ASE Final Iteration – Int_Elligence

Sai Srujan Chinta (sc4401)

Chandana Priya Gandikota (cg3111)

Varun Chanddra (vc2491)

Jiayi Wang(jw3704)

Create Account: As a user, I want to be able to create an account
so that I can sync my schedule and use the system. My conditions of
satisfaction are that system stores my basic profile information and allows me to set a login password.

Use-Case for Create Account:

Title: Create Account

Actor: User of Meeting Scheduler

Description: Customer creates a new account using valid

credentials Precondition: Application displays the create account

page

Basic Flow:

- User enters their first name and last name in the respective fields.
- User enters unique username and valid email address.
- User enters a password of their choice which conforms to the requirement (regex testing) and re-enters it accurately in the confirm password field.
- Upon clicking the create account button, if all the fields are entered correctly, the user account is created and all the details are added to the database.
- The user is now re-directed to the login

page. Alternate Flow:

- If any of the fields are left empty, the application throws an error.
- If the password and confirm password values do not match, an error message is displayed.
- If the username is not unique, the user account creation fails.
- E-mail address needs to match the standard template for email address (regular expression check is performed to enforce this)
- If the password does not satisfy certain pre-defined requirements, an error message is displayed.

Postcondition: User account is created and the details are added to the database.

Sign In: As an authorized user, I want to be able to log-in to my account so that I can access, update my information, find upcoming meetings, schedule new meetings, find best time slot for the meeting and get comprehensive report on things covered. My condition of satisfaction are that the system asks for my login credentials (user name, password) and allows me in on valid entries.

Use-Case for Sign In:

Title: Sign-In

Actor: User of Meeting Scheduler

Description: User logs in to their account using valid

credentials Precondition: Application displays the login page

(default page) Basic Flow:

- User enters their username and password.
- Upon clicking the login button, the user is re-directed to the home

page. Alternate Flow:

- If the user leaves either field empty, the application displays and error message.
- If invalid credentials are entered, the user is prompted to enter the correct credentials.
- If the user is not connected to the internet, an error message will be displayed relaying this information.

~~Forgot-password: As a registered user, I want to be able to reset my password so that I can I don't have to fear the loss of credentials. My condition of satisfaction is the system should send me link for password reset to my email provided at the time of registration.~~

~~Use-Case for Forgot Password:~~

~~Title: Forgot Password~~

~~Actor: User for Meeting Scheduler~~

~~Description: User resets his password after authenticating their identity via~~

~~email. Basic Flow:~~

- ~~User clicks the forgot password button.~~
- ~~An email is sent to the email address associated with that user account.~~
- ~~In the database, the password is modified to the temporary password sent via email to the user.~~
- ~~The user is prompted to enter the temporary password and later change this password.~~

Create groups: As event host/ participant, I want to be able to create new groups for people attending same events/meetings and allow sharing of information within that group. My condition of satisfaction is that I should be able to search for people (based on their usernames) and add them to a new group.

Use Case for Create Groups:

Title: Create

Groups Actor:

Group admin

Description: User creates a group to later schedule a meeting

Precondition: The user clicks the create group button and is currently in the page which lets them create a group

Basic Flow:

- The user types in the partial username of users that he wants to add to a group.
- A list of users with similar usernames is displayed and the user selects the people he wants in their group.
- After adding all the users the admin wishes to, they click the create group button. At this point, a new group is created with all the selected users and this information is stored in the database.

Alternate Flow:

- If the user attempts to create a group without any other members, an error message is displayed which does not allow the group creation.

Post-Condition: The user is re-directed to the home page which displays all the groups that the user belongs to.

Location (from admin perspective): As an admin, I want to set the location of a particular meeting. My conditions of satisfaction are a seemless interface to select the location.

Use case for Location (admin perspective):

Title: Location (Admin

perspective) Actor: Admin

Description: Admin sets the location of the

meeting Precondition: The user is currently in the

meeting tab Direct Flow:

- The admin clicks the set location button upon which a new browser window opens.

- Google Maps will run in this web browser and the admin can navigate and select the location that he wants.
- The latitude and longitude of the selected location gets stored in the database and gets associated with that particular meeting.

Alternate Flow:

- If there is no internet connection, the admin gets an error message detailing the problem.

Postcondition: The user returns to the meeting tab.

Location (User perspective): As a user, I want to be able to access the location of a particular meeting so as to be prepared for it. My conditions of satisfaction is that the system should show me ~~a map location for~~ the meeting venue.

Use case for location (user perspective):

Title: Location (User

Perspective) Actor: User

Description: User accesses the location of the

meeting Direct Flow:

- User clicks the get location button for the meeting of their choice. This opens up a browser window wherein the Google Maps API clearly displays the location of the meeting.

Alternate Flow:

- If the user is not connected to the internet, an appropriate error message will pop up.

~~Directions: As a user, I want to be able to search for directions to a meeting place from my current location so that I can easily get to the venue.~~

~~Use Case for directions:~~

~~Title:~~

~~Directions~~

~~Actor: User~~

~~Description: Get directions from current location to meeting~~

~~location Direct Flow:~~

- ~~User clicks the get directions button and is redirected to a web browser which shows them the directions from their location to the location of the meeting.~~

~~Alternate Flow:~~

- ~~If the user is not connected to the internet, an appropriate error message will pop up.~~

Check-in: As a meeting host/attendee, I want to be able to check-in as soon as I show up in a meeting in order to keep track of all the participants' attendance and for personal record My condition of satisfaction that I am either provided with an option to manually check-in ~~or the system automatically checks-in (based on my current location) and update that info in the system.~~

Use case for check-in:

Title: Check-In

Actor: Admin and Users

Description: Check-In to mark attendance in a

meeting Direct Flow:

- Users and admins can click the check-in button to mark their attendance. If that person is actual at the meeting, that person will be marked as present.

Alternate Flow:

- If the person is not actually present at the meeting but clicks the check-in button, an error message will pop up.
- If the user does not have internet connection, then a different error message pops up.

Scheduling (from organiser perspective): As an organiser, I want to be able to schedule a meeting as per the convenience of all the attendees so that everyone is informed beforehand about the meeting. My conditions for satisfaction are being able to: toggle between taking a consensus on a list of locations for the meeting and enforcing a particular location for the meeting, and being able to bypass rules in case of emergency meetings.

Use Case for Scheduling:

Title: Scheduling from User

perspective Actor: Admin

Description: Deciding if consensus or single-point decision

Direct Flow:

- The admin gets to decide if they want the group to vote on the location of the meeting or if the admin gets to enforce the location of the meeting.

Alternate Flow:

- If the admin leaves this field blank, the application will throw an error message.

Todolist: As a user, I want to be able to add items to a todolist as per my convenience so that a clear-cut agenda is defined for the meeting. My conditions of satisfaction are being able to add a reasonable amount of text per item and having the option to cross off items that have been completed.

Use Case for Todolist:

Title: Todolist

Actor: Member of meeting

Description: Building a todo-list

Direct Flow:

- The user is a member of a meeting and is currently in the meetings tab. The user can now choose the appropriate meeting and click the show todo-list option.
- The user can add a new todo-list item or modify the status of an existing item.
- Upon clicking “Save Changes”, all the changes will be reflected in the database.

Alternate Flow:

- The user is not a member of any team and hence cannot modify or view any todo-list.
- The user tries to add a todo-list item without any text, which is not allowed.

Test Cases

User Creation:

A new user must be able to enter his details like first and last name, email address, password, user- id.

Equivalent classes:

Input	Valid Equivalence classes	Invalid Equivalence classes	Resources
First Name	[Not Empty]	[Empty]	test_UI_op.py
Last Name	[Not Empty]	[Empty]	test_UI_op.py
User Name	[Original] [Not Empty]	[Duplicate] [Empty]	test_database_op.py
Email ID	[Original] [Not Empty] [Valid Regex]	[Duplicate] [Empty] [Invalid Regex]	test_database_op.py test_UI_op.py
Password	[Not Empty] [Valid Regex]	[Empty] [Invalid Regex]	test_UI_op.py
Confirm Password	[Matches with Password]	[Doesn't match with Password]	test_UI_op.py

Login:

A user must be able to enter his username and password to login to his account.

Equivalent classes:

Input	Valid Equivalence classes	Invalid Equivalence classes	Resources
User Name	[Existing Username]	[Empty] [Username doesn't exist]	test_database_op.py test_UI_op.py
Password	[Matching password for the username]	[Empty] [Doesn't match with password for the username]	test_database_op.py test_UI_op.py

Group Creation:

A user must be able to create a group with two or more people.

Equivalent classes:

Input	Valid Equivalence classes	Invalid Equivalence classes	Resources
Group Name	[Not Empty]	[Empty]	test_UI_op.py
Group Type	[Formal] [Informal]	N/A	test_UI_op.py
List of users	[More than one User]	[less than 2 users]	test_UI_op.py

Meetings Creation:

An user must be able to create a meeting with valid values.

Equivalent classes:

Input	Valid Equivalence classes	Invalid Equivalence classes	Resources
GroupID	[Exists]	[Doesn’t Exist]	test_UI_op.py
Location	[Not Empty]	[Empty]	test_UI_op.py

Time	[Not Empty]	[Empty]	test_UI_op.py
Meeting Name	[Not Empty	[Empty]	test_UI_op.py

To-Do List Creation:

A user should be able to create a to-do list.

Input	Valid Equivalence classes	Invalid Equivalence classes	Resources
meetingID	[Exists]	[Doesn't Exist]	test_UI_op.py
List of to-do list items	[Not Empty] [Empty]		test_UI_op.py

Tests:

Coverage tool used: unittest,

coverage Test Suite location: tests/

Link to Github Repository: https://github.com/ASE-Int-Elligence/Meeting_Scheduler