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. We may add or change some features in the future, but this is the baseline we have so far.

Requirements

Functional

1. Events should interact with each other
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 calendars
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

Use Cases	Rq 1	Rq 2	Rq 3	Rq 4	Rq 5	Rq 6	Rq 7	Rq 8	Rq 9	Rq 10	Rq 11	Rq 12	Rq 13	Rq 14	Rq 15	Owner
UC Create New User				🟩			🟩									B C
UC Login							🟩									B C
UC Manage Events	🟩		🟩		🟩			🟩	🟩		🟩				🟩	B C
UC Manage Friends Access						🟩						🟩				B K
UC Filter search results					🟩	🟩		🟩						🟩		J W
UC Set Notifications	🟩		🟩										🟩			J W
UC View Profiles				🟩		🟩						🟩				J W
UC Search Planner					🟩									🟩		Y X
UC Set Setting													🟩			Y D
UC Sponsored Search Results		🟩			🟩										🟩	Y D
UC View Planner						🟩		🟩		🟩	🟩					Y X
UC Manage Services									🟩						🟩	B K
UC Manage Booking	🟩		🟩					🟩	🟩	🟩	🟩				🟩	B K
UC Manage Planner						🟩			🟩	🟩	🟩					Y X
UC Payment															🟩	Y D

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.

Business User

- Service providers

Precondition

- User does not have an account

Postcondition

- A User is created

Main success scenario

1. User selects account type
2. User provides Credentials
3. User is updated

Extensions

a.* Anytime User exits account creation

1. User sent to desktop

1a. User selects account type User

1. User provides Credentials

2. User is updated

1b. User selects account type Business User

1. Business User provides Credentials

2. Business User is updated

ID UC Login

Created By Bryce Clark

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 log in

Postcondition

- User is logged in

Main success scenario

1. User selects login type
2. User provides credentials
3. User is logged in

Extensions

a.* anytime User exits login

1. User sent back to desktop

1a. User selects User login

    1. User provides credentials

    2. User is logged into Planner

1b. User selects Business User login

    1. User provides credentials

    2. Business User is logged into Business Planner

ID UC Manage Events

Created By Bryce Clark

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 an event

Postcondition

- Event is updated

Main success scenario

1. User selects manage events
2. User picks option
3. Option is executed

Extensions

a.* anytime User exits manage event

1. Event is restored before User made any edits

1a. No Events Exist

1. User creates an event

2. User input information

3. Planner is updated

2a. User selects add service

1. User inputs information

2. Planner is updated

2b. User wants to Edit the Event

1. User selects event to edit

2. User inputs information

3. Planner is updated

2b.2a. Edits result in event conflict

1. User receives warning message

2. Event is restored to before user made any edits

2c. User wants to Delete the Event

1. User selects event to delete

2. User confirms deletion

3. planner is updated

2d. User wants to Archive the Event

1. User select event to archive

2. User confirms archive

3. Planner is updated

2e. Business User wants to Sponsor an Event

1. Business User adds sponsorship to event

2. Planner is updated

ID UC Manage Friends Access

Created By

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 premission

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 View 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 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

Buisness owner

- Wants to make profit by advertising services

Buisness employee

- Wants to advertise services that they provide

Precondition

- User with buisness 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

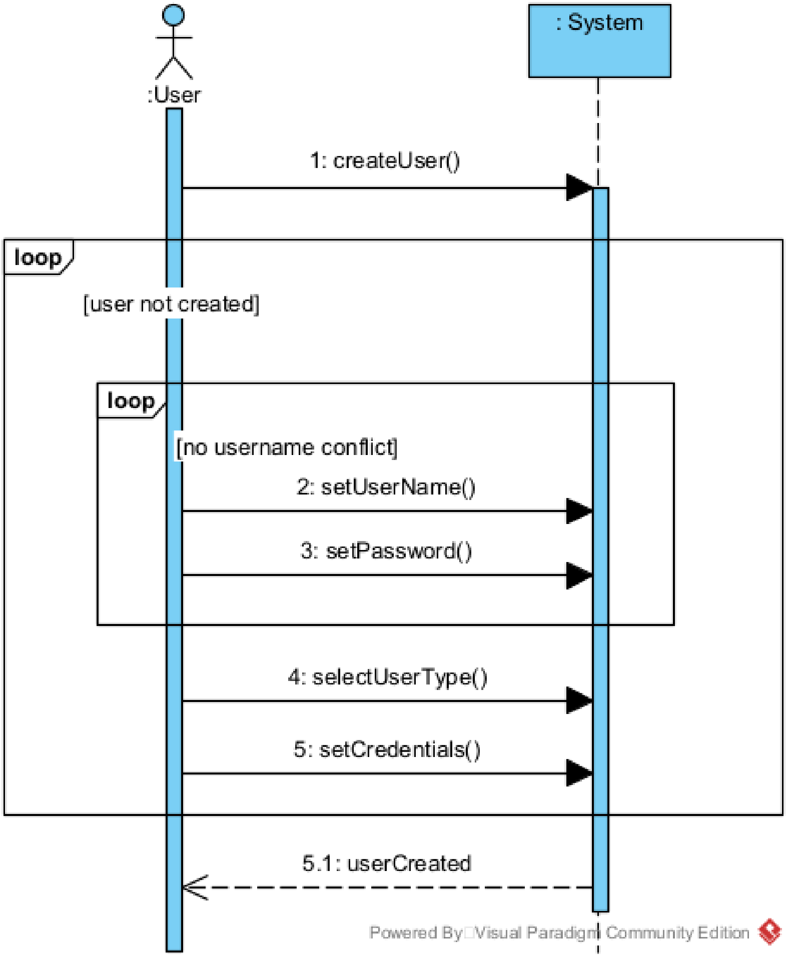
2a. User cannot afford the total

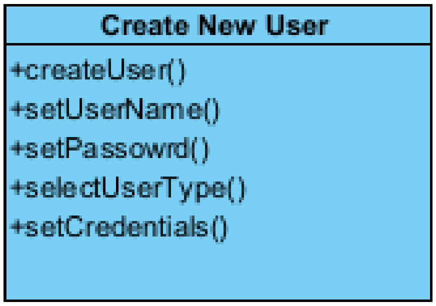
1. User receives a warning message

2. User ends action

SSD and Operation Contracts

UC Create New User

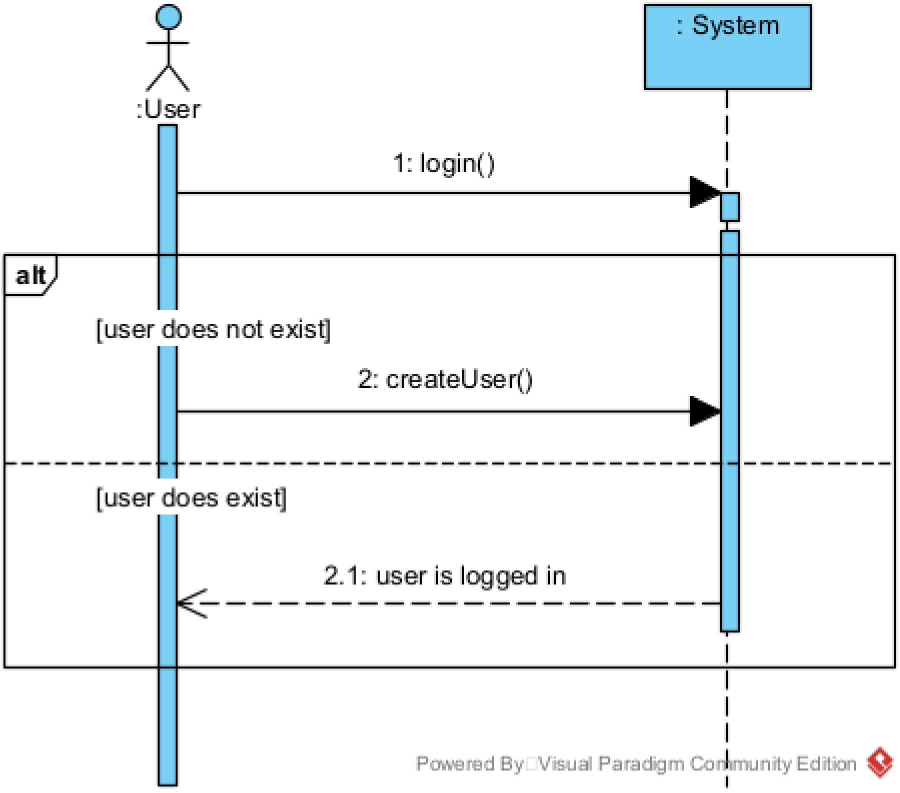


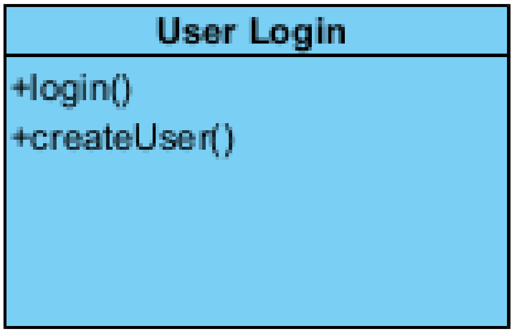


Contract: createUser
Operation: begins the create new user process Cross Reference: user login Pre-conditions: A user does not have an account. Post-condition: A user is created.
Contract: setUserName
Operation: sets the username credential of the new user Cross Reference: user login

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: selectUserType()
Operation: selects the type of a new user Cross Reference: user login Pre-conditions: A new user account does not have a type. Post-condition: A user password is created.
Contract: setCredentials()
Operation: sets the credentials to the values given in the previous functions Cross Reference: user login Pre-conditions: A new user account has no credentials Post-condition: A new user’s credentials are set

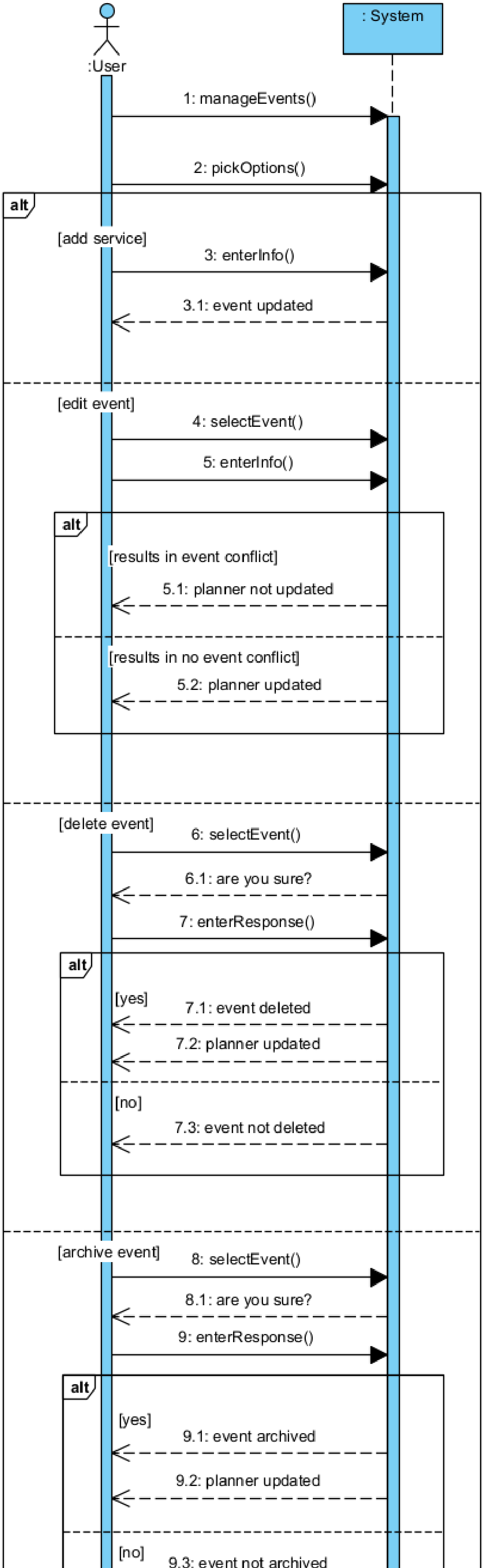
UC Login

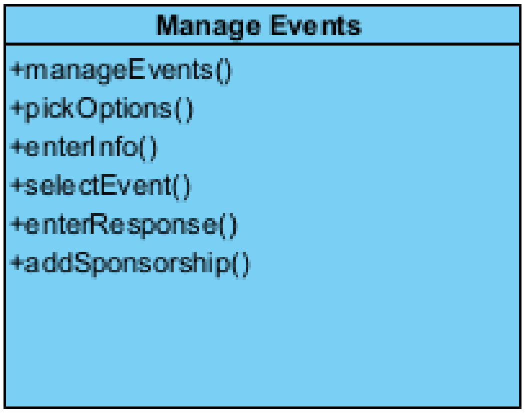




Contract: Login
Operation: user login Cross Reference: Login Pre-conditions: User wants to login. Post-condition: User is logged in.
Contract: createUser
Operation: begins process for new user creation Cross Reference: Login Pre-conditions: User wants to login, but has no account Post-condition: User has an account, may or may not be logged in

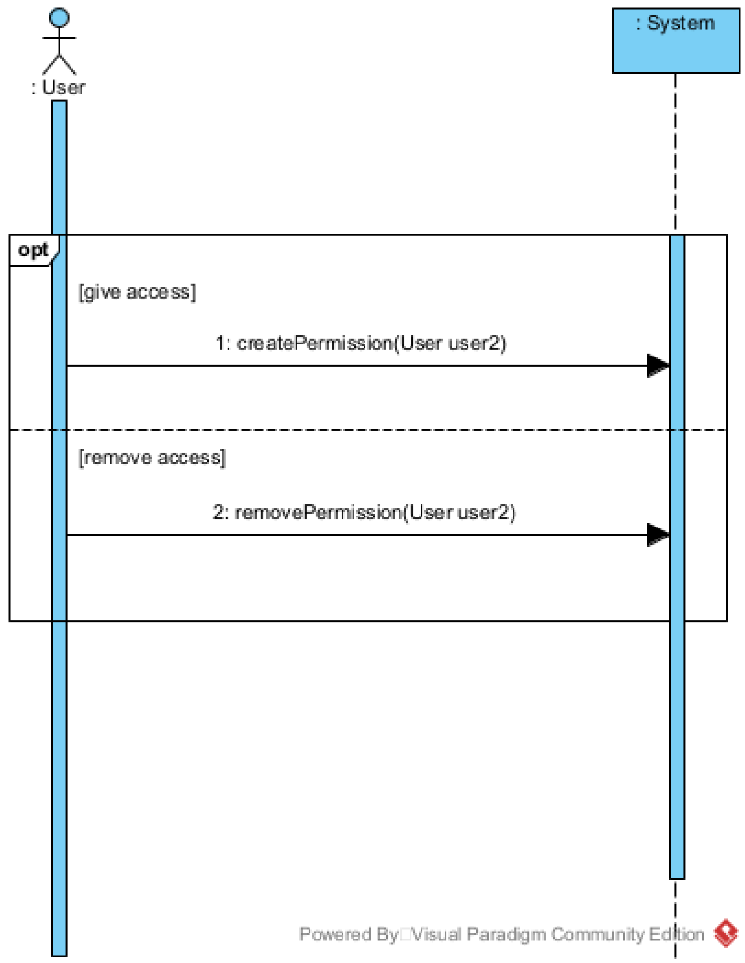
UC Manage Events

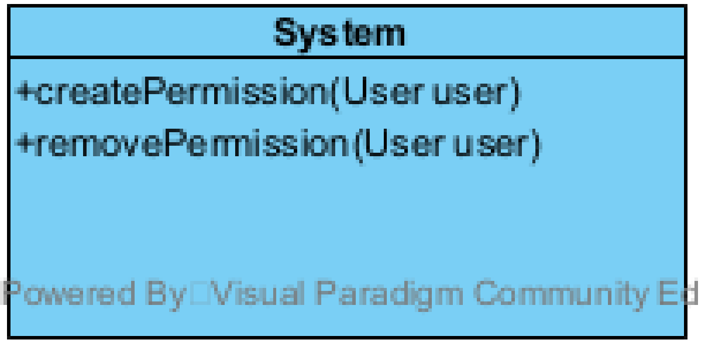

:User
: System
1: manageEvents()
2: pickOptions()
alt
[add service]
3: enterInfo()
3.1: event updated
[edit event]
4: selectEvent()
5: enterInfo()
alt
[results in event conflict]
5.1: planner not updated
[results in no event conflict]
5.2: planner updated
[delete event]
6: selectEvent()
6.1: are you sure?
7: enterResponse()
alt
[yes]
7.1: event deleted
7.2: planner updated
[no]
7.3: event not deleted
[archive event]
8: selectEvent()
8.1: are you sure?
9: enterResponse()
alt
[yes]
9.1: event archived
9.2: planner updated
[no]
9.3: event not archived



Contract: Manage Events
Operation: begins the manage event process Cross Reference: search, view planner Pre-conditions: User wants to manage event. Post-condition: Event is updated.
Contract: pickOptions()
Operation: user chooses an event management option Cross Reference: search, view planner Pre-conditions: User wants to manage an event. Post-condition: Management method is chosen
Contract: enterInfo()
Operation: user inputs information needed for the type of event managment Cross Reference: search, view planner Pre-conditions: User selected event type Post-condition: Event information is updated
Contract: selectEvent()
Operation: user chooses an event management option Cross Reference: search, view planner Pre-conditions: User wants to manage an event. Post-condition: Management method is chosen
Contract: enterResponse()
Operation: user responds to the confirmation prompt Cross Reference: search, view planner Pre-conditions: Event has information to be filled in. Post-condition: User response is recorded
Contract: addSponsorship()
Operation: user adds a sponsorship to an event Cross Reference: search, view planner Pre-conditions: User wants to sponsor an event. Post-condition: Business sponsorship is added to an event

UC Manage Friends Access

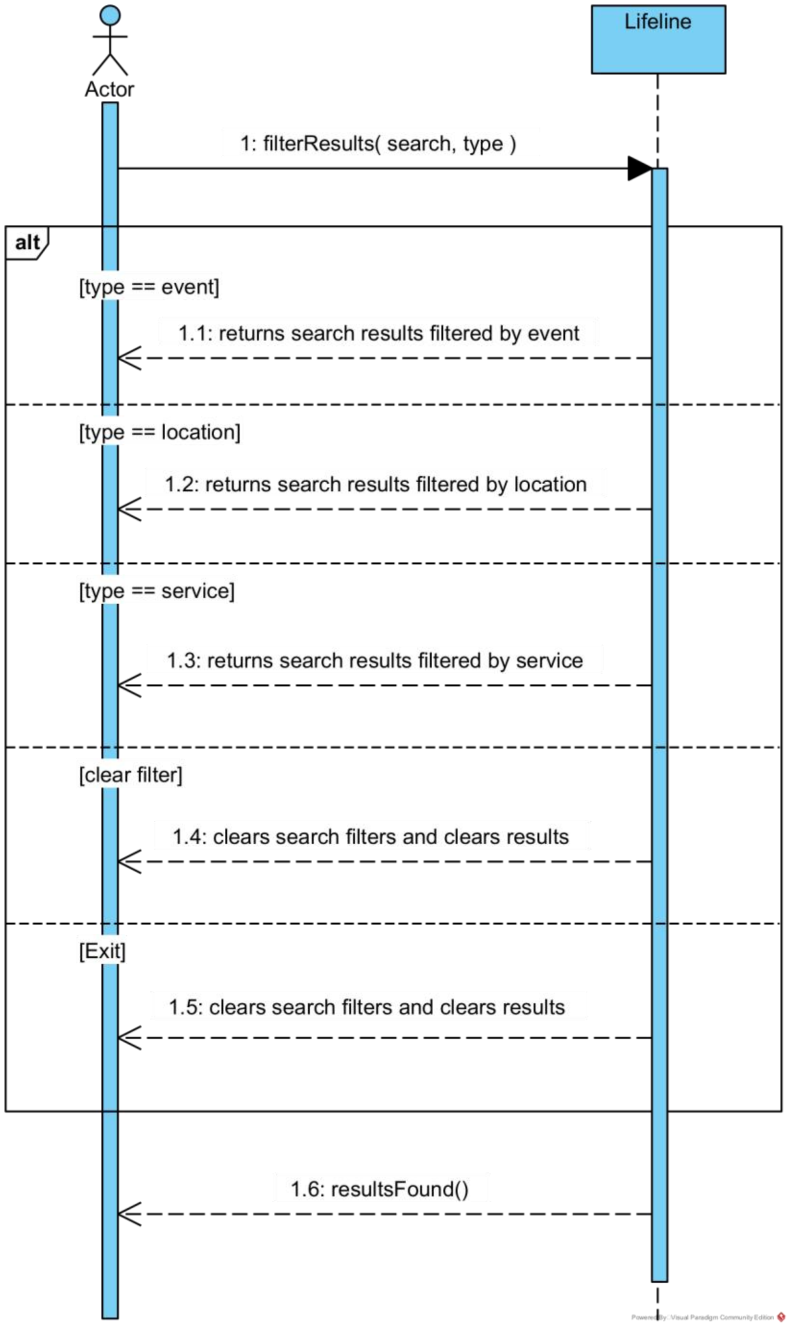

: User
: System
opt
[give access]
1: createPermission(User user2)
[remove access]
2: removePermission(User user2)

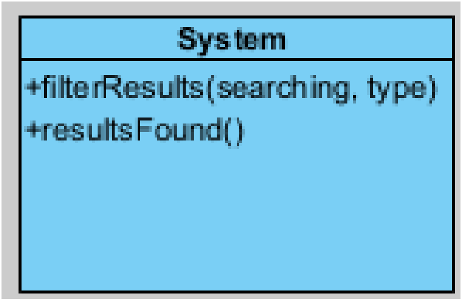


Contract: createPermission
Operation: creates permission for another user to access the active user’s information Cross Reference: Manage Friend Access Pre-conditions: another user exists without access to the active user Post-condition: the other user is granted access to the active users location and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

UC Filter Search Results


Actor
Lifeline
1: filterResults(search, type)
alt
[type == event]
1.1: returns search results filtered by event
[type == location]
1.2: returns search results filtered by location
[type == service]
1.3: returns search results filtered by service
[clear filter]
1.4: clears search filters and clears results
[Exit]
1.5: clears search filters and clears results
1.6: resultsFound()



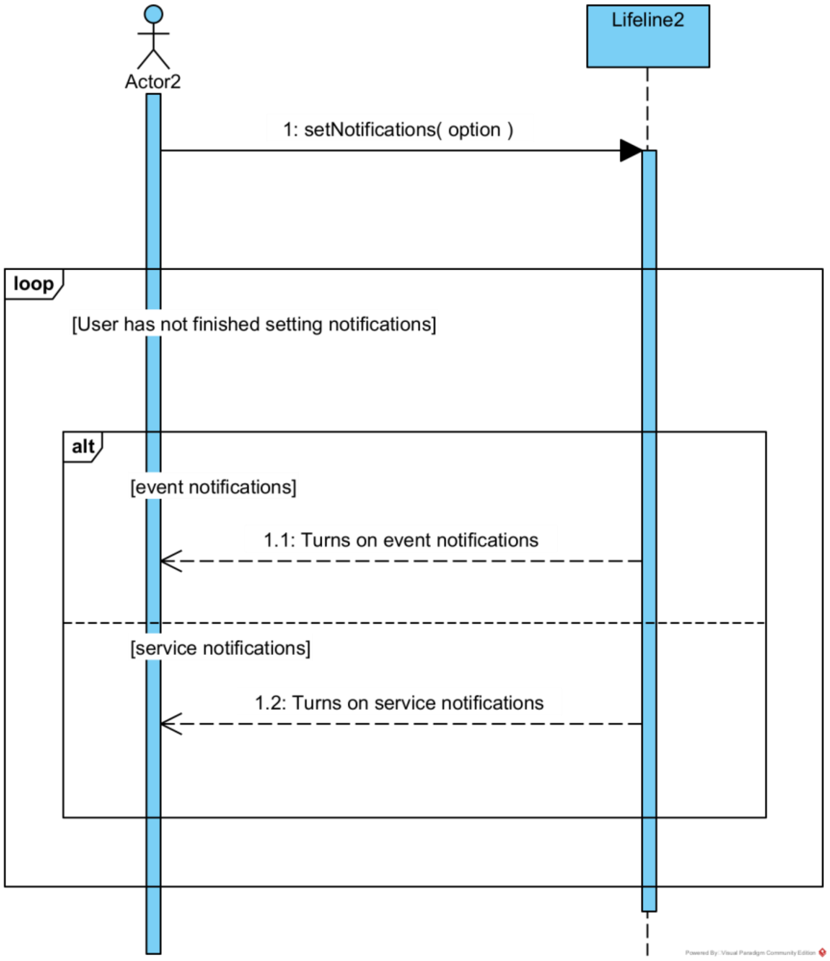
Contract: filterResults(searching, type)
Operation: Filters the results matching “searching” to only those that have the type of “type” Cross Reference:Use Case: Filter Search Results

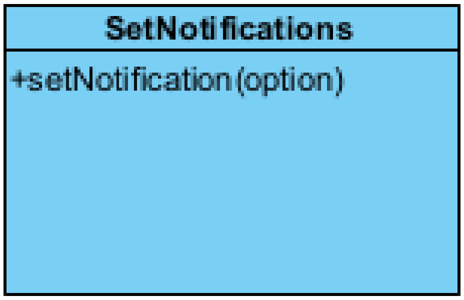
Pre-conditions: User wants to search for a specific result
Post-condition: User receives search results

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

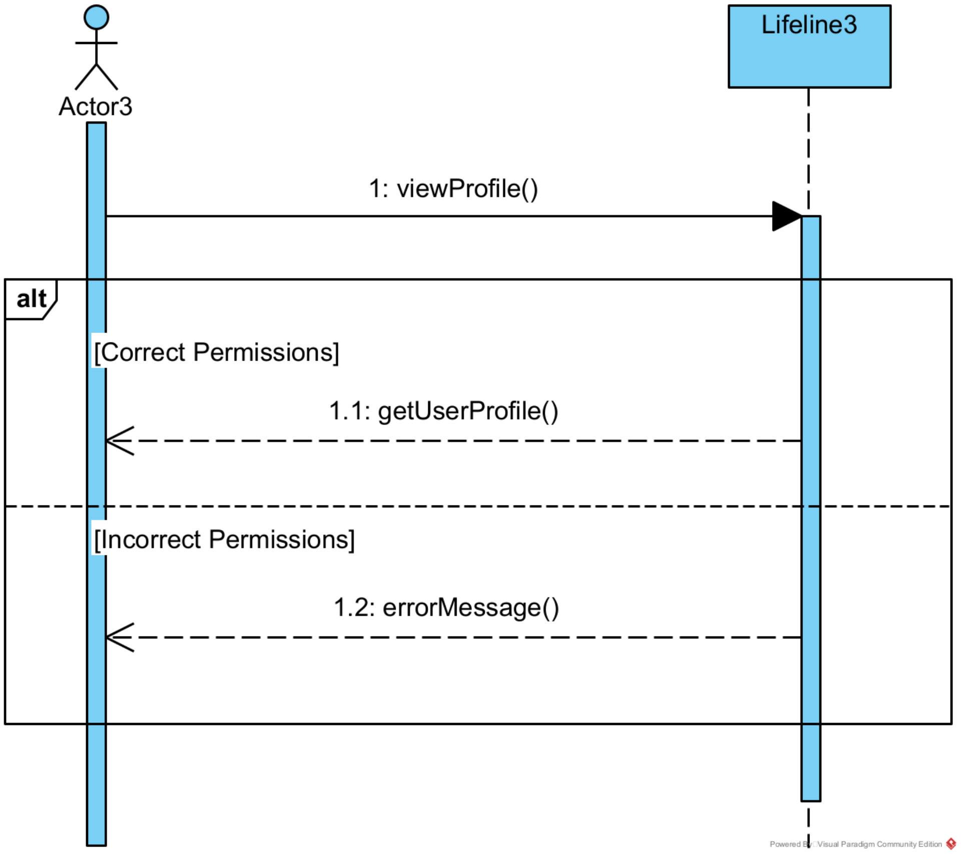

Actor2
Lifeline2
1: setNotifications(option)
loop
[User has not finished setting notifications]
alt
[event notifications]
1.1: Turns on event notifications
[service notifications]
1.2: Turns on service notifications
Powered By: Visual Paradigm Community Edition

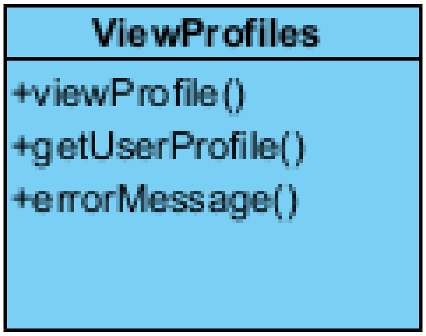


Contract: set Notification (option)

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

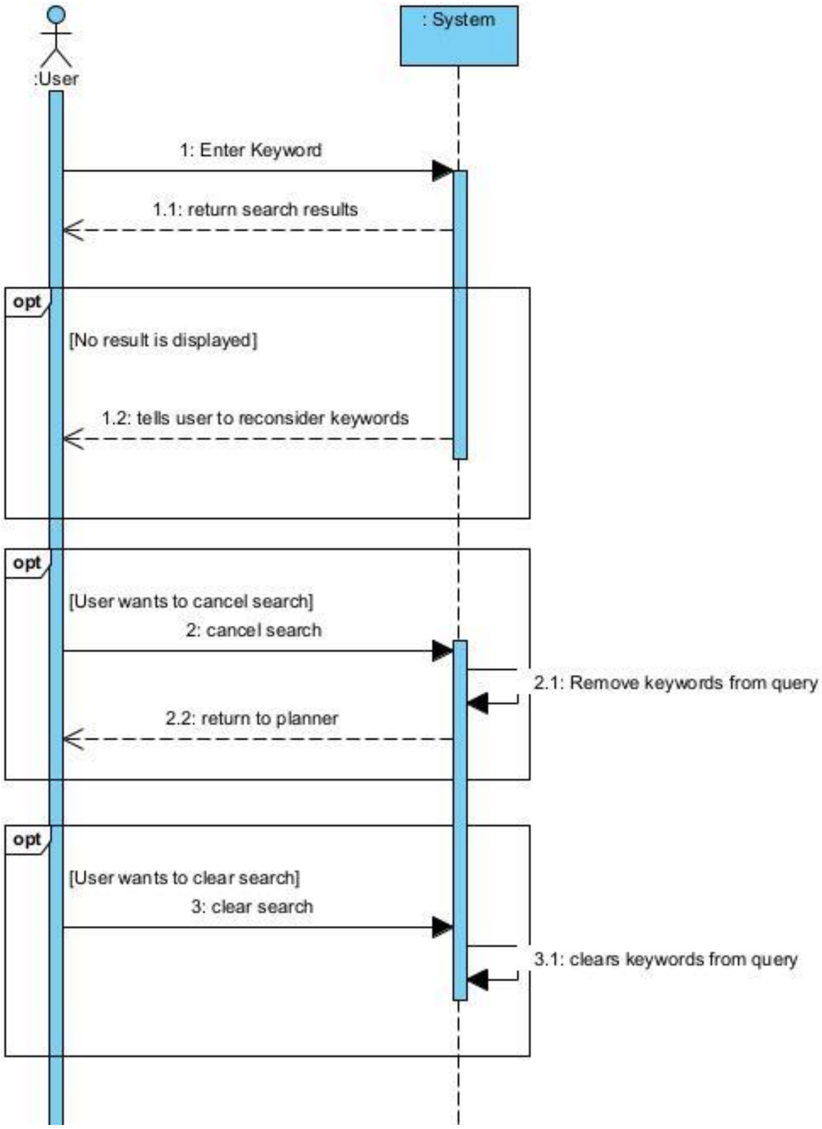




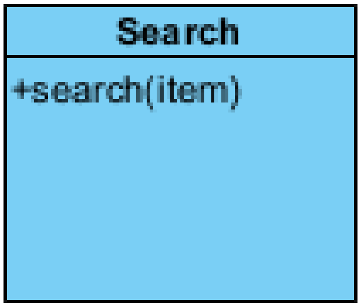
Contract: viewProfile( )
Operation: checks if the user is allowed to view the given profile Cross Reference: Use Case: ViewProfiles Pre-conditions: User wants to open other users' profiles Post-condition: User's permissions are checked with the s
Contract: getUserProfile( )
Operation: acquires the profile 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



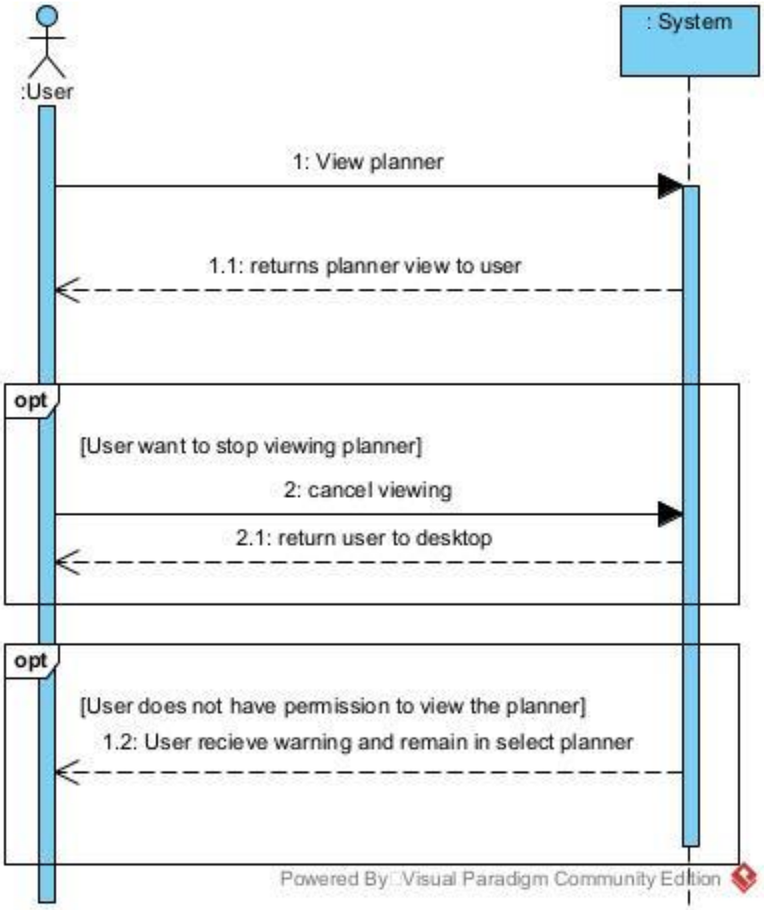
:User
: System
1: Enter Keyword
1.1: return search results
opt
[No result is displayed]
1.2: tells user to reconsider keywords
opt
[User wants to cancel search]
2: cancel search
2.1: Remove keywords from query
2.2: return to planner
opt
[User wants to clear search]
3: clear search
3.1: clears keywords from query

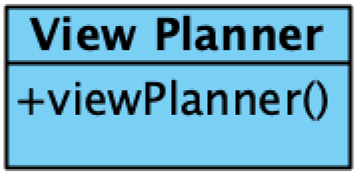


Contract: Search
Operation: searches for an item with a keyword
Cross Reference: Filter Search Result
Pre-conditions: User want to search for resources in Planner

Post-condition: User gets result based on keyword

UC View Planner

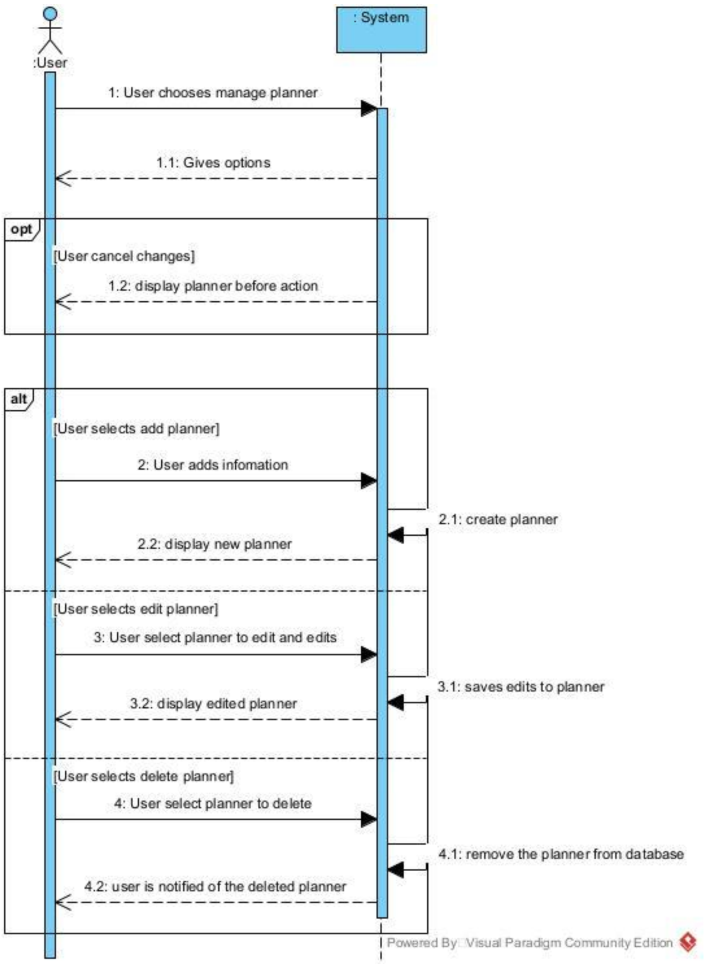

:User
: System
1: View planner
1.1: returns planner view to user
opt
[User want to stop viewing planner]
2: cancel viewing
2.1: return user to desktop
opt
[User does not have permission to view the planner]
1.2: User recieve warning and remain in select planner
Powered By Visual Paradigm Community Edition

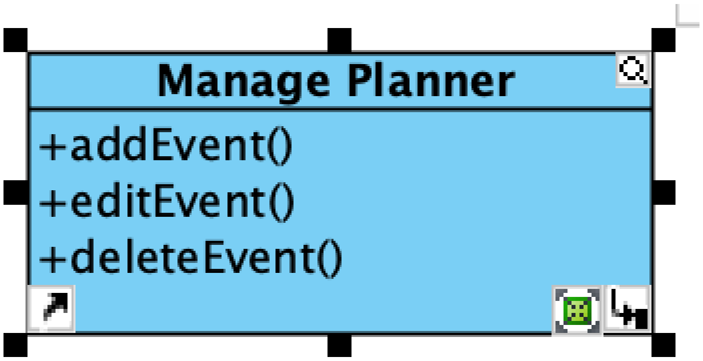


Contract: View Planner

Operation: selects planner to display
Cross Reference: manage planner
Pre-conditions: User want to view other planners
Post-condition: User views other planners

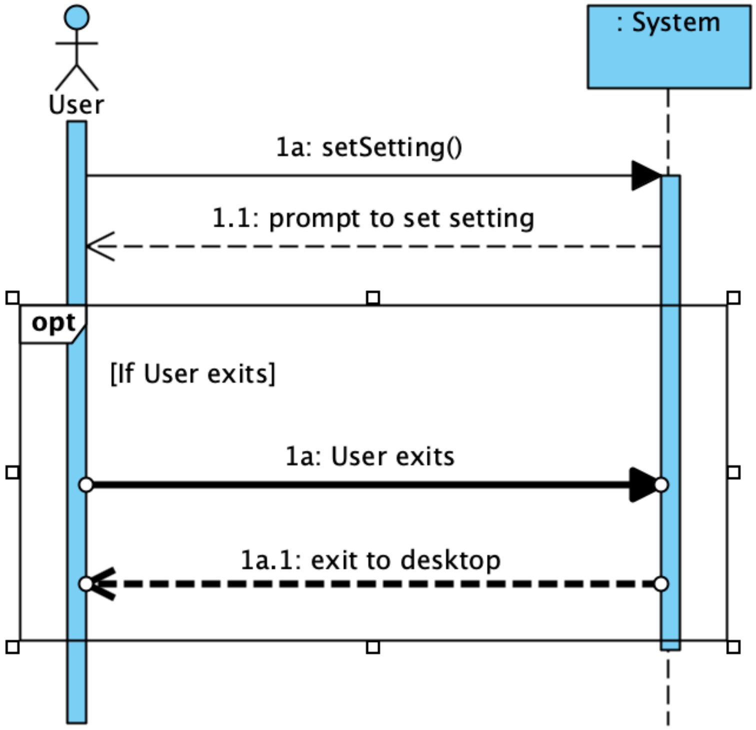
UC Manage Planner

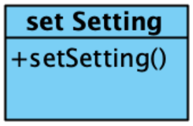

:User
: System
1: User chooses manage planner
1.1: Gives options
opt
[User cancel changes]
1.2: display planner before action
alt
[User selects add planner]
2: User adds infomation
2.1: create planner
2.2: display new planner
[User selects edit planner]
3: User select planner to edit and edits
3.1: saves edits to planner
3.2: display edited planner
[User selects delete planner]
4: User select planner to delete
4.1: remove the planner from database
4.2: user is notified of the deleted planner
Powered By: Visual Paradigm Community Edition



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

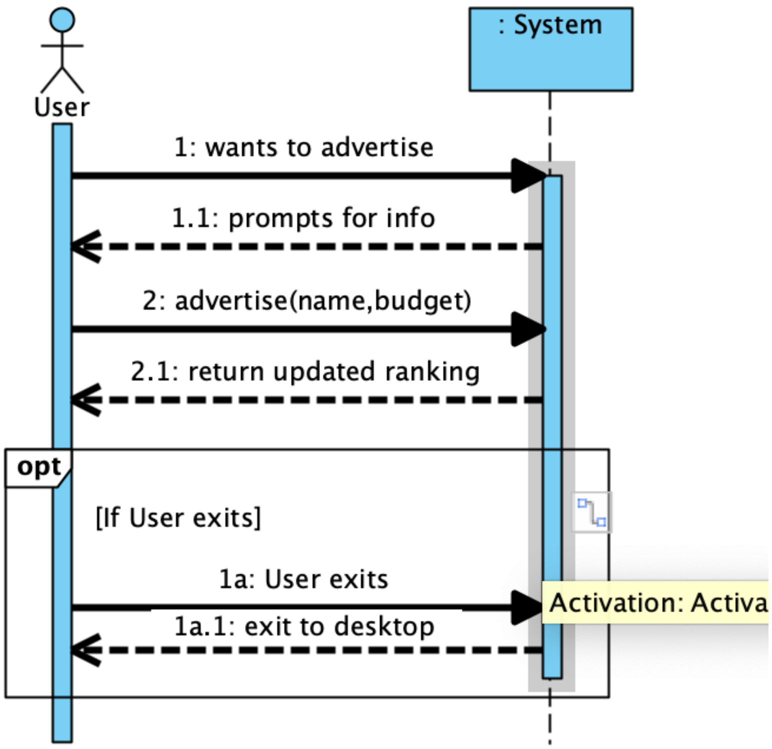
UC Set Settings

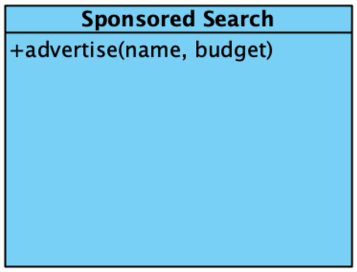




Contract: Set Setting
Operation: Allows User to set setting Cross Reference: Filter search results Pre-conditions: User wants to set settings Post-condition: Settings are updated

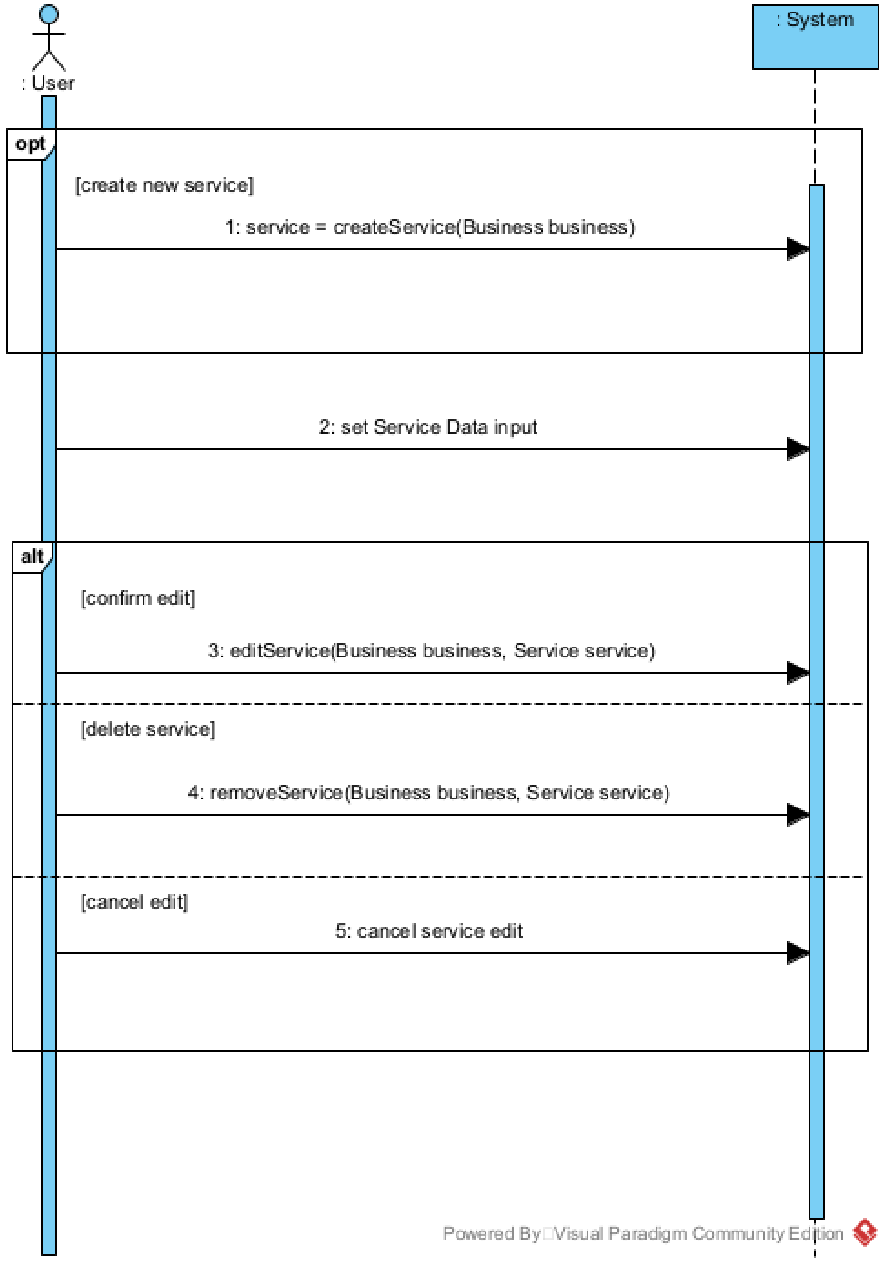
UC Sponsored Search Results

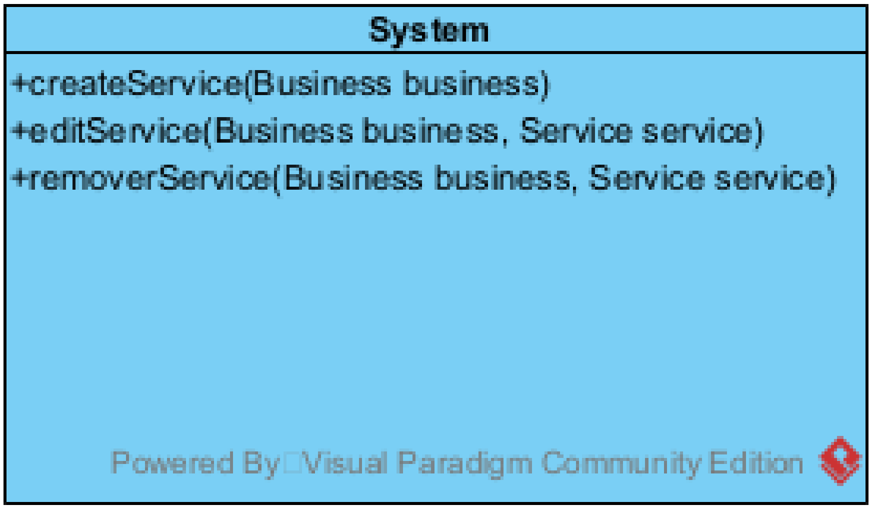




Contract: Sponsored Search Results
Operation: Search results are padded with sponsors Cross Reference: Filer search results Pre-conditions: Business User wants to advertise Post-condition: Rankings is updated

UC Manage Service


: User
: System
opt
[create new service]
1: service = createService(Business business)
2: set Service Data input
alt
[confirm edit]
3: editService(Business business, Service service)
[delete service]
4: removeService(Business business, Service service)
[cancel edit]
5: cancel service edit
Powered By Visual Paradigm Community Edition


System
+createService(Business business)
+editService(Business business, Service service)
+removerService(Business business, Service service)
Powered By Visual Paradigm Community Edition

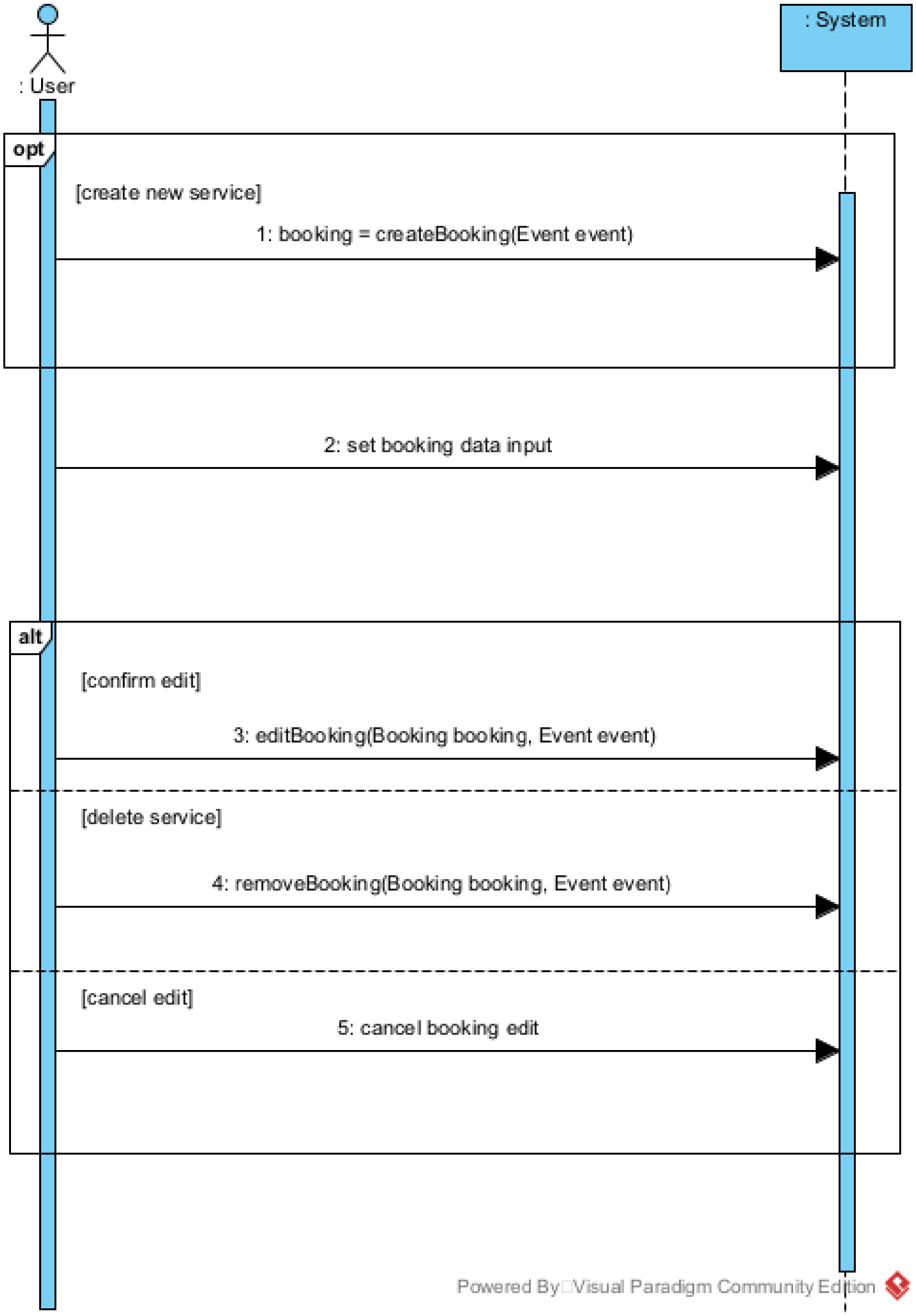
Contract: createService
Operation: creates a service that is sold by a buisness and booked by users for their events

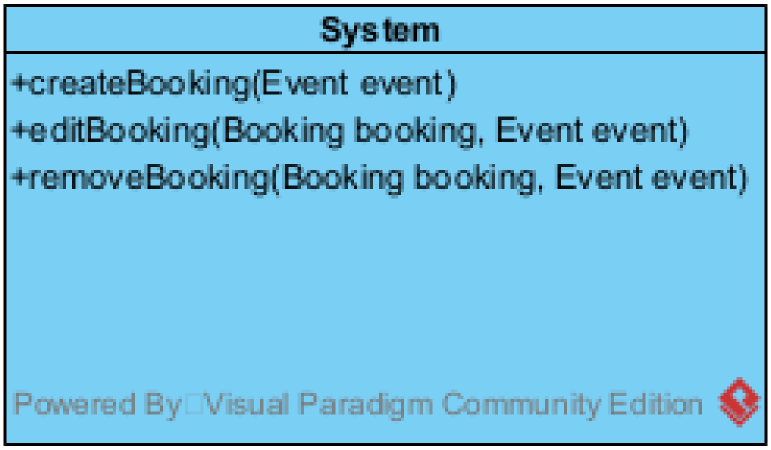
Cross Reference: Manage Service Pre-conditions: the user wants to create a service associated with a business to which they have access. Post-condition: a new service is associated with a buisness

Contract: editService
Operation: edits a service that is sold by a buisness 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 buisness 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 offerd by a business

UC Manage Booking


: User
: System
opt
[create new service]
1: booking = createBooking(Event event)
2: set booking data input
alt
[confirm edit]
3: editBooking(Booking booking, Event event)
[delete service]
4: removeBooking(Booking booking, Event event)
[cancel edit]
5: cancel booking edit

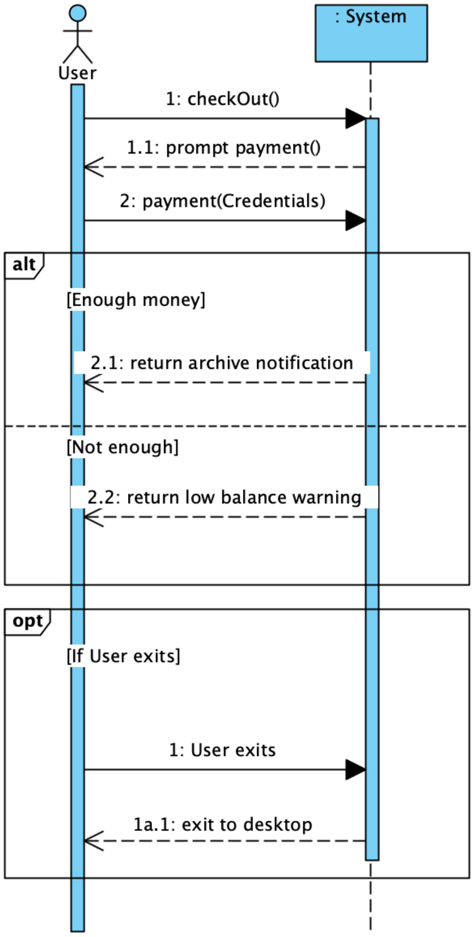

System
+createBooking(Event event)
+editBooking(Booking booking, Event event)
+removeBooking(Booking booking, Event event)

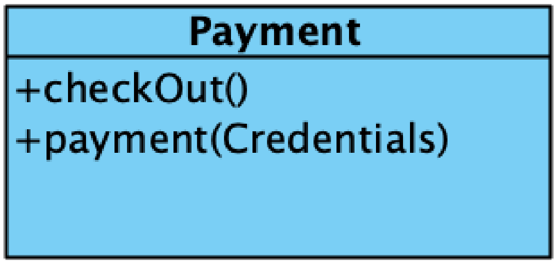
Contract: createBooking
Operation: creates a booking Cross Reference: Manage Booking Pre-conditions: User wants to make a new booking Post-condition: booking is created with some business offering some service

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

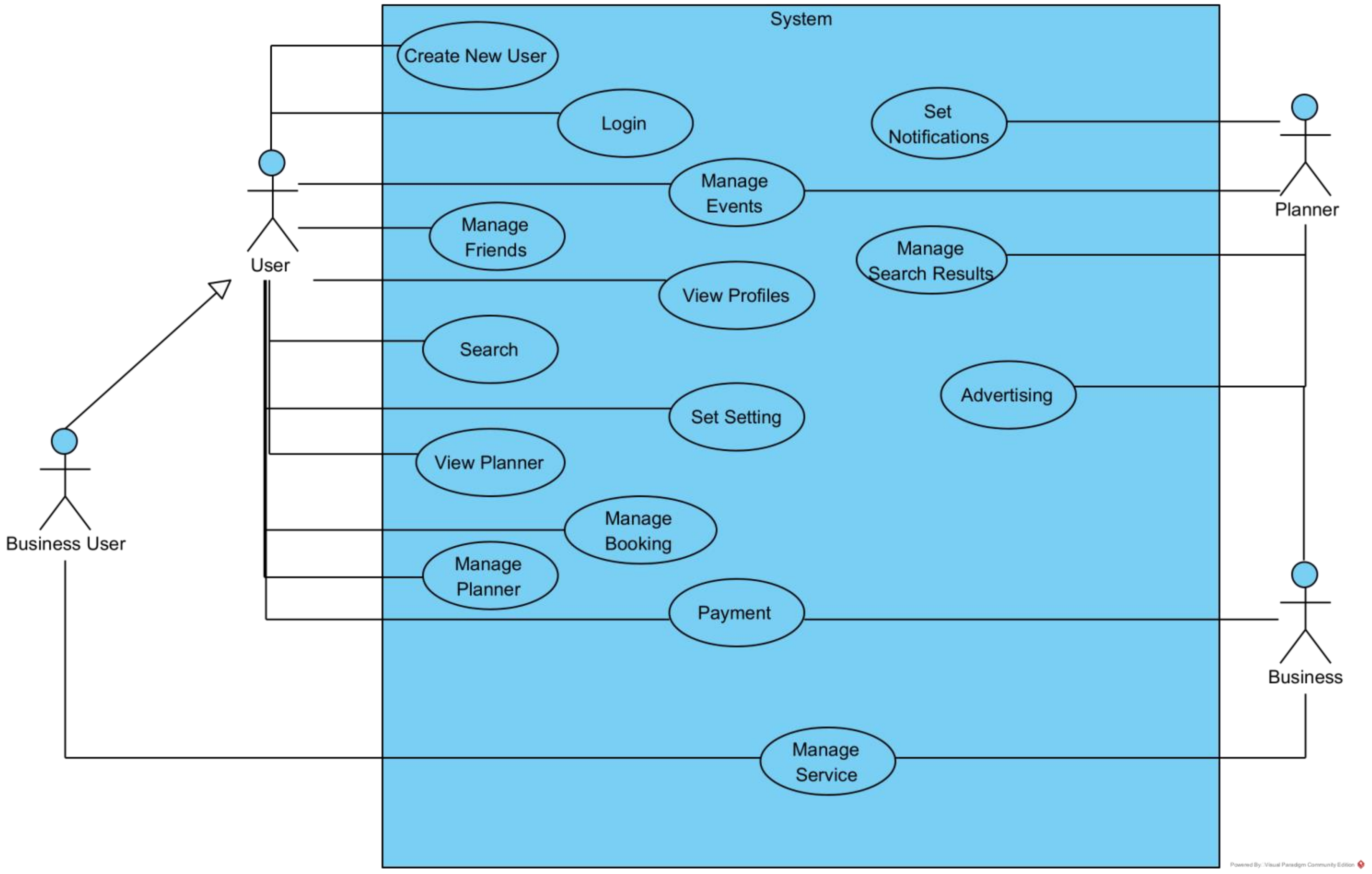




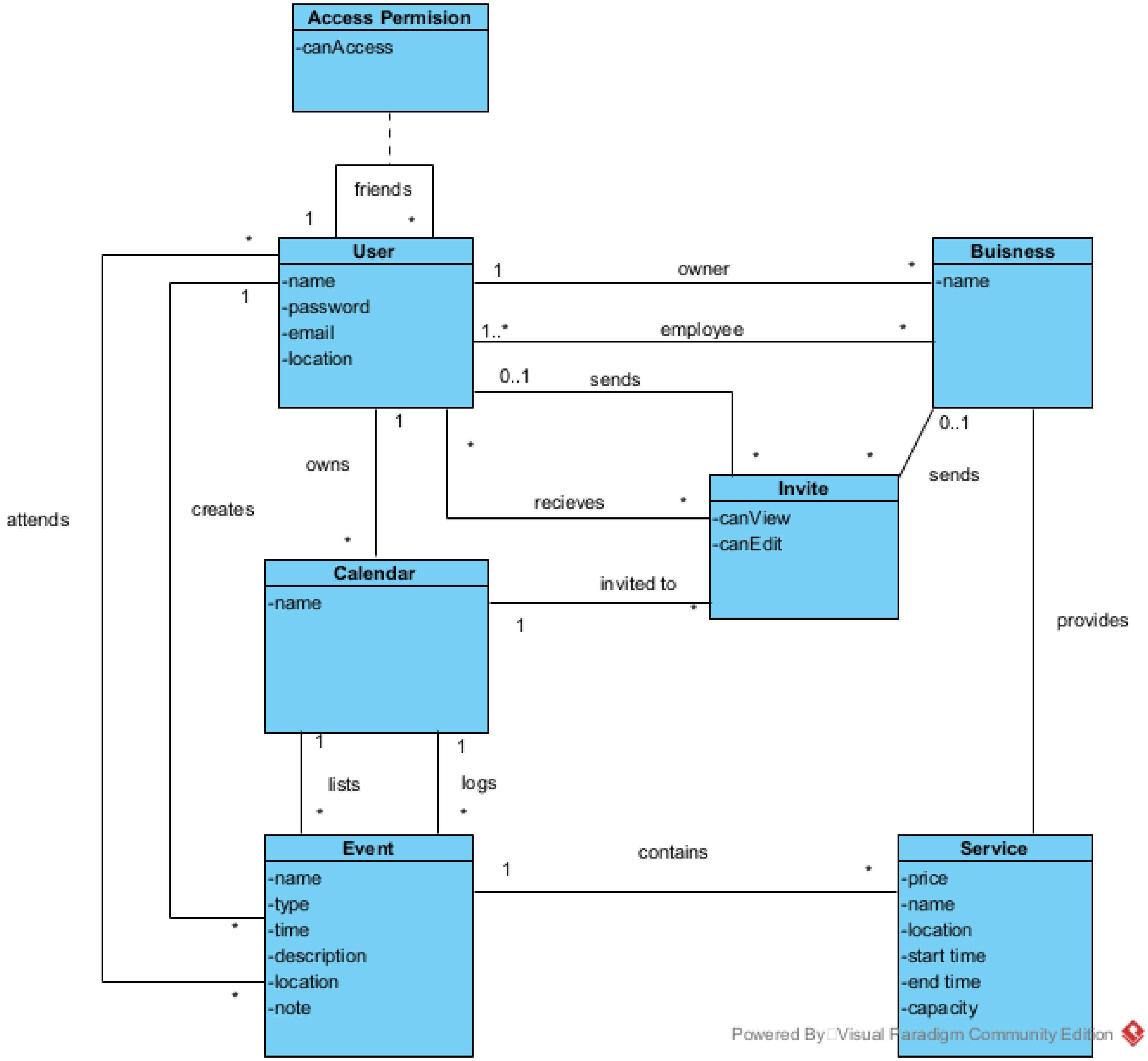
Contract: Payment

Operation: Allows User to pay total
Cross Reference: Manage service
Pre-conditions: User wants to pay total
Post-condition: Total is paid

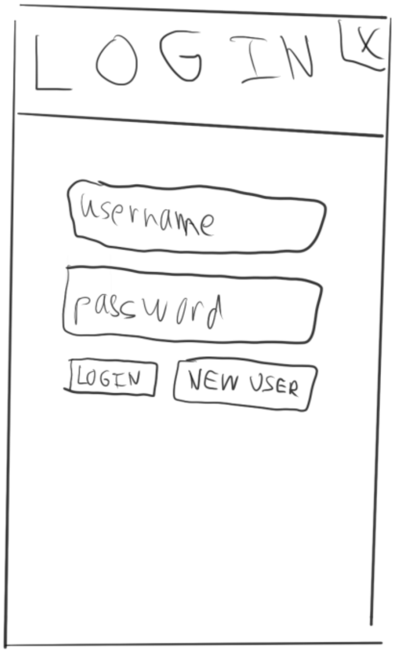
Use Case Diagram

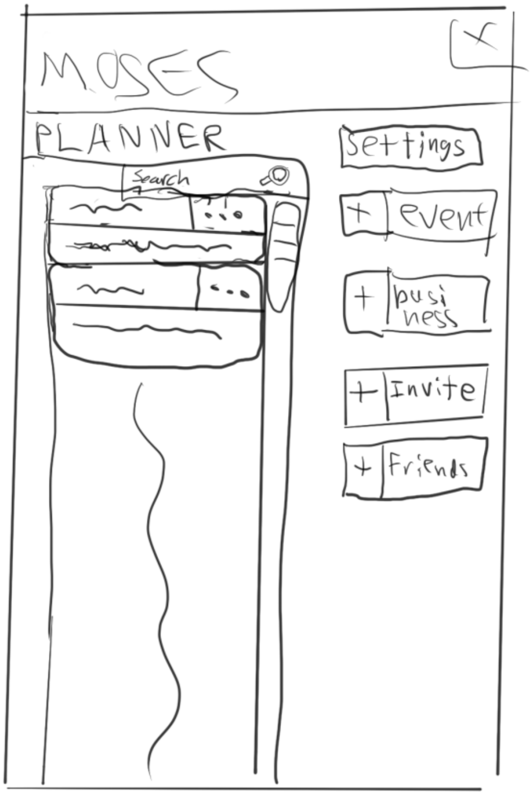


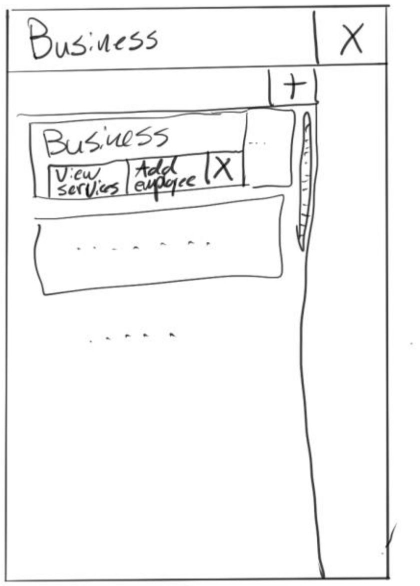
Domain Model

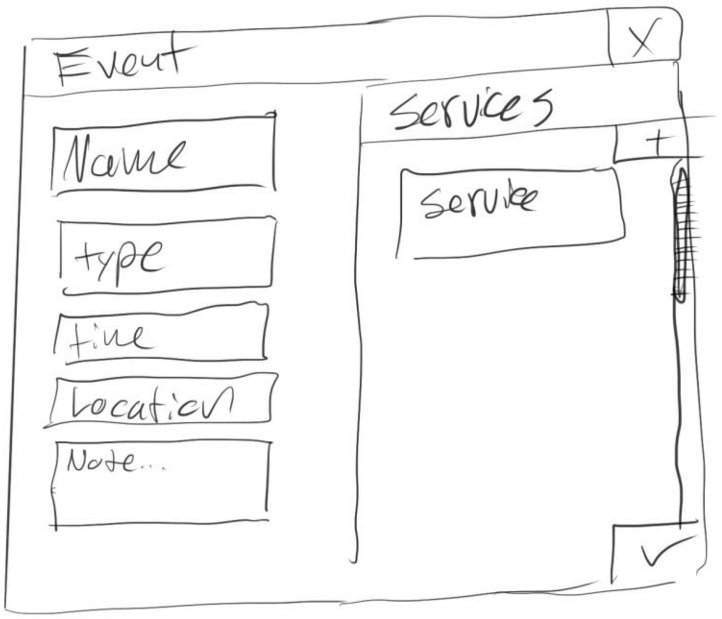


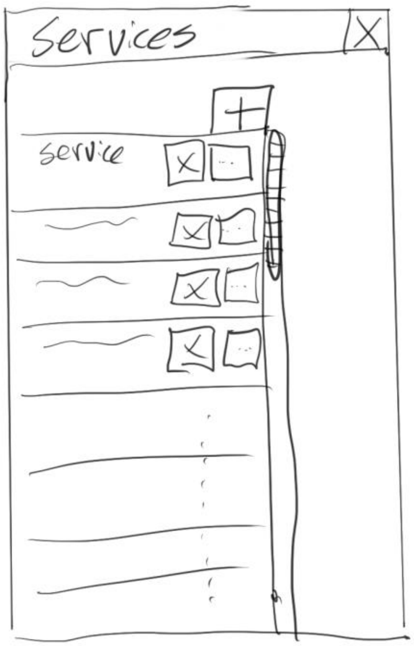
Wire Frame

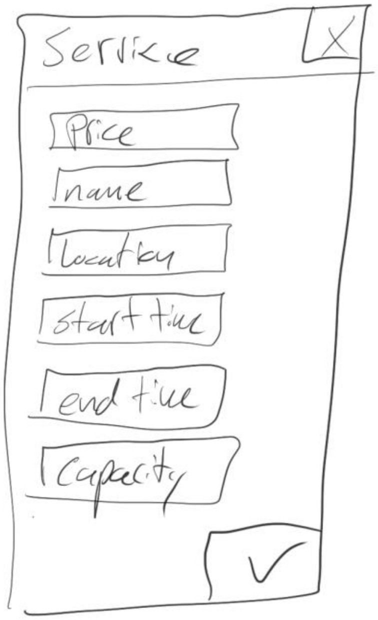

LOGIN
X
username
password
LOGIN
NEW USER


MOSES
X
PLANNER
Search
Settings
+ event
+ business
+ Invite
+ Friends

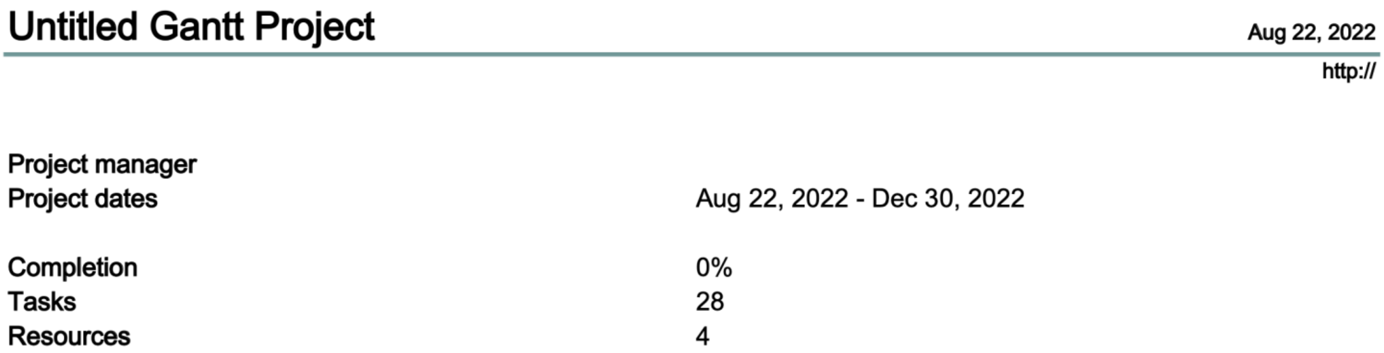

Business
X
+
Business
View services
Add employee
X


Event
X
Name
type
time
Location
Note...
Services
+
Service
✓


Services
X
+
service
X
...


Service
X
Price
name
Location
start time
end time
Capacity
✓

Gantt Diagram

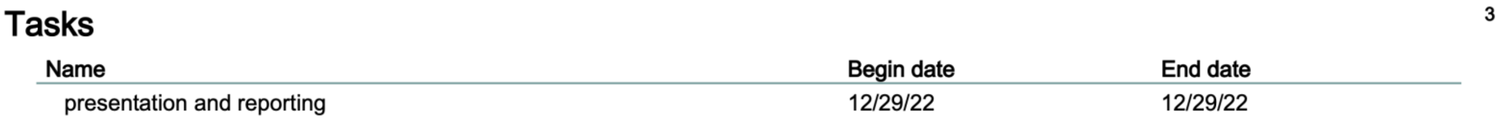


Untitled Gantt Project

Aug 22, 2022

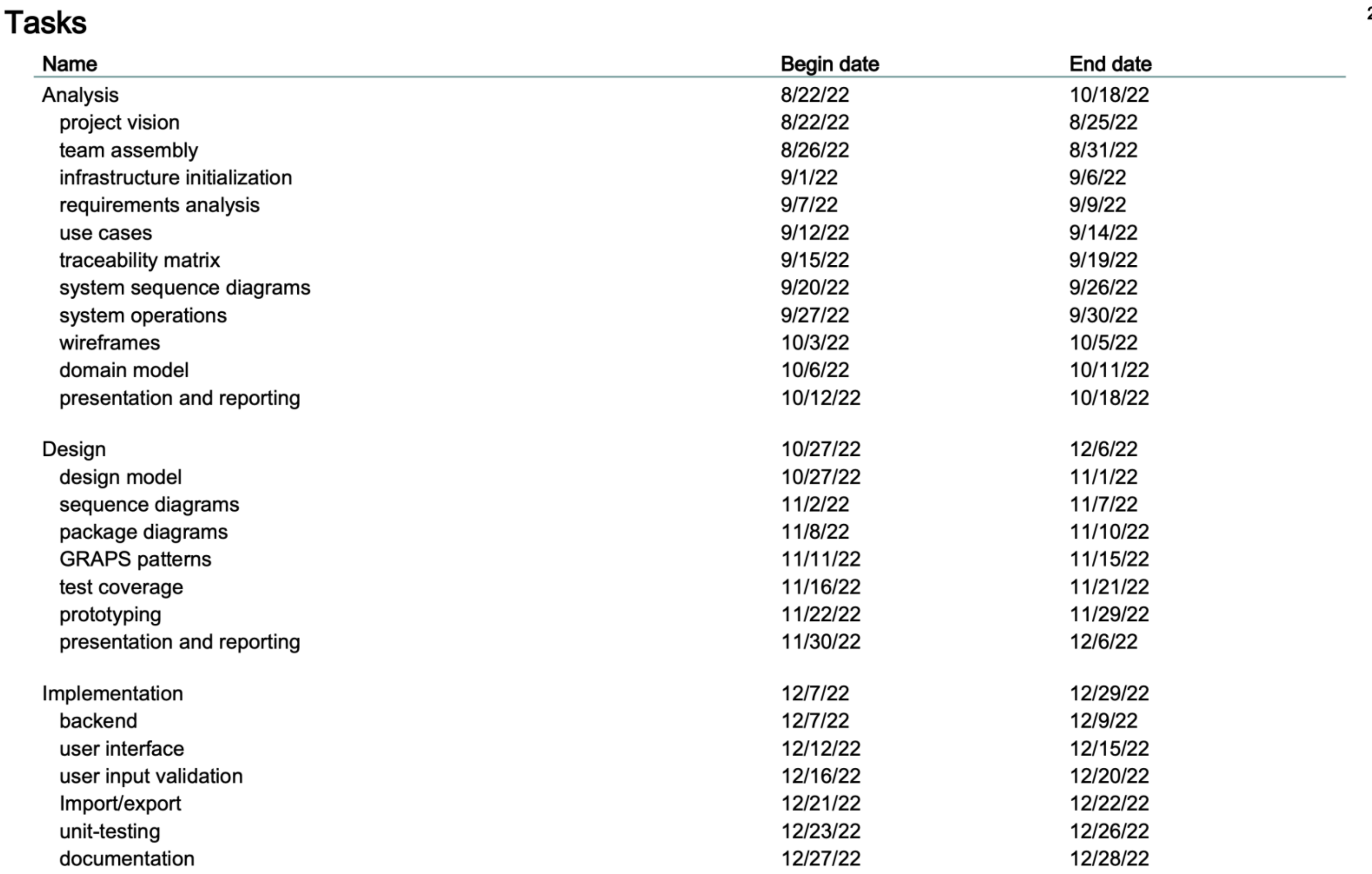
http://

Project manager	
Project dates	Aug 22, 2022 - Dec 30, 2022
Completion	0%
Tasks	28
Resources	4



Tasks

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



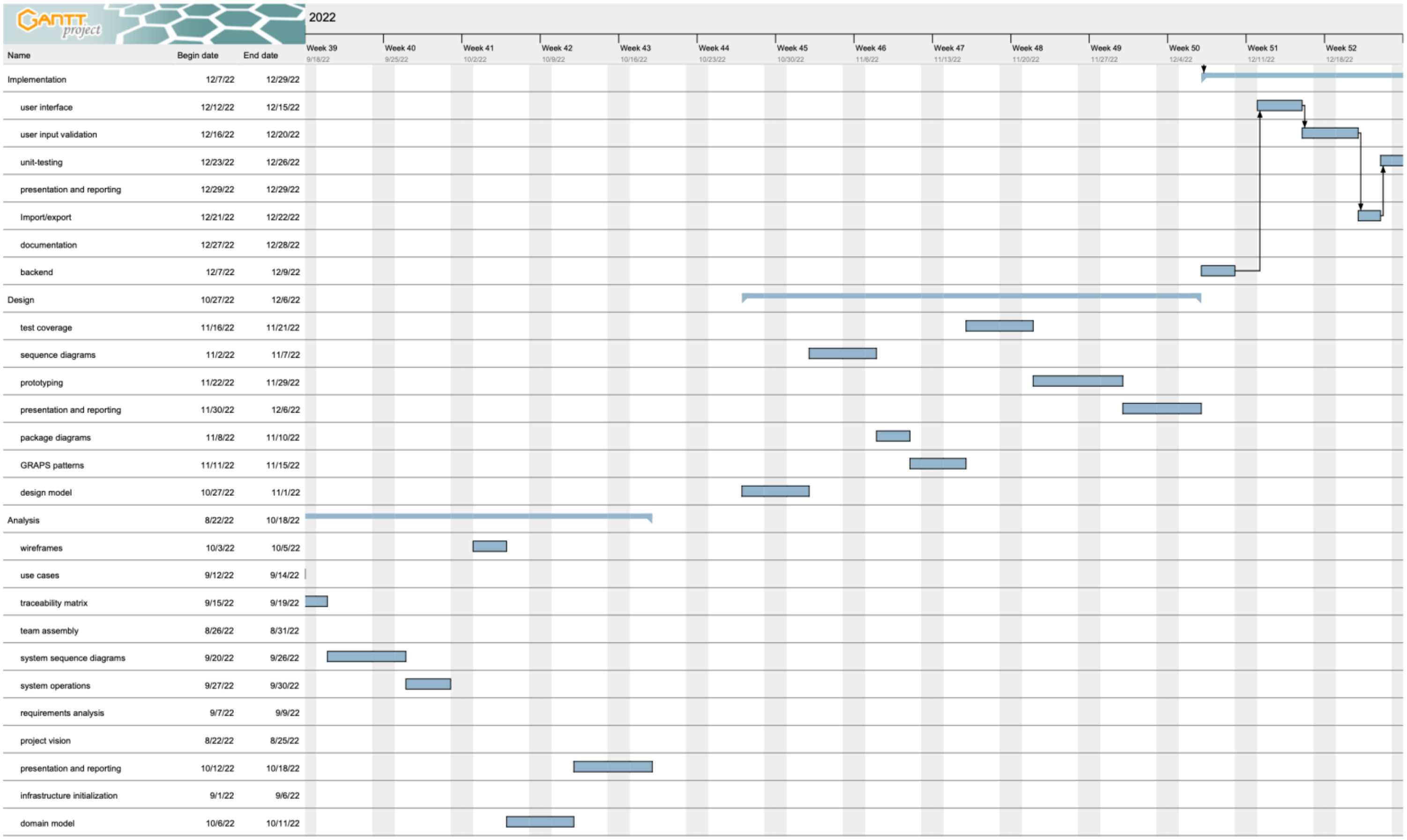
Tasks

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

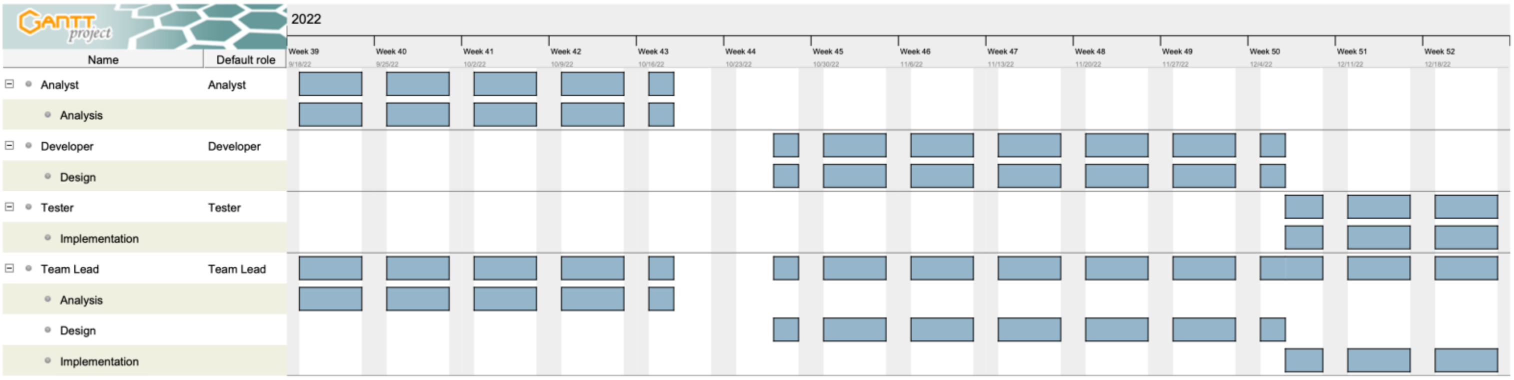
Name	Default role
Analyst	Analyst
Developer	Developer
Tester	Tester
Team Lead	Team Lead

Gantt Chart



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

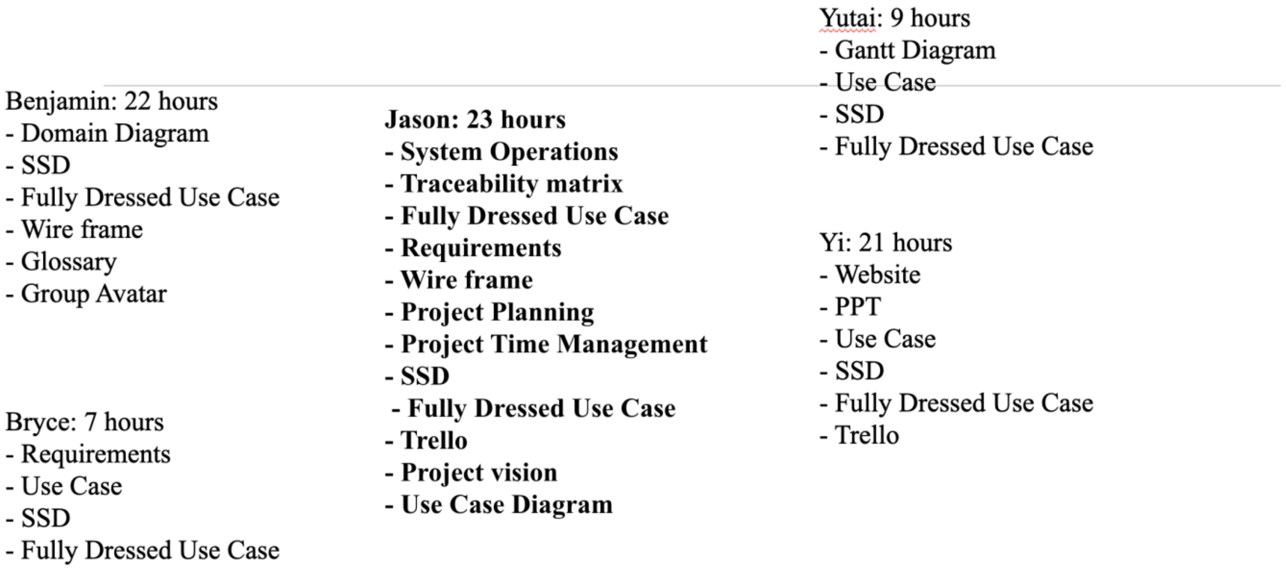
Resources Chart



Name	Default role
Analyst	Analyst
Analysis	
Developer	Developer
Design	
Tester	Tester
Implementation	
Team Lead	Team Lead
Analysis	
Design	
Implementation	

Timecard

GitHub



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