Demo Link: https://www.youtube.com/watch?v=SYEPf38joQk

GitHub Link

https://github.com/jeremymeadows/Mentality

Work Distribution, Point Distribution, Time Spent

Micah Dadson- 6 sequence diagrams, GRASP, UML class diagram, Package diagram, Testing Plan, Chart.java, Feed.Java, MenuBar.Java, Post.java, PostToWall.java, Search.Java, StarRater.java, Dashboard.java, HappinessGraph.java, Report.java, CronScheduler.java, ReportMaker.java, applicationContext.xml,

Points: 8.5/6

Time Spent: 30 hours

Jeremy Meadows- 3 sequence diagrams, Calendar.java, Database.java, Password.java, User.java, Registration.java, CustomUtilities.java, SpringUtilities.java, CustomUtilities.java, Mentality.Java, Runner.java, logo.png, logo.svg

Points:8.5/6

Time Spent:25 hours

Shivani Bobbala- 3 sequence diagrams, Mood.java, DiaryFrame.java

Points: 7/6

Time Spent: 10 hours

Joel Futagawa- 3 sequence diagrams

Points:6/6

Time Spent:5 hours

Shunting Chen- 3 sequence diagrams

Points: 6/6

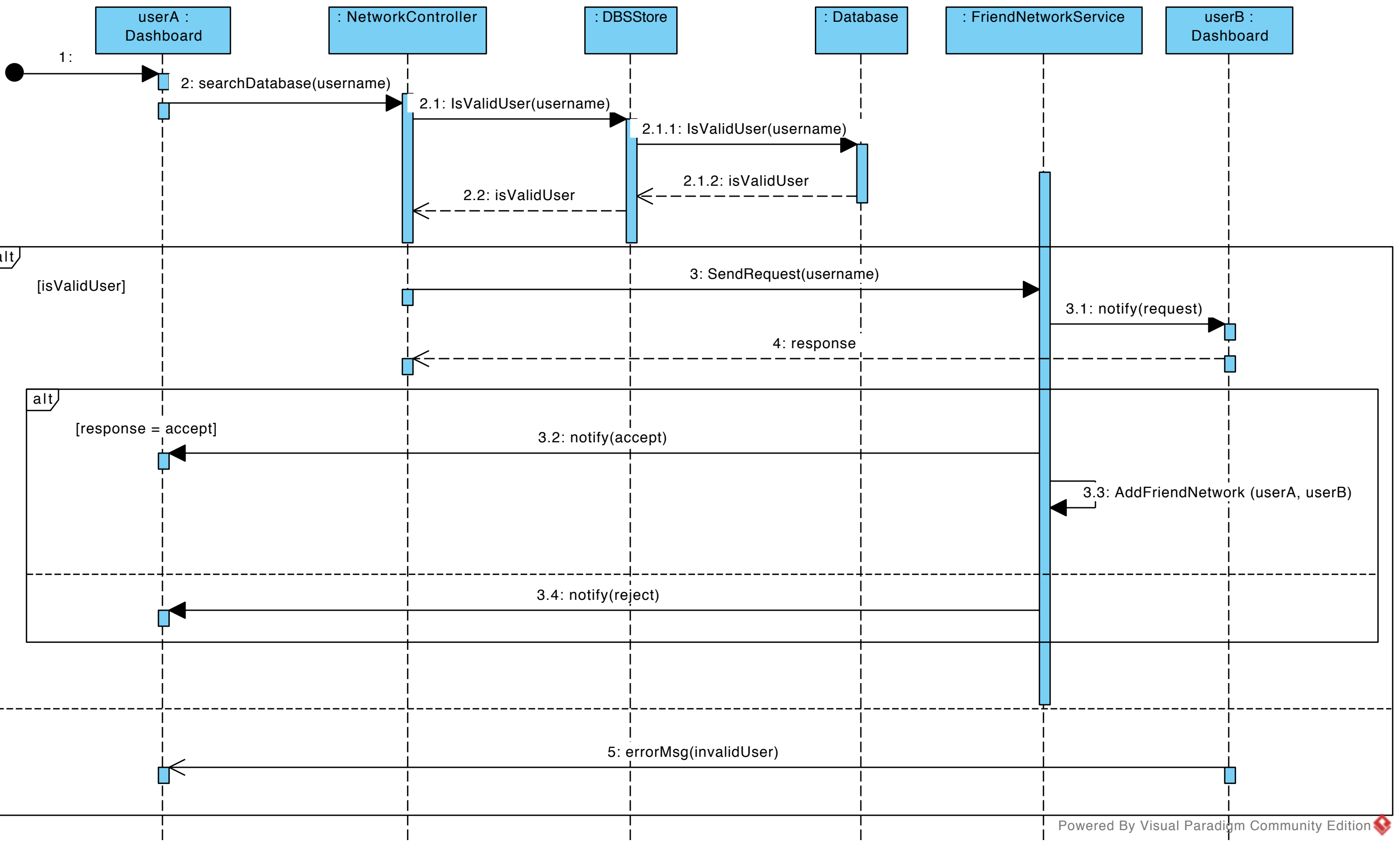
Time Spent: 5 hours

Francis Ning- n/a

Points: 0/6

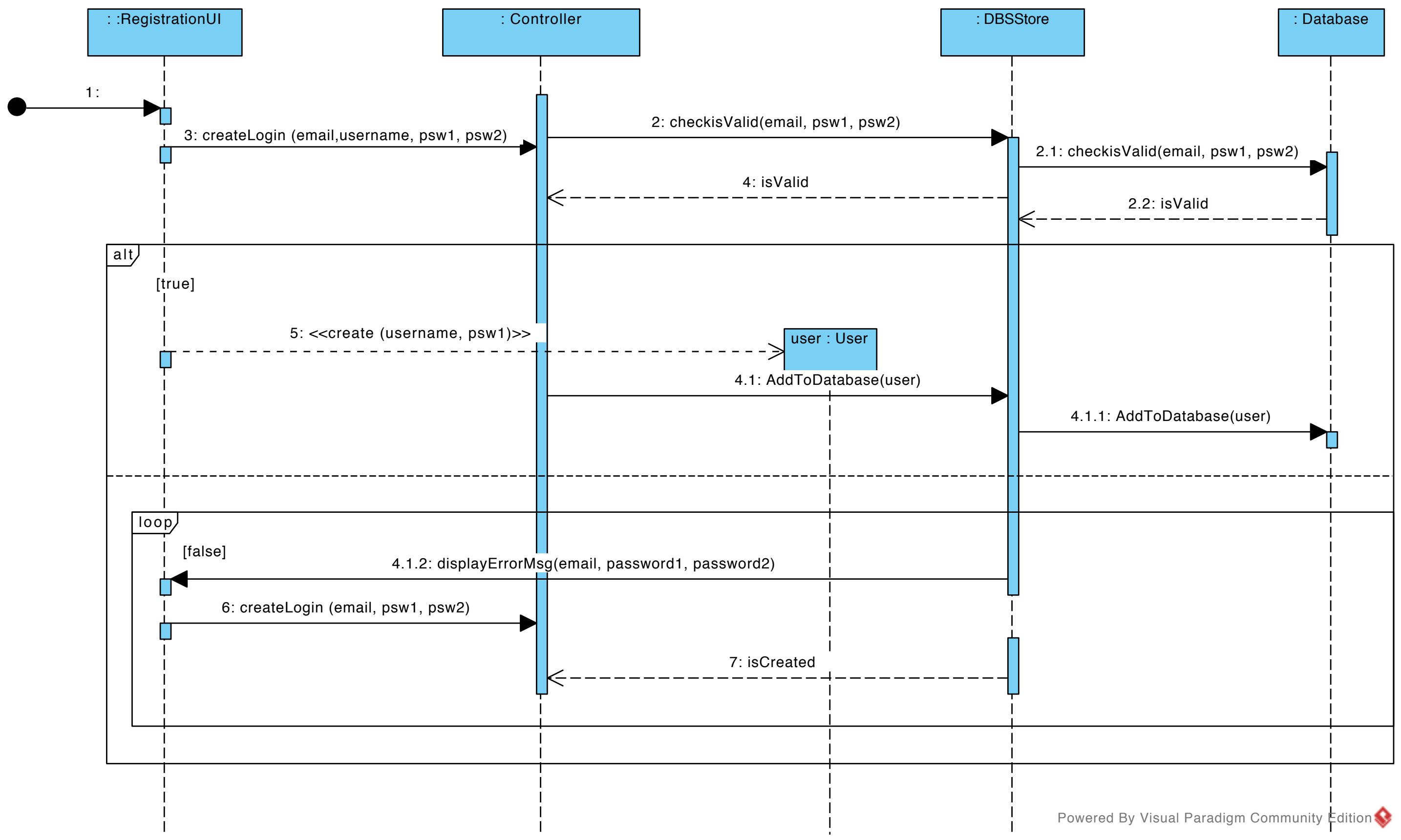
Time Spent: 5 hours

(Sorry we do not have a Gantt diagram, the software would not would on my computer and I did not have access to a different computer)



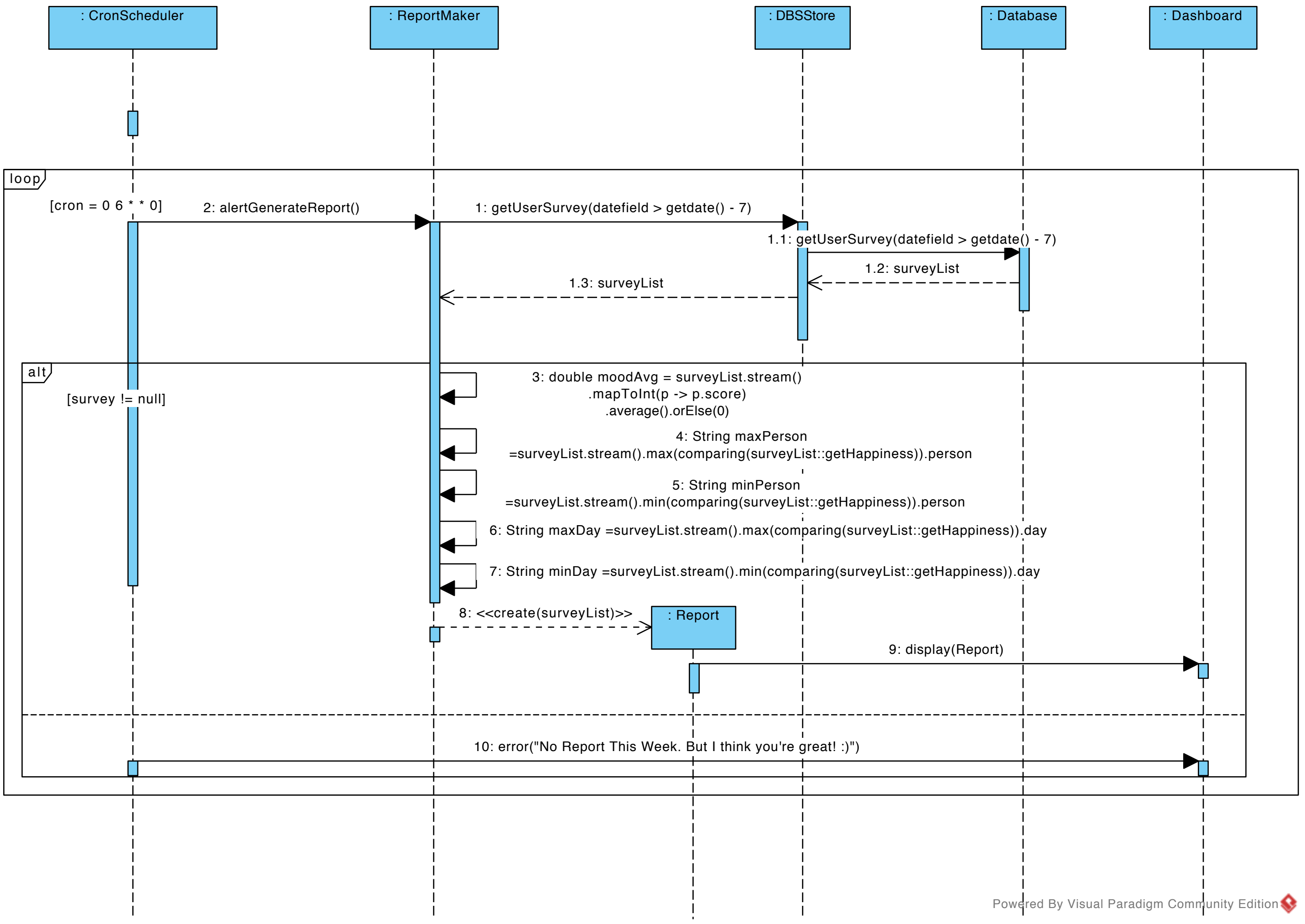
Add a Friend

Micah Dadson


: :RegistrationUI
: Controller
: DBSStore
: Database
1 :
2: checkisValid(email, psw1, psw2)
3: createLogin (email,username, psw1, psw2)
2.1: checkisValid(email, psw1, psw2)
4: isValid
2.2: isValid
alt
[true]
5: <<create (username, psw1)>>
user : User
4.1: AddToDatabase(user)
4.1.1: AddToDatabase(user)
loop
[false]
4.1.2: displayErrorMsg(email, password1, password2)
6: createLogin (email, psw1, psw2)
7: isCreated
Powered By Visual Paradigm Community Edition

Register an Account

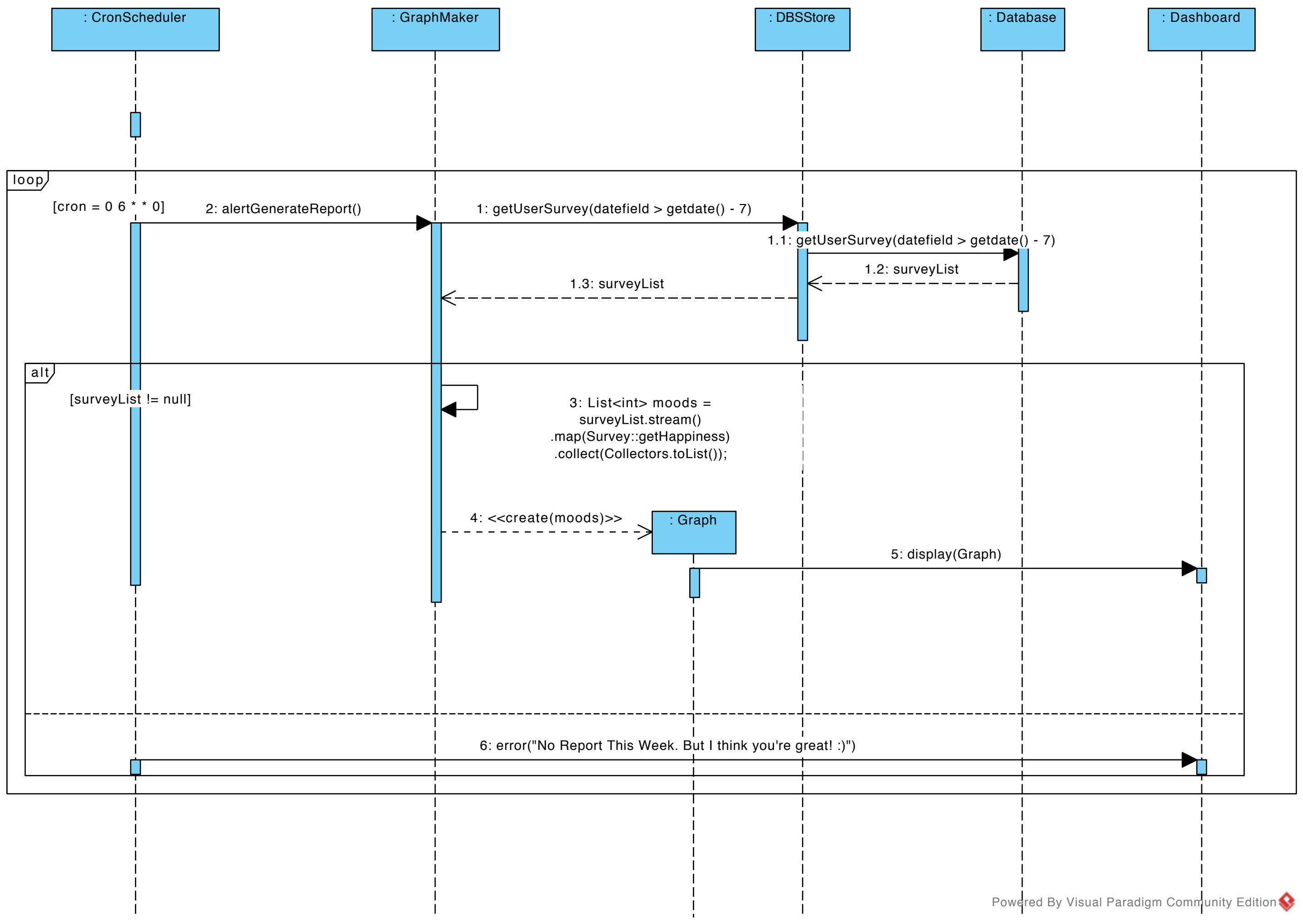
Micah Dadson



: CronScheduler
: ReportMaker
: DBSStore
: Database
: Dashboard
loop
[cron = 0 6 * * 0]
2: alertGenerateReport()
1: getUserSurvey(datefield > getdate() - 7)
1.1: getUserSurvey(datefield > getdate() - 7)
1.2: surveyList
1.3: surveyList
alt
[survey != null]
3: double moodAvg = surveyList.stream()
.mapToInt(p -> p.score)
.average().orElse(0)
4: String maxPerson
=surveyList.stream().max(comparing(surveyList::getHappiness)).person
5: String minPerson
=surveyList.stream().min(comparing(surveyList::getHappiness)).person
6: String maxDay =surveyList.stream().max(comparing(surveyList::getHappiness)).day
7: String minDay =surveyList.stream().min(comparing(surveyList::getHappiness)).day
8: <<create(surveyList)>>
: Report
9: display(Report)
10: error("No Report This Week. But I think you're great! :)")
Powered By Visual Paradigm Community Edition

Generate a Report

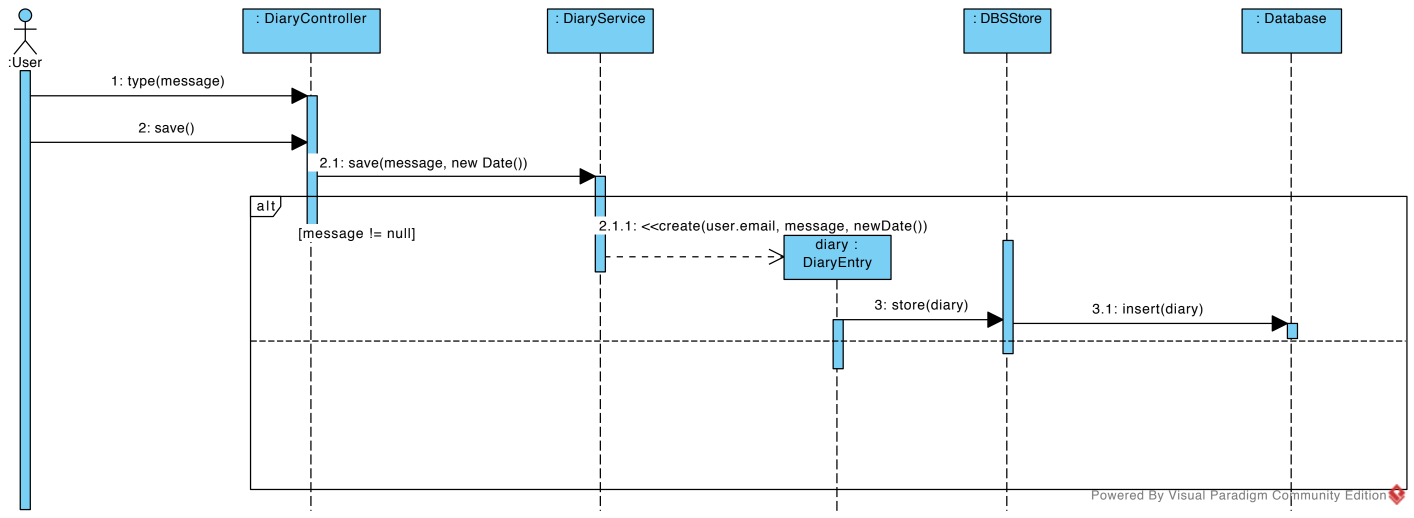
Micah Dadson



: CronScheduler
: GraphMaker
: DBSStore
: Database
: Dashboard
loop
[cron = 0 6 * * 0]
2: alertGenerateReport()
1: getUserSurvey(datefield > getdate() - 7)
1.1: getUserSurvey(datefield > getdate() - 7)
1.2: surveyList
1.3: surveyList
alt
[surveyList != null]
3: List<int> moods = surveyList.stream() .map(Survey::getHappiness) .collect(Collectors.toList());
4: <<create(moods)>>
: Graph
5: display(Graph)
6: error("No Report This Week. But I think you're great! :)")
Powered By Visual Paradigm Community Edition

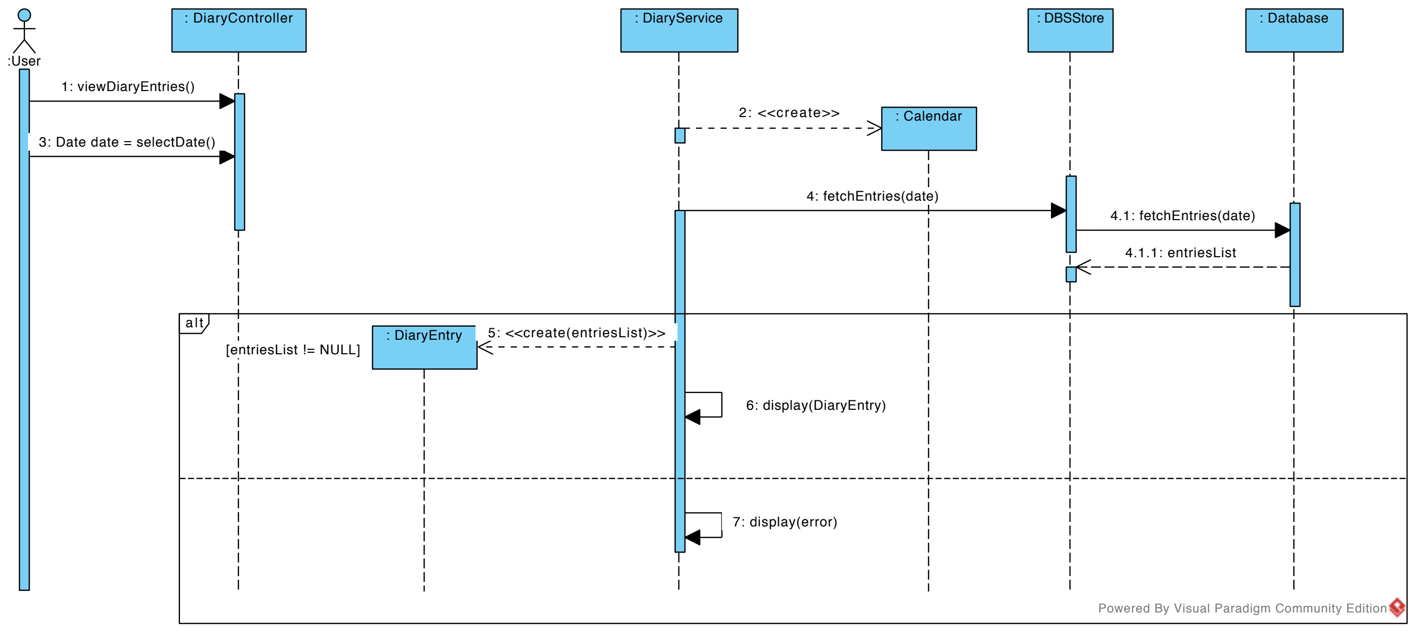
Generate a Happiness Graph

Micah Dadson



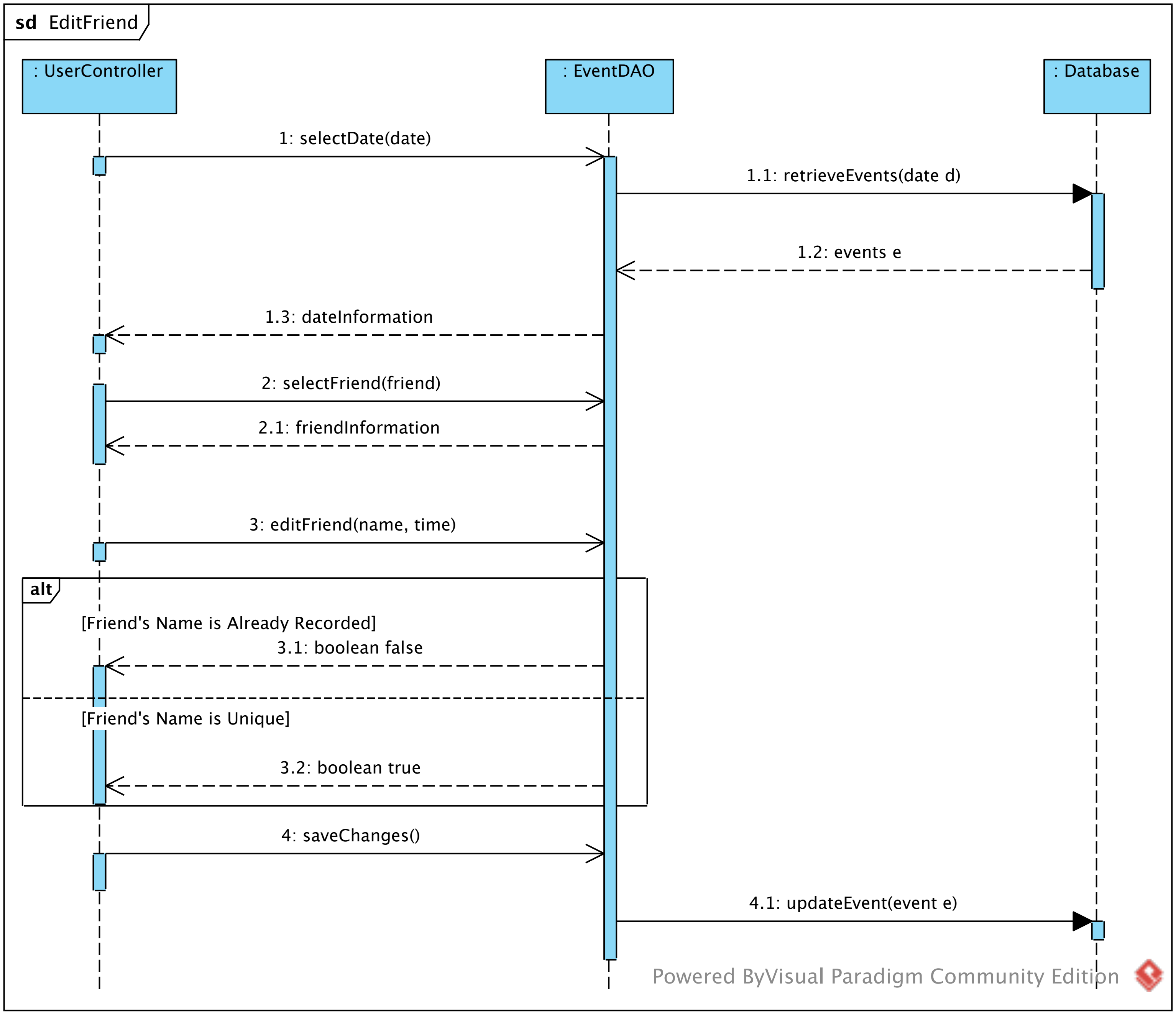
Complete a Diary Entry

Micah Dadson

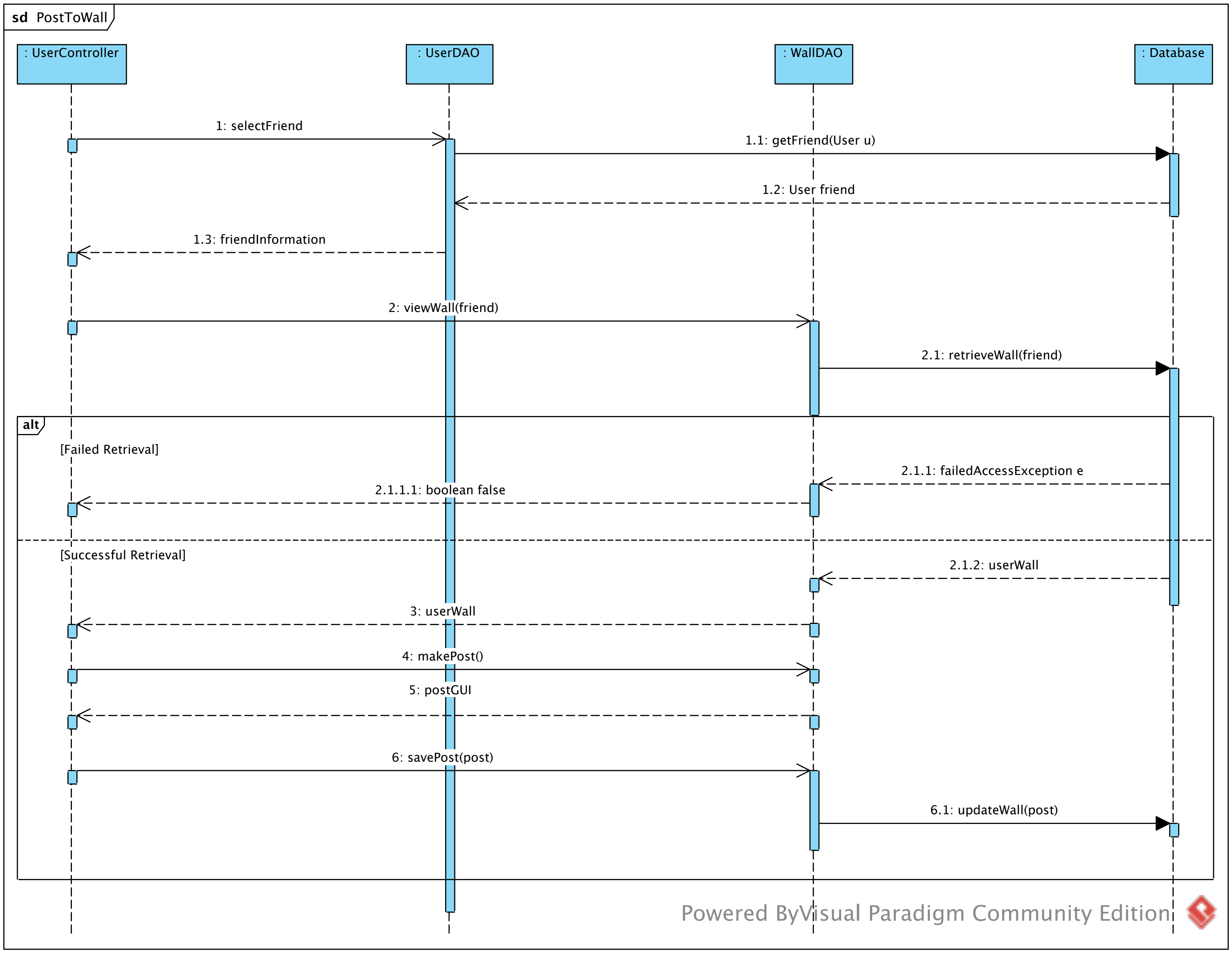


Look Up a Diary Entry

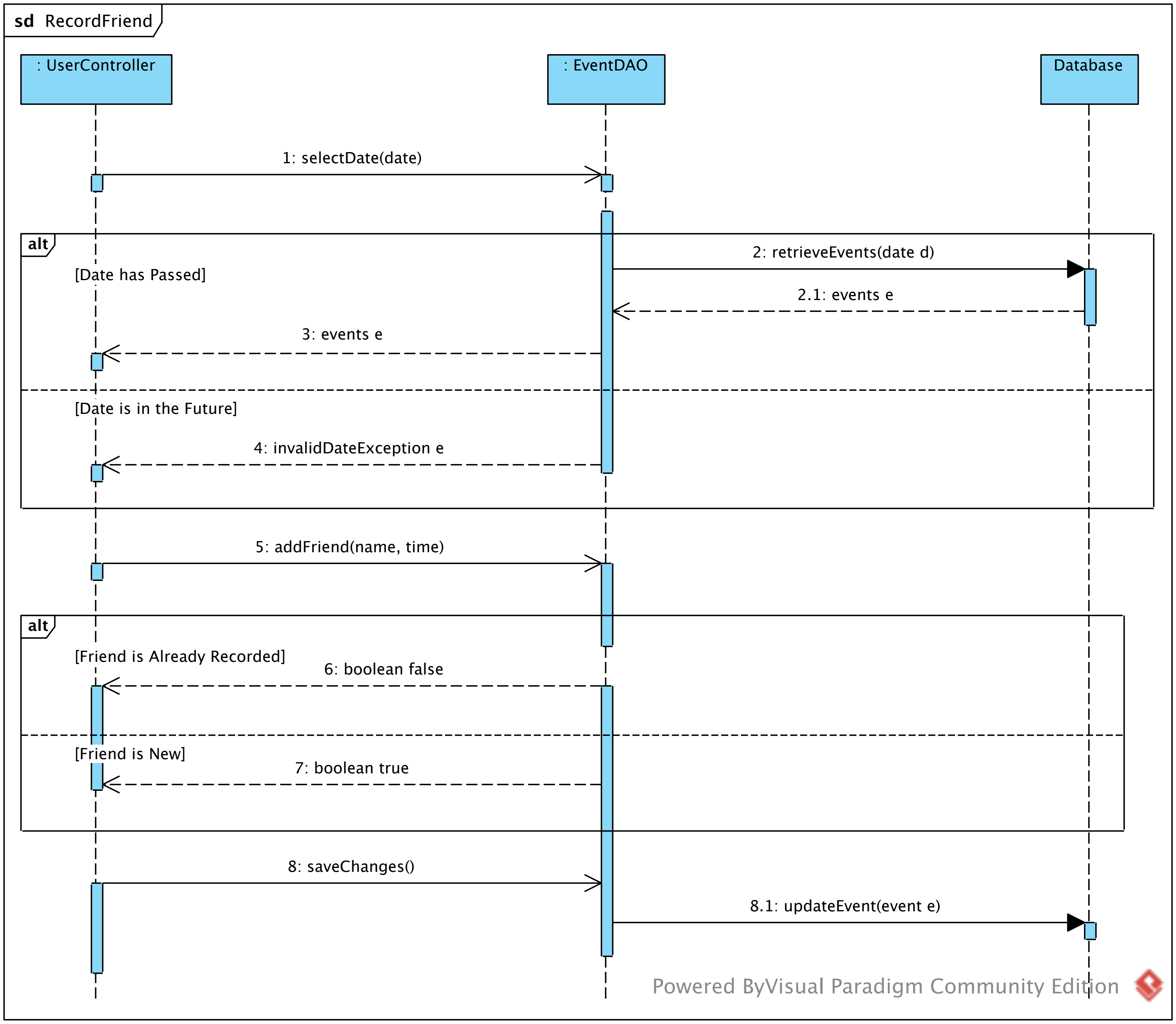
Micah Dadson



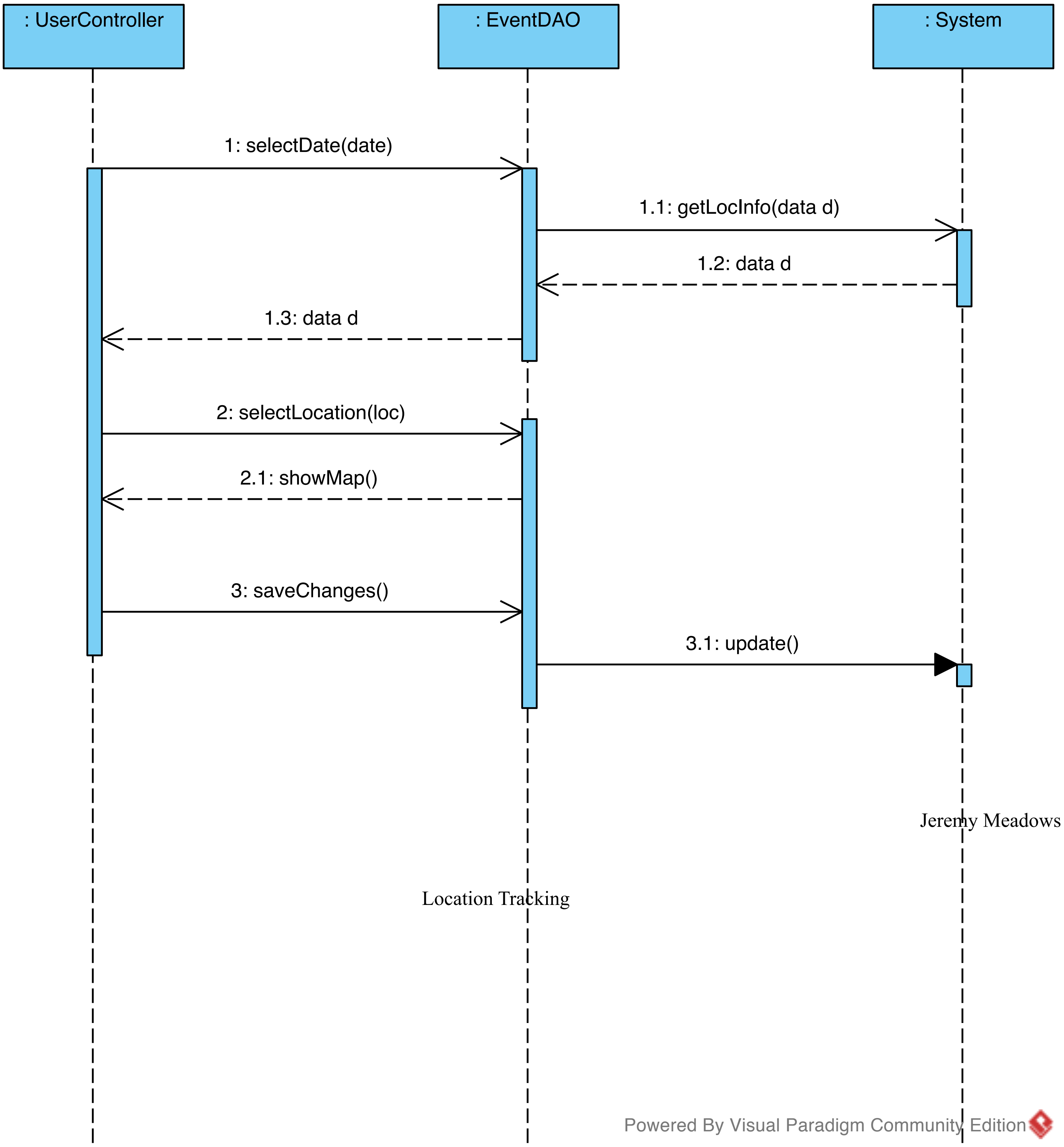
Joel Futagawa

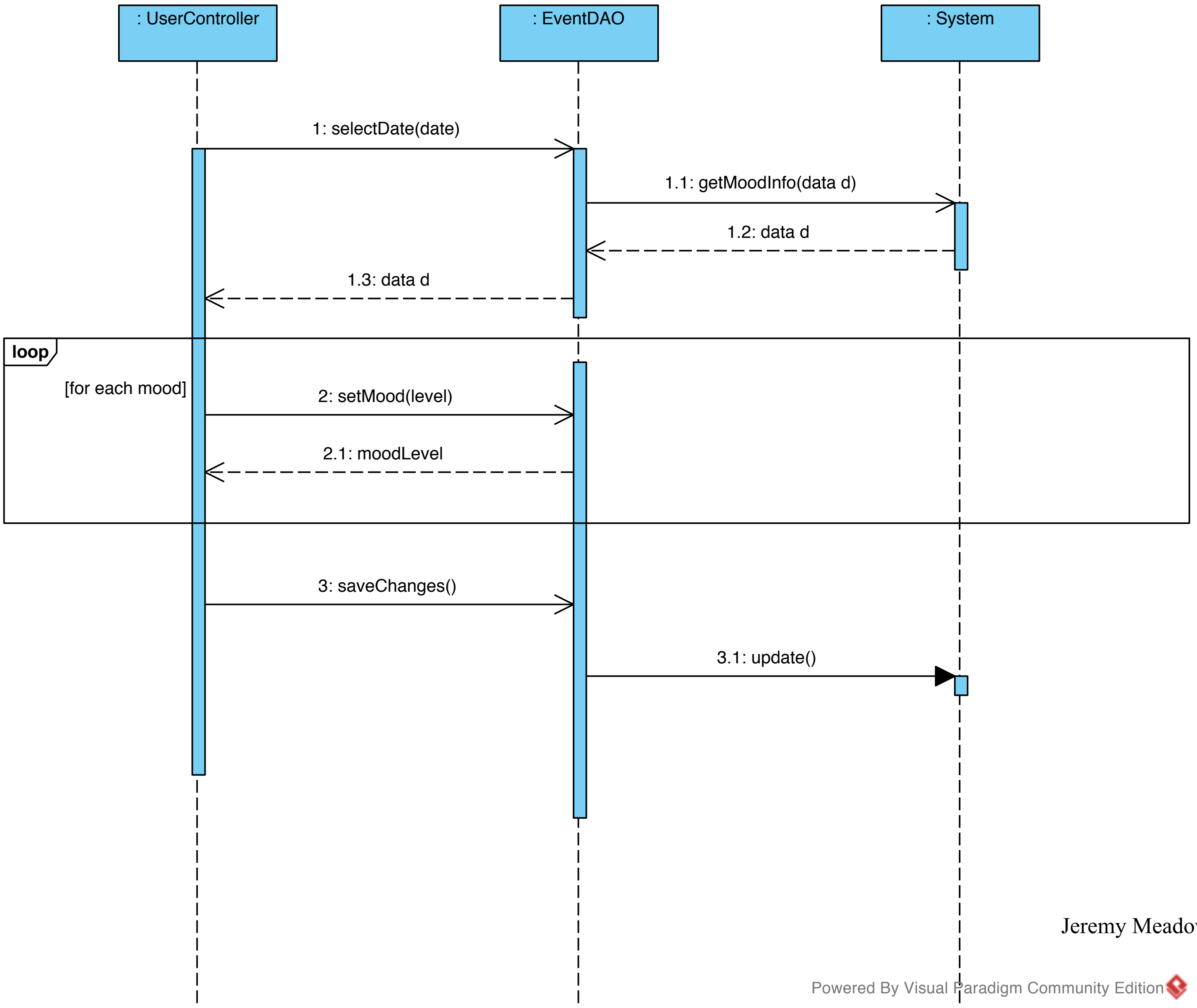


Joel Futagawa



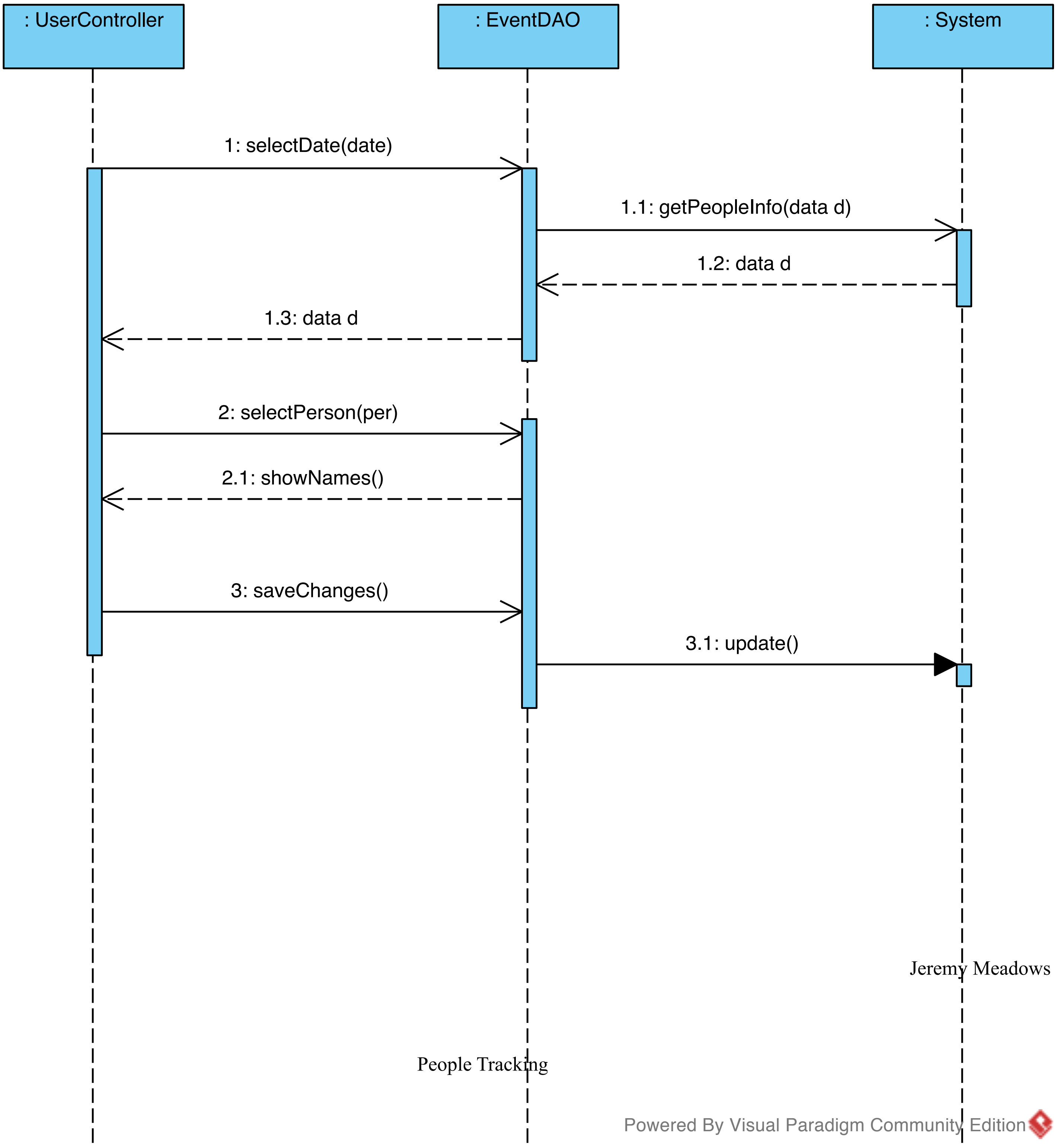
Joel Futagawa

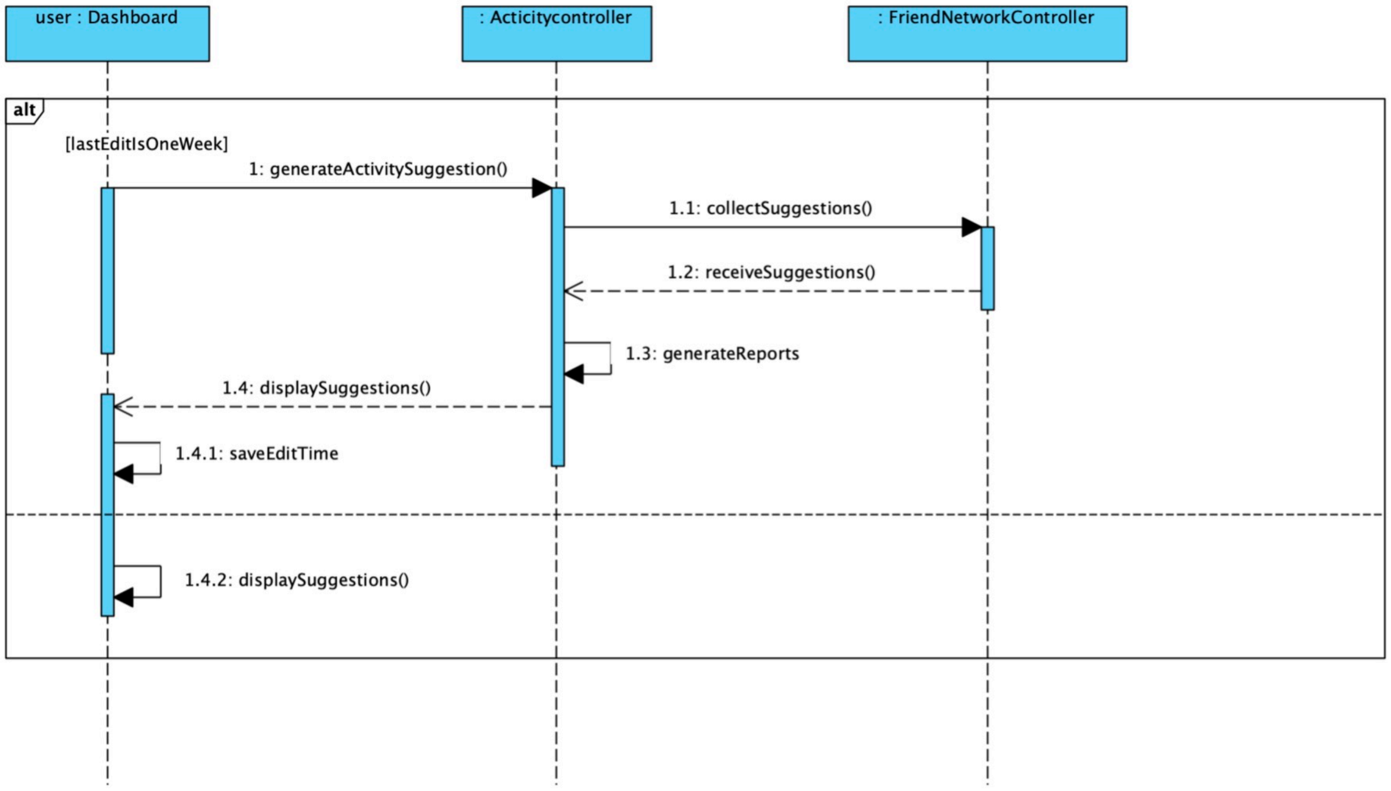

: UserController
: EventDAO
: System
1: selectDate(date)
1.1: getLocInfo(data d)
1.2: data d
1.3: data d
2: selectLocation(loc)
2.1: showMap()
3: saveChanges()
3.1: update()
Jeremy Meadows
Location Tracking
Powered By Visual Paradigm Community Edition



Jeremy Meadows

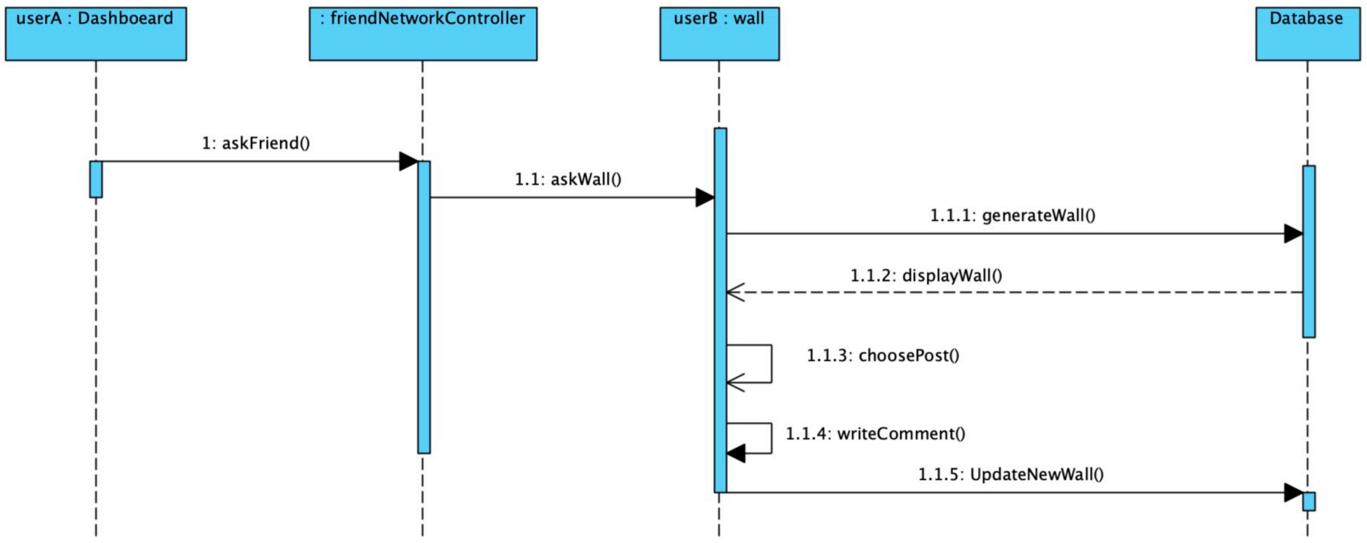
Mood Tracking


: UserController
: EventDAO
: System
1: selectDate(date)
1.1: getPeopleInfo(data d)
1.2: data d
1.3: data d
2: selectPerson(per)
2.1: showNames()
3: saveChanges()
3.1: update()
Jeremy Meadows
People Tracking


user : Dashboard
: Acticitycontroller
: FriendNetworkController
alt
[lastEditIsOneWeek]
1: generateActivitySuggestion()
1.1: collectSuggestions()
1.2: receiveSuggestions()
1.3: generateReports
1.4: displaySuggestions()
1.4.1: saveEditTime
1.4.2: displaySuggestions()

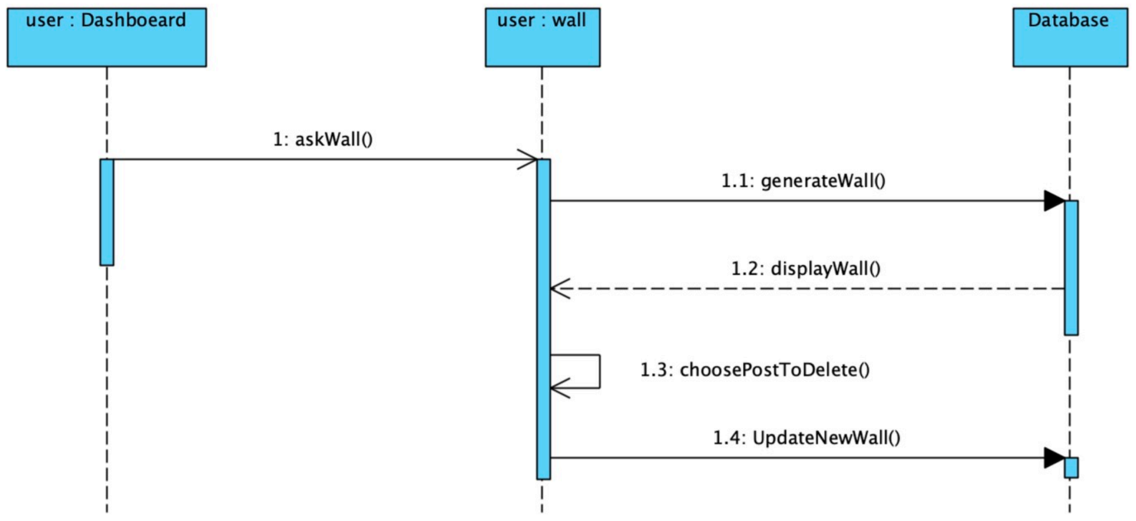
Shunting Chen

Activity Suggestion



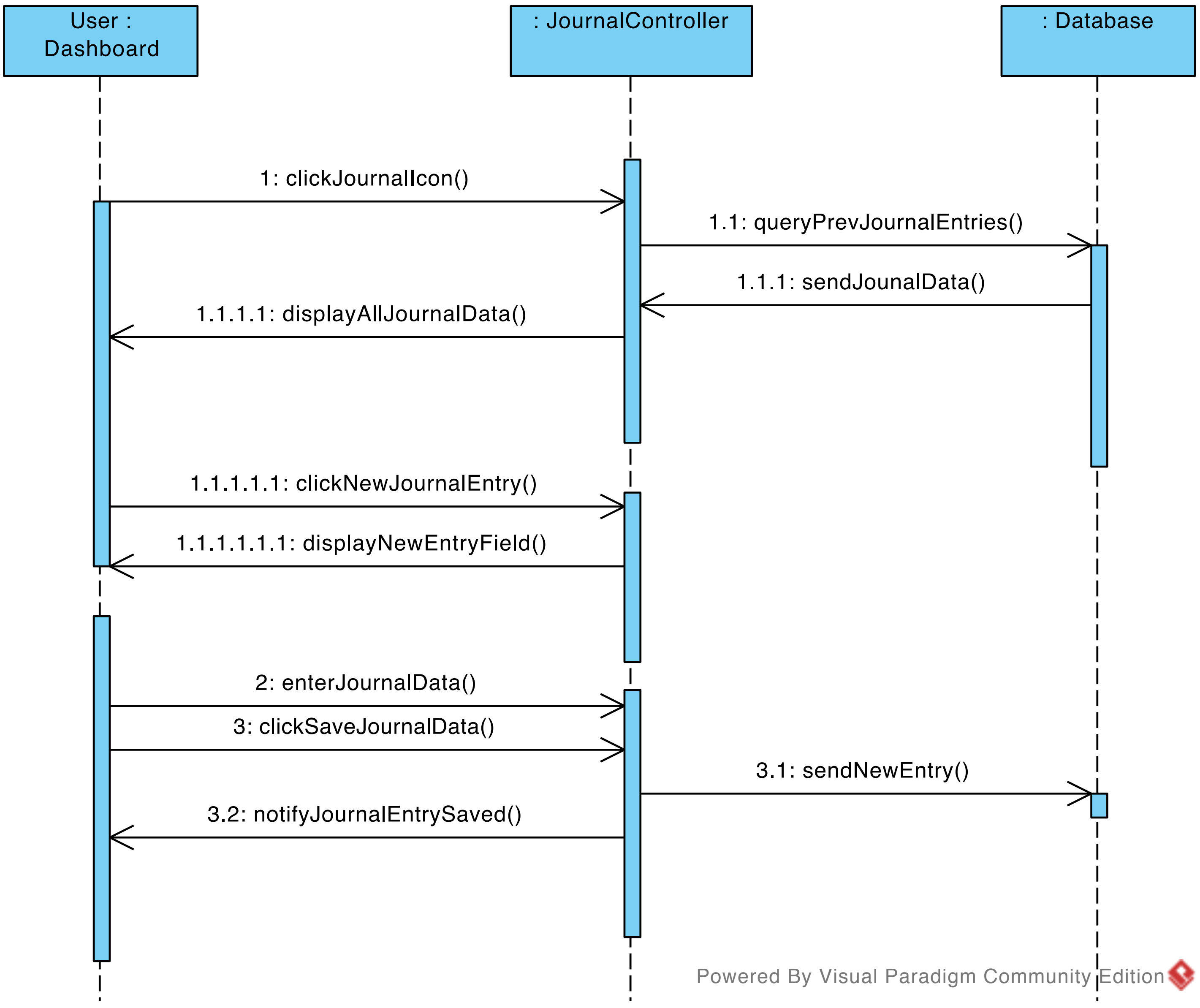
Shunting Chen

Comment Post



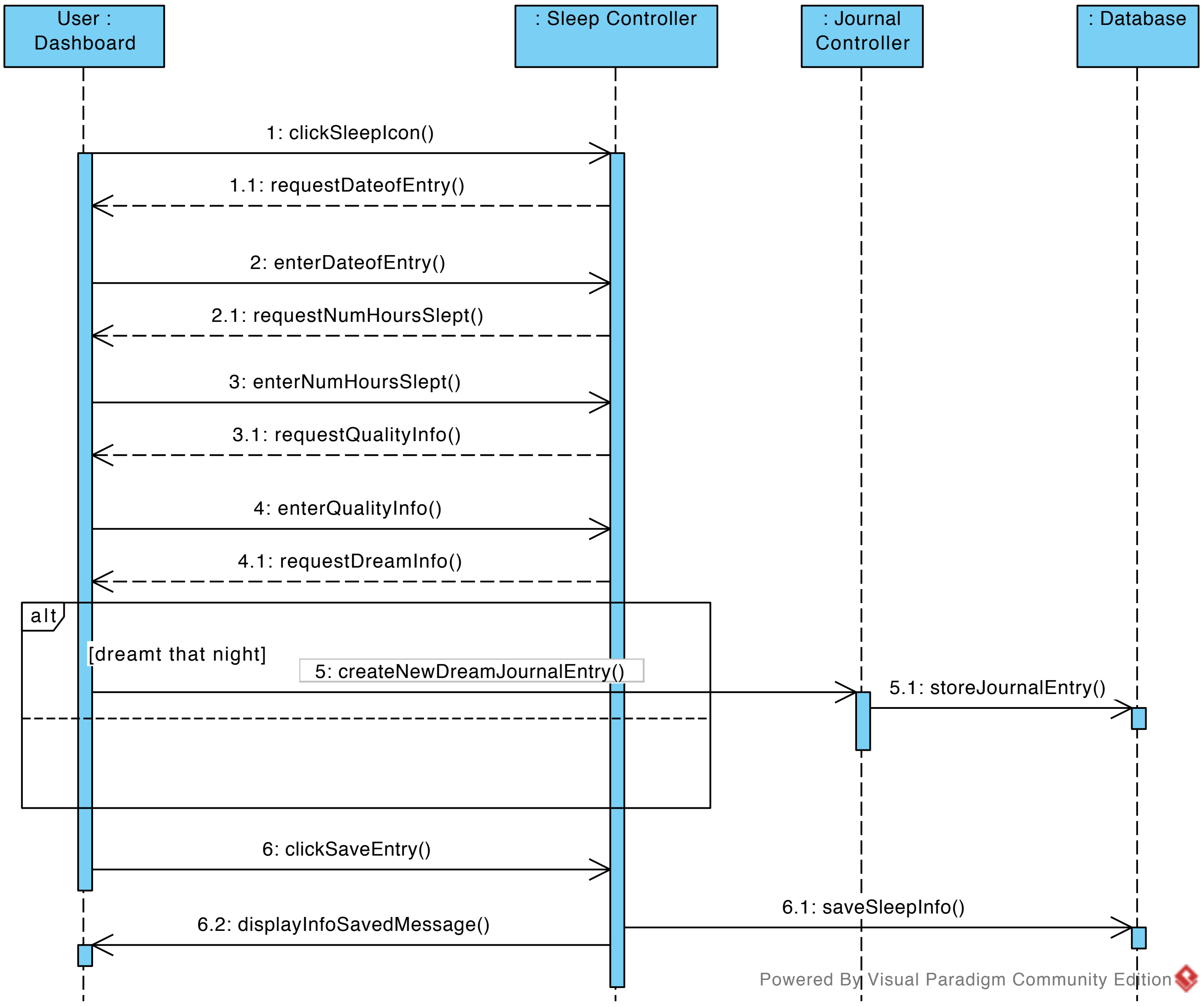
Shunting Chen

Delete Post


User : Dashboard
: JournalController
: Database
1: clickJournalIcon()
1.1: queryPrevJournalEntries()
1.1.1: sendJounalData()
1.1.1.1: displayAllJournalData()
1.1.1.1.1: clickNewJournalEntry()
1.1.1.1.1.1: displayNewEntryField()
2: enterJournalData()
3: clickSaveJournalData()
3.1: sendNewEntry()
3.2: notifyJournalEntrySaved()
Powered By Visual Paradigm Community Edition

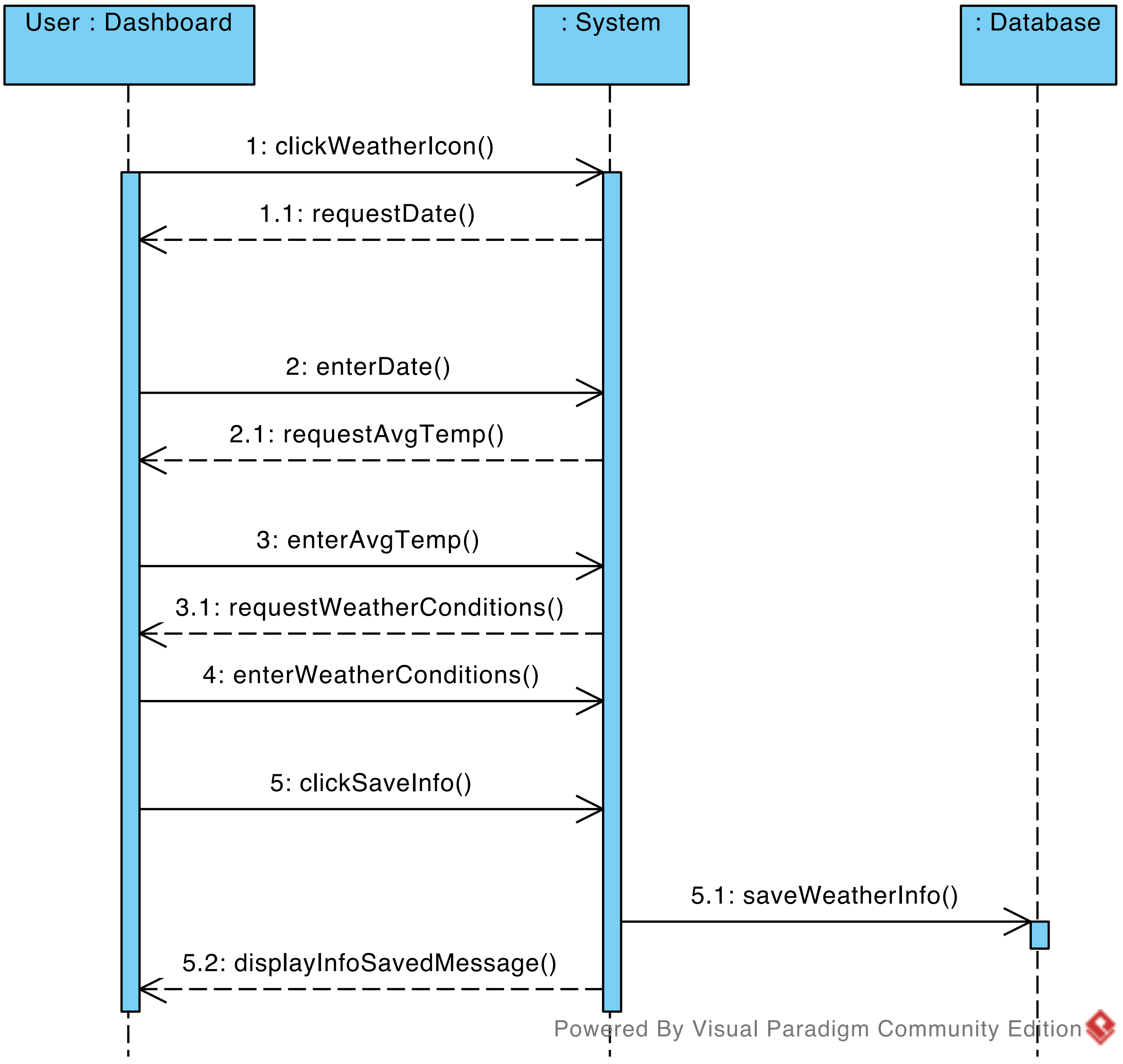
Shivani Bobbala

Journal



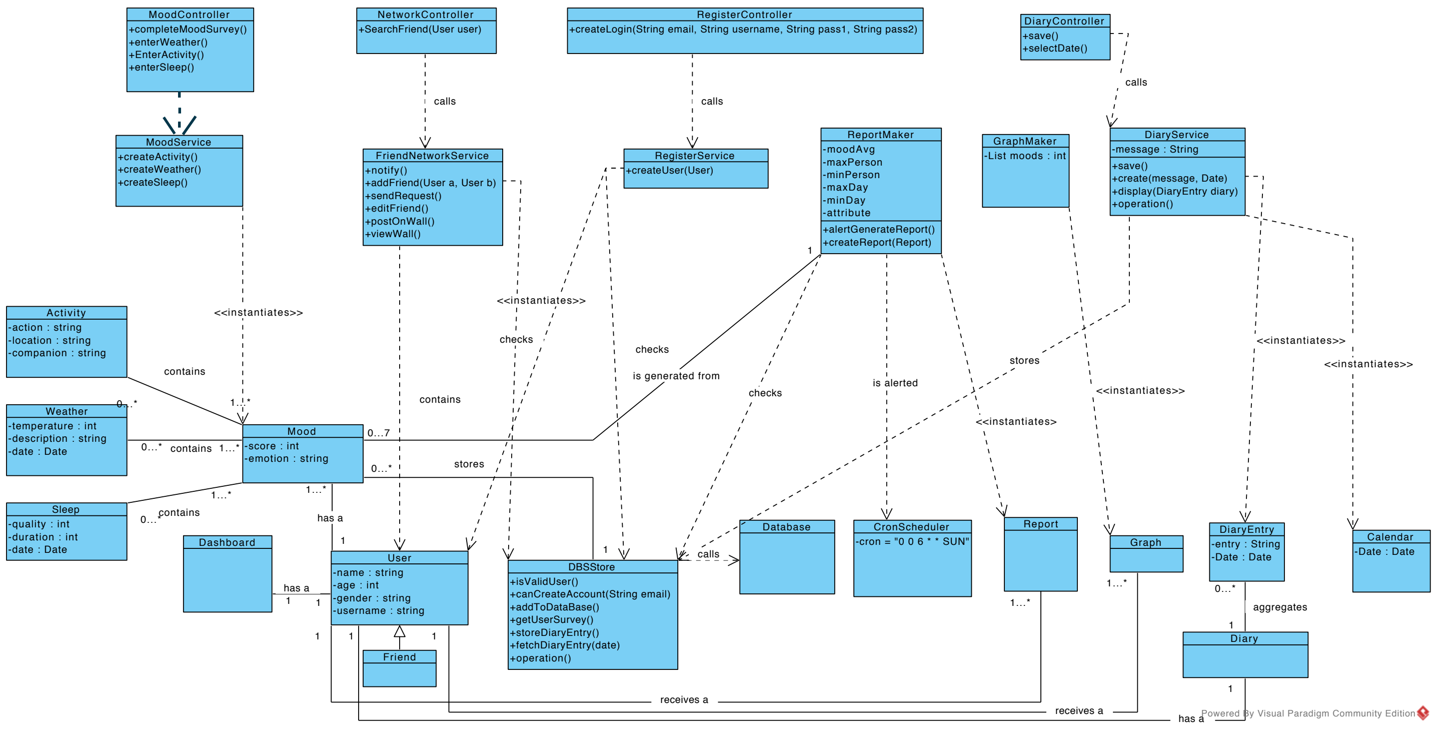
Shivani Bobbala

Enter Sleep Information



Enter Weather Information

Shivani Bobbala


MoodController
+completeMoodSurvey()
+enterWeather()
+EnterActivity()
+enterSleep()
NetworkController
+SearchFriend(User user)
RegisterController
+createLogin(String email, String username, String pass1, String pass2)
DiaryController
+save()
+selectDate()
calls
calls
calls
MoodService
+createActivity()
+createWeather()
+createSleep()
FriendNetworkService
+notify()
+addFriend(User a, User b)
+sendRequest()
+editFriend()
+postOnWall()
+viewWall()
RegisterService
+createUser(User)
ReportMaker
-moodAvg
-maxPerson
-minPerson
-maxDay
-minDay
-attribute
+alertGenerateReport()
+createReport(Report)
GraphMaker
-List moods : int
DiaryService
-message : String
+save()
+create(message, Date)
+display(DiaryEntry diary)
+operation()
<<instantiates>>
<<instantiates>>
checks
checks
is generated from
checks
is alerted
stores
<<instantiates>>
<<instantiates>>
<<instantiates>>
<<instantiates>
Activity
-action : string
-location : string
-companion : string
contains
Weather
-temperature : int
-description : string
-date : Date
contains
Sleep
-quality : int
-duration : int
-date : Date
contains
Mood
-score : int
-emotion : string
0...*
1...*
0...*
1...*
1...*
0...*
0...7
0...*
1
contains
stores
has a
1...*
1
Dashboard
has a
1
1
User
-name : string
-age : int
-gender : string
-username : string
Friend
DBSStore
+isValidUser()
+canCreateAccount(String email)
+addToDataBase()
+getUserSurvey()
+storeDiaryEntry()
+fetchDiaryEntry(date)
+operation()
1
calls
Database
CronScheduler
-cron = "0 0 6 * * SUN"
Report
Graph
DiaryEntry
-entry : String
-Date : Date
Calendar
-Date : Date
1...*
1...*
0...*
aggregates
1
Diary
1
1
1
1
receives a
receives a
has a
Powered By Visual Paradigm Community Edition

Testing Plan

Registration

- Register with email already in database
  - Test passes if registration is unsuccessful
- Register with passwords that do not match
  - Test passes if registration is unsuccessful
- Register with email that is not in database and with passwords that match
  - Test passes if registration is successful and stored in database

FriendNetwork

- Add a friend not in network
  - Test passes if add is successful
- Add a friend already in network
  - Test passes if add is unsuccessful and stored in database

ReportMaker

- Make report with one or less stored mood surveys
  - Test passes if report cannot be generated
- Make report with at least two mood surveys stored
  - Test passes if report is generated

HappinessGraphMaker

- Make graph with one or less stored mood surveys
  - Test passes if report cannot be generated
- Make graph with at least two mood surveys stored
  - Test passes if report is generated

DiaryService

- Make diary entry with no user input
  - Test passes if diary entry is not created and stored in database
- Make diary entry with user input
  - Test passes if diary entry is created and stored in database
- Look up diary entry for date with null diary entries
  - Test passes if error message is generated
- Look up diary entry for date with at least one diary entries
  - Test passes if all diary entries are presented

MoodSurvey

- If any field is left empty
  - Test passes if validation check prevents user from submitting survey
- If all fields are complete
  - Test passes if serialized object is stored in the database

Behavioral Suggestions

- If user does not have friends
  - Test passes if user receives suggestions from robot
- User has friends
  - Test passes if user is recommend the activities from his/her friend with the greatest average mood

WeatherSurvey

- If temperature is not entered
  - Test passes if validation check prevents user from submitting survey
- If temperature is entered
  - Test passes if the serialized object is stored in the database

SleepSurvey

- If duration is not entered
  - Test passes if validation check prevents user from submitting survey
- If duration is entered
  - Test passes if the serialized object is stored in the database

ActivitySurvey

- If description is not entered
  - Test passes if validation check prevents user from submitting survey
- If description is entered
  - Test passes if the serialized object is stored in the database

Cron Scheduler

- Test passes if system event occurs only at the specified cron

Feed

- Delete Post
  - Test passes if all conditions are meant: the post is removed from the feed, is removed from the database, and the post is replaced with a different post from the feed

- Post on Wall
    - Test passes if the post replaces a different post on the feed, and if the post is added to the database
- Comment on Post
    - Test passes if comment is saved to feed and database

GRASP

1. The Creator:
    a. HappinessGraph
        i. constructor creates Chart because Happiness Graph aggregates Chart objects
    b. PostToWall
        i. constructor creates Post object (by current user) because PostToWall has all the initializing data that will be passed to Post when it is created
    c. Feed
        i. updateFeed creates Post object because Feed contains three Post objects
    d. Dashboard
        i. constructor creates Feed because Dashboard contains a Feed object
    e. Dashboard
        i. constructor creates PostToWall because Dashboard contains a PostToWall object
    f. Post
        i. constructor creates StarRater because a Post object has a Star Rater object
    g. Dashboard
        i. constructor creates Search because it contains a Search object
    h. DiaryService
        i. create() is responsible for creating a DiaryEntry object because it has all of the necessary information to do so
    i. ReportMaker
        i. createReport() is responsible for creating a Report object because it has all of the necessary information to do so
    j. RegistrationService
        i. createUser() is responsible for creating a User object because it has all of the necessary information to do so
    k. MoodService
        i. createMood() is responsible for creating a Mood object because it has all of the necessary information to do so

2. Information Expert:
    a. Chart
        i. constructor is responsible for creating a basic graph that displays mood history for a 7 day interval because it has all of the necessary information to do so
    b. Feed

        i. updateFeed is responsible for updating the posts displayed on the wall after a new post is made because it has access to Post objects
    c. ReportMaker
        i. createReport creates a report based off of user data because it has all of the necessary information to do so
    d. FriendNetworkService
        i. addFriend() will be responsible for notifying DBSStore to add userA and userB to the same network because it will have the necessary information (email addresses) to do so
        ii. sendRequest() will be responsible for inviting userB to be friends with userA because it has access to userB's email
        iii. notify() is responsible for altering userB that userA wants to be friends because it has access to userB's email
3. Low Coupling and High Cohesion:
    a. Though DashBoard depends on many classes, it redirects to other JPanels or opens JDialogues. As a result, a change to one of its dependencies does not grossly affect its functionality.
    b. HappinessGraph is coupled to Chart because it contains a Chart object and displays it on its JPanel. Because it simply adds Chart to its JPanel, a change to Chart should not affect the functionality of HappinessGraph.
    c. Likewise, Feed contains Post objects. It must access a Post object's member variables, resulting in High Coupling. Because the purpose of Feed is to update Posts displayed on a Dashboard, the high coupling was justified as it allows for high cohesion.
4. PureFabrication
    a. DBSStore will have access to Database and program logic to maintain high cohesion and low coupling
5. Controller
    a. RegistrationController
        i. createLogin() is responsible for altering RegisterService after taking in user input to limit its responsibilities
    b. NetworkController
        i. calls searchFriend() which will alert FriendNetworkService to call DBSStore to check if the user exits. Because it only receives a username as a parameter, it has insufficient information to do anything else
    c. UserController will handle the adding of an event. We are assigning this responsibility as a use case scenario
    d. UserController will handle the posting to a wall. We are assigning this responsibility as a use case scenario

    e. DiaryController
        i. save() is responsible for altering DiaryService to save the diaryEntry
6. Law of Demeter: Our classes only know about each other if one class must somehow manipulate another.