1. Create 1 document per team that describes how at least three features within your finished product will be tested.
2. The test plans should include specific test cases (user acceptance test cases) that describe the data and the user activity that will be executed in order to verify proper functionality of the feature.
3. The test plans should include a description of the test data that will be used to test the feature.
4. The test plans should include a description of the test environment that will be used to test the feature.
5. The test plans should include a description of the test results that will be used to test the feature.
6. The test plan should include information about the user acceptance testers. Please note that the deployed application can only be tested on the CU Boulder campus. So make sure to select your testers accordingly.

Using JMockit

What needs to be tested

- **Register**
- **Login**
- Trail Search
- Friend Search
- Edit Profile
- Create Profile
- Friend Request
- **Messaging**
- Individual Trail Page
- Individual Profile Page

- **Acceptance Criteria Definition 1**: "Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder."

Test Environment: Development Environment

Register

User instructions

- Test case 1: User inputs a valid new username, password is above 5 chars (then submits)

- {

“username”: dokr2508
“password”: poop420LOL

}
User expectations: The user should expect that their username + hashed password is stored in the users table. They should be able to login using the login page with this new password. The database should display a row for the new username in addition to a **hashed** password.

- Test case 2: User inputs a new username, a password that is less than 5 characters (then submits)
  User expectations: The user should expect to get an error message and to remain on the registration page. They should NOT be able to login with their input on the login page.
  {
  “username”: “dokr2508”
  “password”: “lol1”
  }
- Test case 3: User inputs a username that already exists, password long enough (then submits)
  {
  “username”: “dokr2510”
  “password”: “ballsToTheWalls”
  }
  User expectations: The user should get a warning message that this username exists already and be recommended to redirect themselves to the login page.
- Test case 4: User leaves one of the fields empty and submits.
  {
  “username”: “”
  “password”: “beastmode”
  }
  User expectations: The user should get a warning that one of the fields in invalid. They should NOT be able to login with their input.

Login

Test Case 1: Successful login:
User Instructions:

- User will pass a username and password of an account already created into the login page and submit the information
  - Ex: data{
    “username”:”climberman”
    “Password”:”password123”
    }

Where a profile with username climberman and password password123 exists in our database.

- User will make sure the path properly executes and they are properly linked to their account

Test Environment:

- The landing page for our website which is the login page
- It will tested in the development environment

Test Results:

- If properly executed, the user should be taken to their profile page

Testers:

- Ideally the tester would be climbers, but anyone on campus could test this

<u>Test Case 2: Incorrect password</u>

User Instructions:

- User will pass the username of an account that has already been created but will pass a password that is not linked with the account in the login page

  Ex: data{

  “username”:”climberman”

  “Password”:”passwordWRONG”

  }

  Where a profile with username climberman and password password123 exists in our database.
- User will make sure they stay on the login page and an error message appears

Test Environment:

- The landing page for our website which is the login page
- It will tested in the development environment

Test Results:

- If properly executed, the user should stay on the login page and an error should appear

Testers:

- Ideally the tester would be climbers, but anyone on campus could test this

<u>Test Case 3: User not found</u>

User Instructions:

- User will pass a username that is not in the database and a random password into the login page

  Ex: data{

  “username”:”WRONGclimberman”

  “Password”:”passwordWRONG”

  }

  Where a profile with username climberman and password password123 exists in our database.

- User will make sure they stay on the login page and an error message telling them to create an account appears

Test Environment:

- The landing page for our website which is the login page
- It will tested in the development environment

Test Results:

- If properly executed, the user should stay on the login page, and an error message recommending them to create an account should appear

Testers:

- Ideally the tester would be climbers, but anyone on campus could test this

Messaging

- Test Case 1: Successful Message:
  - Message to be sent 'Hi!'
  - Existing user: 'johndoe'
  - The user will send a message containing content to a currently existing user. A success is expected and the message will be uploaded to the database. The database will be checked before giving the success.
- Test Case 2: Empty Message
  - Message to be sent: ''
  - Existing user: 'johndoe'
  - User will send an empty message to a currently existing user. The message will throw an error, as empty messages are useless, and the message will not be added to the database.
- Test Case 3: Invalid Receiving User
  - Message to be sent: 'Hi!'
  - Non-existent user: 'johndoe1'
  - The user will send a message containing content to a non-existent user. This will throw an error and the message will not be added to the database.

User acceptance Testers:

- **CU Students who rock climb**