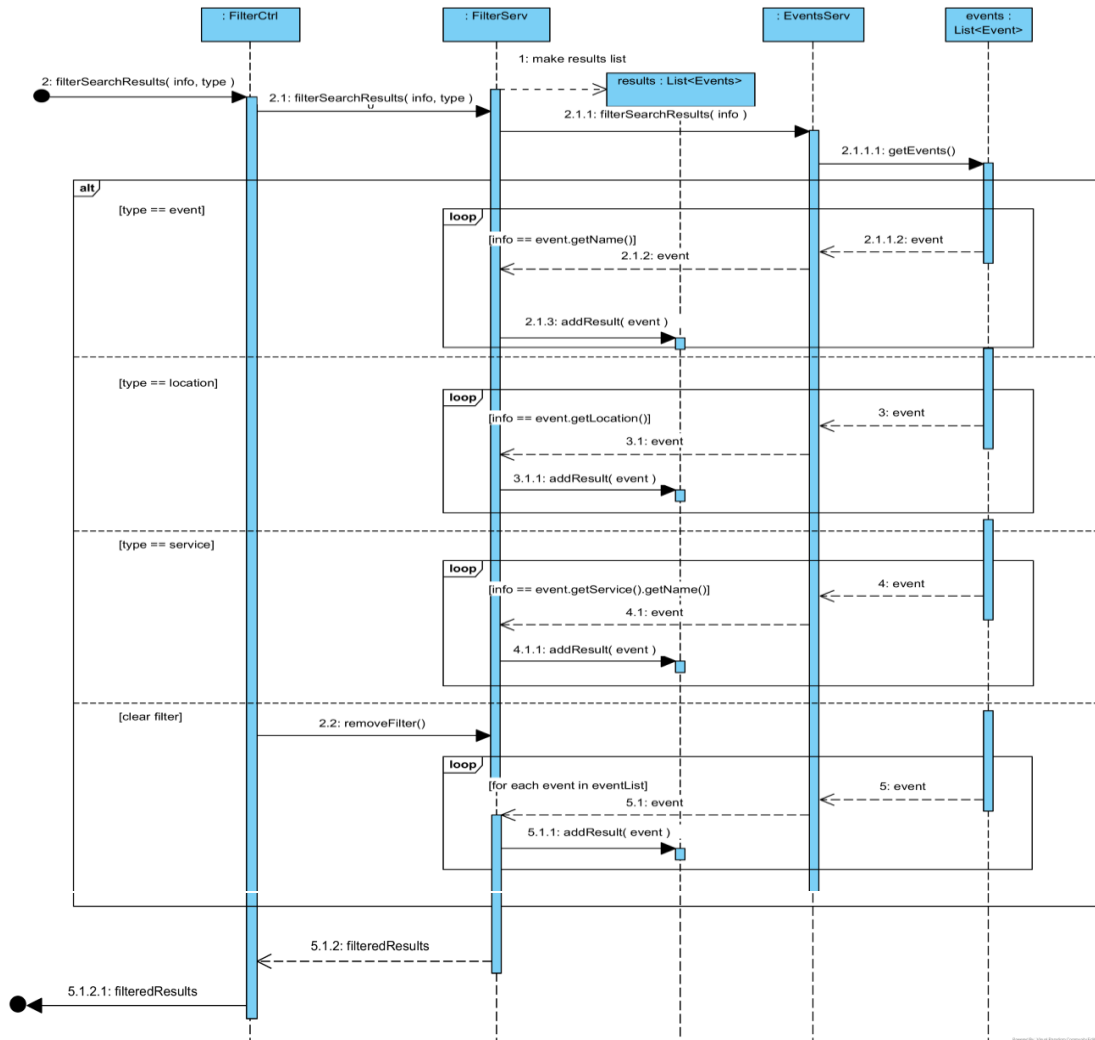
SD Manage Friends – Benjamin



Contract: createPermission
Operation: creates permission for another user to access the active user’s information
Cross Reference: Manage Friend Access
Pre-conditions: another user exists without access to the active user
Post-condition: the other user is granted access to the active users location and email.
Contract: removePermission
Operation: removes permission for another user to access the active user’s information
Cross Reference: Manage Friend Access
Pre-conditions: another user exists with access to the active user
Post-condition: the other user’s permission to view the active user’s location and email is revoked.

UC Filter Search Results

Filter Search Results – Jason



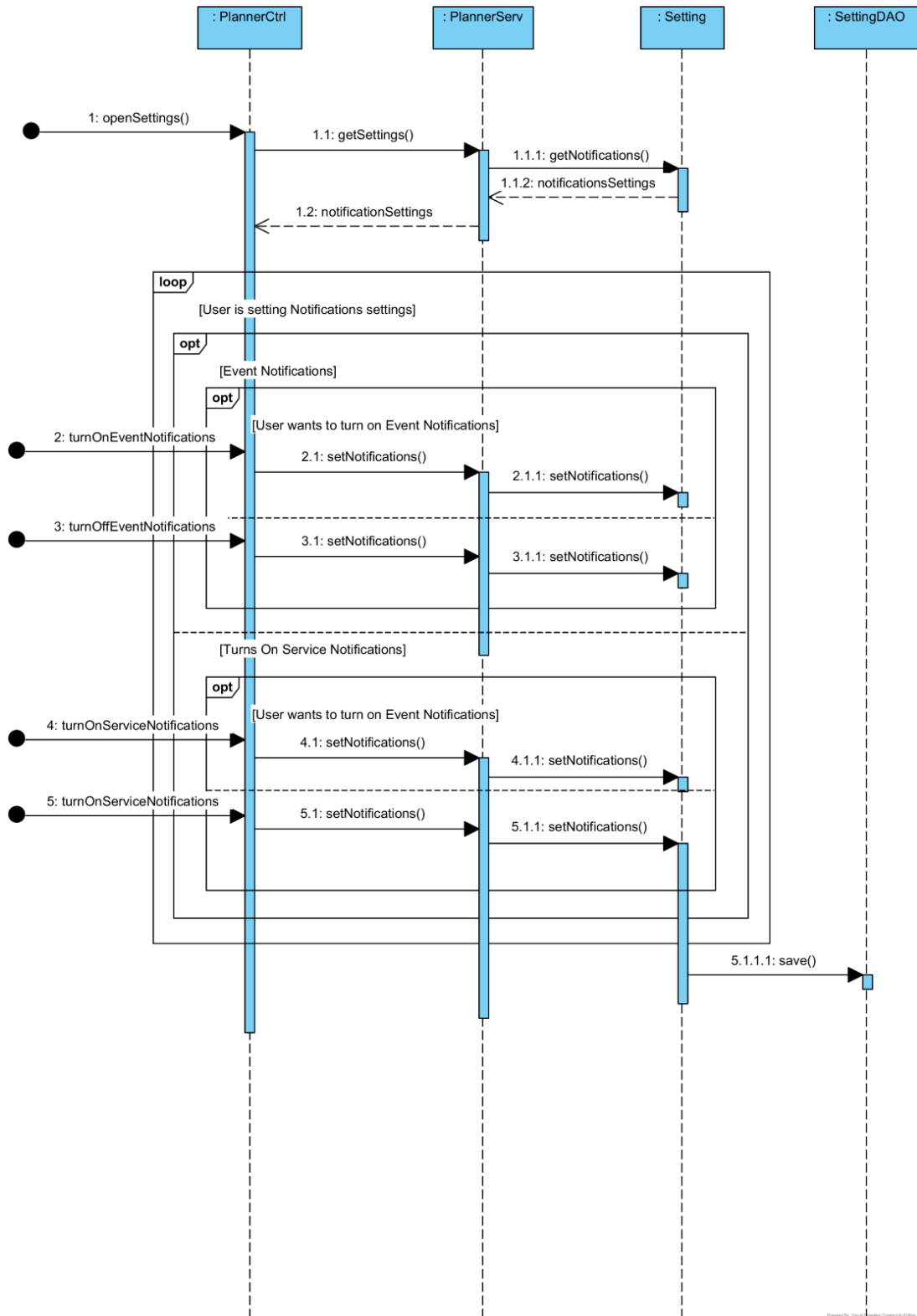
Contract: filterResults(searching, type)

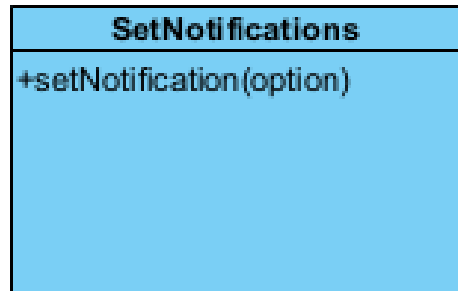
Operation: Filters the results matching “searching” to only those that have the type of “type”

Cross Reference: Use Case: Filter Search Results Pre-conditions: User wants to search for a specific result Post-condition: User receives search results
Contract: resultsFound
Operation: Prints the results returned by filterResults Cross Reference: Use Case: Filter Search Results Pre-conditions: results aren't visible to user Post-condition: results are printed for user to see

UC Set Notifications

Set Notifications – Jason

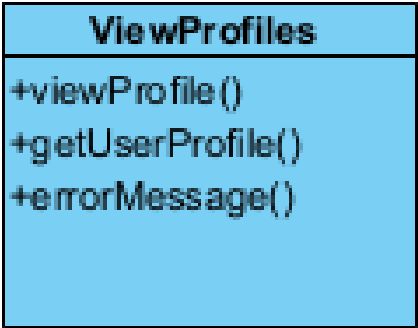
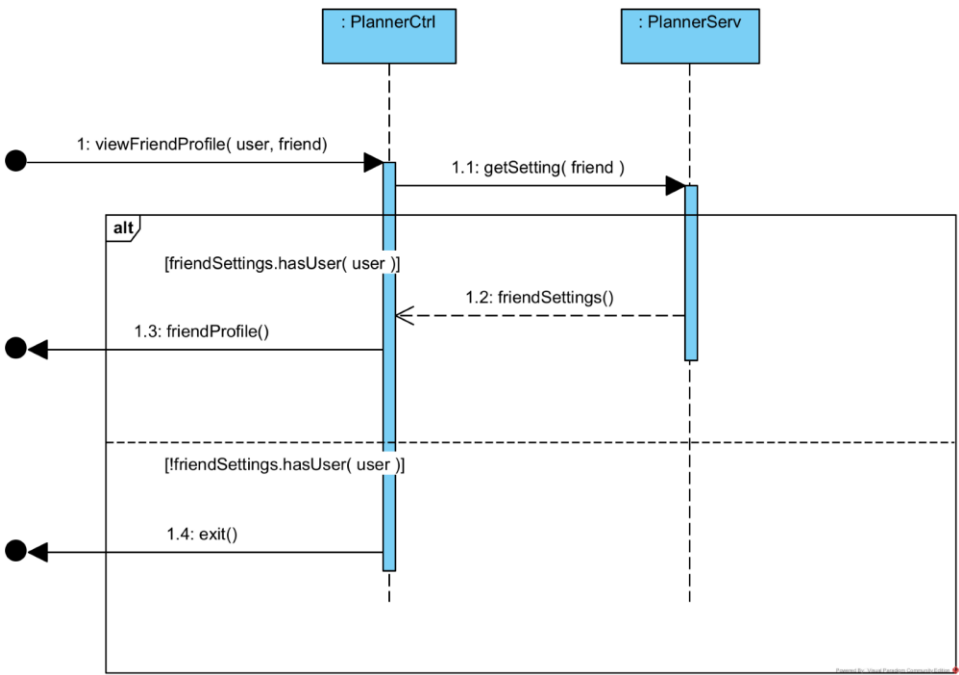




Contract: set Notification (option)
Operation: sets notifications for the planner
Cross Reference: Use Case: Set Notifications
Pre-conditions: Notifications aren't properly configured
Post-condition: Notifications are configured

UC View Profiles

ViewProfiles – Jason

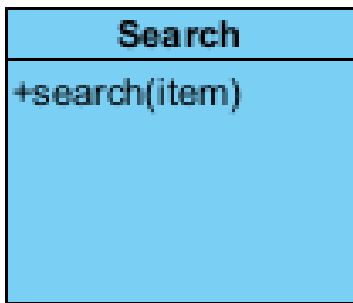
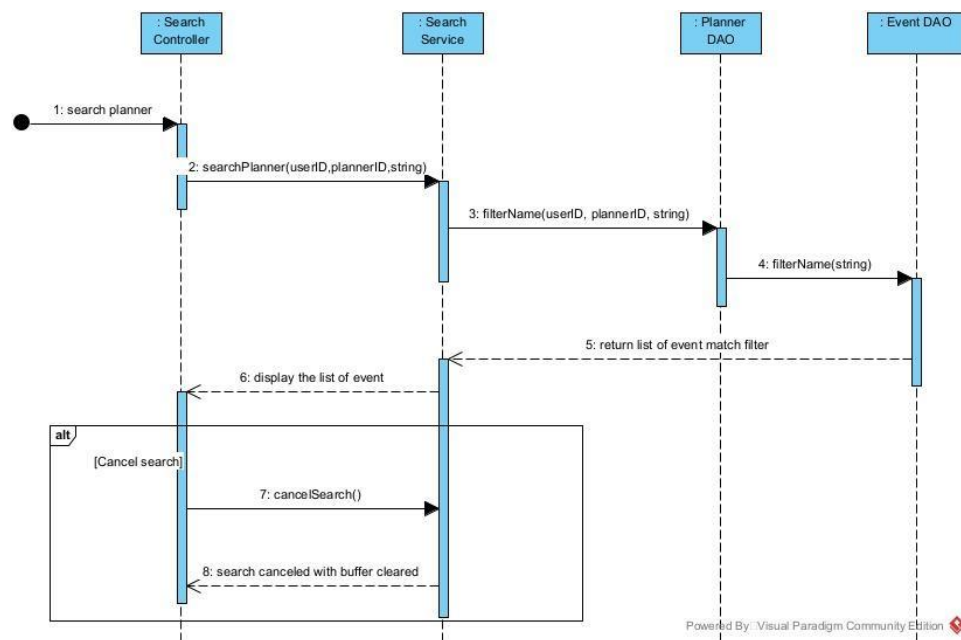


Contract: viewProfile()
Operation: checks if the user is allowed to view the given profile
Cross Reference: Use Case: ViewProfiles
Pre-conditions: User wants to open other users' profiles
Post-condition: User's permissions are checked with the s
Contract: getUserProfile()
Operation: acquires the profile of a user and displays it
Cross Reference: Use Case: ViewProfiles
Pre-conditions: User has correct permissions to view the profile
Post-condition: Profile is displayed
Contract: errorMessage()
Operation: sends an error message if the User has incorrect
Cross Reference: Use Case: errorMessage()

Pre-conditions: User has incorrect permissions for viewing the profile
Post-condition: Error message is displayed

UC Search Planner

Search – Yutai



Contract: Search

Operation: searches for an item with a keyword

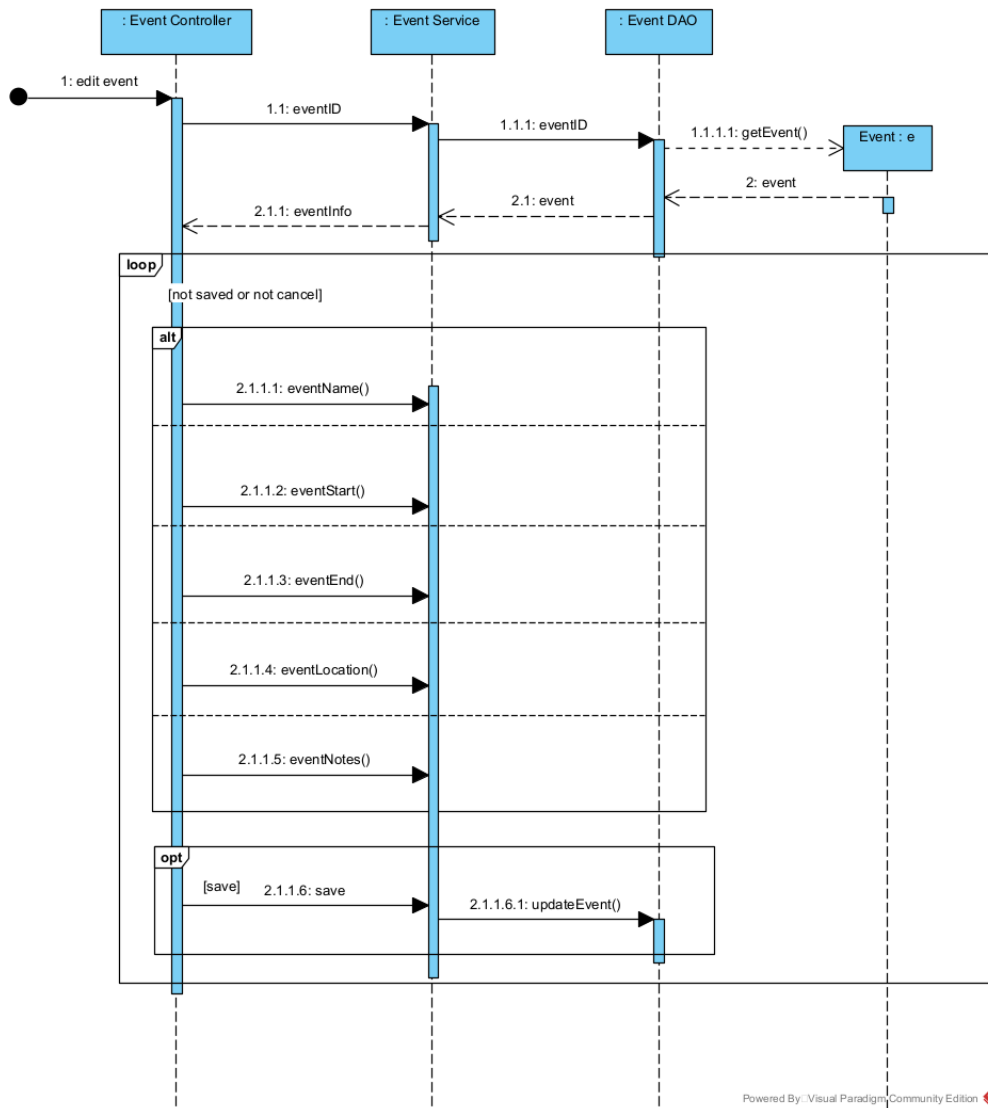
Cross Reference: Filter Search Result

Pre-conditions: User want to search for resources in Planner

Post-condition: User gets result based on keyword

UC Edit Event

Edit Event– Yutai



Contract: eventName

Operation: Updates the event with the changes you have made

Cross Reference: manage planner

Pre-conditions: User want to edit an event's name

Post-condition: User has completed their editing process

Contract: eventStart

Operation: Updates the event with the changes you have made

Cross Reference: manage planner

Pre-conditions: User want to edit an event's start date

Post-condition: User has completed their editing process

Contract: eventEnd

Operation: Updates the event with the changes you have made

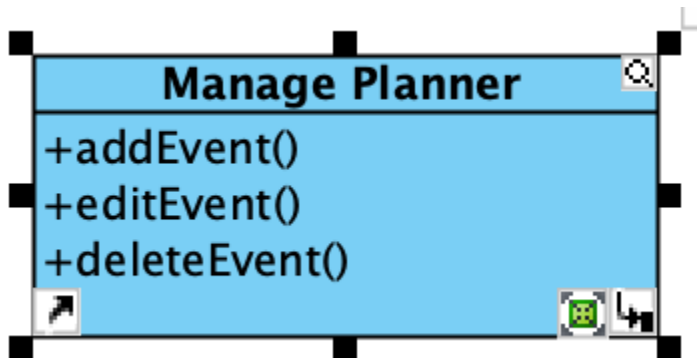
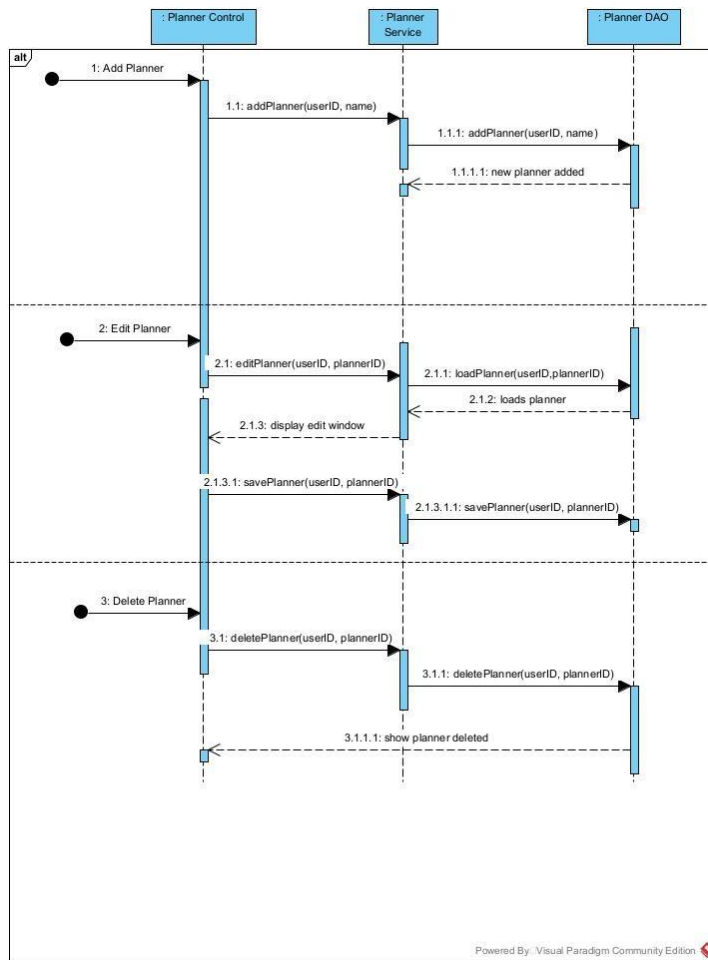
Cross Reference: manage planner

Pre-conditions: User want to edit an event's end date

Post-condition: User has completed their editing process
Contract: eventLocation
Operation: Updates the event with the changes you have made Cross Reference: manage planner Pre-conditions: User want to edit an event's location Post-condition: User has completed their editing process
Contract: eventNotes
Operation: Updates the event with the changes you have made Cross Reference: manage planner Pre-conditions: User want to edit an event's notes Post-condition: User has completed their editing process
Contract: updateEvent
Operation: Updates the event in the databse with the changes you have made Cross Reference: manage planner Pre-conditions: User has finished editing their event Post-condition: User has completed their editing process

UC Manage Planner

Manage Planner – Yutai

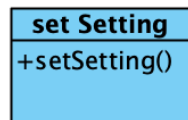
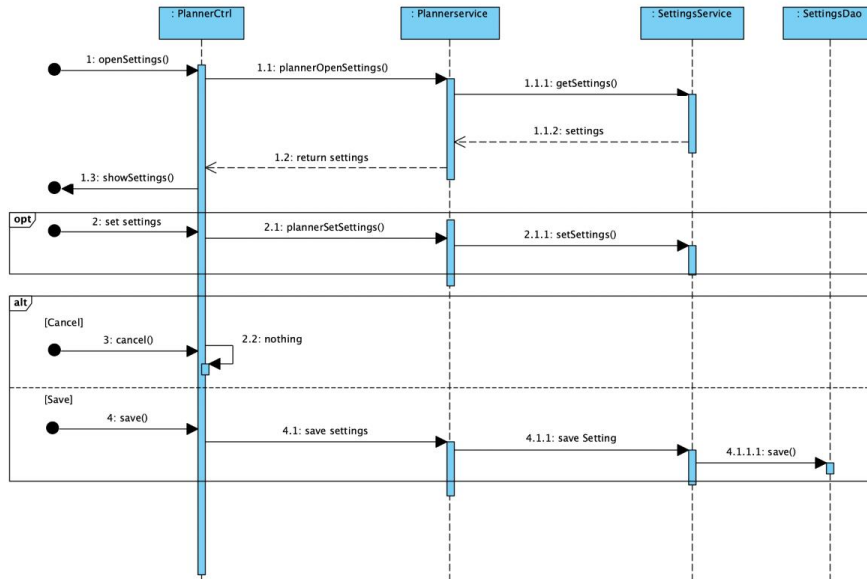


Contract: Manage Planner
Operation: Allows for add, delete, or edit events from a planner
Cross Reference: Manage booking
Pre-conditions: User want to manage a planner
Post-condition: Planner selected has been updated

UC Set Settings

Settings – Yi Ding

[sd [Set Settings] /



Contract: Set Setting

Operation: Allows User to set setting

Cross Reference: Filter search results

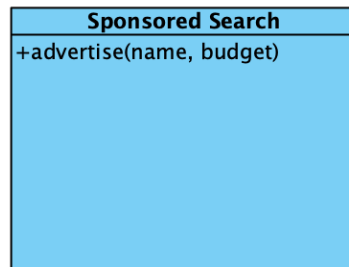
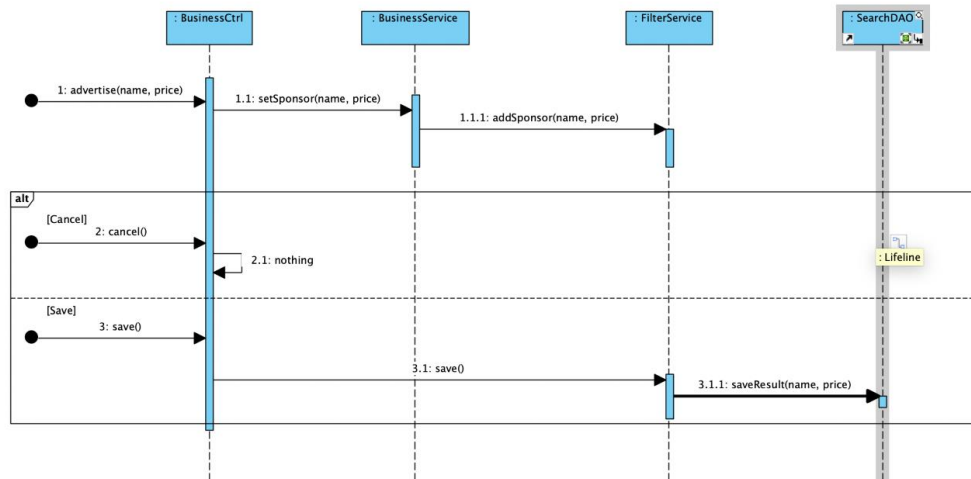
Pre-conditions: User wants to set settings

Post-condition: Settings are updated

UC Sponsored Search Results

Sponsored Search – Yi Ding

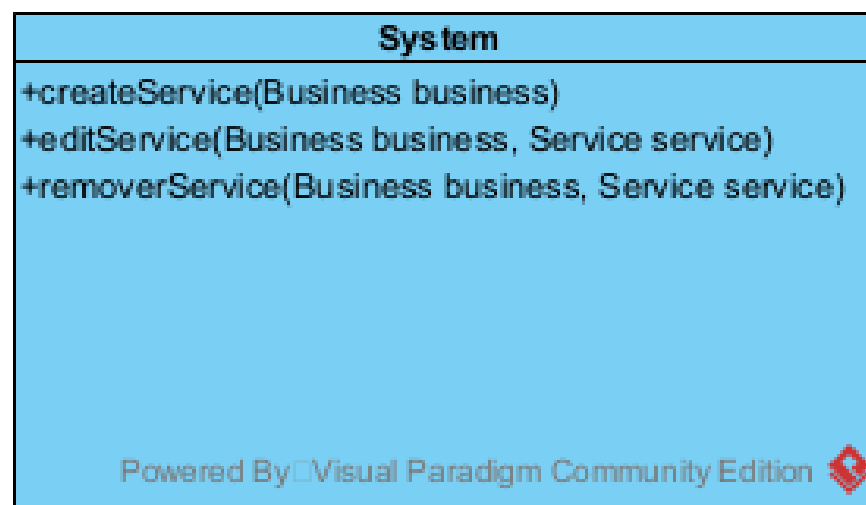
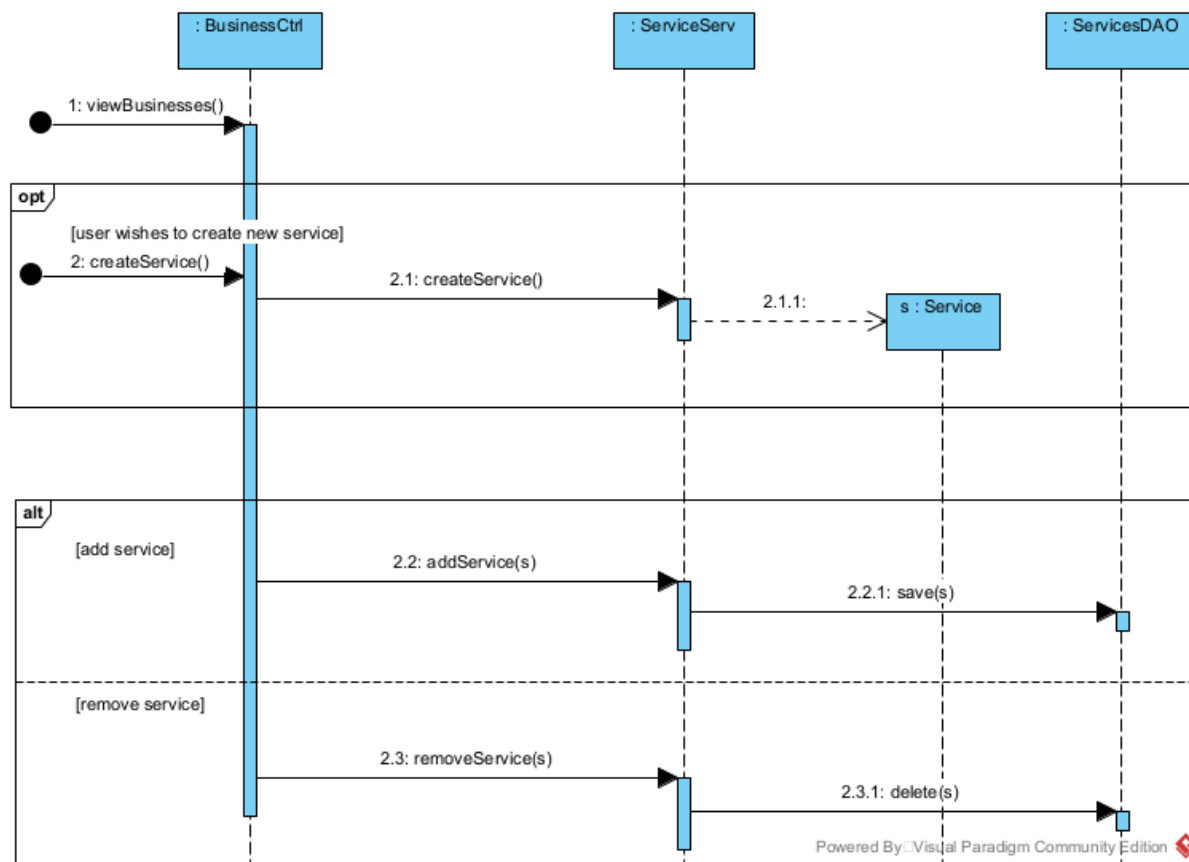
sd [Sponsored Search Result]



Contract: Sponsored Search Results
Operation: Search results are padded with sponsors
Cross Reference: Filer search results
Pre-conditions: Business User wants to advertise
Post-condition: Rankings is updated

UC Manage Service

SD Manage Service - Benjamin



Contract: createService
Operation: creates a service that is sold by a buisness and booked by users for their events
Cross Reference: Manage Service
Pre-conditions: the user wants to create a service associated with a business to which they have access.
Post-condition: a new service is associated with a buisness

Contract: editService

Operation: edits a service that is sold by a business and booked by users for their events

Cross Reference: Manage Service

Pre-conditions: the user wants to edit a service associated with a business to which they have access.

Post-condition: a pre-existing service is edited

Contract: removeService

Operation: removes a service that is sold by a business and booked by users for their events

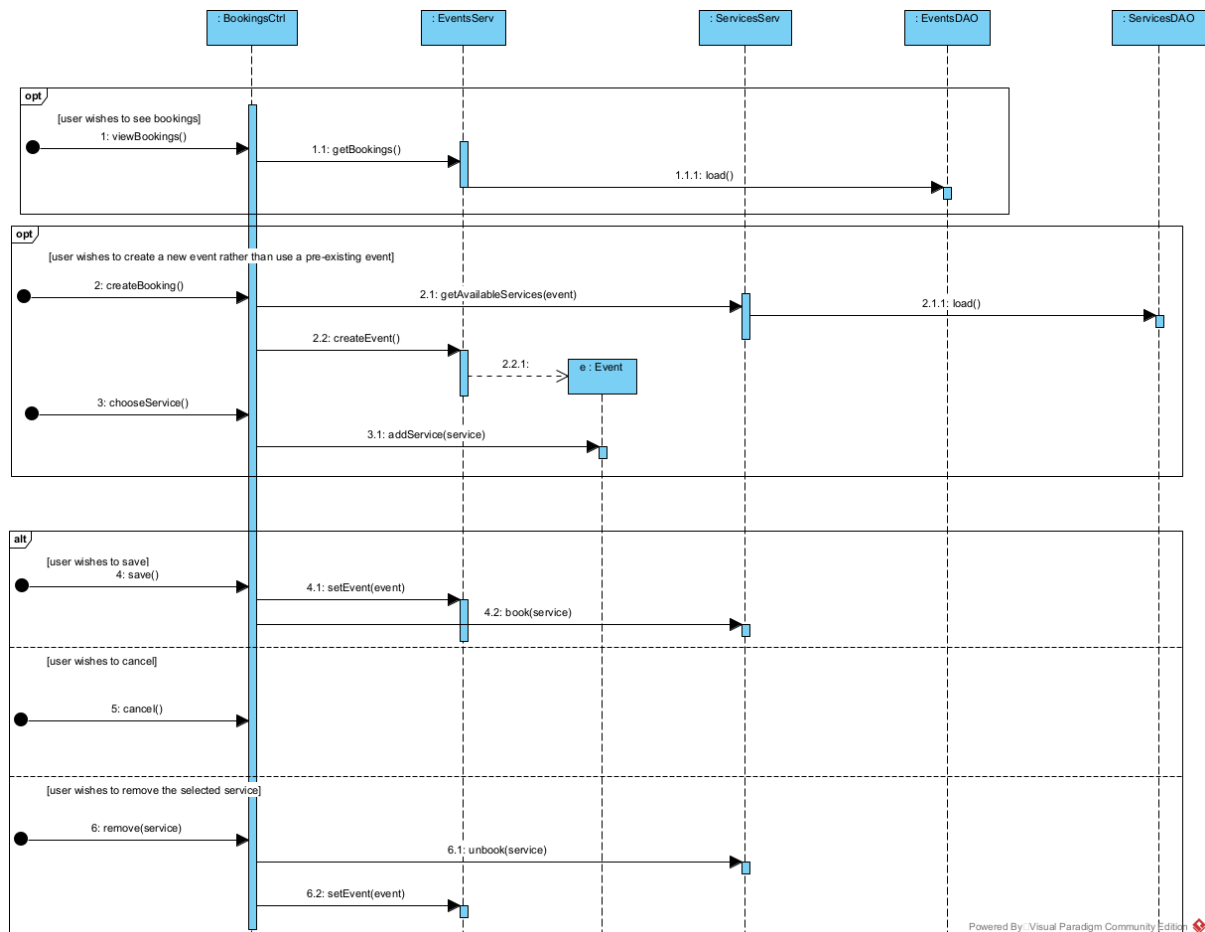
Cross Reference: Manage Service


Pre-conditions: the user wants to remove a pre-existing service associated with a business to which they have access.

Post-condition: a pre-existing service is removed from those offered by a business

UC Manage Booking

SD Manage Booking - Benjamin



System
+createBooking(Event event) +editBooking(Booking booking, Event event) +removeBooking(Booking booking, Event event)
Powered By  Visual Paradigm Community Edition

Contract: createBooking
Operation: creates a booking Cross Reference: Manage Booking Pre-conditions: User wants to make a new booking Post-condition: booking is created with some business offering some service

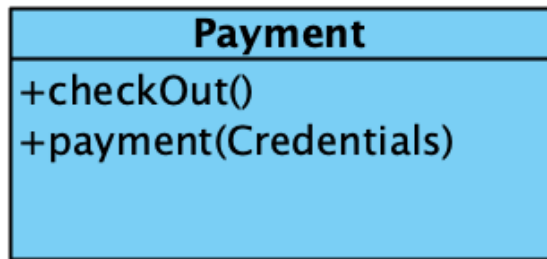
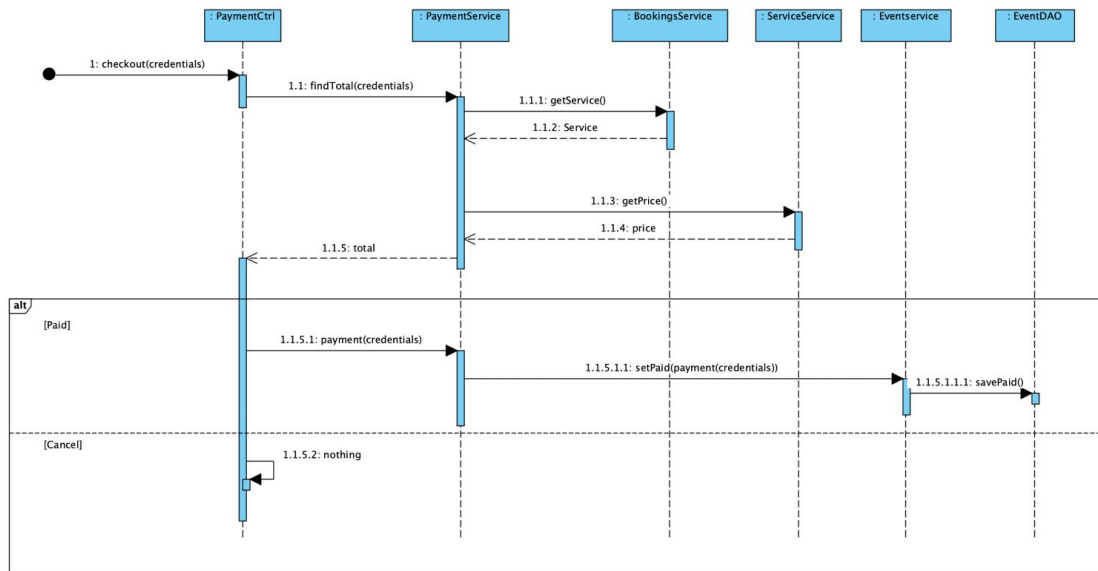
Contract: editBooking
Operation: edits a booking Cross Reference: Manage Booking Pre-conditions: User wants to edit a pre existing booking Post-condition: booking is changed buisness is notified of change

Contract: removeBooking
Operation: remove a booking Cross Reference: Manage Booking Pre-conditions: User wants to make a new booking Post-condition: booking is created

UC Payment

Payment – Yi Ding

sd [Payment]



Contract: Payment

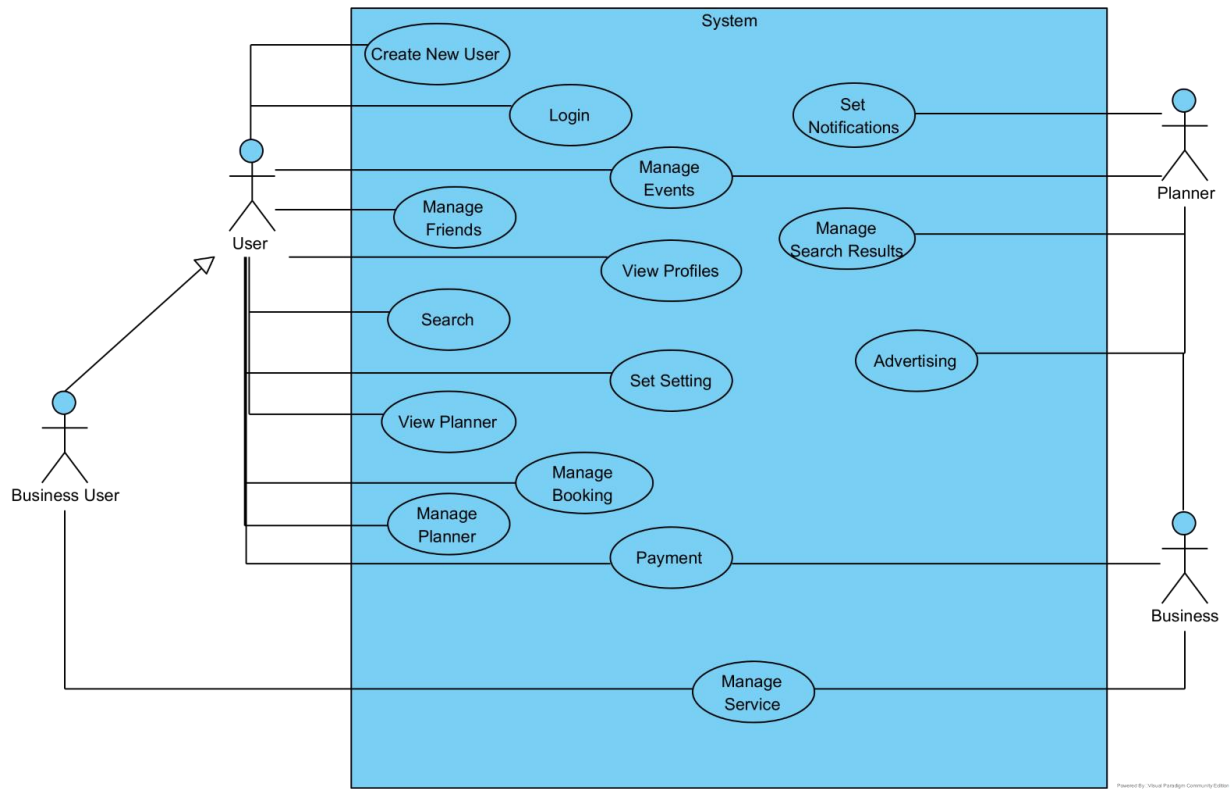
Operation: Allows User to pay total

Cross Reference: Manage service

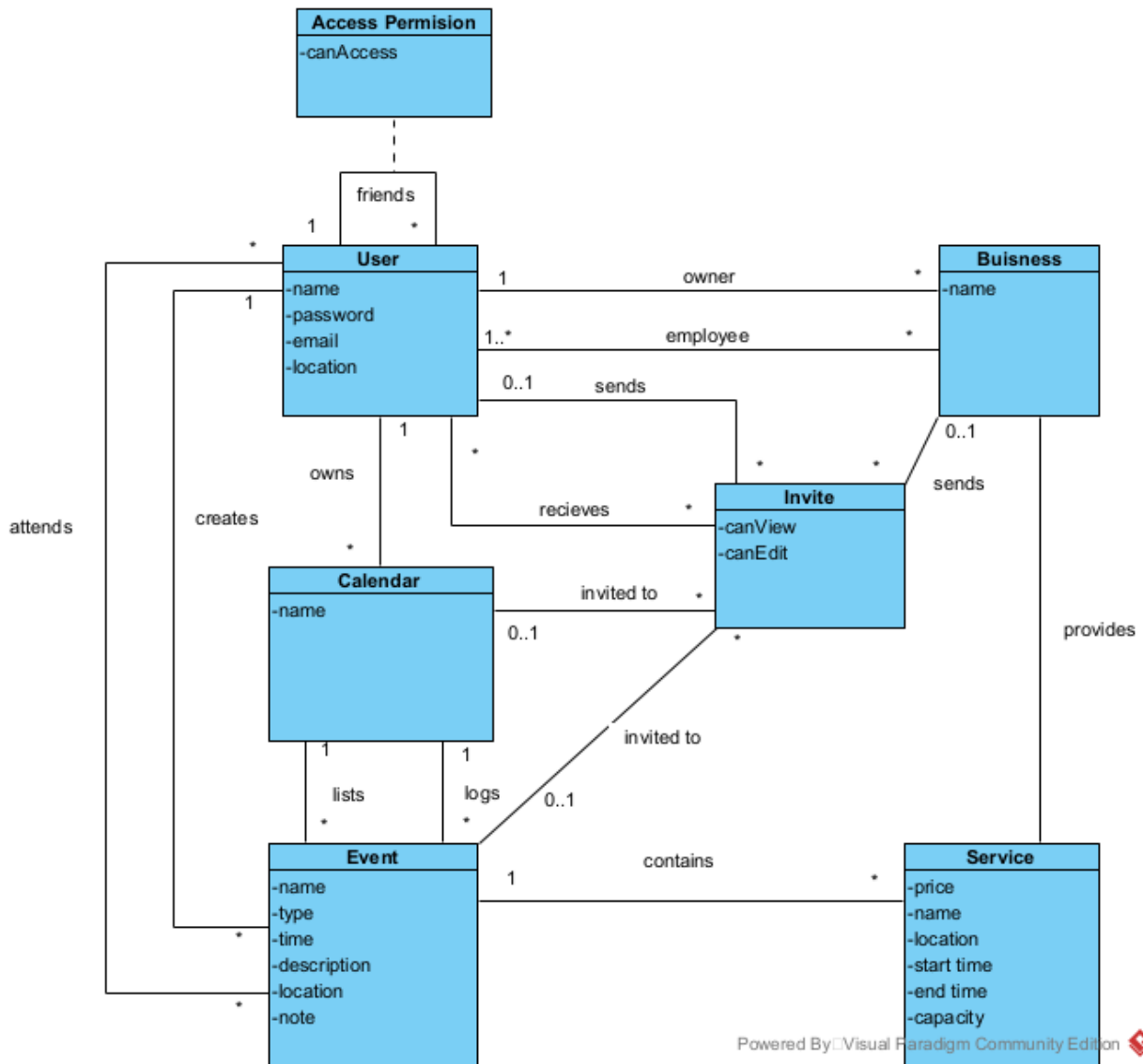
Pre-conditions: User wants to pay total

Post-condition: Total is paid

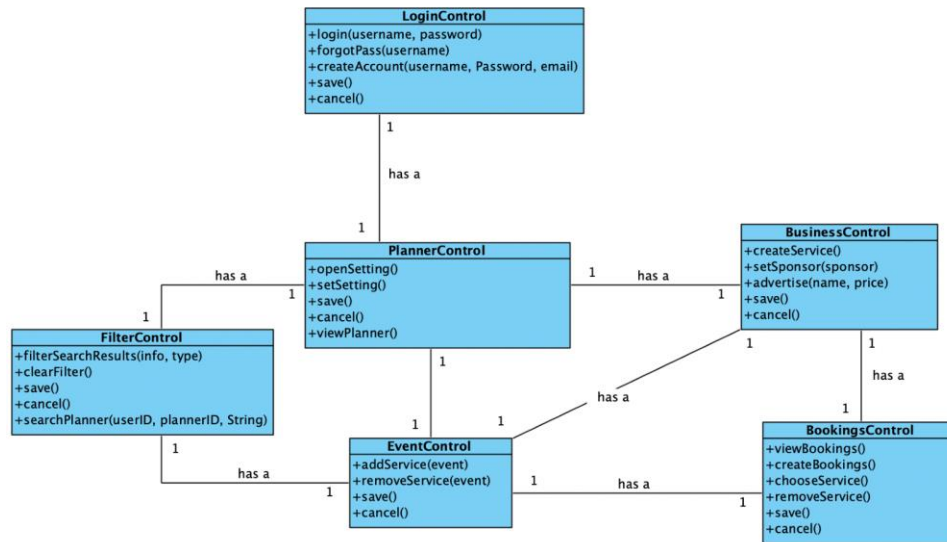
Use Case Diagram



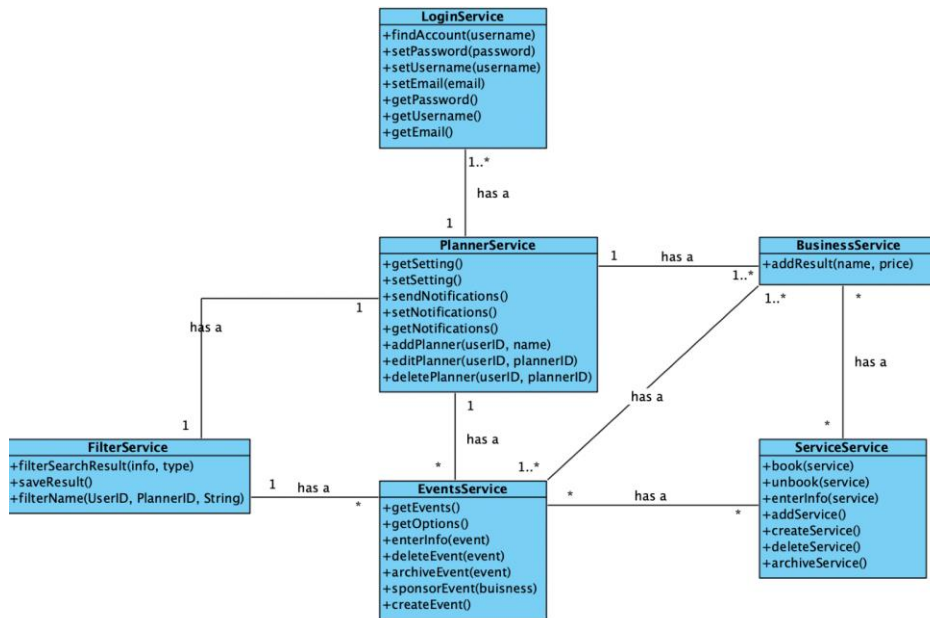
Domain Model



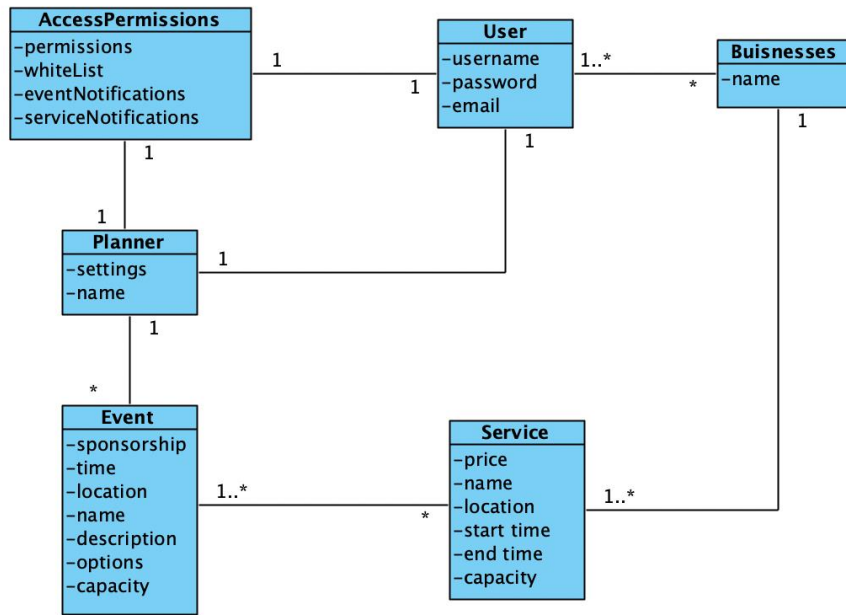
UI Design Diagram



System Design Diagram



Persistence Design Diagram



Testing Coverage

[illegible]

[illegible]

GRASP Patterns

Pattern: Information Experts

These classes are responsible for the management of other classes. The Planner allows interaction between the User and Event classes. The Event's job stores the relation between Users, Dates, and Services. The Planner's job is to store the relation between Users, Events, and Businesses. The Planner and Event classes are Information Experts, as they hold information on all the other classes. We plan to implement them so that each Planner will know what Events it has, and each Event will know the Users and Services connected to it.

Pattern: Creator

The main Creator in this software is the Planner, which creates Events, Services, Businesses, Users, and Permissions.

Pattern: Low Coupling & High Cohesion

Originally, Login and Registration functions were directly connected to the planner, but we decided to split the Login function into its own class to maintain Lower Coupling and High Cohesion. Most of our classes are self-contained, and don't require explicit knowledge of other classes to work as intended.

High coupling exists between our Planner, Event and Business classes. We have set it up this way so that we could decouple the User class from Events and Businesses and increase cohesion by separating Users from the Events and Services. The Planner has a list of Events, which also has a list of Services. Planners are directly connected to Users, so each member of Planner is, by proxy, a member of User. Both Events and Services cannot function without the existence of the User class. However, the other classes seem to be coupled only for the sake of a particular Use Case, so I believe the application has low coupling overall.

Pattern: Controller

The Controllers for this app will be the UI classes, which handle and react to input for different use cases. We chose to have controllers for Login, Planners, Filters, Business, Bookings, and Events. Bookings are closely tied to Services, so we set up a Controller to help visualize the ways in which booking works.

Pattern: Polymorphism

Polymorphism is demonstrated through our UI.

Pattern: Dynamic Binding

This exists because the UI all have `createAndShowGUI()` functions and extend `JFrame` through different implementations.

Pattern: Pure Fabrication

We have multiple classes that demonstrate this pattern. The first is the Event class, which represents the connection between Users, dates, locations, and Services. The class that

exemplifies this GRASP pattern is the Event class, which doesn't really represent anything that exists in reality. Event represents the relationship between Users and Dates.

Pattern: Indirection

Indirection is used many times throughout this software. One example is the Planner class which mediates between the Users and Events. There's also a lot of indirection through the UI, which handles User input and directs it to the other classes.

Pattern: Law of Demeter

This law is satisfied in multiple ways. Firstly, by the indirection between Users and Services, and between Services and Events.

Project manager

Project dates

Aug 22, 2022 - Dec 30, 2022

Completion

0%

Tasks

28

Resources

4

Tasks

3

Name	Begin date	End date
presentation and reporting	12/29/22	12/29/22

Tasks

2

Name	Begin date	End date
Analysis	8/22/22	10/18/22
project vision	8/22/22	8/25/22
team assembly	8/26/22	8/31/22
infrastructure initialization	9/1/22	9/6/22
requirements analysis	9/7/22	9/9/22
use cases	9/12/22	9/14/22
traceability matrix	9/15/22	9/19/22
system sequence diagrams	9/20/22	9/26/22
system operations	9/27/22	9/30/22
wireframes	10/3/22	10/5/22
domain model	10/6/22	10/11/22
presentation and reporting	10/12/22	10/18/22
Design	10/27/22	12/6/22
design model	10/27/22	11/1/22
sequence diagrams	11/2/22	11/7/22
package diagrams	11/8/22	11/10/22
GRAPS patterns	11/11/22	11/15/22
test coverage	11/16/22	11/21/22
prototyping	11/22/22	11/29/22
presentation and reporting	11/30/22	12/6/22
Implementation	12/7/22	12/29/22
backend	12/7/22	12/9/22
user interface	12/12/22	12/15/22
user input validation	12/16/22	12/20/22
Import/export	12/21/22	12/22/22
unit-testing	12/23/22	12/26/22
documentation	12/27/22	12/28/22

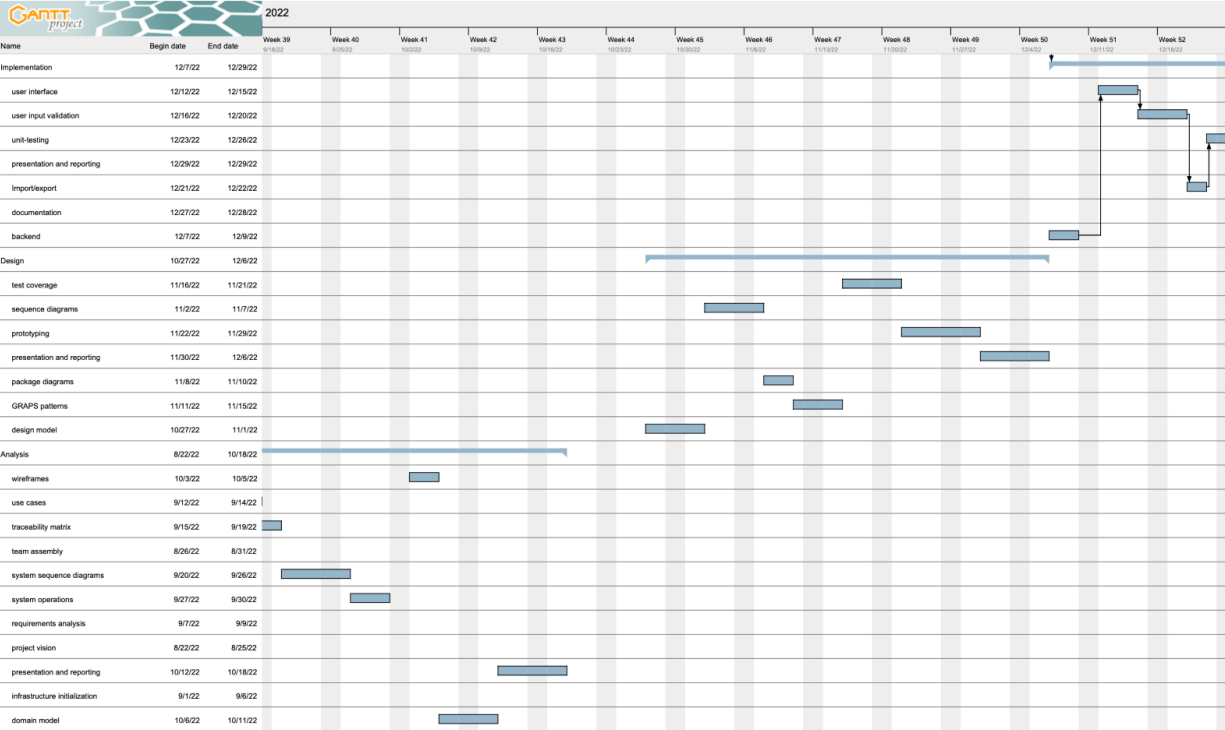
Resources

4

Name	Default role
Analyst	Analyst
Developer	Developer
Tester	Tester
Team Lead	Team Lead

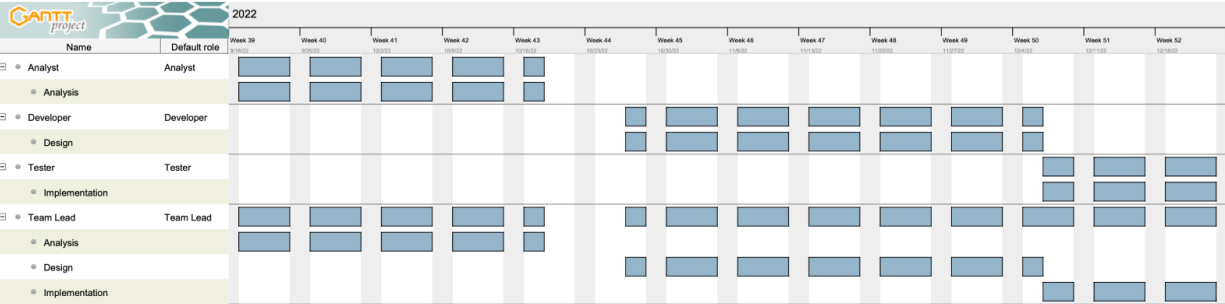
Gantt Chart

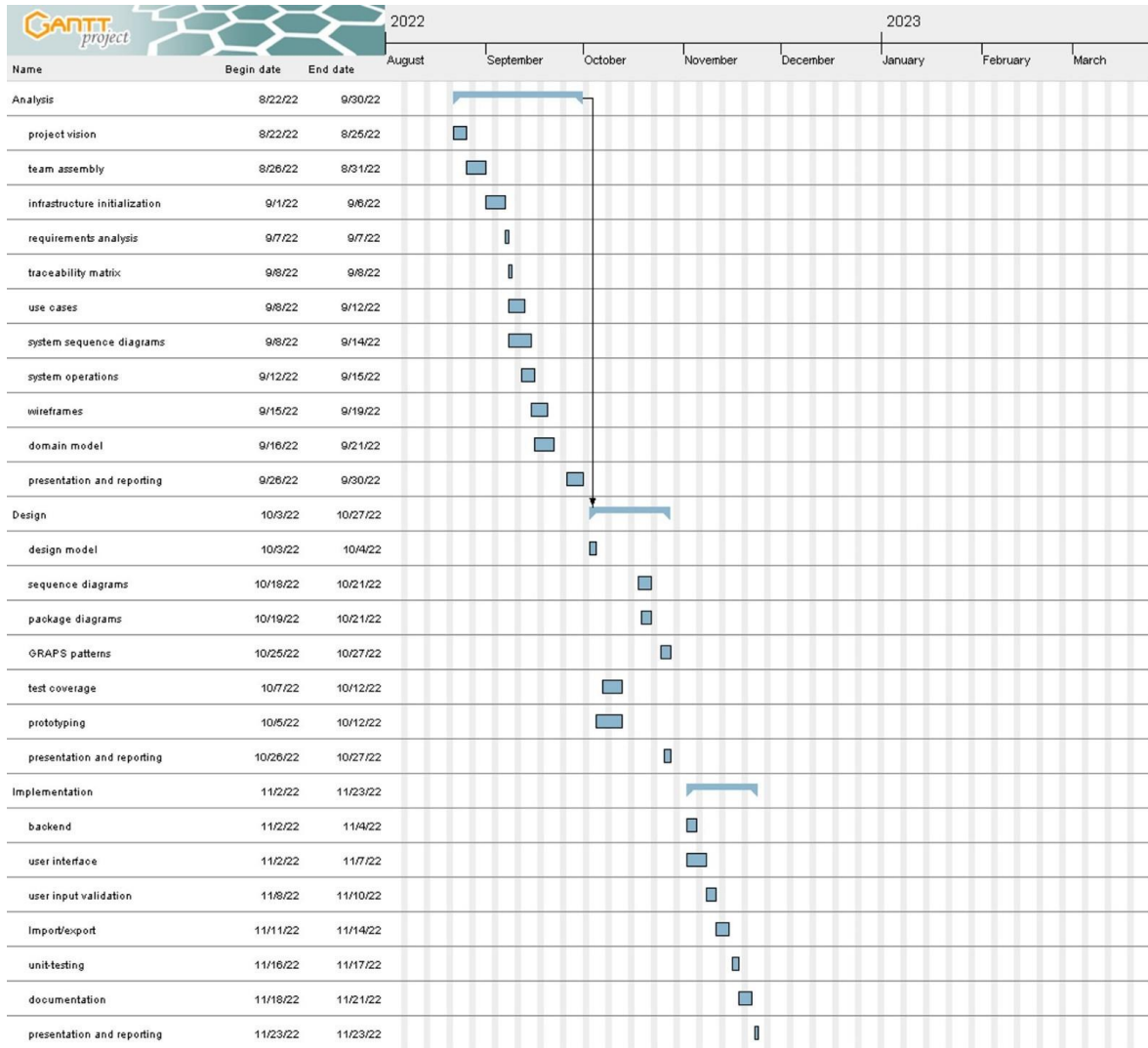
5



Resources Chart

6





Timecard

Benjamin: 22 hours

- Domain Diagram
- SSD
- Fully Dressed Use Case
- Wire frame
- Glossary
- Group Avatar

Bryce: 7 hours

- Requirements
- Use Case
- SSD
- Fully Dressed Use Case

Jason: 23 hours

- **System Operations**
- **Traceability matrix**
- **Fully Dressed Use Case**
- **Requirements**
- **Wire frame**
- **Project Planning**
- **Project Time Management**
- **SSD**
- **Fully Dressed Use Case**
- **Trello**
- **Project vision**
- **Use Case Diagram**

Yutai: 9 hours

- Gantt Diagram
- Use Case
- SSD
- Fully Dressed Use Case

Yi: 21 hours


- Website
- PPT
- Use Case
- SSD
- Fully Dressed Use Case
- Trello

GitHub

Git Commit Tracking from IntelliJ

The screenshot displays the GitHub Desktop interface with a dark theme. At the top, there's a search bar and navigation controls. The main area shows a commit history list on the right and a graphical commit log on the left. The commit log uses colored lines to represent branches and dots for commits. The commit history list includes details like the commit message, author, date, and time taken.

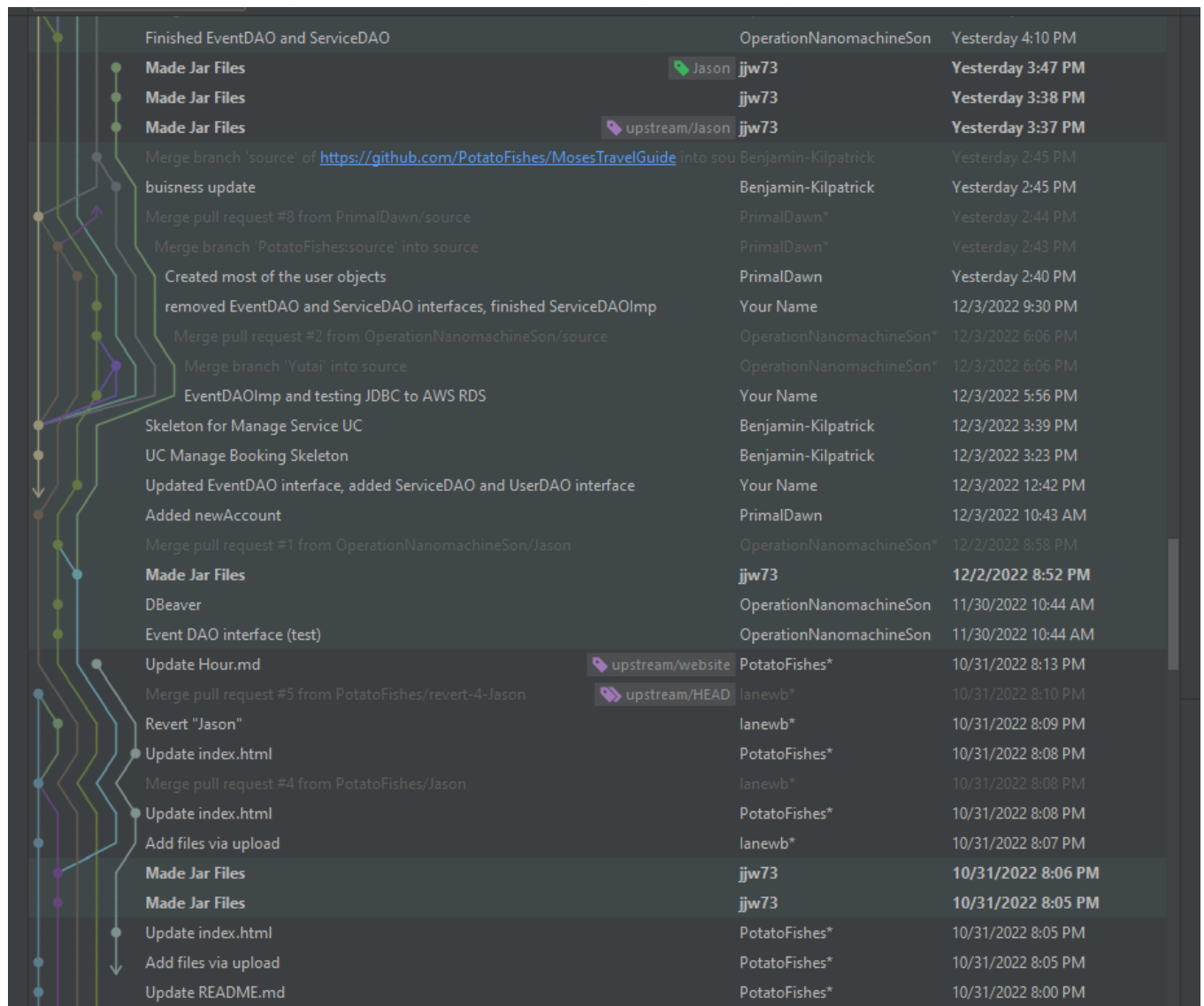
Commit Message	Author	Date	Time Taken
Merge remote-tracking branch 'upstream/source' into upstream & source	jww73	19 minutes ago	
changed EditEventDialog	jww73	19 minutes ago	
Merge pull request #25 from PrimalDawn/source	PrimalDawn*	33 minutes ago	
changed planner layout	PrimalDawn	57 minutes ago	
InviteEvents almost done	PrimalDawn	Today 3:32 PM	
changed ServiceDAOImp	jww73	Today 2:13 PM	
Merge remote-tracking branch 'upstream/source' into source	jww73	Today 2:13 PM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 2:08 PM	
fixed settings	PotatoFishes	Today 2:08 PM	
Merge pull request #24 from PrimalDawn/source	PrimalDawn*	Today 1:56 PM	
inviteEvents	PrimalDawn	Today 1:54 PM	
Merge pull request #23 from PotatoFishes/Yutai	OperationNanomachineSon*	Today 1:51 PM	
Changed up NotificationService	OperationNanomachineSon	Today 1:37 PM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 1:29 PM	
Merge pull request #22 from PotatoFishes/Yutai	OperationNanomachineSon*	Today 1:28 PM	
Edit EventDAOImp and SearchService	OperationNanomachineSon	Today 1:28 PM	
rename	PotatoFishes	Today 1:28 PM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 1:23 PM	
Email works	PotatoFishes	Today 1:23 PM	
Merge remote-tracking branch 'upstream/source' into source	jww73	Today 1:20 PM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 1:14 PM	
BusinessUI update	Benjamin-Kilpatrick	Today 1:12 PM	
edit errors in preferencesdao	OperationNanomachineSon	Today 1:11 PM	
Merge pull request #21 from PotatoFishes/Yutai	OperationNanomachineSon*	Today 1:09 PM	
Merge branch 'Yutai' of https://github.com/PotatoFishes/MosesTravelGuide into Yutai	OperationNanomachineSon	Today 1:08 PM	
New Preferences class	OperationNanomachineSon	Today 1:08 PM	
Merge remote-tracking branch 'origin/source' into Yutai	OperationNanomachineSon	Today 12:48 PM	
Adding some more fuctions to EventDAO and ServiceDAO	OperationNanomachineSon	Today 12:47 PM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 12:12 PM	
BusinessUI update	Benjamin-Kilpatrick	Today 12:12 PM	
changed ServiceDAOImp	jww73	Today 11:52 AM	
changed AddServiceDialog, EditEventDialog, ServiceDAOImp, ServiceServ	jww73	Today 11:47 AM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 11:22 AM	
Finished Search Until Dao	PotatoFishes	Today 11:22 AM	
changed AddServiceDialog, EditEventDialog, ServiceServ	jww73	Today 11:01 AM	
Merge pull request #20 from PotatoFishes/source	OperationNanomachineSon*	Today 10:40 AM	
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 10:19 AM	
User can naw view incoming friend info	Benjamin-Kilpatrick	Today 10:18 AM	



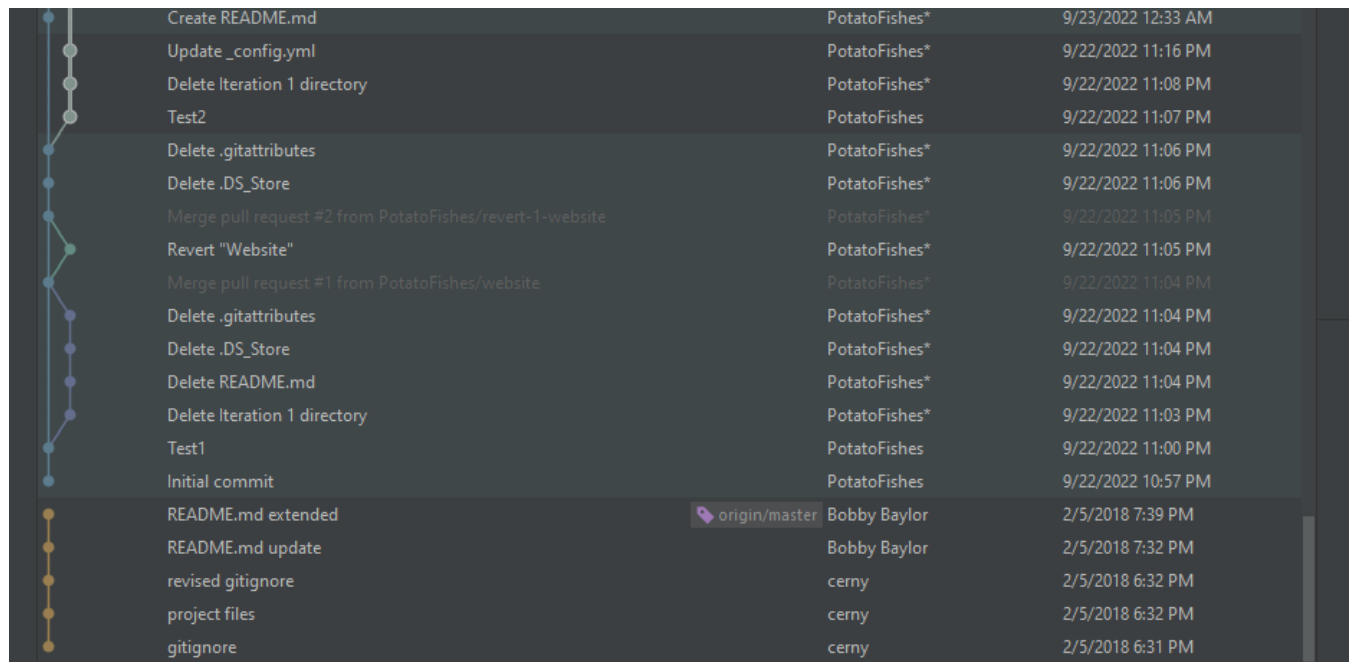
User can now view incoming friend info	Benjamin-Kilpatrick	Today 10:18 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 9:24 AM
Finished Payment	PotatoFishes	Today 9:24 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 9:10 AM
Implemented the Follower class	Benjamin-Kilpatrick	Today 9:08 AM
changed EventsDAOImp, EventsServ, Planner,	jjw73	Today 8:38 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 8:11 AM
blah	Benjamin-Kilpatrick	Today 7:46 AM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Today 7:22 AM
changed AddEvent, EditEventDialog, Event EventsDAOImp, EventsServ, Login, Ne	jjw73	Today 7:19 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	PotatoFishes	Today 6:58 AM
Made Settings, Missing database access	PotatoFishes	Today 6:57 AM
Added BusinessUI	Benjamin-Kilpatrick	Today 5:39 AM
Remove friends from database	Benjamin-Kilpatrick	Today 5:19 AM
User is limited to only add friends who exist	Benjamin-Kilpatrick	Today 5:01 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 4:50 AM
Users can now add followers.	Benjamin-Kilpatrick	Today 4:49 AM
changed AddEvent, EditEventDialog, EventsDAOImp, EventsServ Planner UserDA	jjw73	Today 4:49 AM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Today 2:51 AM
changed AddEvent (Added EventService.createEvent), AddServiceDialog (tryna fix	jjw73	Today 2:50 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 2:45 AM
FriendsManagerUI update but still not finished	Benjamin-Kilpatrick	Today 2:44 AM
Merge pull request #19 from PotatoFishes/Yutai	OperationNanomachineSon*	Today 1:17 AM
InviteDAO (half way done, too sleepy, going to finish morning)	OperationNanomachineSon	Today 1:16 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Today 12:17 AM
Removed the ability to change id of event manually	Benjamin-Kilpatrick	Today 12:16 AM
changed AddEvent, AddServiceDialog, EditEventDialog, Event, Planner, Service	jjw73	Today 12:16 AM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Today 12:01 AM
changed AddEvent, AddServiceDialog, EditEventDialog, Event, Planner, Service	jjw73	Today 12:00 AM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 11:55 PM
Fix the UserDao	Benjamin-Kilpatrick	Yesterday 11:53 PM
Merge pull request #18 from PotatoFishes/Yutai	OperationNanomachineSon*	Yesterday 11:08 PM
fix UserLoginService	Benjamin-Kilpatrick	Yesterday 11:00 PM

fix UserLoginService	Benjamin-Kilpatrick	Yesterday 11:00 PM
Fix UserDAO query	OperationNanomachineSon	Yesterday 10:57 PM
Merge pull request #17 from PotatoFishes/source	OperationNanomachineSon*	Yesterday 10:32 PM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Yesterday 10:21 PM
re-push committed	jjw73	Yesterday 10:21 PM
Merge pull request #16 from PotatoFishes/Yutai	OperationNanomachineSon*	Yesterday 10:19 PM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Yesterday 10:17 PM
Merge remote-tracking branch 'origin/source' into Yutai Fixing some SQL lines and Usi	OperationNanomachineSon	Yesterday 10:17 PM
AddServiceDialog changed EditEventDialog changed	jjw73	Yesterday 10:16 PM
Fixing SQL queries	OperationNanomachineSon	Yesterday 10:16 PM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 10:14 PM
Implemented Booking	Benjamin-Kilpatrick	Yesterday 10:14 PM
Merge remote-tracking branch 'upstream/source' into source	jjw73	Yesterday 9:56 PM
EditEventDialog added, AddServiceDialog changed, EditEventDialog changed	jjw73	Yesterday 9:55 PM
Merge remote-tracking branch 'origin/source' into Yutai Includ an updated version of t	OperationNanomachineSon	Yesterday 9:54 PM
EditEventDialog added, AddServiceDialog changed, EditEventDialog changed	jjw73	Yesterday 9:54 PM
Updated UserDAO, included a follow table.	OperationNanomachineSon	Yesterday 9:53 PM
Merge pull request #15 from PrimalDawn/source	PrimalDawn*	Yesterday 9:52 PM
sql error	PrimalDawn	Yesterday 9:50 PM
Load events from database when starting the app	Benjamin-Kilpatrick	Yesterday 8:58 PM
Updated the UserLoginService	Benjamin-Kilpatrick	Yesterday 8:18 PM
Created user Login Service to store user info	Benjamin-Kilpatrick	Yesterday 8:06 PM
EditEventDialog added	jjw73	Yesterday 7:39 PM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 5:54 PM
Event added new constructor for no event ID input	jjw73	Yesterday 5:54 PM
springUtils 7 rows 1 col	jjw73	Yesterday 5:47 PM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 5:43 PM
Merge pull request #14 from PrimalDawn/source	PrimalDawn*	Yesterday 5:38 PM
Finished UserDAO	OperationNanomachineSon	Yesterday 5:37 PM
manageEvents part 2	PrimalDawn	Yesterday 5:35 PM
Merge branch 'PotatoFishes:source' into source	PrimalDawn*	Yesterday 5:23 PM
mangaeEvents class	PrimalDawn	Yesterday 5:22 PM
Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 5:18 PM

Merge branch 'source' of https://github.com/PotatoFishes/MosesTravelGuide into sou	Benjamin-Kilpatrick	Yesterday 5:17 PM
Merge pull request #13 from PotatoFishes/Yutai	OperationNanomachineSon*	Yesterday 5:16 PM
Delete test2.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete test.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete roundBorder.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete UserDao.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete User.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete SpringUtilities.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete ServicesServ.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete ServiceDAOImp.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete Service.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete Planner.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete Login.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete NewAccount.java	OperationNanomachineSon*	Yesterday 5:14 PM
Delete EventsServ.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete EventUI.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete EventDAOImp.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete Event.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete EditDialog.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete ButtonColumn.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete Booking.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete AddServiceDialog.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete AddLineDialog.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete AddEvent.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete AddBusiness.java	OperationNanomachineSon*	Yesterday 5:13 PM
Delete AbstractTableAction.java	OperationNanomachineSon*	Yesterday 5:12 PM
Merge branch 'Yutai' of https://github.com/PotatoFishes/MosesTravelGuide into Yutai	OperationNanomachineSon	Yesterday 5:09 PM
Add files via upload	OperationNanomachineSon*	Yesterday 5:05 PM
Finished UserDao	OperationNanomachineSon	Yesterday 4:51 PM
Friends management and other updates	Benjamin-Kilpatrick	Yesterday 4:25 PM
UserDao edit	OperationNanomachineSon	Yesterday 4:21 PM
Merge pull request #3 from PotatoFishes/source	OperationNanomachineSon*	Yesterday 4:11 PM
Merge branch 'Yutai' into source	OperationNanomachineSon*	Yesterday 4:11 PM



	Made Jar Files	jjw73	10/31/2022 7:58 PM
	Add files via upload	PotatoFishes*	10/31/2022 7:57 PM
	Add files via upload	OperationNanomachineSon*	10/31/2022 7:33 PM
	Add files via upload	lanewb*	10/31/2022 4:33 PM
	Add files via upload	PrimalDawn*	10/30/2022 11:51 PM
	Add files via upload	PrimalDawn*	10/30/2022 11:50 PM
	Add files via upload	PotatoFishes*	10/30/2022 11:05 PM
	Delete DesignDiagram.vpp	PotatoFishes*	10/30/2022 11:05 PM
	Add files via upload	PotatoFishes*	10/30/2022 8:19 PM
	Delete RevisedITR1.pdf	PotatoFishes*	10/30/2022 2:09 PM
	Updated Login, AddLineDialog, EditDialog, Event, Login, Planner, and Service	jjw73	10/28/2022 10:39 AM
	Merge pull request #3 from PrimalDawn/source	PrimalDawn*	10/28/2022 9:14 AM
	A waste of time.	PrimalDawn	10/28/2022 9:11 AM
	Add files via upload	PotatoFishes*	10/28/2022 5:15 AM
	Create README.md	PotatoFishes*	10/28/2022 5:15 AM
	Add files via upload	PotatoFishes*	10/28/2022 3:38 AM
	Update README.md	PotatoFishes*	10/28/2022 3:26 AM
	Update README.md	PotatoFishes*	10/28/2022 3:26 AM
	Add files via upload	PotatoFishes*	10/28/2022 3:19 AM
	Create README.md	PotatoFishes*	10/28/2022 3:19 AM
	Added Settings Button	jjw73	10/28/2022 2:54 AM
	Added Settings Button	jjw73	10/28/2022 2:46 AM
	Updated Service Class	jjw73	10/26/2022 2:15 PM
	Updated Service Class	jjw73	10/26/2022 1:33 PM
	Updated Planner, Event, and User Classes Added AddLineDialog, AbstractTableActi	jjw73	10/26/2022 5:23 AM
	Added stuff from Saturday 10/1/2022	jjw73	10/4/2022 11:43 AM
	Added Stuf worked on Saturday	jjw73	10/4/2022 11:36 AM
	added folders for project using .keep	Benjamin-Kilpatrick	9/30/2022 7:58 PM
	added Java Maven project	Benjamin-Kilpatrick	9/30/2022 5:21 PM
	Update index.html	PotatoFishes*	9/25/2022 10:27 PM
	Update index.html	PotatoFishes*	9/25/2022 10:23 PM
	Add files via upload	PotatoFishes*	9/25/2022 10:22 PM
	Delete UseCasesCSI3471.pdf	PotatoFishes*	9/25/2022 10:22 PM



Total Commits:

Jason (jjw73 + ianewb): 47

Will (PotatoFishes): 46

Benjamin (Benjamin-Kilpatrick): 40

Yutai (OperationNanoMachineSon + Your Name): 38

Bryce (PrimalDawn): 20

Hours:

Jason – 25 hrs

Benjamin – 27 hrs

Yutai – 20 hrs

Will – 20

Bryce – 20