SafeHouse

System and Unit Test Report

Team Speakeasy

System Tests Scenarios:

Sprint 1: Not Applicable/No testable code produced

Sprint 2:

User Stories:

As a user, I want the ability to register for an account.
As a user, I want the ability to log in to my account/stay logged in to my account.
As a user, I want the ability to log out of my account.

Test Procedure:

1. Start the SafeHouse App, select Register for an account
    a. Name = <testUser1>
    b. Password = <testUser1>
    c. Email = <testUser1@gmail.com>
    d. Press register key
    e. User should be logged into account, and able to access “/desk” page
2. Select log out
    a. User should be redirected to login page
    b. Change URL to “/desk” page
    c. User should be redirected to login page
3. Select Login option
    a. Email = <testUser1@gmail.com>.
    b. Password = <testUser1>
    c. User should be logged into account

Sprint 3:

User Stories:

As a user, I want to send letters to other users
As a user, I want to review my sent letters
As a user, I want to receive letters and review my received letters
As a user, I want to respond to letters sent to me
As a user, I want to delete/archive letters from my inbox

Test Procedure:

1. Start the SafeHouse App, select Register for an account
    a. Name = <testUser1>
    b. Password = <testUser1>
    c. Email = <testUser1@gmail.com>
    d. Press register button
    e. User should be logged into account, and able to access “/desk” page
2. Select Send a letter option
    a. User should be redirected to “/letter” page
    b. Letter = <this is a letter from testUser1>
    c. Press send button
    d. User should get a confirmation that the letter has been sent
    e. Click ok button
    f. User should be redirected to “/desk” page
3. Select log out
    a. User should be redirected to login page
4. Select Register for an account
    a. Name = <testUser2>
    b. Password = <testUser2>
    c. Email = <testUser2@gmail.com>
    d. Press register button
    e. User should be logged into account, and able to access “/desk” page
5. Select Respond to Letter option
    a. User should see a list of letters to respond to, including the previous letter sent by testUser1
    b. Click respond to letter button
    c. User should be redirected to “/response” page
    d. Response = <this is a response by testUser2>
    e. User should get a confirmation that the response has been sent.
    f. Click ok button
    g. User should be redirected to “/desk” page
6. Select log out
    a. User should be redirected to login page
7. Select Login option
    a. Email = <testUser1@gmail.com>.
    b. Password = <testUser1>
    c. User should be logged into account
8. Select View inbox page
    a. User should see the previous sent letter and it’s response from testUser2

Sprint 4: Not Applicable/No testable code produced

Unit Tests:

Sarah

Ran user navigation testing on frontend.
Navigation Test
Required inputs: UserValidated (boolean)
Tests:

1. UserValidated == True
   -> navigating to “/login” redirects to “/desk”
   -> navigating to “/desk”, “/request”, or “/inbox” is allowed
2. UserValidated == False
   -> navigating to “/login” is allowed
   -> navigating to “/desk”, “/request”, or “/inbox” redirects to “/login”

Dhanush

Ran integration testing (from above).

Harmanjot:

Tested backend middleware for user verification and valid info, HTTP get requests for “getinboxresponses” and “getinboxletters”.

Authorization middleware
Required inputs: JWT Token storing user id as encrypted payload
Expected outputs: 403(Not authorized), nothing
Tests:

1. JWT Token is validated
   -> nothing is returned, user ID is stored and passed to next request
2. JWT token is malformed
   -> status code 403, error message “Not Authorized”
3. JWT token is missing
   -> status code 403, error message “Not Authorized”

ValidInfo middleware
Required Inputs: Email, Password, next request,Username (dependent on next request)
Expected outputs: 401(Missing credential), 401(invalid email)
Tests:

1. Next request = register, Email is invalid, everything else ok
   -> status code 401, error message “Invalid email”
2. Next request = register, password or email or username is missing
   -> status code 401, error message “Missing credential”
3. Next request = register, all credentials exist and email is valid
   -> nothing is returned, credentials passed to next request
4. Next request = login, Email is invalid, everything else ok
   -> status code 401, error message “Invalid email”

5. Next request = login, password or email is missing
   -> status code 401, error message “Missing credential”
6. Next request = login, password and email exist and email is valid
   -> nothing is returned, credentials passed to next request

* email invalid meaning email not in email form xxx@x.xxx

GetInboxLetters GET request
Required Inputs: User ID
Expected Outputs: 200(success), 400(no letters), 403(invalid credentials)
Tests:

1. User ID valid, user has sent letters
   -> status code 200
   -> all letters sent by user are returned
2. User ID is valid, user has not sent letters
   -> status code 400, error message “no letters sent”
3. User ID is not valid
   -> Status code 403, error message “Not authorized”

GetInboxRequests GET request
Required Inputs: User ID, LetterID
Expected Outputs: 200(success), 400(no responses), 403(invalid credentials)
Tests:

4. User ID valid, Letter has responses
   -> status code 200
   -> all responses for specific letter are returned
5. User ID is valid, Letter has no responses
   -> status code 400, error message “no responses recieved”
6. User ID is not valid
   -> Status code 403, error message “Not authorized”

Eric
Tested login, register, send letter and send response HTTP post requests and request letters HTTP get request.

Register POST request
required inputs: username, email, password
Expected output: status code 200(success), 401 (invalid credentials)
Tests:

1. All valid inputs
   -> status code 200
   -> select statement in Postgres server shows user inserted
2. Username, password okay, email already in user table
   -> status code 401, error message “user already exists”
   -> select statement in Postgres server shows no user inserted

3. Email not an email
   -> status code 401, error message “not a valid email”
   -> select statement in Postgres server shows no user inserted
4. Any missing inputs
   -> status code 401, error message “invalid credentials”
   -> select statement in Postgres server shows no user inserted

Login POST request
required inputs: email, password
Expected output: status code 200(success), 401 (invalid credentials)
Tests:

1. All valid inputs (user in user table)
   -> status code 200
   -> valid JWT token returned
2. email, password don’t match any user in user table
   -> status code 401, error message “invalid credentials”
3. Email not an email
   -> status code 401, error message “not a valid email”
4. Any missing inputs
   -> status code 401, error message “invalid credentials”

Send Letter POST Request
Expected inputs: User ID, Letter
Expected output: status code 200(success), 403 (not authorized)
Tests:

1. All Valid Inputs (user id is validated)
   -> status code 200
   -> select statement in Postgres server shows letter inserted
2. User ID is malformed
   -> status code 403, error message “Not Authorized”
   -> select statement in Postgres server shows no letter inserted

Send Response POST Request
Expected inputs: User ID, Letter ID, Response
Expected output: status code 200(success), 403 (not authorized)
Tests:

1. All Valid Inputs (user id is validated)
   -> status code 200
   -> select statement for response table shows response inserted
   -> select statement for letter table shows original letter’s response value incremented
2. User ID is malformed
   -> status code 403, error message “Not Authorized”
   -> select statement in Postgres server shows no response inserted, no letter updated

Request Letters GET Request
Expected Inputs: User ID
Expected output: status code 200(success), 400 (no letters) ,403 (not authorized)
Tests:

1. User ID is valid, database has valid letters to send to user
   -> status code 200
   -> List of letters returned
      Letters are not from user
      Letters haven't been responded to by user
2. User ID is valid, database has no valid letters to send to user
   -> status code 400, error message "No Valid Letters"
3. User ID is malformed
   -> status code 403, error message "Not Authorized"