Second Iteration Submission

GroupUs

Rui Ge rg3105

Ruimin Zhao rz2390

Hao Fang hf2345

Shenxiu Wu sw3196

November 27, 2018

1 Section One: Implementation MVP

- Pre-commit configuration
  - Run tests: The maven test suites will be run before commit.
  - Check files: Default checks such as checking file name will be run.
- Post-commit configuration
  - We configured Jinkens pipeline as described in our first iteration report.

2 Section Two: Use Cases

2.1 Sign in/up

2.1.1 Sign up

- Note: step 2 to 4 could change the order, but here we assume the order to be 2, 3, 4 for simplification. The steps that could contain wrong actions are step 2, 3, and 4. Theoretically there could be 8 possible alternate flows, but there we list 3 symbolic ones where each has one wrong step.
- Title: Sign up an account
- Actor: Unregistered User
- Description: Part of this group up system. The user registers an account with a unique Columbia email address as the username and a password with valid content.
- Precondition: The program is available on user's local laptop.
- Triggers: The user execute the program and get sign up/in window.
- Basic Flow:
  - 1. User starts the system and the system prompts out a window for the user to enter registration information.

  - 2. User enter the email address with correct format (with the suffix being "@columbia.edu", only consisting of numbers and normal letter characters, and with prefix pattern being either "2 letters + 4 digits" or "3 letters + 4 digits") which has never been used to register before.
  - 3. User enter the password with correct format (with the input name starting with letter (character), and with length no more than 20).
  - 4. User enter the name with correct format (with the input password starting with letter (character), and with the length between 6 to 18, and only consisting of characters/numbers/underlines).
  - 5. User click on the "Sign Up" button on the window, and complete the registration.

- Alternate Flow 1:
  - Note: Step 2 has error, the other steps are the same as basic flow (except the result for final step).
  - 2. User enter the email address with wrong format (with the suffix being "@gmail.edu"/with special characters other than letters and digits/with prefix pattern being neither "2 letters + 4 digits" nor "3 letters + 4 digits"), or which has already been used to register before.
  - 5. User click on "Sign up", then the registration fails and user is re-directed to step 1 user interface.

- Alternate Flow 2:
  - Note: Step 3 has error, the other steps are the same as basic flow (except the result for final step).
  - 3. User enter the password with wrong format (with the input name starting with number/with length more than 20).
  - 5. User click on "Sign up", then the registration fails and user is re-directed to step 1 user interface.

- Alternate Flow 3:
  - Note: Step 4 has error, the other steps are the same as basic flow (except the result for final step).
  - 4. User enter the name with wrong format (with the input password starting with number/with the length out of 6 to 18 range/containing special characters).

- 5. User click on "Sign up", then the registration fails with an alert window prompted out, and the user is re-directed to step 1 user interface.

2.1.2 Sign in

- Title: Sign in to an account
- Actor: A registered user who has already signed up.
- Description: Part of this group up system. The user who has already registered an account logs in to the account with correct combination of username and password.
- Basic Flow:
  - 1. User starts the system and the system prompts out a window for the user to enter registration information.
  - 2. User enter the email and password he/she used for registration.
  - 3. User click on the "Sign In" button on the window, and log in.
- Alternate Flow:
  - Note: Step 2 has error, the other steps are the same as basic flow (except the result for final step).
  - 2. User enters wrong combination of password and email which do not fit any account stored in database/any wrong format input itself.
  - 3. User click on the "Sign In" button on the window, then the sign in fails with an alert window prompted out, and the user is re-directed to step 1 user interface.

2.2 Post

- Title: Post an event
- Actor: A registered user who has already logged in
- Description: A user who wants to post an event, he/she at the post screen with specific properties including category, subject, time, location, memo and description.

- Basic Flow: The user is currently in the post screen now.
  - 1. The user presses the "Post" button at the status screen and the system will be redirected to the post screen for posting events.
  - 2. The user clicks on the drop down box to select the event category among "study", "eat out" and "go home". The default category setting is "study".
  - 3. The user enters a subject of event with the correct format(length between 1 to 20)
  - 4. The user enters a valid location for the event(length between 1 to 200 and can pass the check of external geology application)
  - 5. The user clicks on the drop down box to choose proper start date and end date of the event.There are two drop down boxes for each. Select a certain day and specify the accuracy of time to one minute.
  - 6. The user enters content into the field of "memo" and "description" with correct format(length between 1 to 200)
- Alternate Flow1: The user comes to the post page. Because the default category is "study", this field will not cause any conflict even the user don't select it.
  - 1. After choosing any category among those three categories, he or she type a subject which exceed 20 bytes, which obviously is not a correct format in our standard. Another condition is that user input some special character will are not allowed. This will trigger a same alert as over-long length. Thus, here, we combine the 2 situations to have a discussion.
  - 2. The user enters a valid location for the event, here we assume this location can pass the check of external geology API.
  - 3. The user clicks on the drop down box to choose proper start date and end date of the event.There are two drop down boxes for each. Select a certain day and specify the accuracy of time to one minute.
  - 4. The user enters content into the field of "memo" and "description" with correct format(length between 1 to 200)
  - 5. Once the user click the "post" button, the system will check the input information and finally find out the subject is invalid. The system will pop out an alert window indicating the user some error information and

tell them how the subject input violated our rules and how to fix it. Finally, once the user click the "OK" button on the alert window, our system will return the user to step 1.

- Alternate Flow2:
  - 1. Similar with the assumption of alternate flow 1, but this time, the user input a invalid location while other required information is valid.
  - 2. System will check the inputs and call the geology API to check and later find out the location input is invalid. The system will pop out an alert window indicating the user that the location input should be revised and then return the user to alternate flow step 1.
- Alternate Flow3:
  - 1. Similar with the assumption of alternate flow 1, but this time, the user input a invalid description and/or memo while other required information is valid.
  - 2. Once the user click the "post" button, the system will check the input information and finally find out the description and/or memo is invalid. The system will pop out an alert window indicating the user some error information and tell them how the input violated our rules and how to fix them. Finally, once the user click the "OK" button on the alert window, our system will return the user to alternate flow step 1.
- Alternate Flow3:
  - 1. Quite Similar with the analysis of aforementioned flow 1, but this time, the user input both invalid location and description memo while other required information is valid.
  - 2. Once the user click the "post" button, the system will follow the following order to check the input information and return some error info to ask user to revise through alter window if exists. The check order is:

$$\begin{aligned} &\textit{subject text} \Rightarrow \textit{location text} \Rightarrow \textit{location validation} \\ &\qquad \Rightarrow \textit{memo text} \Rightarrow \textit{description text}. \end{aligned} \tag{1}$$

Once any one or some check failed, the system will pop out an alert window indicating the user what error happened and tell them how the input violated our rules and how to fix them. Finally, once the user click the "OK" button on the alert window, our system will return the user to alternate flow step 1.

2.3 Search

- Title: category based search
- Actor: A user who wants to see the events appealing to him/her.
- Description: A user who wants to see the existent events, he/she signed up already.
- Basic Flow: The user is in the search screen now. He/She needs to select search by category.
  - 1. The user selects one category in the all three possible categories: Study , Eat Out, Go Home.
  - 2. The user also enters his/her current location, then clicks the search button.
  - 3. The system will redirected to a new screen showing the events in that category. The results will be in the order of the distance from user's current location to the events' locations.
  - 4. The user could see other critical information like start time, end time about this event and make his/her favourite choice.
  - 5. The expired events will not be displayed.
- Alternate Flow1:
  - Note: 1.In step 2, if the user doesn't enter his/her current location, there will be no error.
  - 2.The system will still redirect to a new screen. The user can still see the events in one specific category.
  - 3.The other information about this event is still available, but the events are not in the order of the distance.
- Alternate Flow2:

- Note: 1.In the step 2, if the entered location is invalid, there will be no result.
- 2.The system won't redirect to a new screen, instead a error message alert will be shown to the user. The user will know the entered location is invalid and he needs to enter a valid location or just skip this blank.
- 3.The user can't see the events in the corresponding category absolutely.

2.4 Join

- Title: Join a event
- Actor: A user who wants to join an event
- Description: The user wants to join a group study event, he/she signed up already and never used our desktop app before.
- Basic Flow: The user could see the searched results now.
  - 1. The user would like to select a suitable event to join. He/She could just select a event and click the join button.
  - 2. System will redirect to the user's profile page after joining into the event. The user could manage other events he posted or joined here.
- Alternate Flow1:
  - Note: 1.In step 1, if the user is reviewing a joined event and wrongly click the join button.
  - 2.The system will not redirect to a new screen. Instead, an error message will be shown to the user, which is "you joined this event already!".
- Alternate Flow2:
  - Note: 1.In step 1, if the event is posted by the user, the similar error will be shown.
  - 2.when the user searched the events by category, he/she can also see the events posted by himself/herself.
  - 3.If the user wrongly click join button to his/her posted event. The same error occurred, because when someone posted a event he/she is assumed to join this event.

3 Section Three: Test Plan

3.1 Sign in/up

- Test case: email
  - Length:
    - Valid equivalence partitions: the length of the email is between 1 to 20. e.g.“ab1234@columbia.edu”
    - Invalid equivalence partitions: the length of the email is outside the range of 1 to 20. e.g.“abcd1234@columbia.edu”
    - Boundary condition: the boundary condition of the email length is just at, below and above the two sides of length which are 0,-1,1,20,19,21. e.g.valid(“abc3456@columbia.edu”) invalid(“a3@columbia.edu”)
  - Character:
    - Valid equivalence partitions: the valid email only allows characters in 26 letters, numbers and the separator“@”. e.g. “ty9805@columbia.edu”
    - Invalid equivalence partitions: the email contains any character that is beyond the range of 26 letters, numbers and “@” is invalid. e.g. “$gy3452@columbia.edu”
    - Boundary condition: email with at least one invalid character is the invalid side of the boundary condition. e.g. “gh%3567@columbia.edu”
  - Suffix:
    - Valid equivalence partitions: the email that has the exact suffix format “@columbia.edu” e.g. “gg9604@columbia.edu”
    - Invalid equivalence partitions: email with any suffix other than “@columbia.edu” e.g. “bu3702@gmail.com”
    - Boundary condition: email with the suffix format that is same or not same as the format “@columbia.edu”
  - Prefix:
    - Valid equivalence partitions: email with the prefix format of 2-3 characters first and 4 numbers following. e.g. “abc1234@columbia.edu”
    - Invalid equivalence partitions: email with any prefix other than the given pattern. e.g. “432rt@columbia.edu”, “abcd12@columbia.edu”

    * Boundary condition: email with 1/4 normal characters or 3/5 numbers in the prefix. e.g. “b1234@columbia.edu”, “efgh3456@columbia.edu”, “hf123@columbia.edu”

- Test case: password
  - Length:
    * Valid equivalence partitions: the password with the length of 6-18 e.g. “ase12345”
    * Invalid equivalence partitions: the password outside the given length range which is smaller than 6 or larger than 18. e.g. “1234”, “1234567890abcdefghijkl”
    * Boundary condition: the password with length of 4,5,6,17,18,19. e.g. “gogogo”
  - Character:
    * Valid equivalence partitions: the password that start with letters and contains only normal characters, numbers and underlines. “ abc_123”
    * Invalid equivalence partitions: the password that contains special characters other than the specific types above or does not start with letters. e.g. “abc$123”, “123456”
    * Boundary condition: the password that contains at least one invalid characters or contains all types of valid characters. e.g. “abc_123”, “%12345”

- Test case: name
  - Length:
    * Valid equivalence partitions: the length of name is between 1 to 20. e.g. “Jimmy”
    * Invalid equivalence partitions: the length of name is 0 or larger than 20. e.g. “GoodhappyactiveniceJacky”
    * Boundary condition: the length of name is 0,1,2,19,20,21. e.g. “A”,“testlengthtestlength”
  - Character:
    * Valid equivalence partitions: the name starts with at least one character among 26 letters and contains only 26 letters, numbers and special characters among “_%&;=?” e.g. “Name_&123”

        * Invalid equivalence partitions: the name that contains characters other than the specific range or does not start with at least one character among 26 letter. e.g. “123jack”
        * Boundary condition: the name that does not start with 26 letters/ contains letters out of given range. e.g. “ *Harry*”

3.2 Search

- Test case: location
    - Length:
        * Valid equivalence partitions: the length of the string location is smaller than 100.
        * Invalid equivalence partitions: the length of the string location is smaller than 100.
        * Boundary condition: length = 49, length = 50, length = 51.
    - Character:
        * Valid equivalence partitions: string location only consists of letters or numbers
        * Invalid equivalence partitions: string location contains some special characters
        * Boundary condition: string with $.
    - Distance Check method in Distance Class:
        * Valid equivalence partitions: Locations enable Distance check method returns true.
        * Invalid equivalence partitions: Locations enable Distance check method returns false.
        * Boundary condition: string = null; string = ttt.

3.3 Join

- Test case: eventId
    - whether the event is joined by the user or not
        * Valid equivalence partitions: event exists in the user’s joined list

    * Invalid equivalence partitions: event doesn't exist in the user's joined list
    * Boundary condition: eventId = any eventId not in the user's joined list like "test@columbia.eduTue Nov 20 20:39:06 EST 2019"

3.4 Post

Here, the attributes we tested include the user inputs including Subject, Location, Description, and Memo. Here the Location check is the same as the one described in Search and the Description and Memo follow the same check routine as Subject except the length boundary is larger than that of Subject. Therefore, we only provide the detailed explanation for Subject here.

- Test case: subject
  - Length:
    * Valid equivalence partitions: the length of the subject should range from 1 to 20. e.g. "Eat McDonald"
    * Invalid equivalence partitions: the length of the subject is 0 or larger than 20. e.g "we want to hold an activity that is very good and interesting"
    * Boundary condition: the boundary condition corresponding to the max and min length is length 21/20/19 and 2/1/0. e.g. "testlength-testlength", NULL
  - Character:
    * Valid equivalence partitions: the characters of the subject can only contain 26 letters,numbers or space. e.g. "Review Maths123"
    * Invalid equivalence partitions: the subject contains other special characters besides 26 letters,numbers or space. e.g. "%Go home%"
    * Boundary condition: subject with any one character that is outside the given range. e.g. "%Discuss ASE"

4 Section Four: Branch Coverage

- Overview:
  In terms of branch coverage, we focus on the back-end code. And the files we wrote tests for are those inside the folder "impl", which contains all the back-end methods are directly called by the front-end code to interact with the database with the user input. And the two particular files we focused on are "UserServiceImpl" and "EventServiceImpl" which together cover all the methods directly called by front-end. The other files are some middle methods used for making the logic clearer in this work-flow with database operation involved.

- Explanation:
  As can be seen in the screenshots below, all the classes within the "impl" folder have branch coverage of 100%. We achieved this by executing "run with coverage (jacoco code coverage configuration)" locally in IDE while modifying the code for more check branches and adding unit tests.

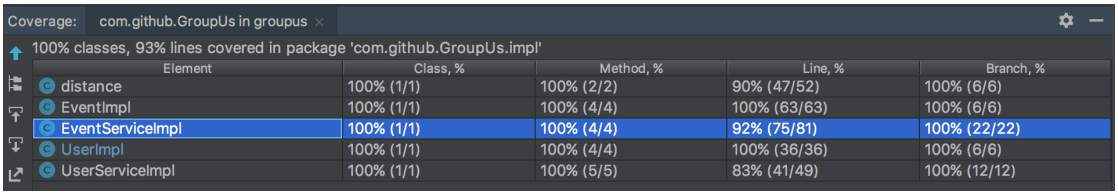

Figure 1: Local IDE Result

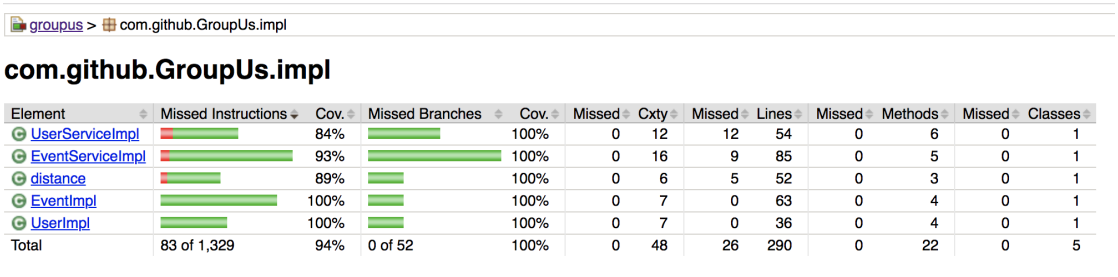

Figure 2: Jinkens Build Result