**CS673 Software Engineering** 

**Team 5 - GearOnTheGo**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Lazaro Perez | QA Leader | *Lazaro Perez* | 09/23/2023 |
| Ahnaf Tajwar | Security Leader | Ahnaf Tajwar | 09/23/2023 |
| Shajee Ur Rehman | Configuration Leader | Shajee ur Rehman | 09/23/2023 |
| Jian Song | Architecture and Design | *Jian Song* | 09/23/2023 |
| Samantha Mathis | Team Leader | *Samantha Mathis* | 09/23/2023 |
| Saahil Vashishta | Requirement Leader | Saahil Vashishta | 09/23/2023 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 0.1 | Saahil Vashishta | 9/23/2023 | 1. [First Draft - Testing Summary](#_heading=h.gjdgxs) 2. [Manual Testing - User registration](#_heading=h.30j0zll) plus template creation 3. [Added Google sign-in and sign up to user registration and login](#_heading=h.30j0zll) |
| 0.2 | Lazaro Perez | 09/24/2023 | 1. [Manual Testing - Search Functionality](#_heading=h.30j0zll) |
| 0.3 | Jian Song | 09/24/2023 | 1. [Manual Testing - Make Reservation functionality](#_heading=h.30j0zll) |
| 0.4 | Samantha Mathis | 09/25/2023 | 1. [Manual Testing](#_heading=h.30j0zll)  - Hosting Equipment |
| 0.5 | Samantha Mathis | 09/25/2023 | 1. [Testing Metrics](#_heading=h.3znysh7) |
| 0.6 | Lazaro Perez | 09/25/2023 | 1.Testing Metrics - |
| 0.7 | Shajee Ur Rehman | 09/25/2023 | 1. Manual Testing - Checkout Process |
| 1.0 | Saahil Vashishta | 10/07/2023 | Document Revision:   1. Addition of [Automated Testing](#_heading=h.1fob9te) 2. Adding Automated tests for Google Registration |
| 1.1 | Ahnaf Tajwar | 10/08/2023 | 1. Adding Automated tests for User Login using Selenium |
| 1.2 | Saahil Vashishta | 10/09/2023 | 1. Addition of Functional Tests for Google registration    1. Google register    2. Google Login    3. Google update 2. Addition of unit test for registration - database model |
| 1.3 | Jian Song | 10/09/2023 | Document Revision:   1. Addition of Reservation tests 2. Testing Metrics |
| 1.4 | Samantha Mathis | 10/09/2023 | 1. Manual Testing Report 2. Automated Testing Report |
| 1.5 | Lazaro | 10/10/2023 | 1. Automated Testing for search. Section 4 |
| 2.0 | Samantha Mathis | 10/10/2023 | Iteration 2 version |
| 2.1 | Lazaro | 10/15/2023 | Testing Metrics |
| 3.0 | Samantha Mathis | 10/15/2023 | Iteration 3 version |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Summary

In this section, you will summarize what was tested, who is involved in testing, testing techniques used, and testing result. You may have the following tests

* + Unit Testing
  + Integration testing
  + System Testing
  + Acceptance Testing
  + Regression Testing

As part of Iteration 1, the team didn’t conduct automated testing and primarily focussed on manual testing. Each of the features were tested based on the acceptance criteria set for each user story and the functionality was tested against the requirements and acceptance criteria before being merged with the main branch. As part of future iterations, the team will utilize selenium to conduct automated tests. Unit testing will also be used as part of the next iterations. This iteration will focus on:

**System testing**: Requirement testing: Each user story will link to a function added on github and shall be tested at a high level, against the requirements.

**Acceptance testing:** Each function will be tested against the acceptance criteria set for the corresponding user story.

**Robustness testing:** Wherever inputs are required, a variety of inputs will be used to make sure the code doesn’t break.

# Manual Testing Report

In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.

Here is a sample template that can be used for each test case. For system tests or acceptance tests, you may also include some screenshots.

* Test case ID, name
* New or old:
* Test items: (what do you test )
* Test priority (high/medium/low)
* Dependencies (to other test case/requirement if any):
* Preconditions: (if any)
* input data:
* Test steps:
* Postconditions:
* Expected output:
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:

(You can use an additional spreadsheet for this section as well)

**User Registration:**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0001 | Save user email and password to database | Database in ElephantSQL connected | Email, Password | Go to Register page, Enter email, Enter password, Click Register | Entered user email and password is saved to database | User email and password saved to database | Pass |
| 0002 | Password is saved as hash | Database in ElephantSQL connected, Password can be saved | Email, password | Go to Register page, Enter email, Enter password, Click Register | Entered user email and password. Password is saved as hash | Pass is saved as hash | Pass |
| 0003 | Successful Login requirement and acceptance criteria test | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Google email, (password is not applicable since google doesn’t provide access) | 1. Go to IDE (Vs code) 2. Run npm start in terminal 3. Click on sign-up with google 4. Enter correct credentials for Google 5. Submit Form | The email (if new) shall be entered into the database along with a string that specifies: “Google account, password not available” | Email was added to User table in database | Pass |
| 0004 | Unsuccessful Login requirement and acceptance criteria test | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Incorrect Google email, (password is not applicable since google doesn’t provide access) | 1. Go to IDE (Vs code) 2. Run npm start in terminal 3. Click on sign-up with google 4. Enter incorrect credentials for Google 5. Submit Form | Google should show an error | Prompt from the Oauth API: “Couldn’t find your google account” | Pass |
| 0005 | Successful Log in requirement and acceptance criteria test - Database duplication protection | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Google email, (password is not applicable since google doesn’t provide access) | 1. Go to IDE (Vs code) 2. Run npm start in terminal 3. Click on sign-up with google 4. Enter correct credentials for Google (used email) 5. Submit Form | The email being used twice shouldn;t added to database again | Successful login, non duplication verified on ELephant SQl | Pass |
| 0006 | Successful logout to refresh session | Logged in using API | Click on button | 1. Click on the logout button | Should have gone back to log in screen | Verified, the program went back to login screen | Pass |
| 0007 |  |  |  |  |  |  |  |

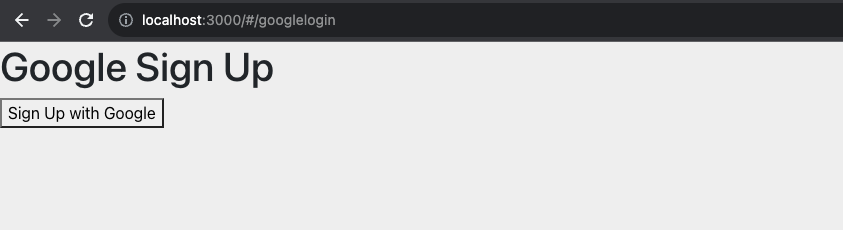


Fig1: Google Sign-up on local host

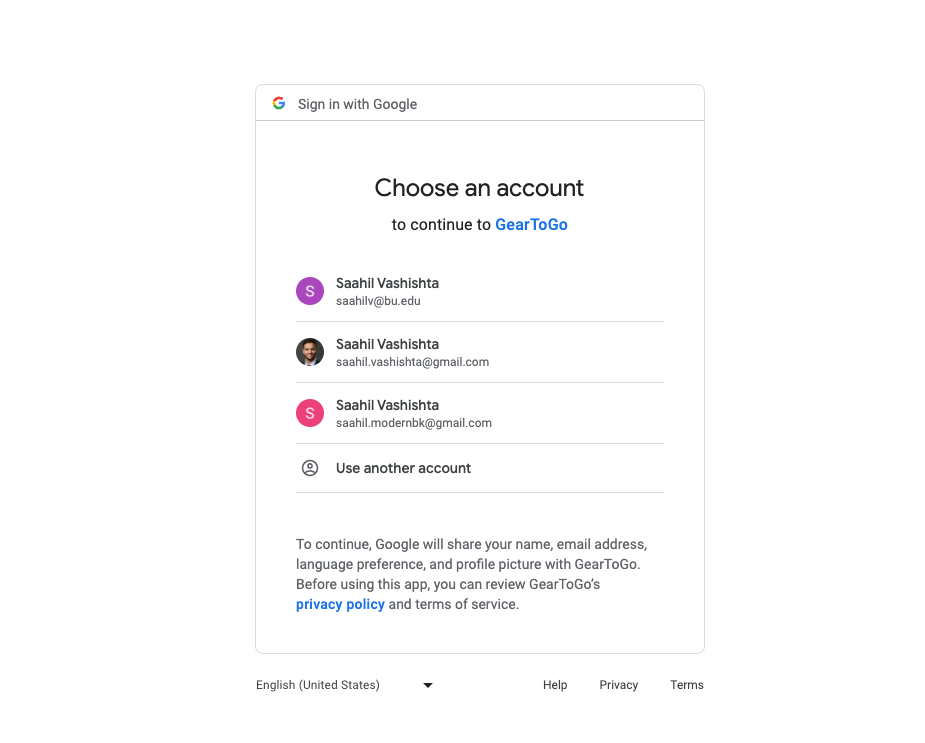
****

Fig2: Gmail account options

****

Fig3: Successful login with session information and logout option

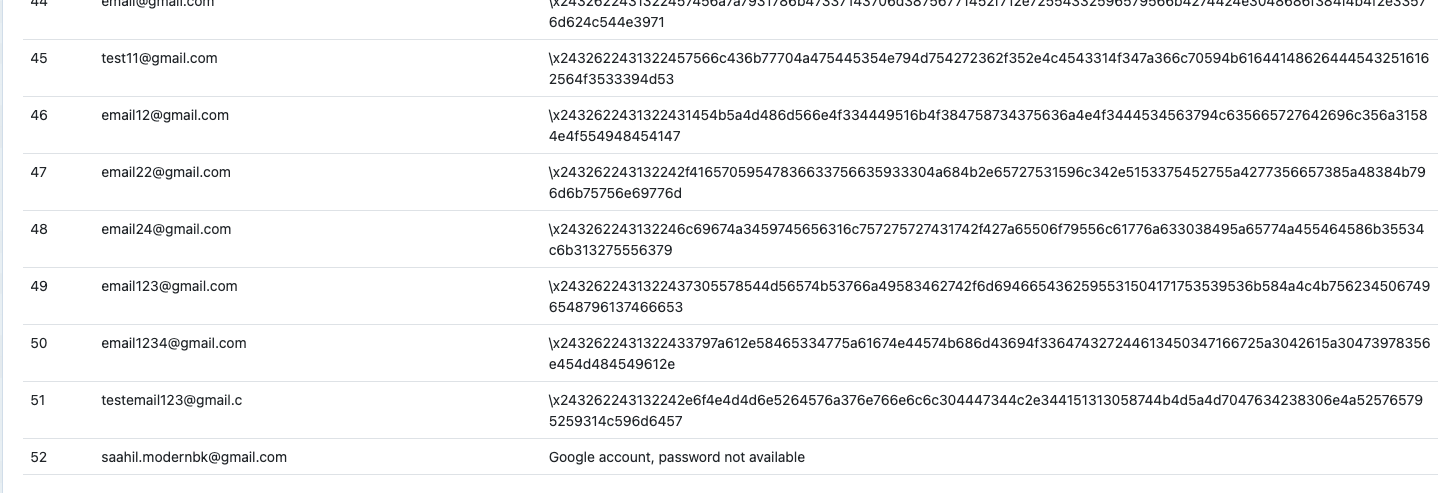


Fig4: Entry of google email chosen to database on Elephant SQL

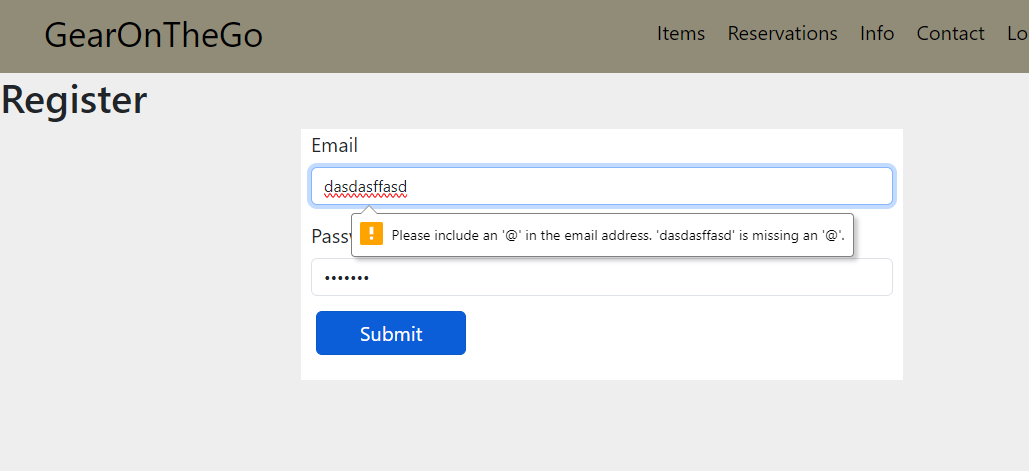


Fig6: Register page on React site. Incorrect email format entered warning

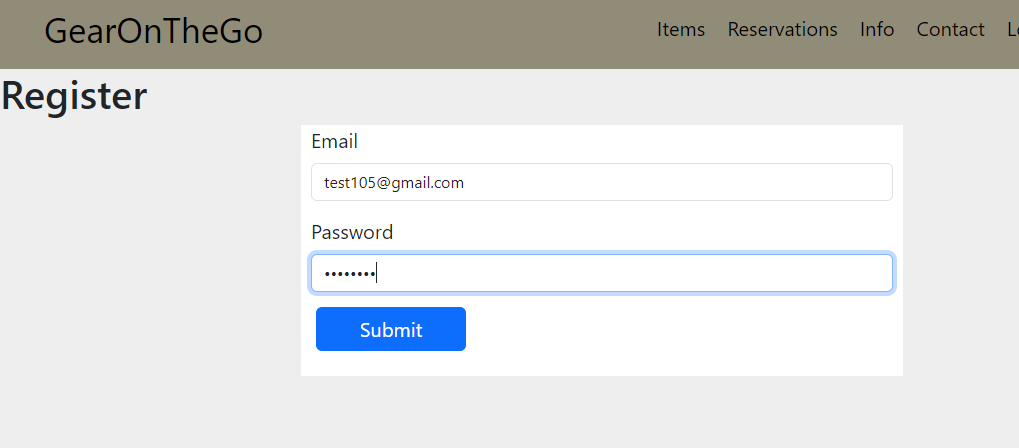


Fig7: Entering appropriate credentials

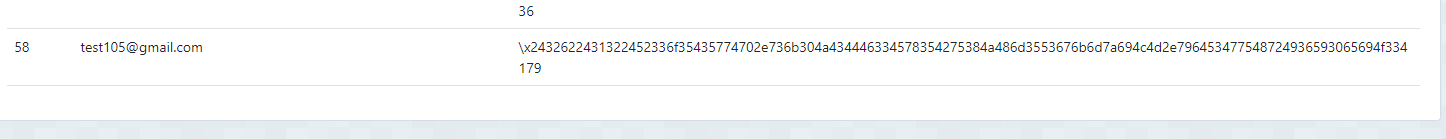


Fig8: User credentials stored in database with hashed password

**User Login:**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0008 | Login with existing user credentials | User already registered with email and password.  Database connection | Email, password | Go to the Login page and enter email and password. Click Login | User is logged in and redirected to dashboard | User is logged in and redirected to dashboard | Pass |
| 0009 | Login with wrong credentials | Database connection | Email, password | Go to the Login page and enter the wrong combination of email and password. Click Login | User is not logged in | User is not logged in | Pass |
| 0010 | Successful login with Google | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Email, password | 1. Go to IDE (Vs code) 2. Run ndp start in terminal 3. Click on sign-up with google 4. Enter correct credentials for Google 5. Submit Form | User logs in | User logs in | Pass |
| 0011 | Incorrect password on Google login | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Email, password | 1. Go to IDE (Vs code) 2. Run ndp start in terminal 3. Click on sign-up with google 4. Enter incorrect credentials for Google 5. Submit Form | Login failure | Login failure | Pass |
| 0012 | Incorrect email in Google Login | Database in ElephantSQL connected, Oath API connecting to Google sign-up, Google developer portal set up correctly, Database tables set up correctly | Email, password | 1. Go to IDE (Vs code) 2. Run ndp start in terminal 3. Click on sign-up with google 4. Enter incorrect credentials for Google 5. Submit Form | Login failure | Login failure | Pass |

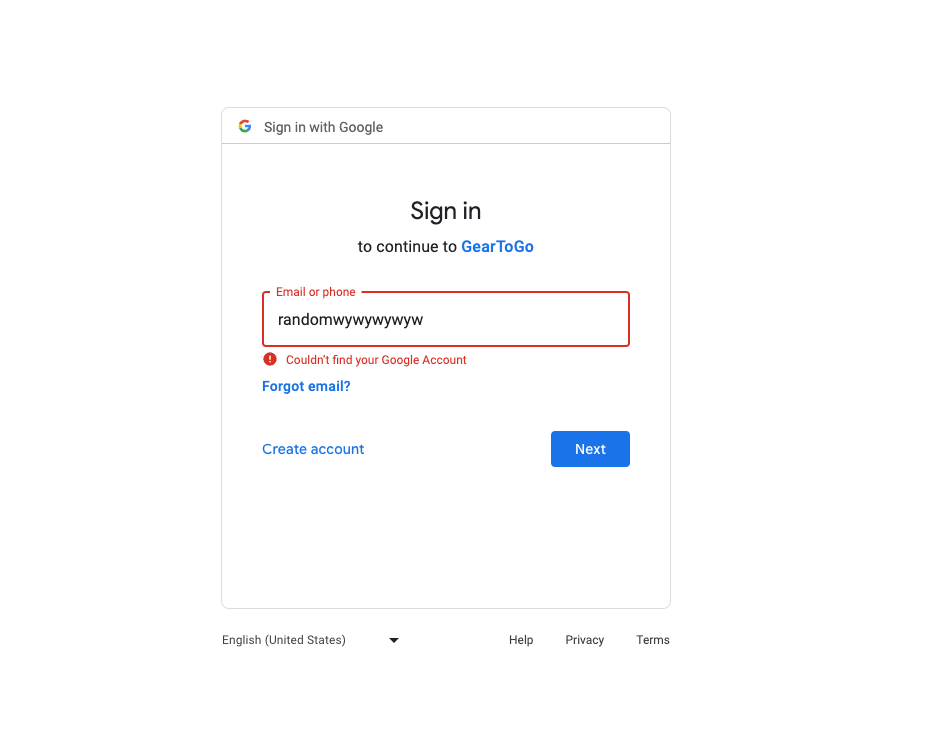


Fig8: Incorrect email error (Google sign-in)

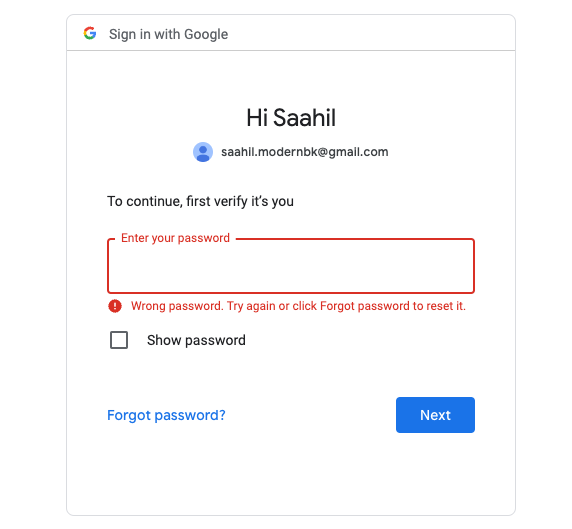


Fig9: Incorrect Password sign-in Google

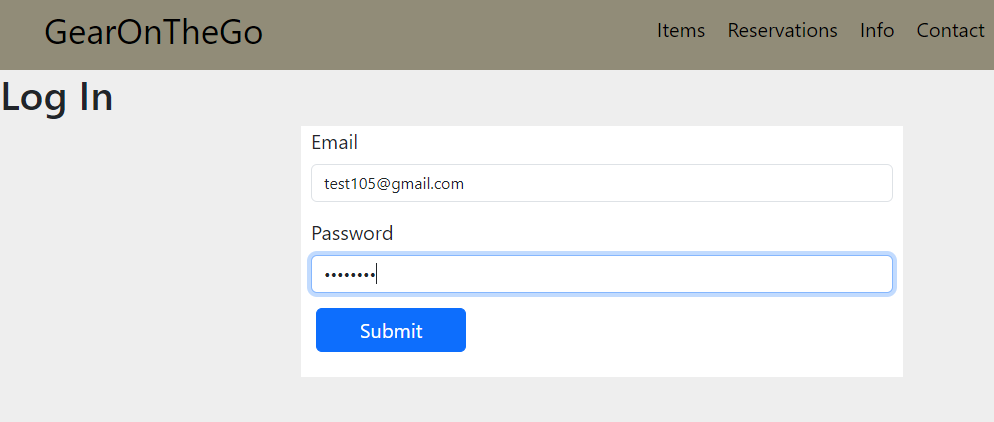


Fig10: Logging in with valid credentials

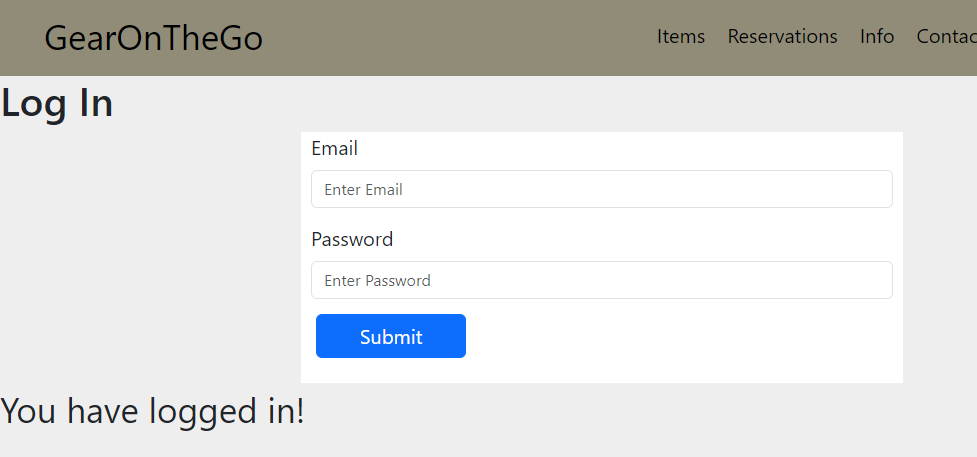
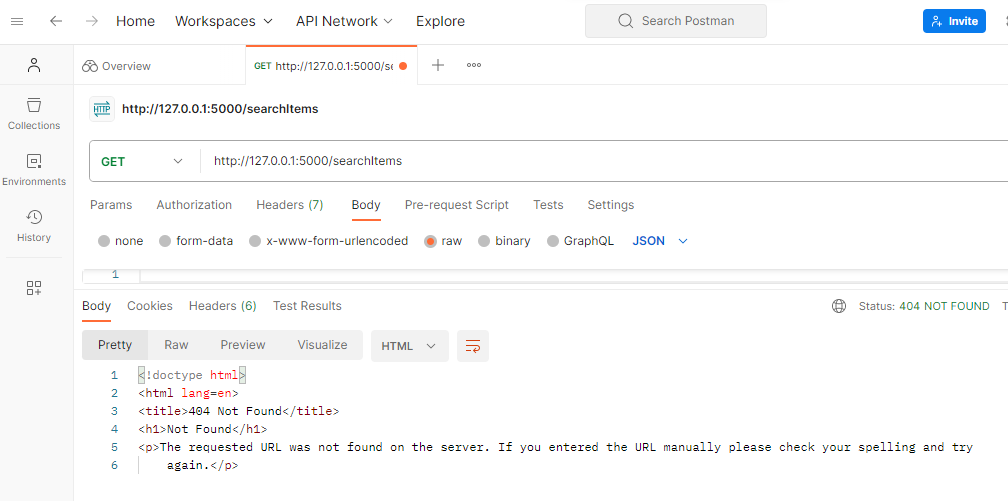
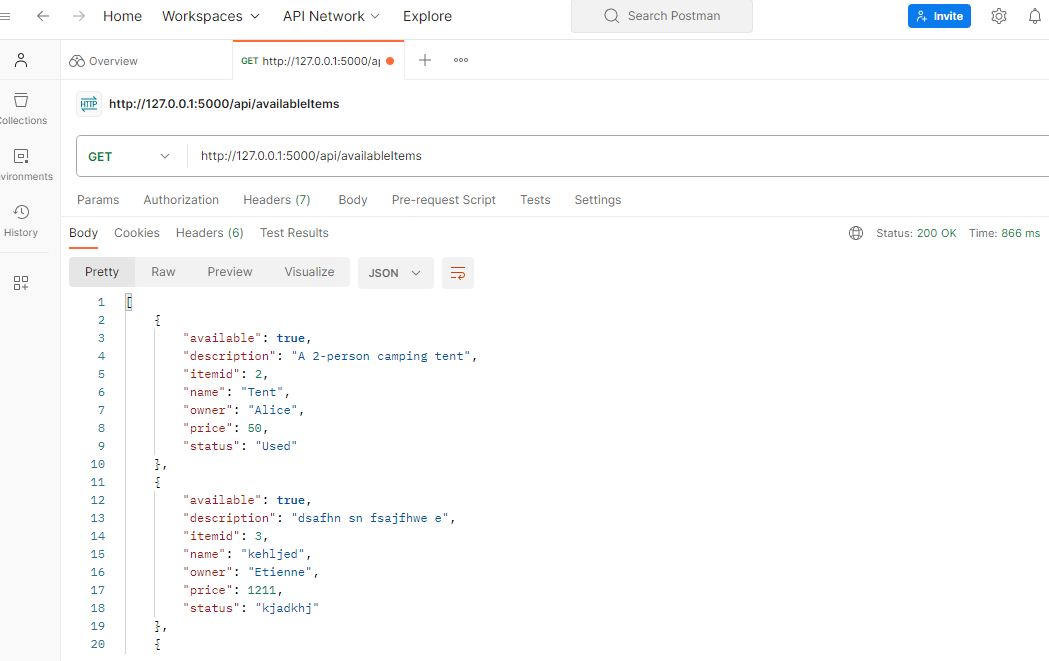


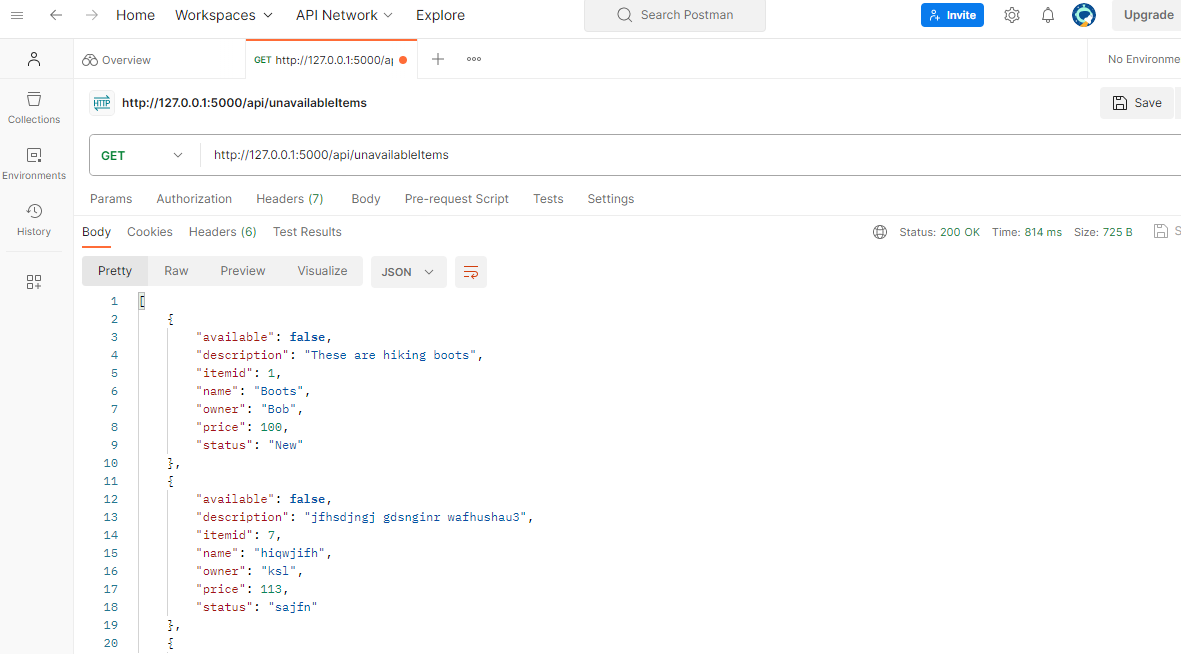
Fig11: Displays logged in message

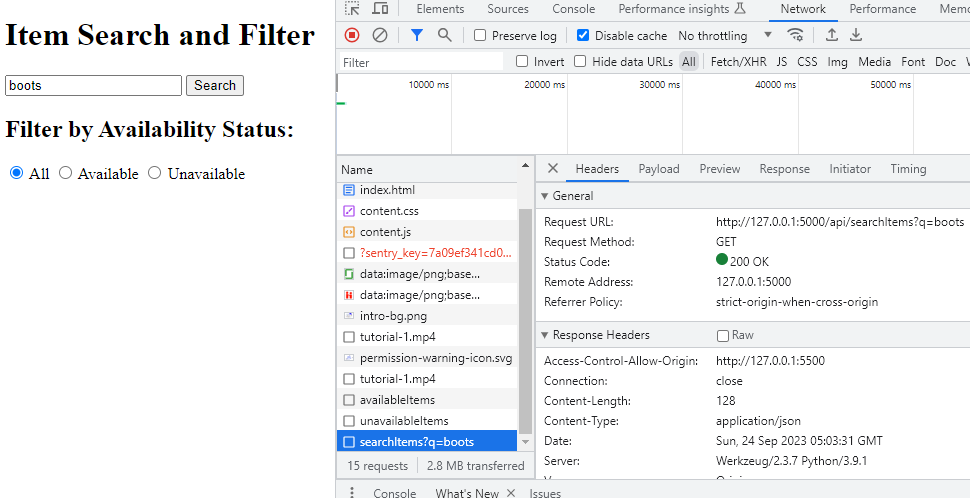
**Search Items Availability and Unavailability:**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0013 | Search Items API Testing in Postman | Database in ElephantSQL attached with Pyscopg2 | Items that are listed in the Item tables | Run Flaks in the CLI, Apply the port and test the endpoint on Postman | Retrieve JSON Data from the endpoint | Retrieved Error from the API GET request | Fail |
| 0014 | Search Items API Testing in Postman | Database in ElephantSQL attached with Pyscopg2 | Items that are listed in the Item tables | Run Flaks in the CLI, Apply the port and test the endpoint on Postman | Retrieve JSON Data from the endpoint | Retrieved Data from the Endpoint | Pass |
| 0015 | Available Items API Testing in Postman | Database in ElephantSQL attached with Pyscopg2, | Items that are available in the items table | Run Flaks in the CLI, Apply the port and test the endpoint on Postman | Retrieve JSON Data from the endpoint | Retrieved Data from the Endpoint | Pass |
| 0016 | Unavailable Items In Postma | Database in ElephantSQL attached with Pyscopg2, | Items that are unavailable in the items table | Run Flaks in the CLI, Apply the port and test the endpoint on Postman | Retrieve JSON Data from the endpoint | Retrieve JSON Data from the endpoint | Pass |
| 0017 | Testing Connection to the API through the Front End by analyzing connection through the Network Web Dev Tools | Database in ElephantSQL attached with Pyscopg2, HTML with Javascript file to test | All items in the table database | Run Flaks in the CLI, Apply the port and then test the onClick handlers from the HTML Code | Make connection to the API and check to make sure I received the 200 OK status | Made connection to the API and check to make sure I received the 200 OK status | Pass |
| 0018 | Testing the rendering of items on the UI in the frontend | Database in ElephantSQL attached with Pyscopg2, React Applicaiton and NPM Start | Items in the table | Run Flaks in the CLI, run NPM Start in another CLI and open Live Server and test to make sure the functionality works | Items to be rendered on the  screen | Items Rendered on the screen | Pass |

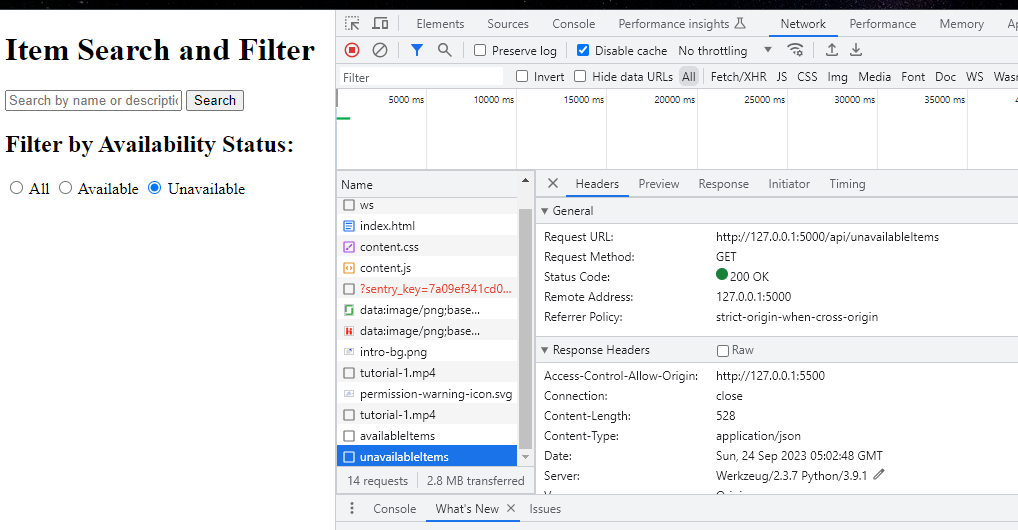
****

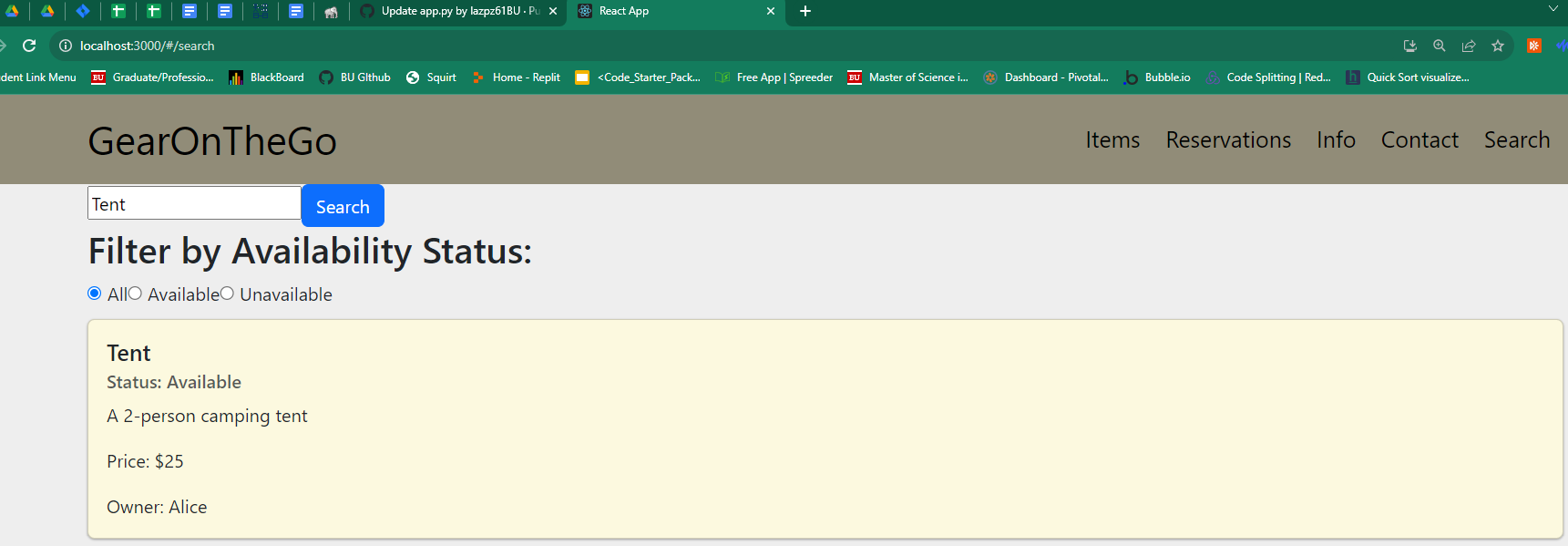
****

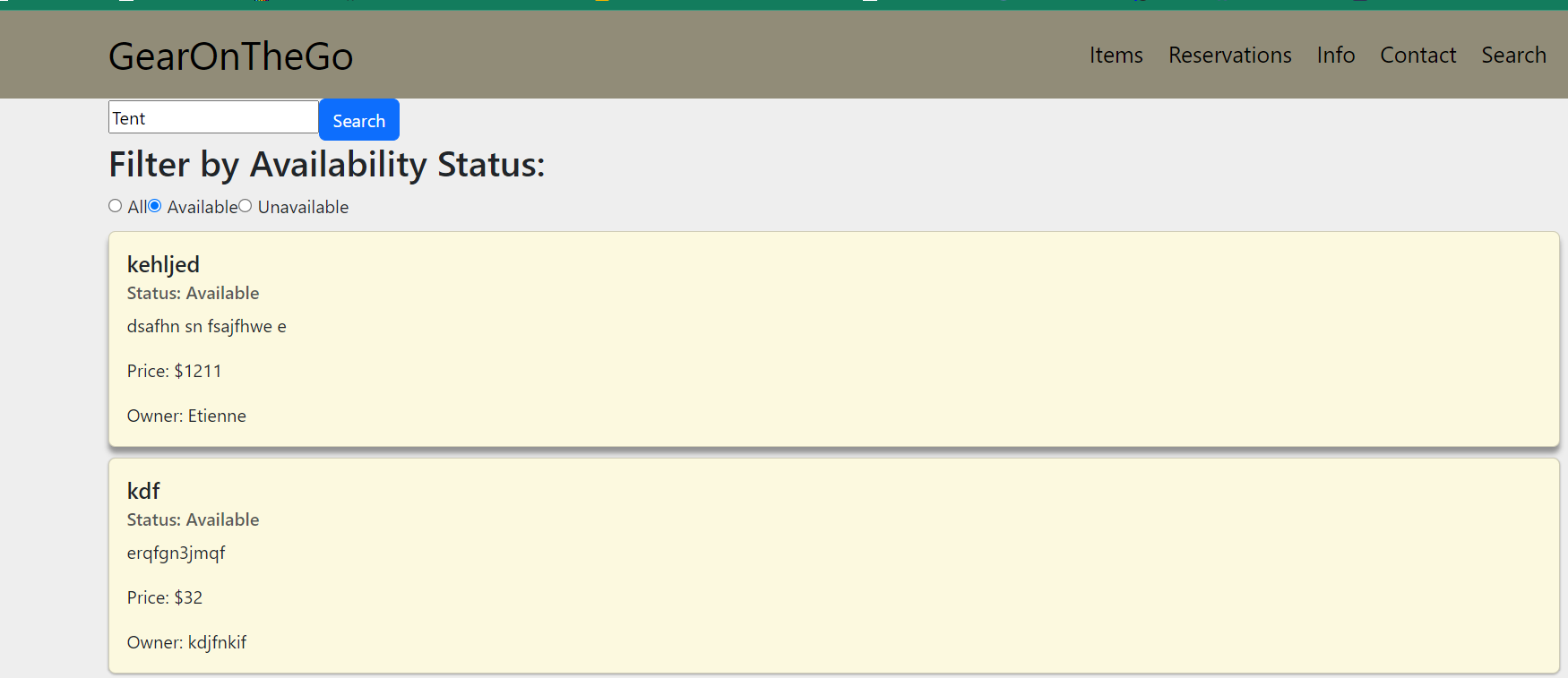
****

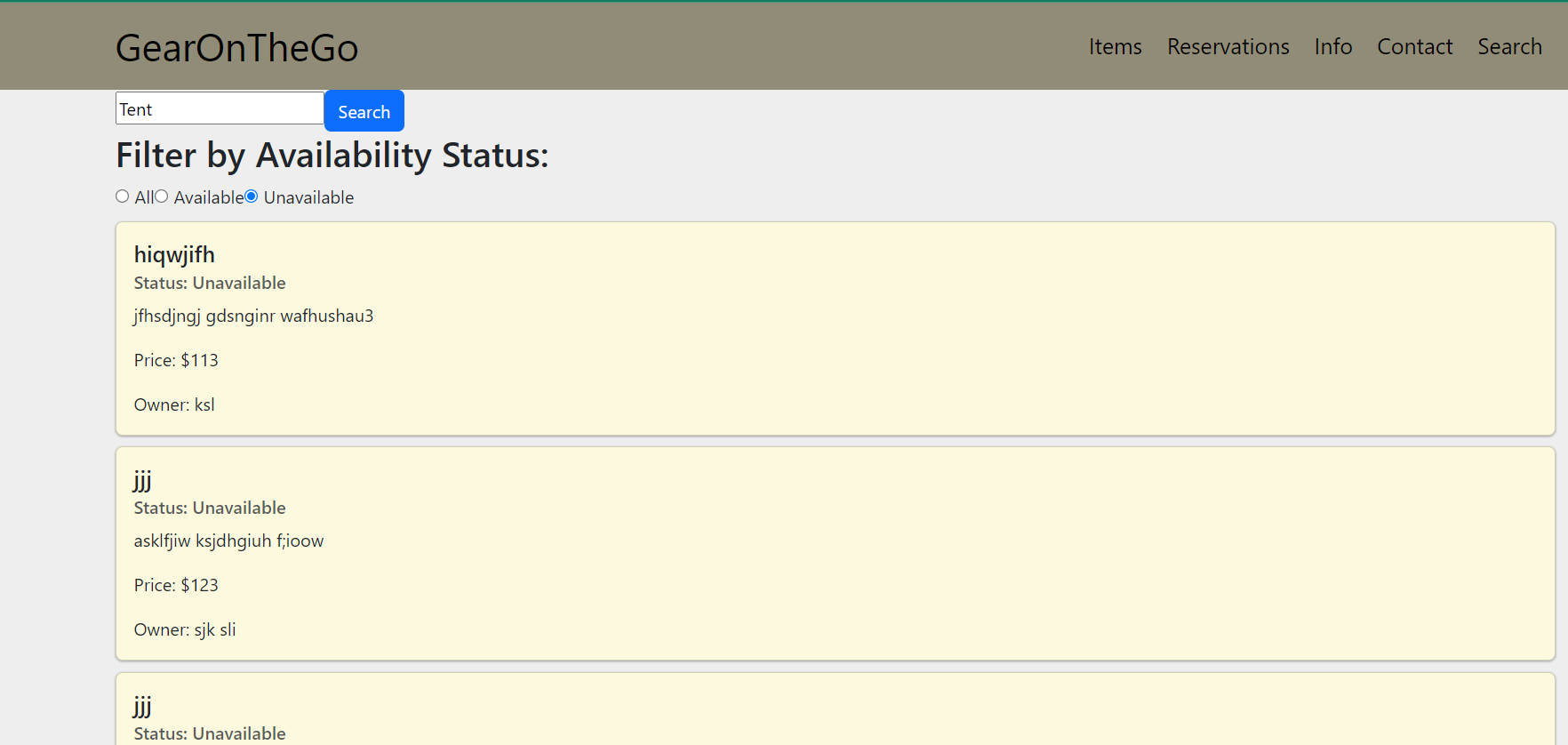
****

****

****

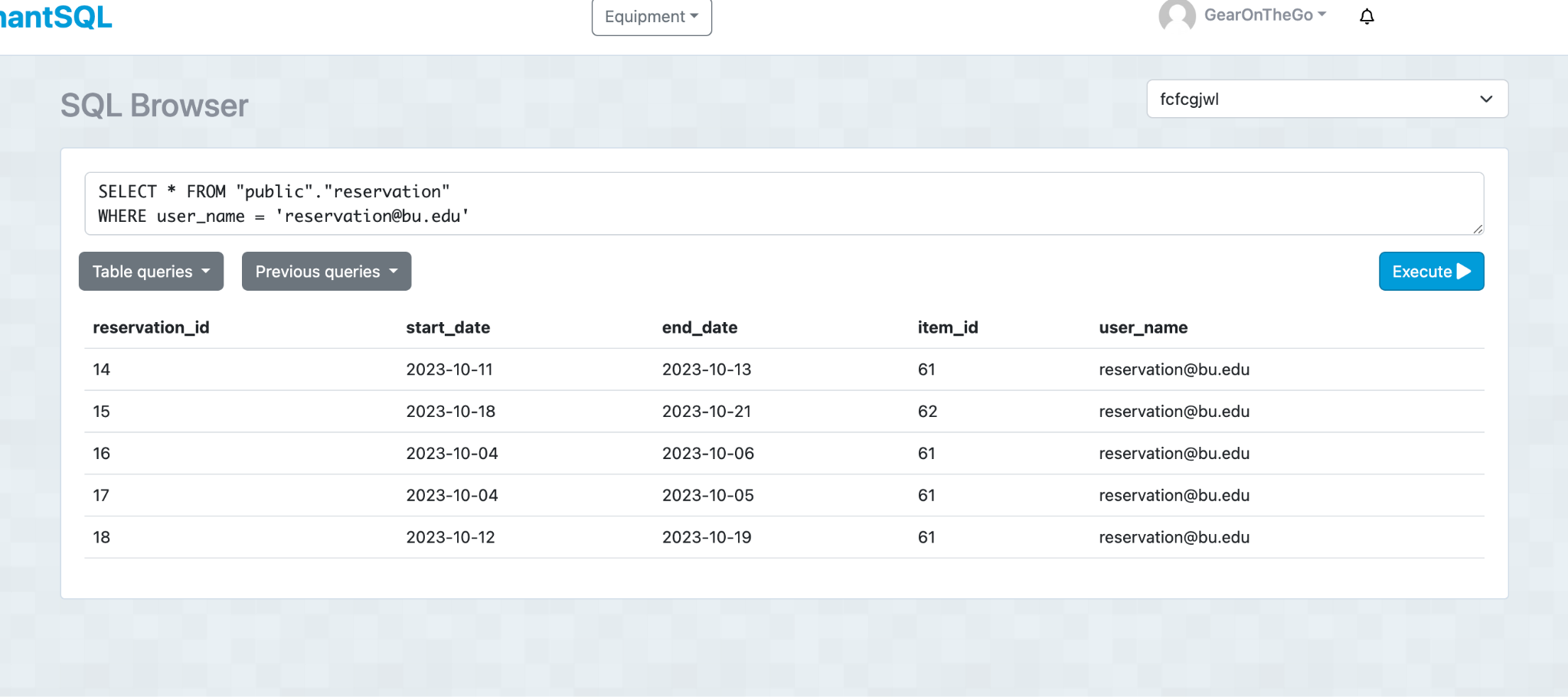
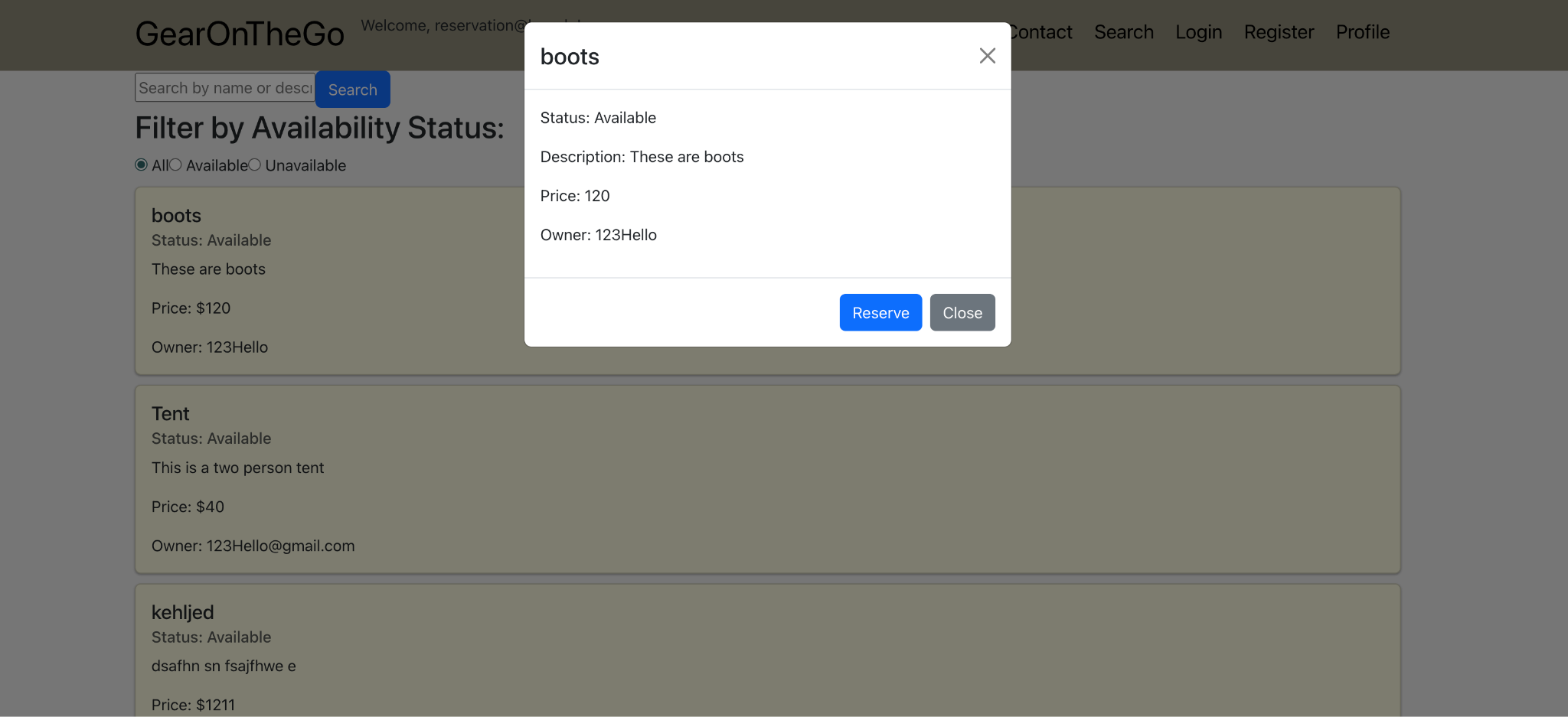
****

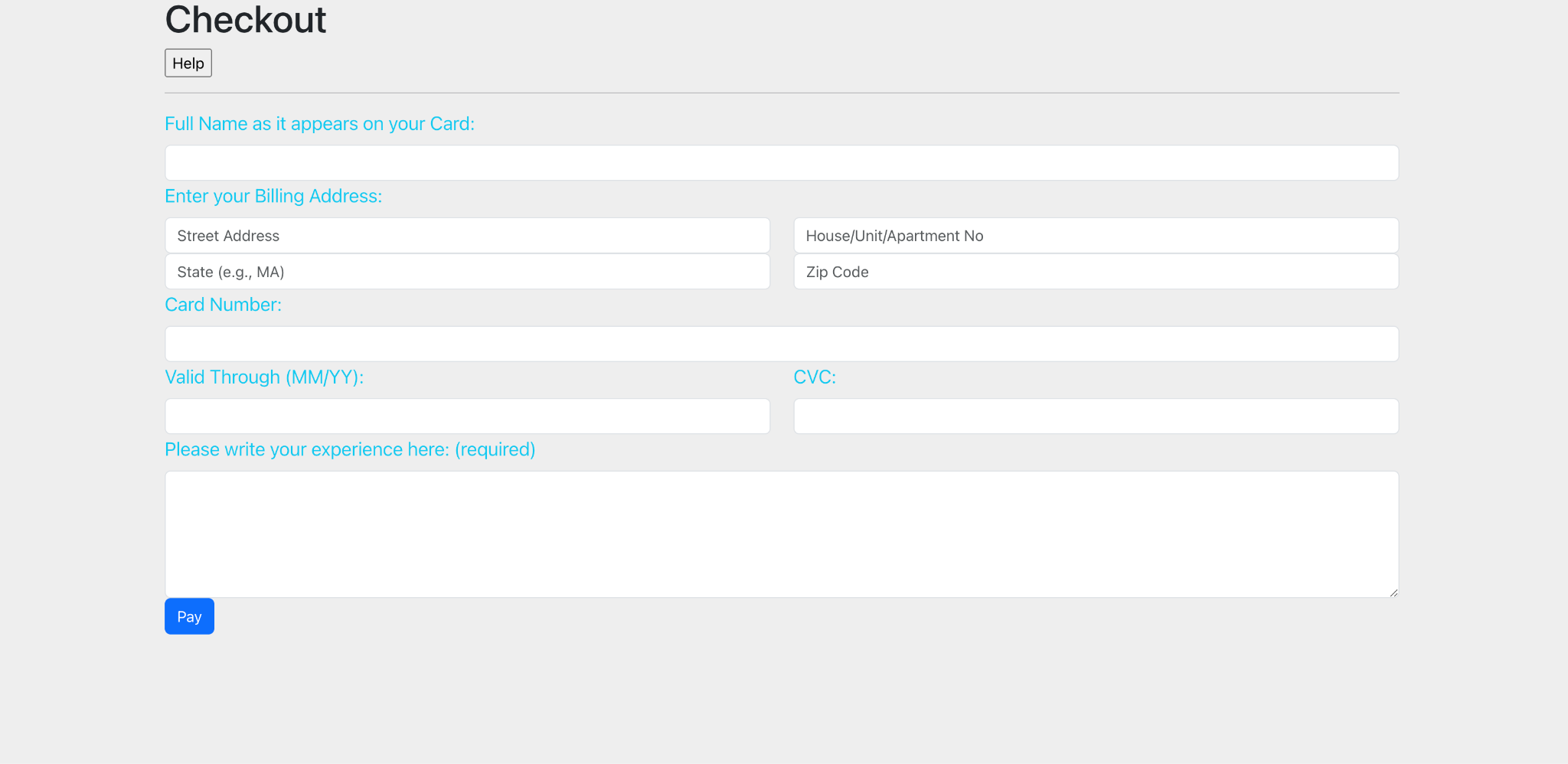
****

****

**Make Reservation Process**

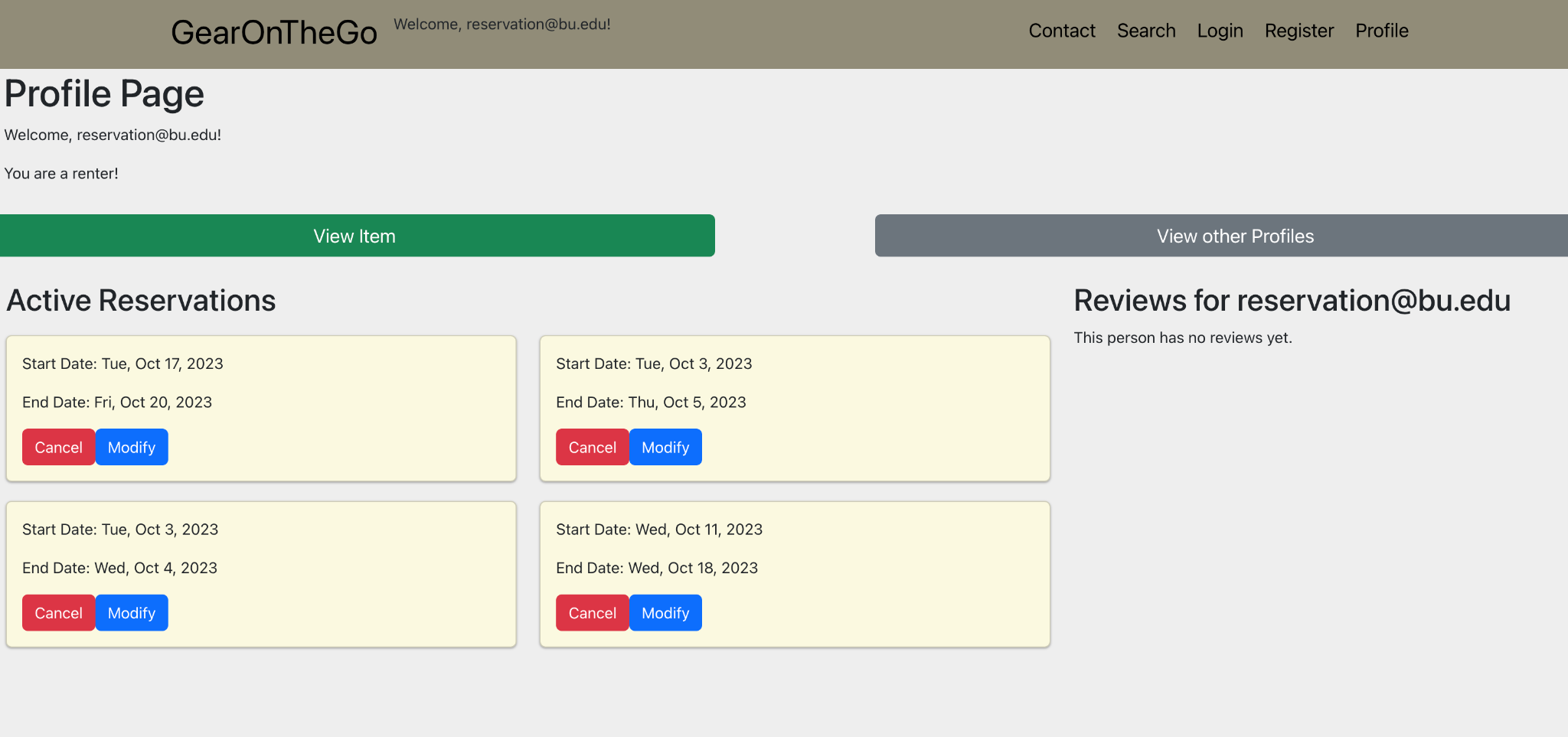
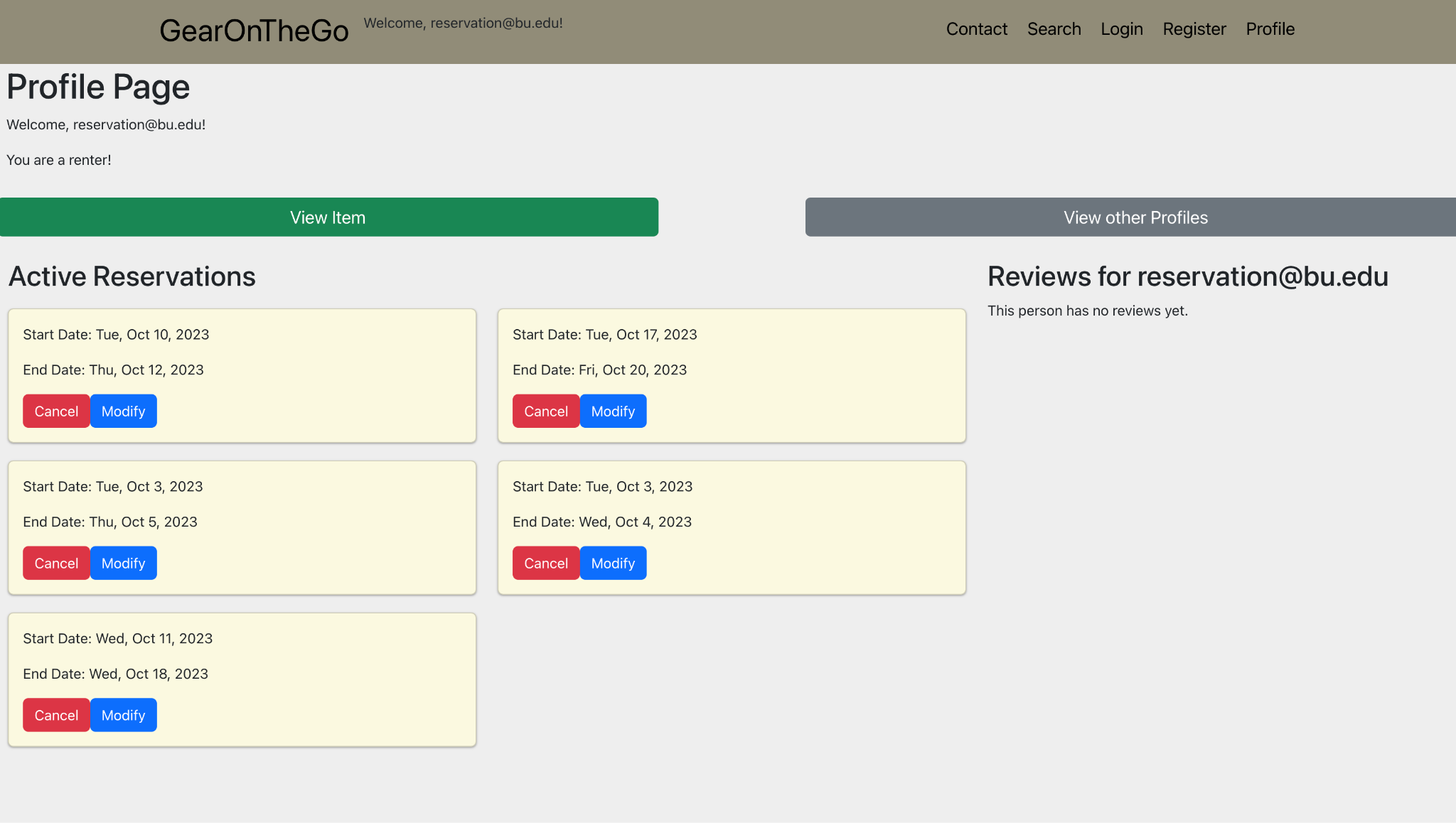
| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0034 | Navigate from the Search page to the Reservation page | Search Page and Database in ElephantSQL connected | Search Data | Go to Search page, Click an Item card, Click Reserve | Navigated to the Reservation page directly | Navigated to the Reservation page directly | Pass |
| 0035 | Save reservation start date ,end date, item id and username to database | Database in ElephantSQL connected | Start date, end date | Go to Reservation page, Enter start date, Enter end date, Click Checkout | Entered start date and end date is saved to database, along with context passed in item id and username | Entered start date and end date is saved to database, along with passed in item id and username | Pass |
| 0036 | Navigate from the Reservation page to Checkout Page | Reservation Page and Database in ElephantSQL connected | Start date, end date, item id, username | Go to Reservation page, Enter start date, Enter end date, Click Checkout | Navigated to the Checkout page directly | Navigated to the Checkout page directly | Pass |

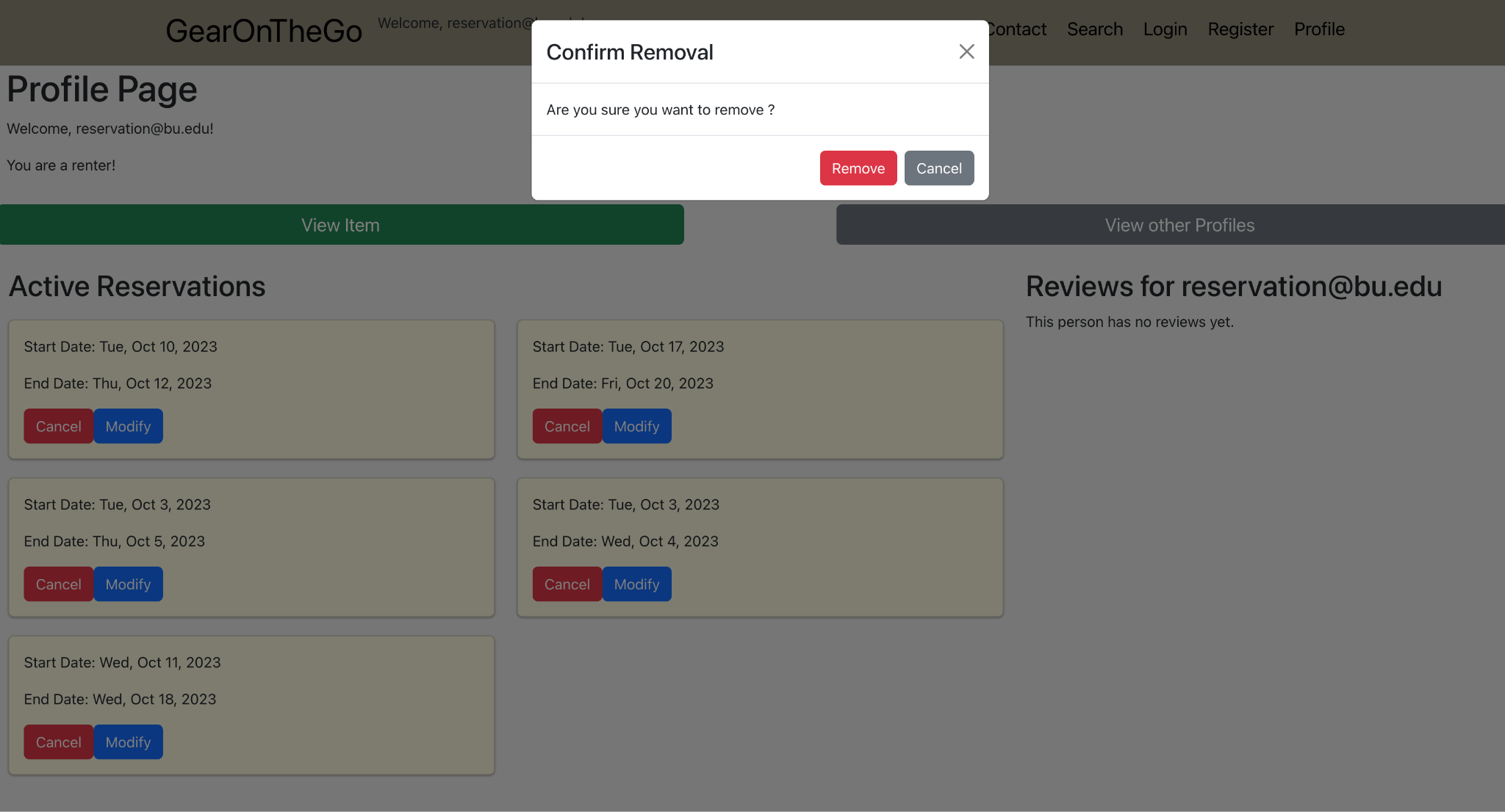
****

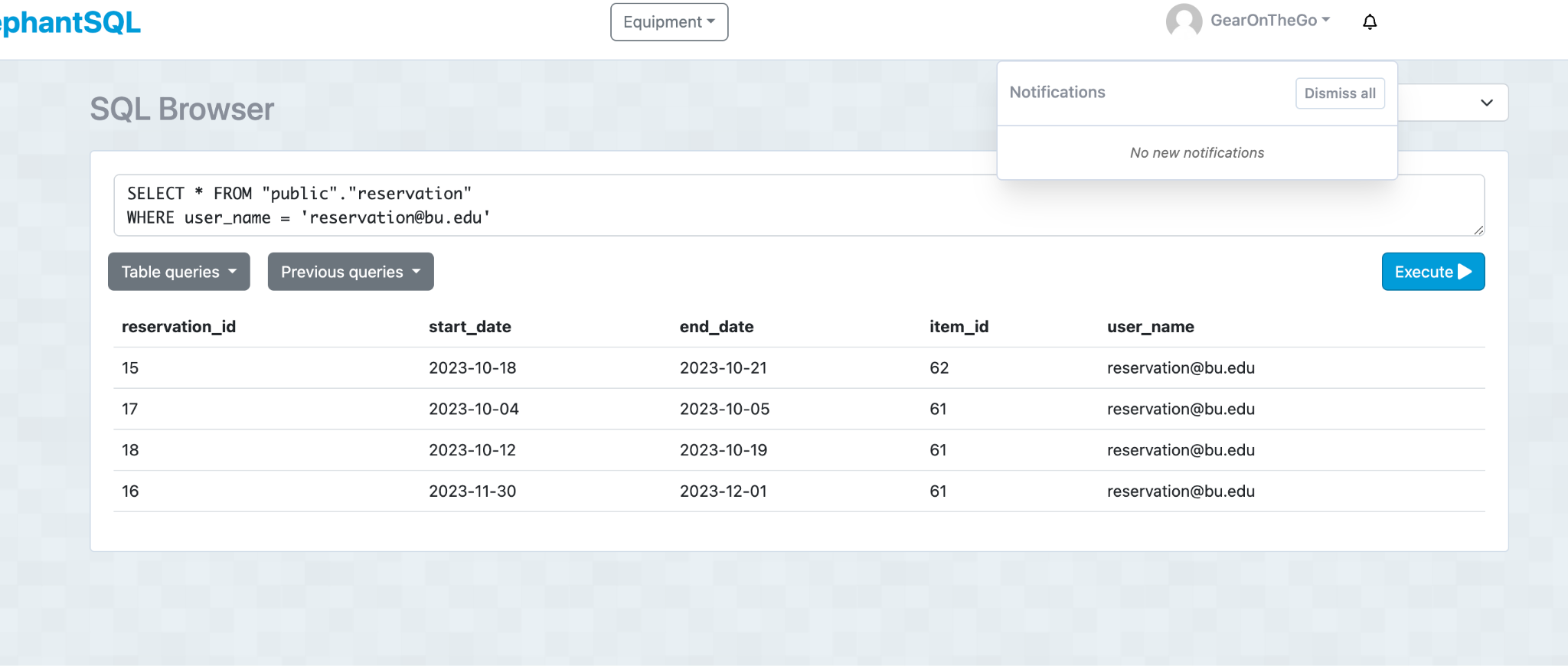
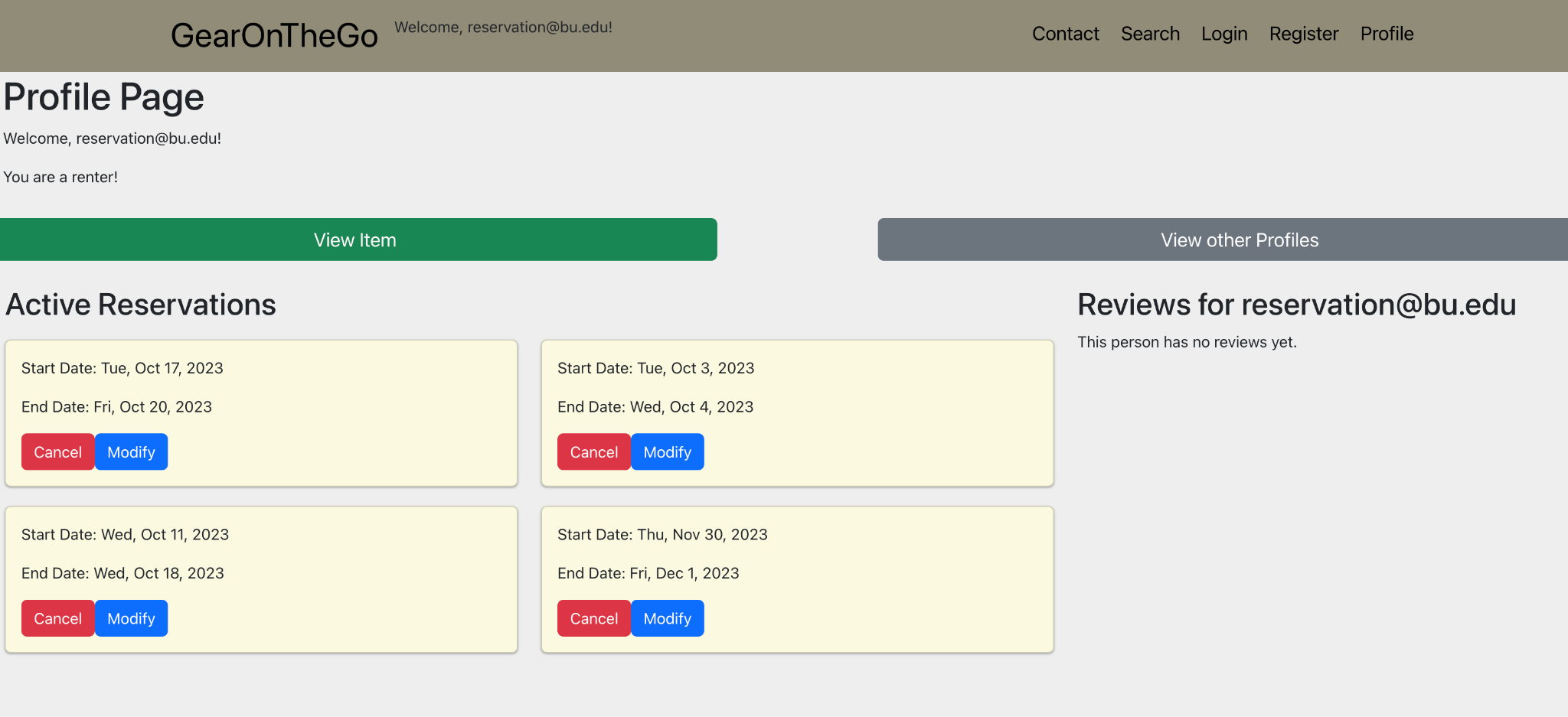
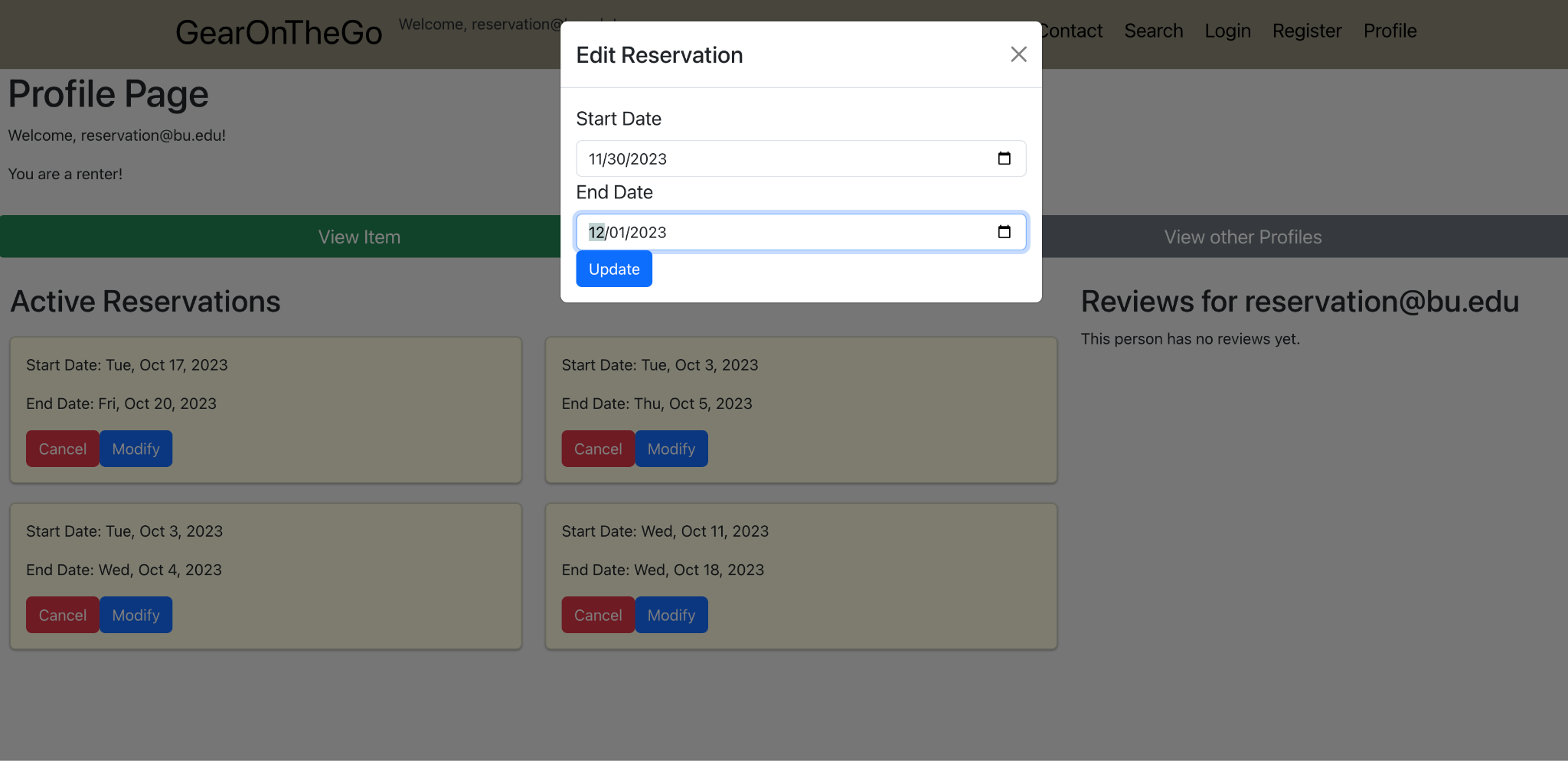
****

**Modify/Cancel Reservation Process**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0037 | Navigate to the Profile page to check all Active Reservations | Database in ElephantSQL connected | None | Log in with a renter user type account with existing reservations, Go to Profile page, check cards under Active Reservations | Active Reservation cards display | Active Reservation cards display | Pass |
| 0038 | Cancel reservations without error | Profile page, Database in ElephantSQL connected | None | Log in with a renter user type account with existing reservations, Go to Profile page, Click a card, Click Cancel, Click Remove | Card disappears, reservation removed from the database | Card disappears, reservation removed from the database | Pass |
| 0039 | Modify reservations without error | Reservation Page and Database in ElephantSQL connected | Start date, end date | Log in with a renter user type account with existing reservations, Go to Profile page, Click a card, Click Modify, Input new Start Date and new End Date, Click Update | New Dates display in the card, database changed accordingly | New Dates display in the card, database changed accordingly | Pass |

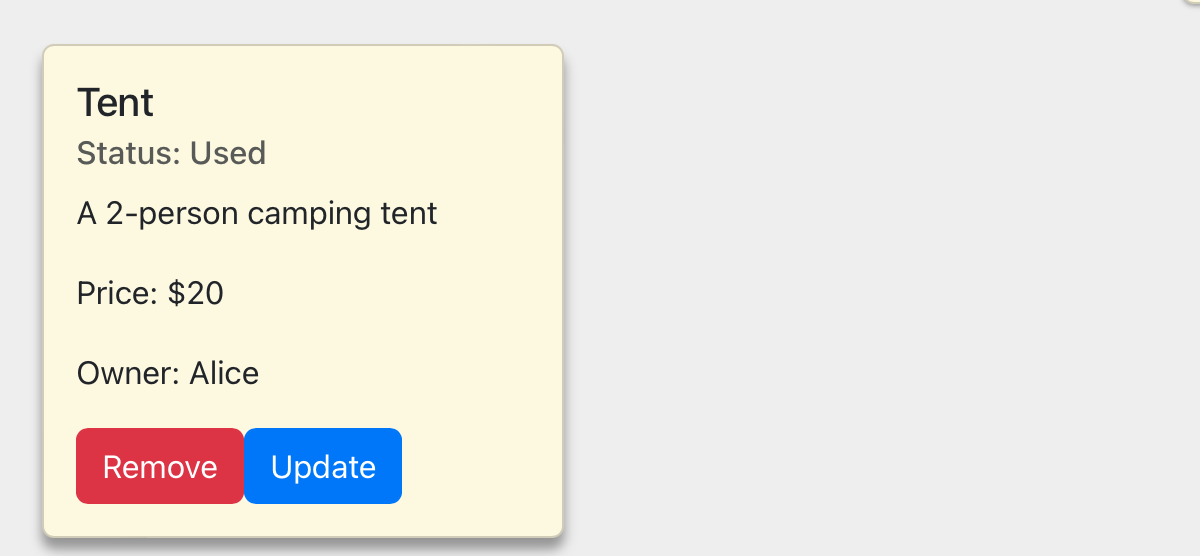
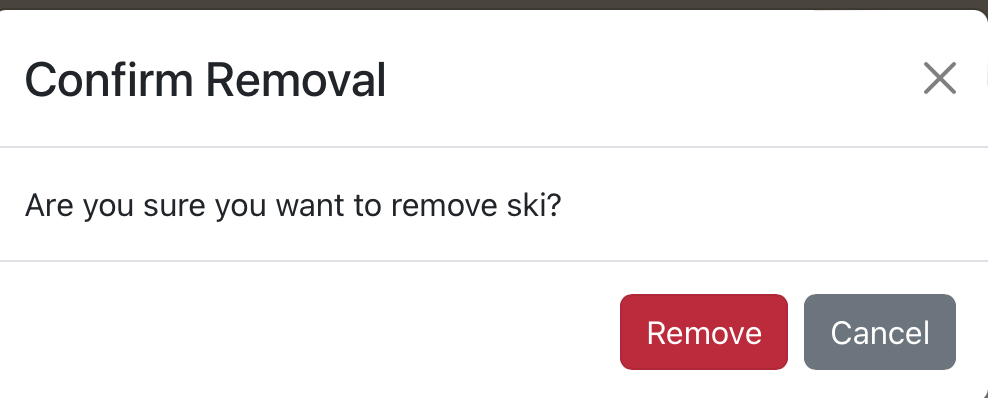
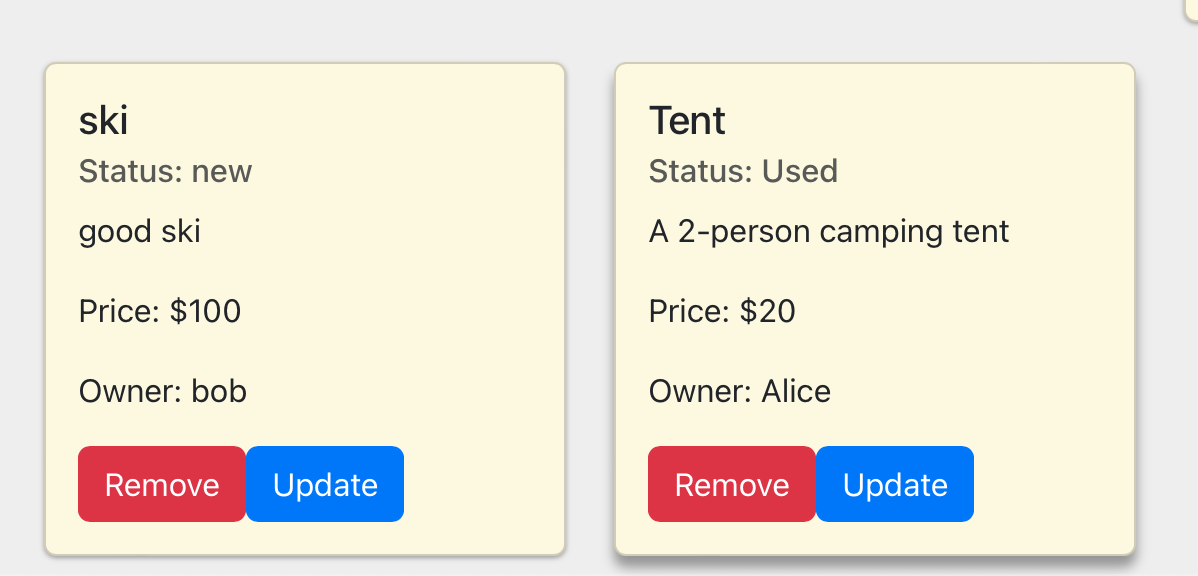
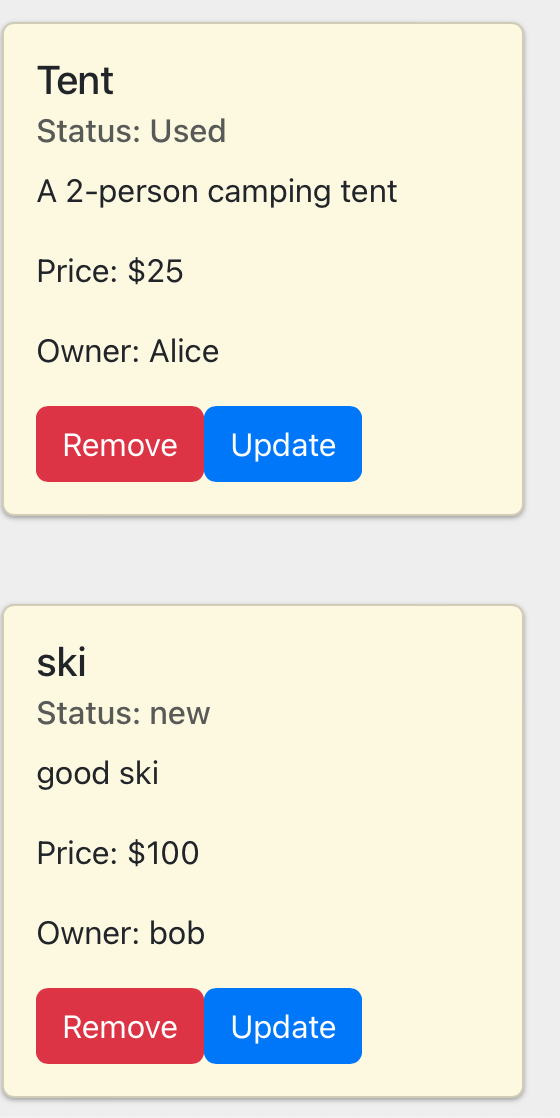
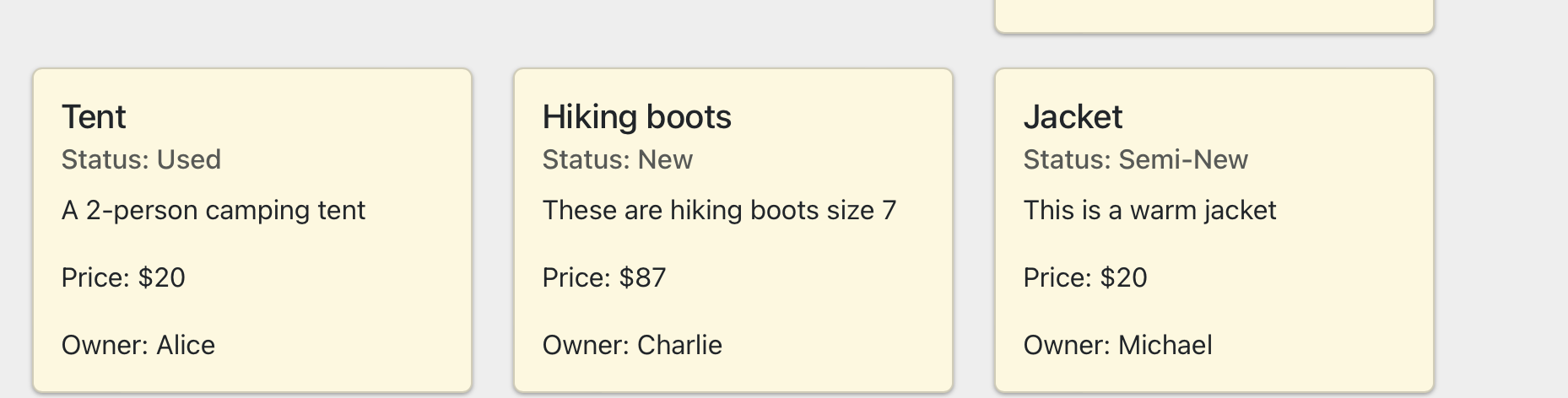
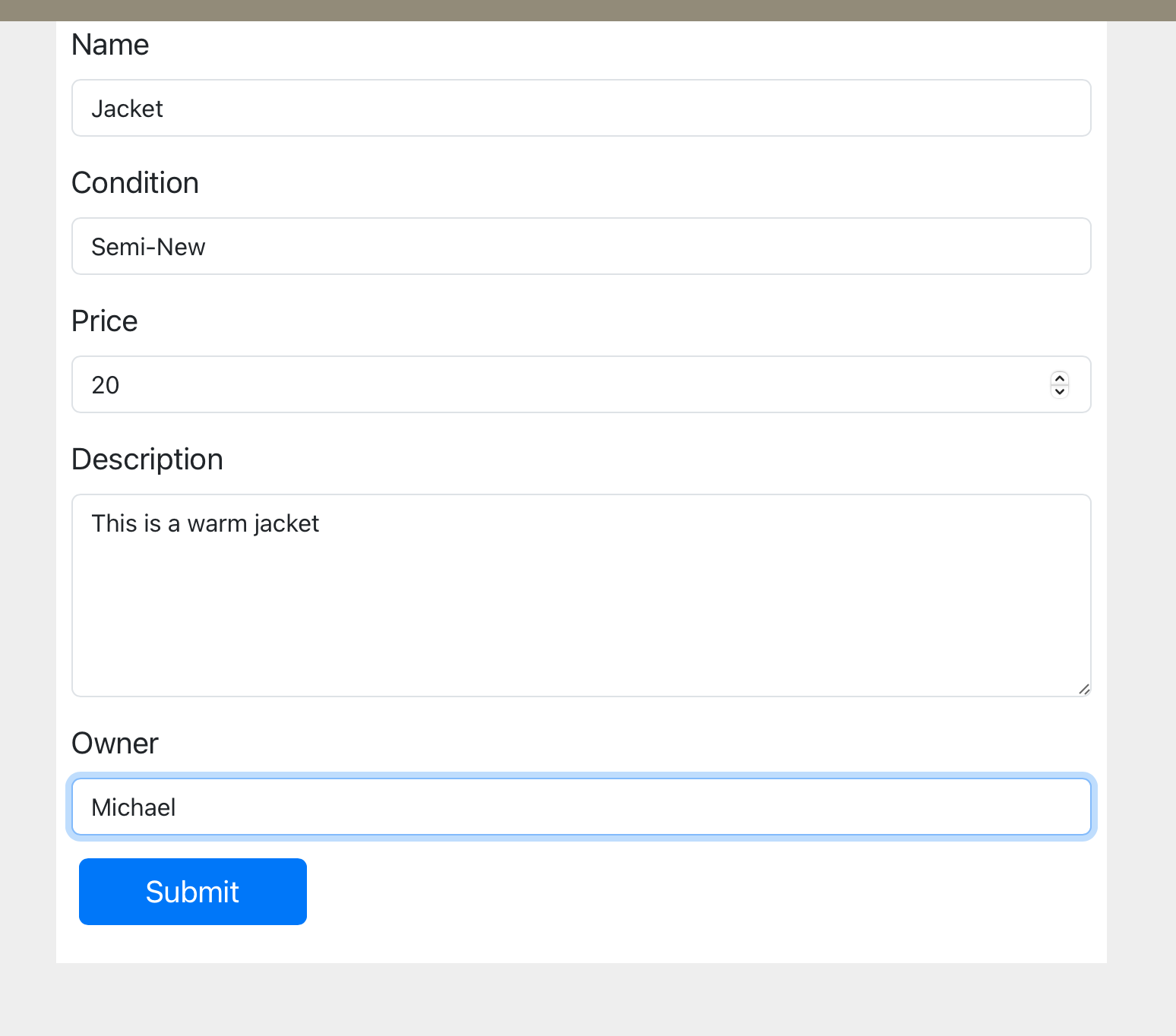
****

****

****

**Hosting Equipment**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0020 | Add Items to equipment database | Database in ElephantSQL connected | Name of Item, Description, Status, Price, Availability, User | Go to add Items, fill in the form about the information, Submit | Confirmation message displayed on the page. Correct information stored as an entry in the database | Confirmation message displayed on the page. Correct information stored as an entry in the database | Pass |
| 0021 | Remove Items from equipment database | Database in ElephantSQL connected, The database needs to have entries in it | Click the remove button on the item card that you want to delete | Click the remove button on the item card that you want to delete, Click remove if you are sure | Database will be updated without that item | Item no longer appears when you view all items | Pass |
| 0022 | Update fields of the items | Database in ElephantSQL connected, The database needs to have entries in it | Update any of these fields Name of Item, Description, Status, Price, Availability, | Click the update button on the item card that you want to update then fill out the form with the new information | Database will be updated with the new information | Item’s display the new information | Pass |

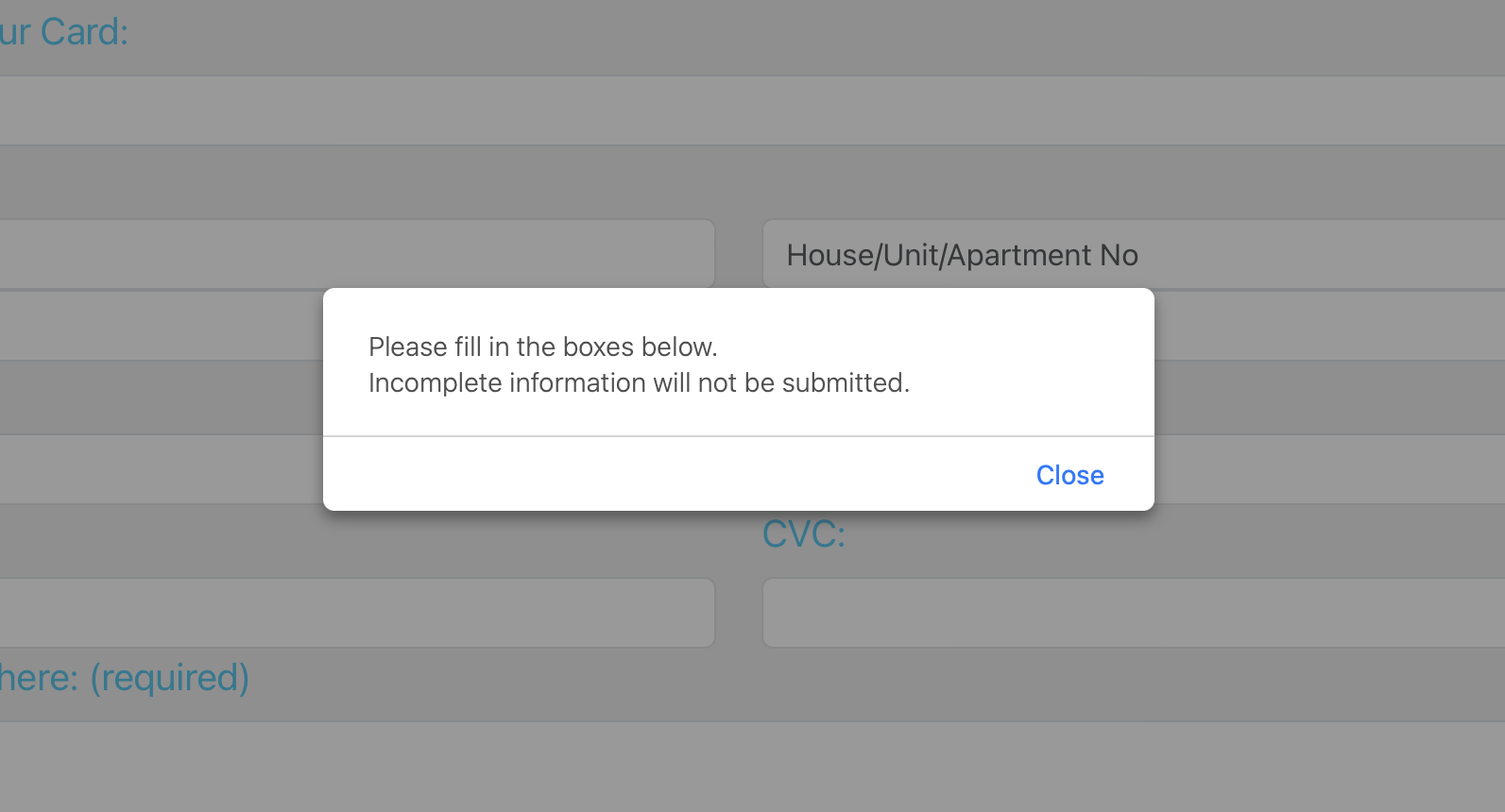
****

**Checkout Process Testing**

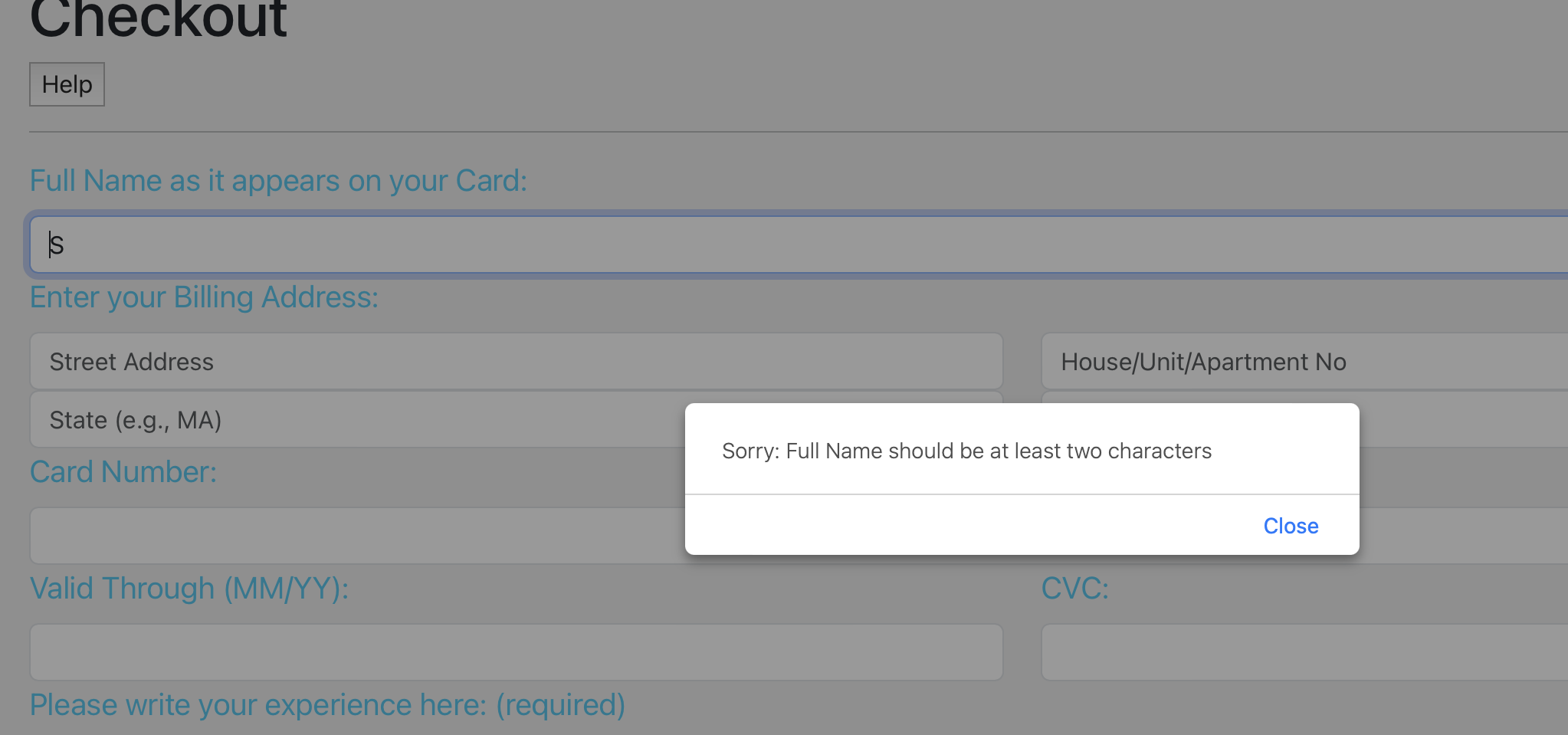
The Checkout Process is reliant on the Renting User’s Name on Card, Billing Address, Card Information etc. It is essential that the Checkout process does not allow the payment to be submitted until the Renting User has inputted the appropriate information including a valid Card Number.

This functionality was tested at each textbox to ensure that the Checkout page continues to prompt the user to correctly fill out each textbox and then informs the user if the Card Number was valid or invalid. Below are screenshots of the multiple test cases:

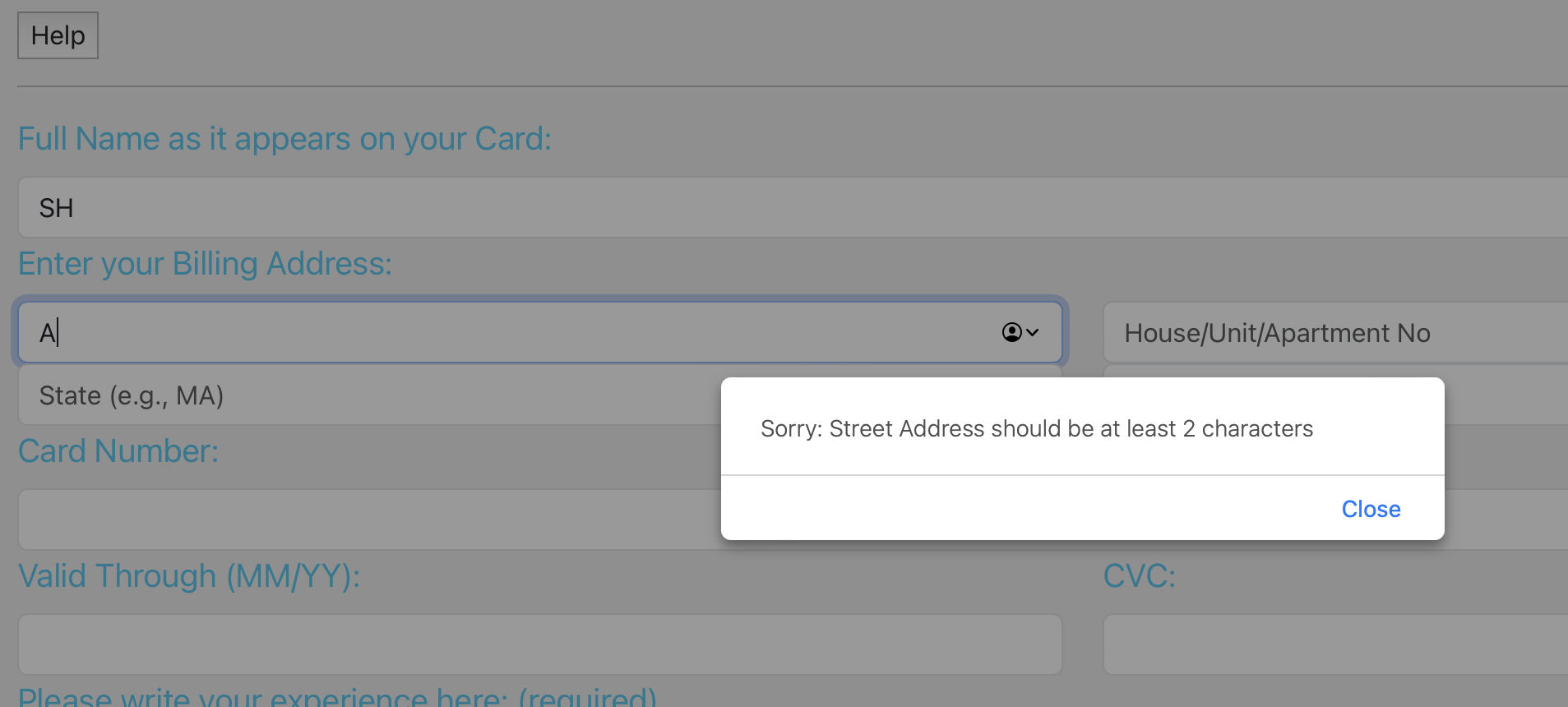
**Testing the help button:**

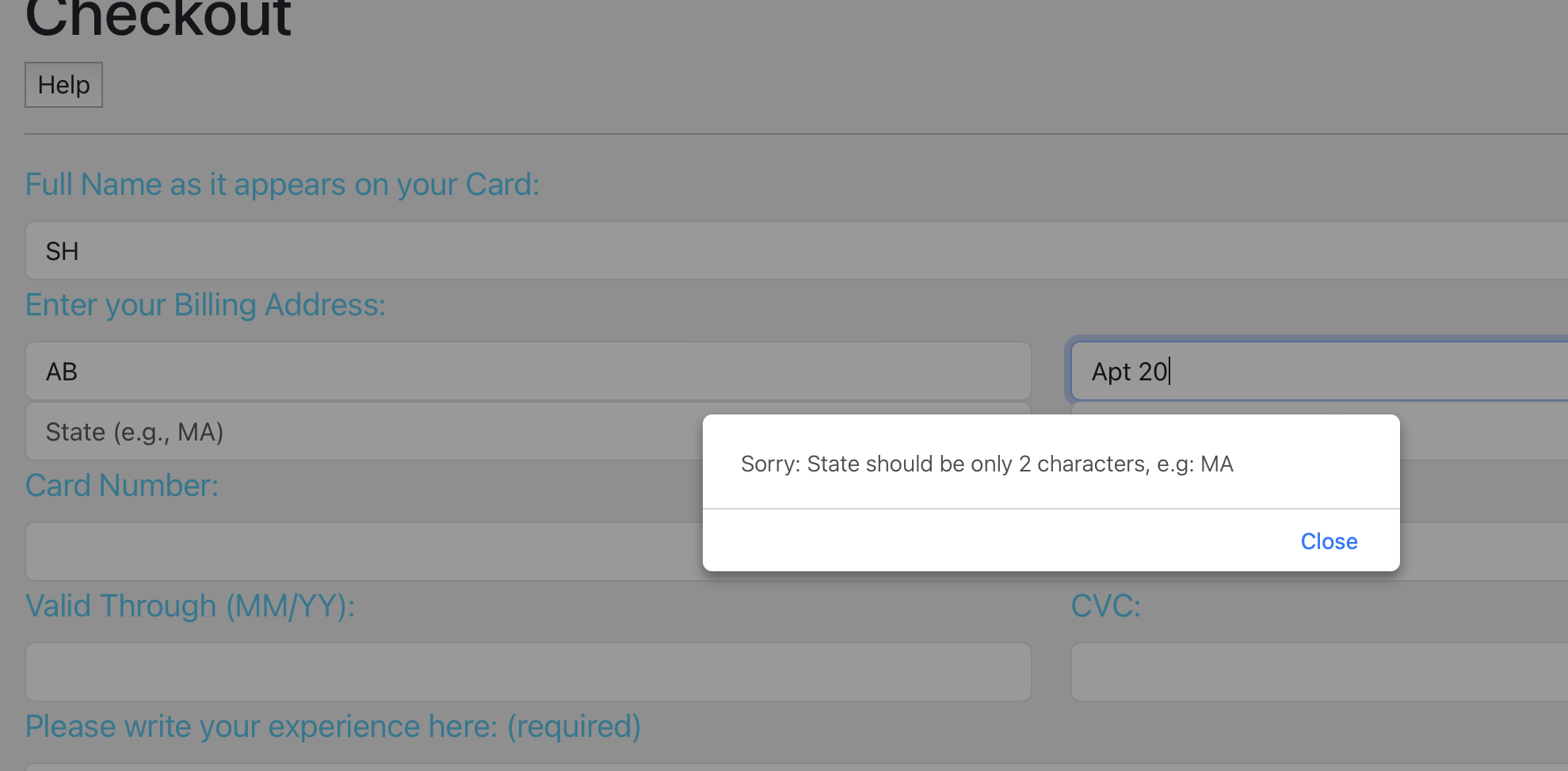
****

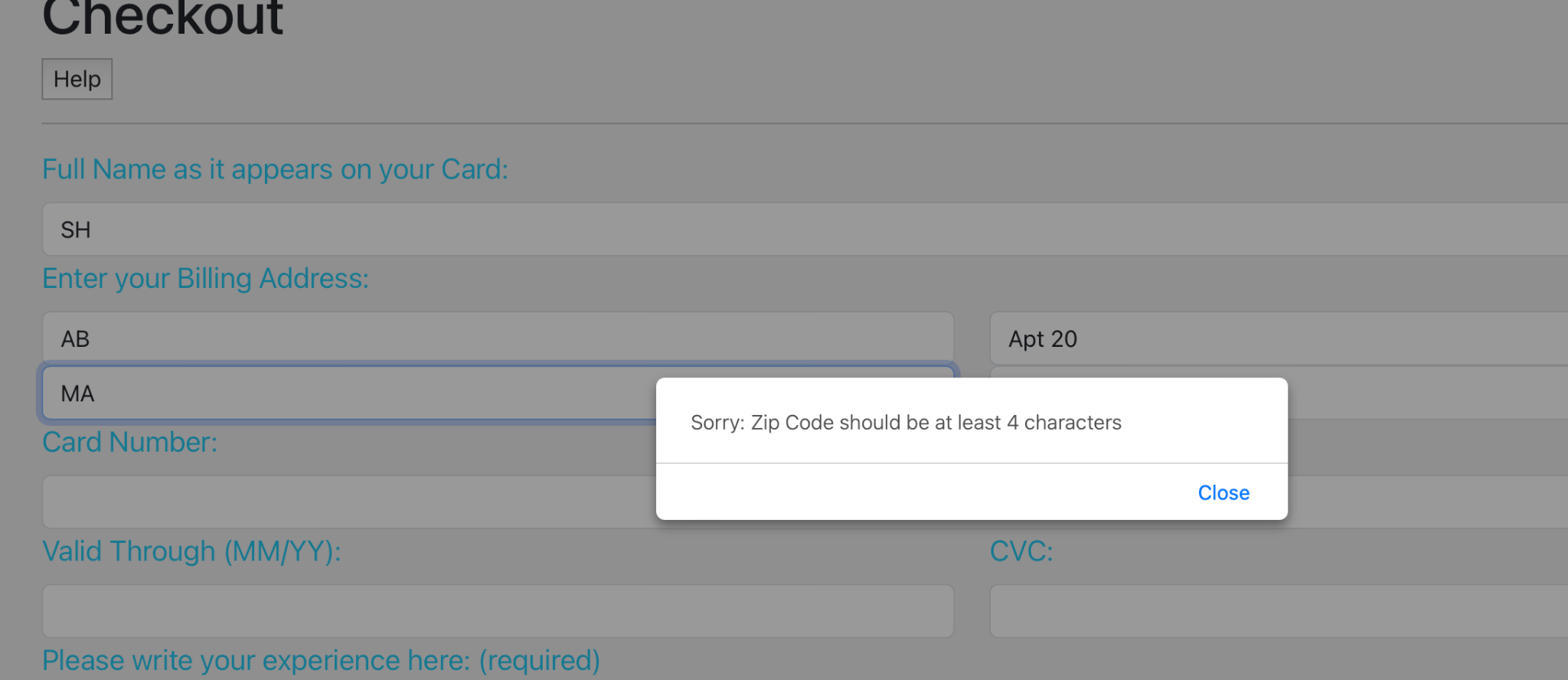
**Testing the Full Name on Card field:**

****

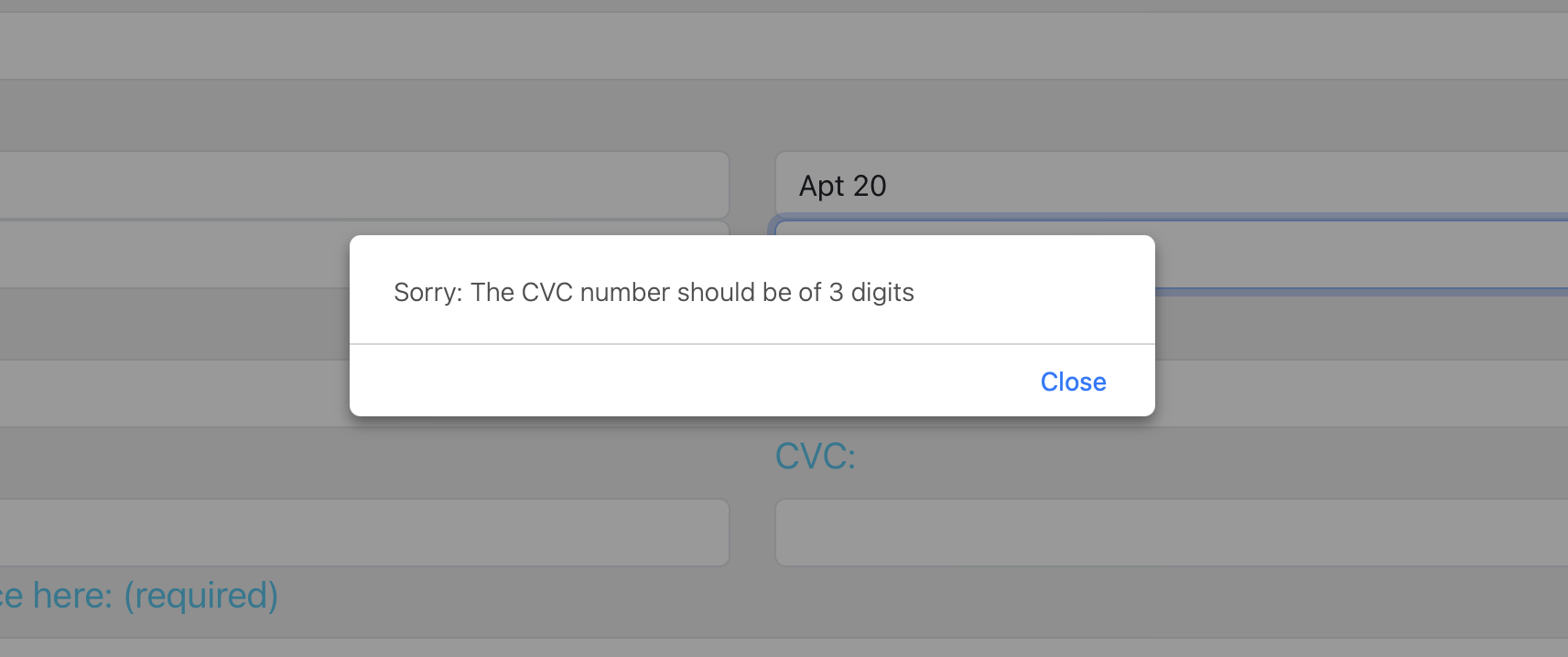
**Testing the different Billing Address fields:**

****

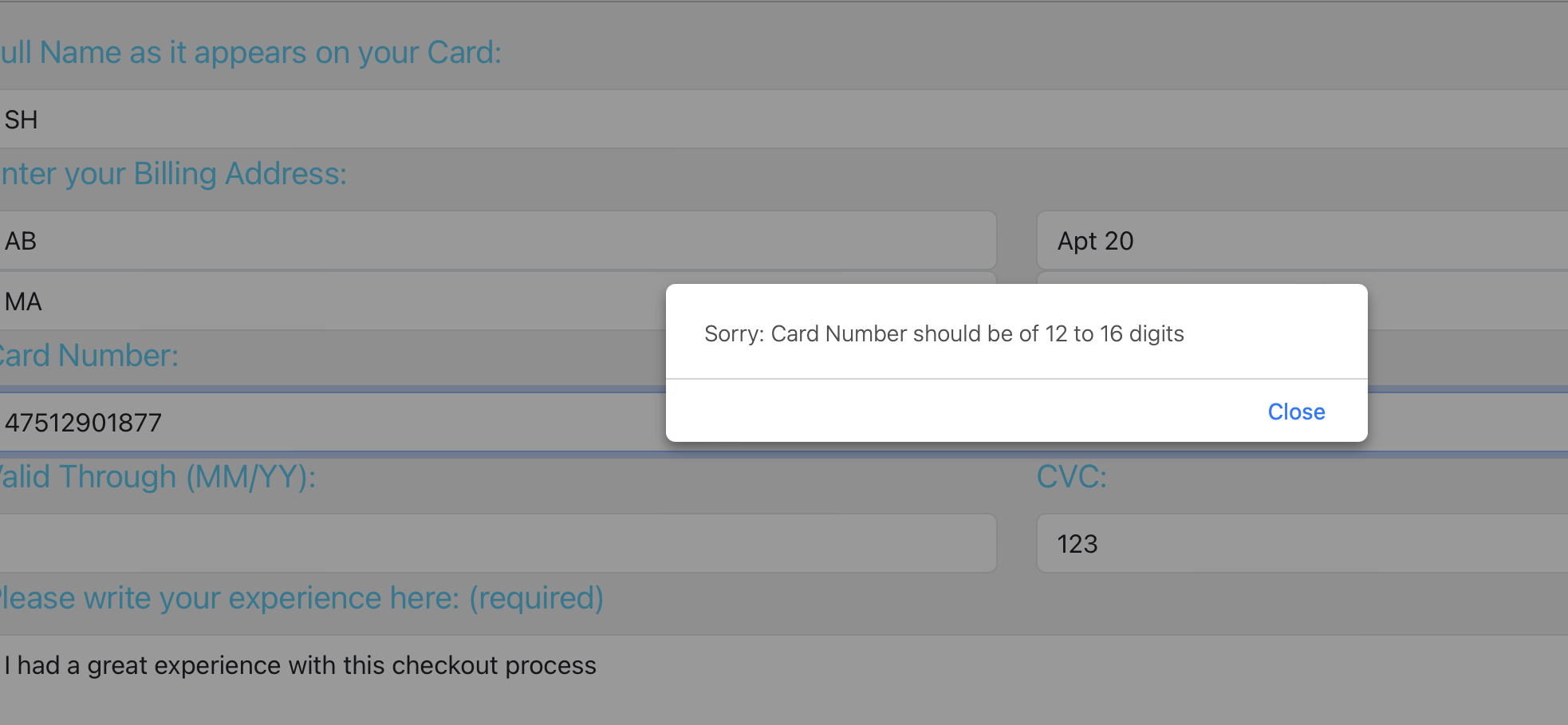
****

****

**Testing the Card Information fields:**

****

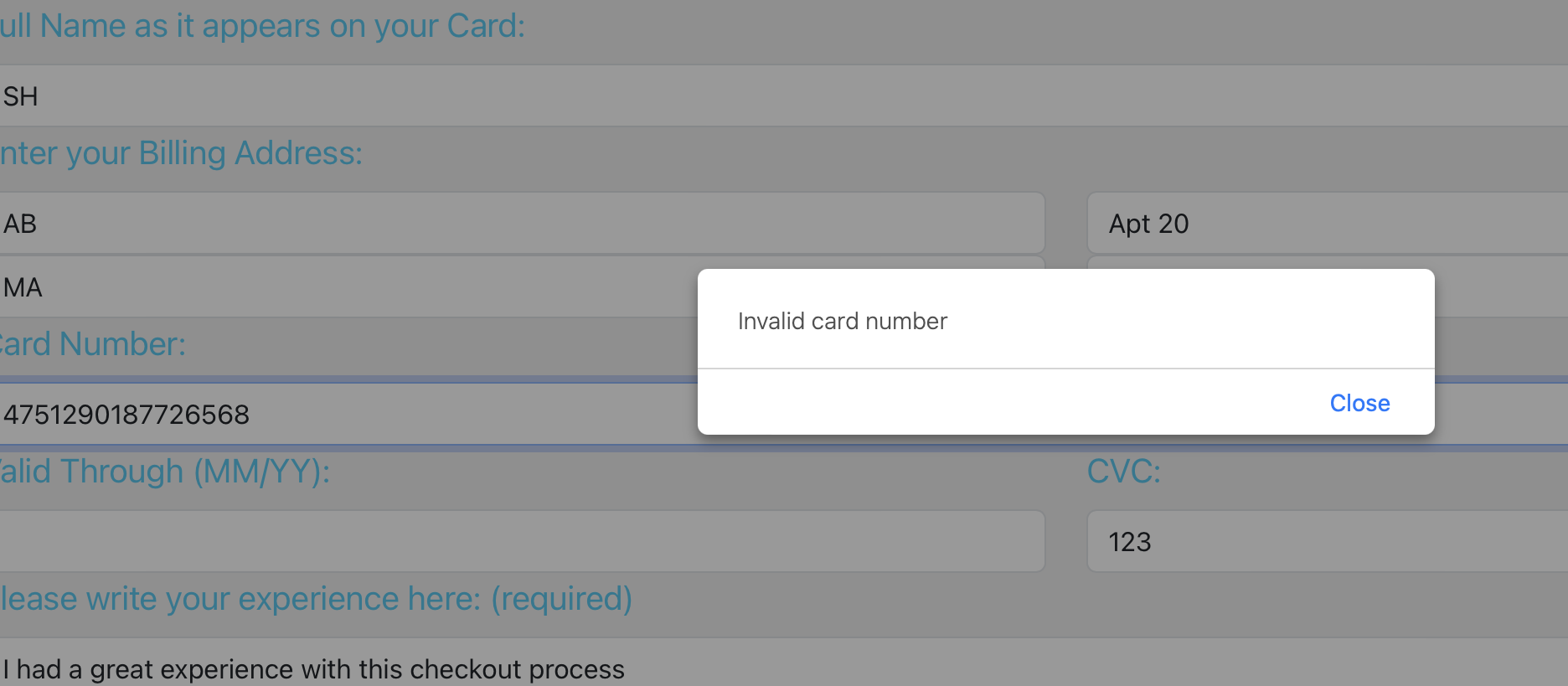
**Card Number - Using less than 12 digits:**

****

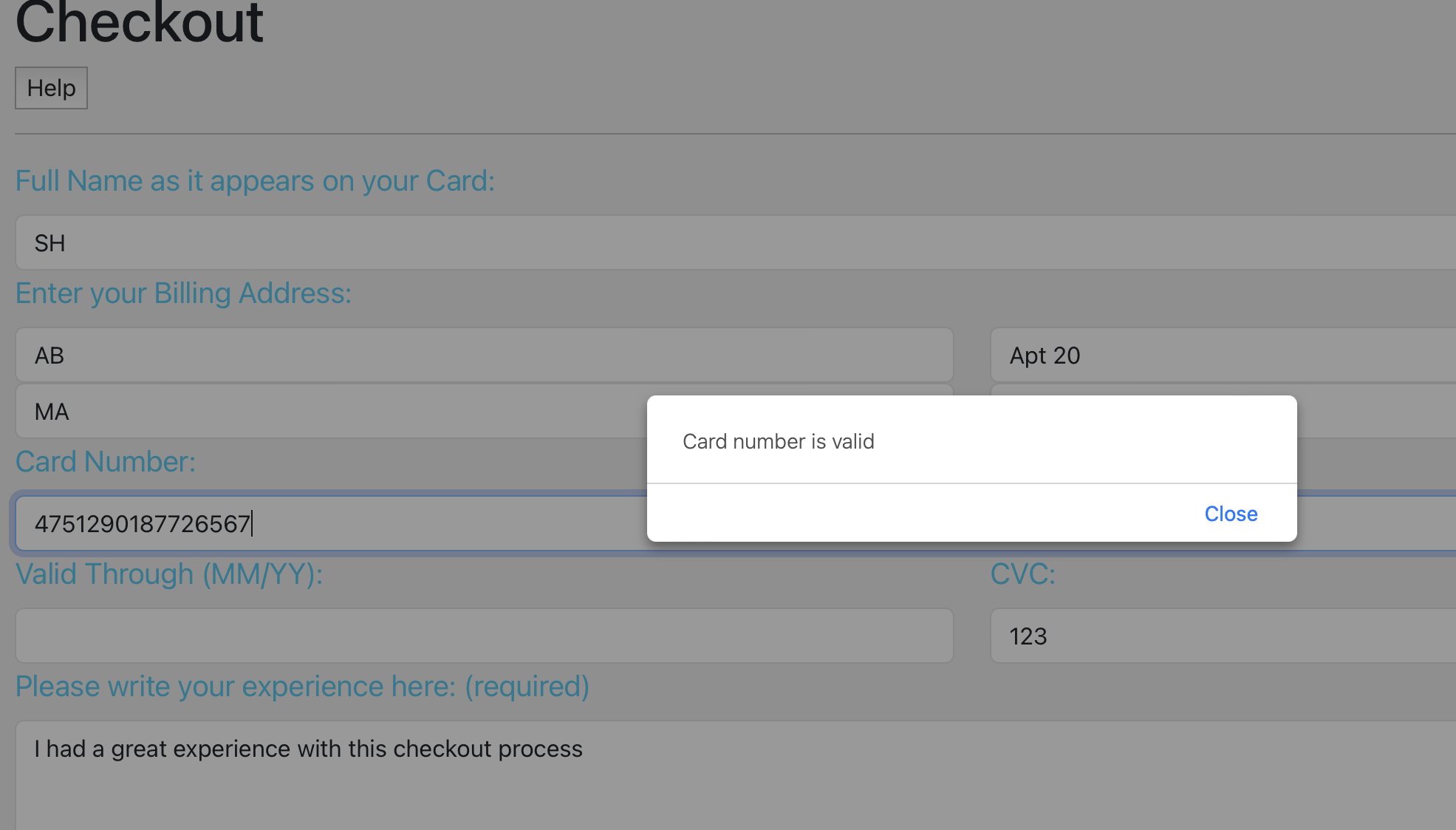
**Card Number - Using more than 16 digits:**

****

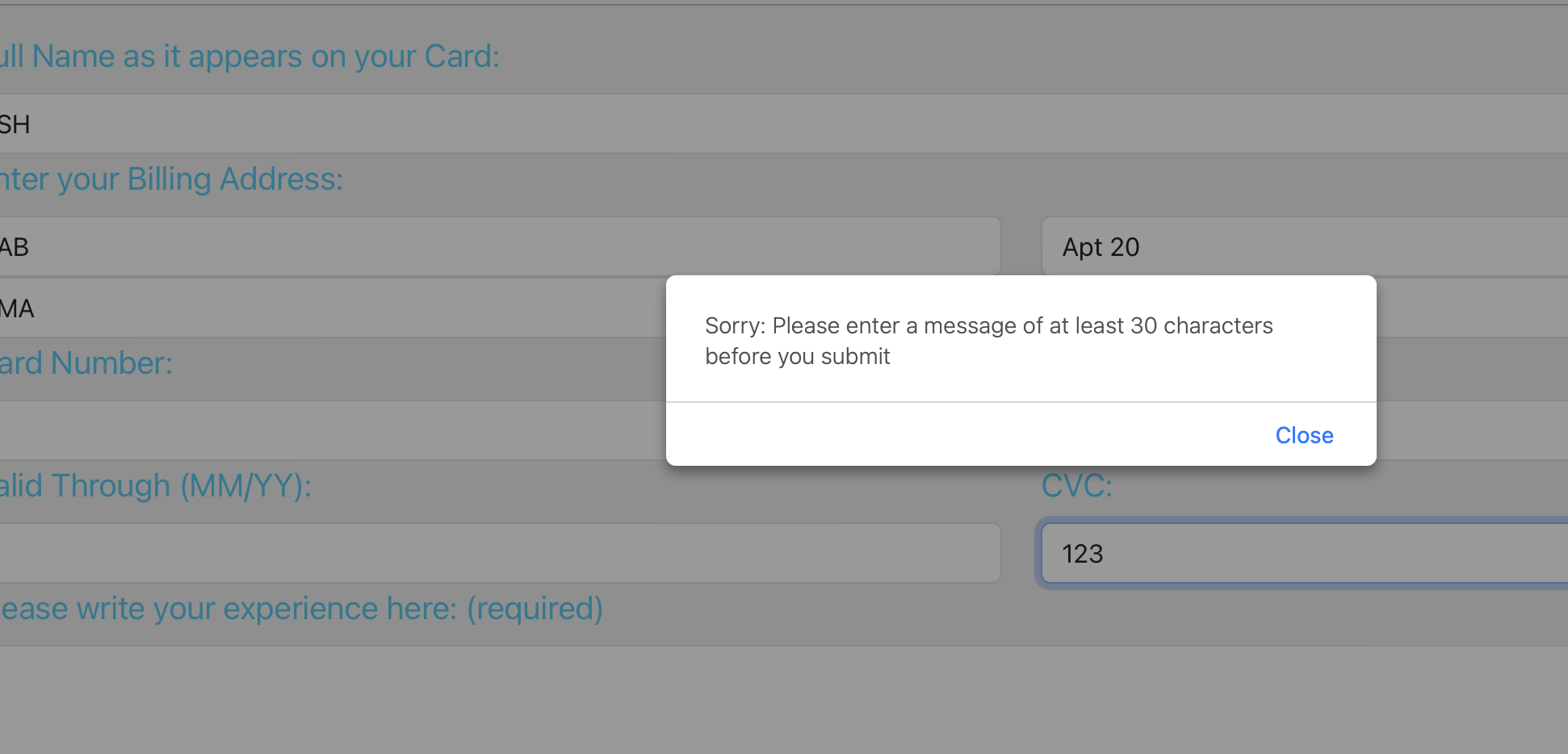
**Correct amount of digits but invalid card number:**

****

**Correct digits AND a valid card number:**

****

**Testing the short message text box:**

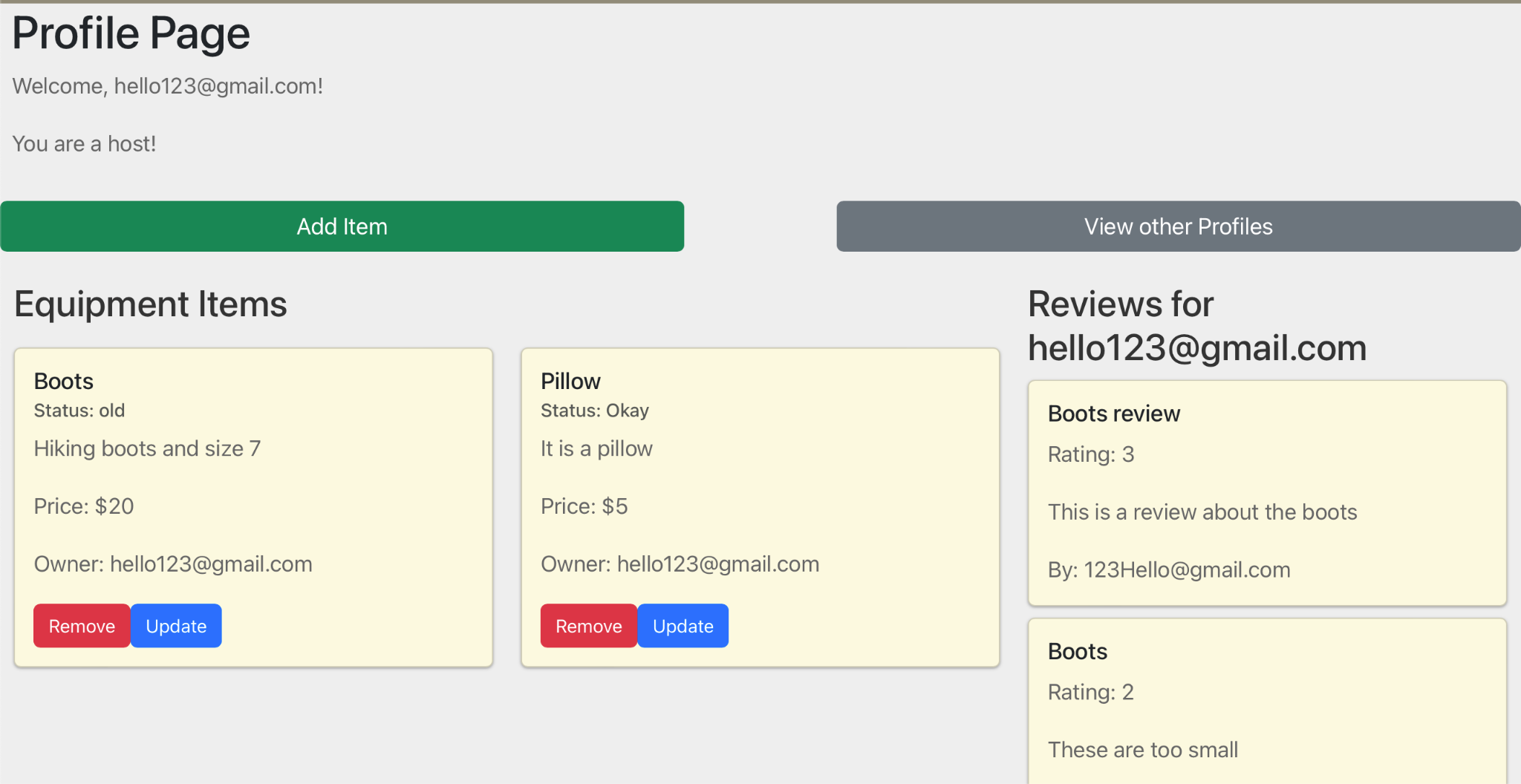
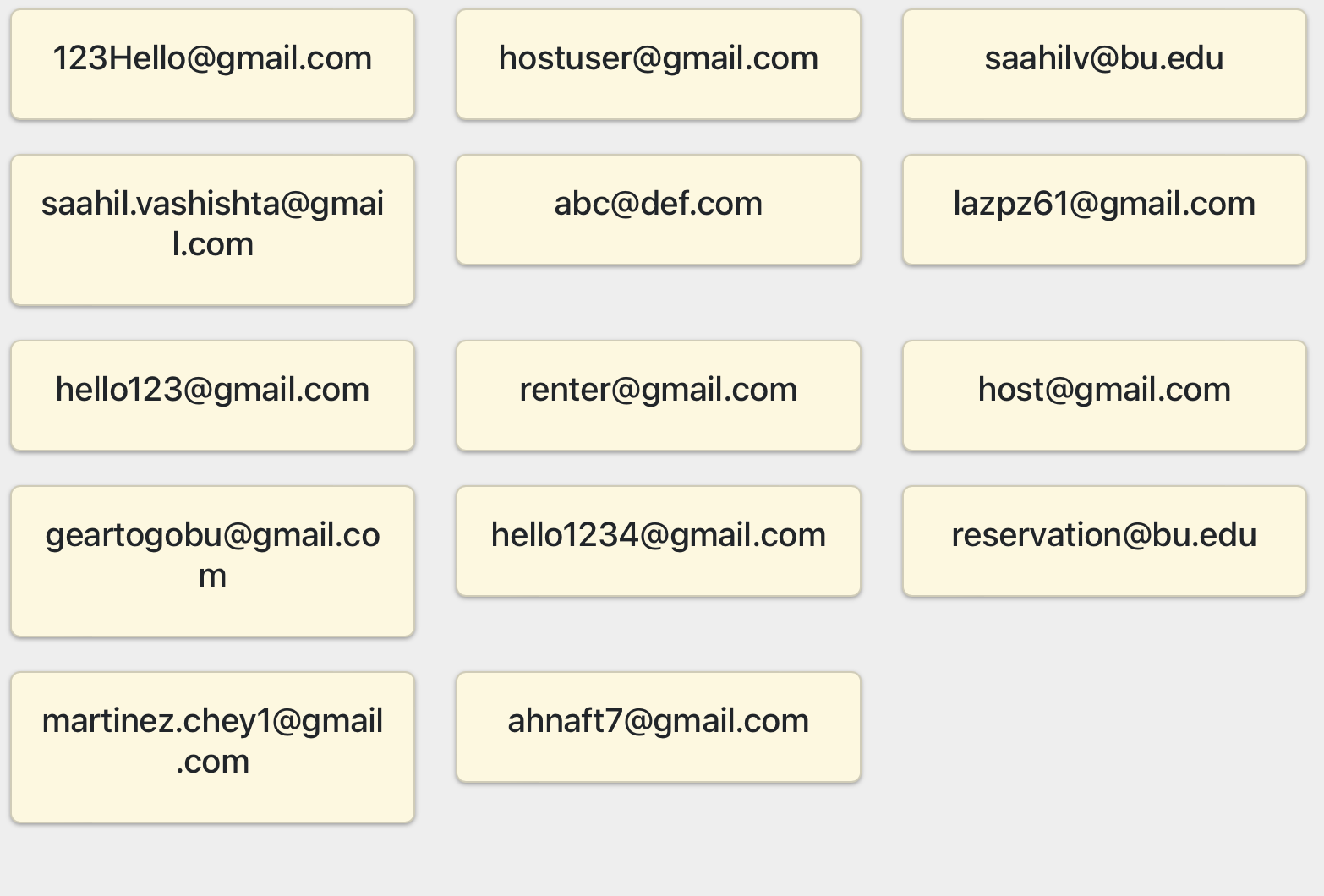
****

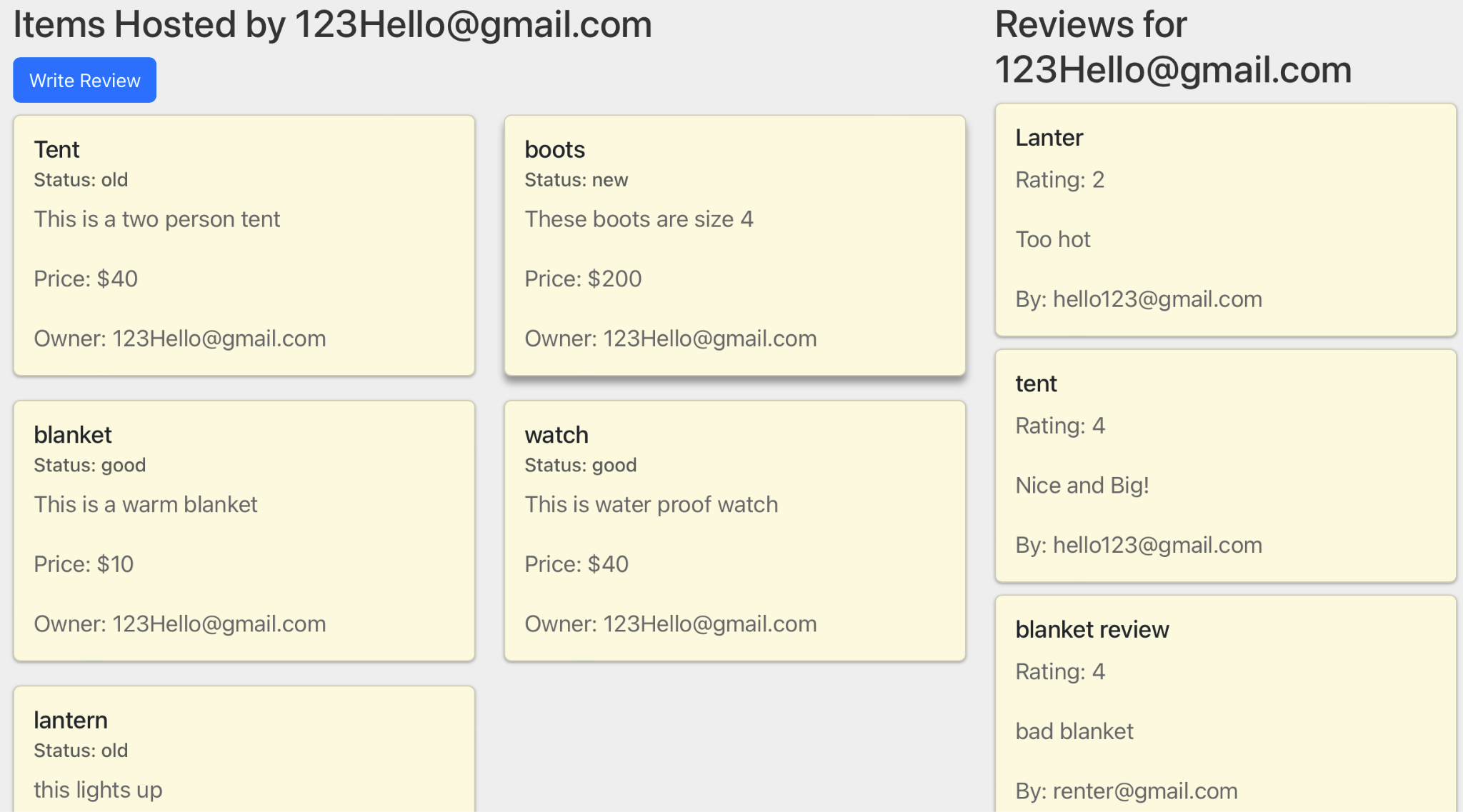
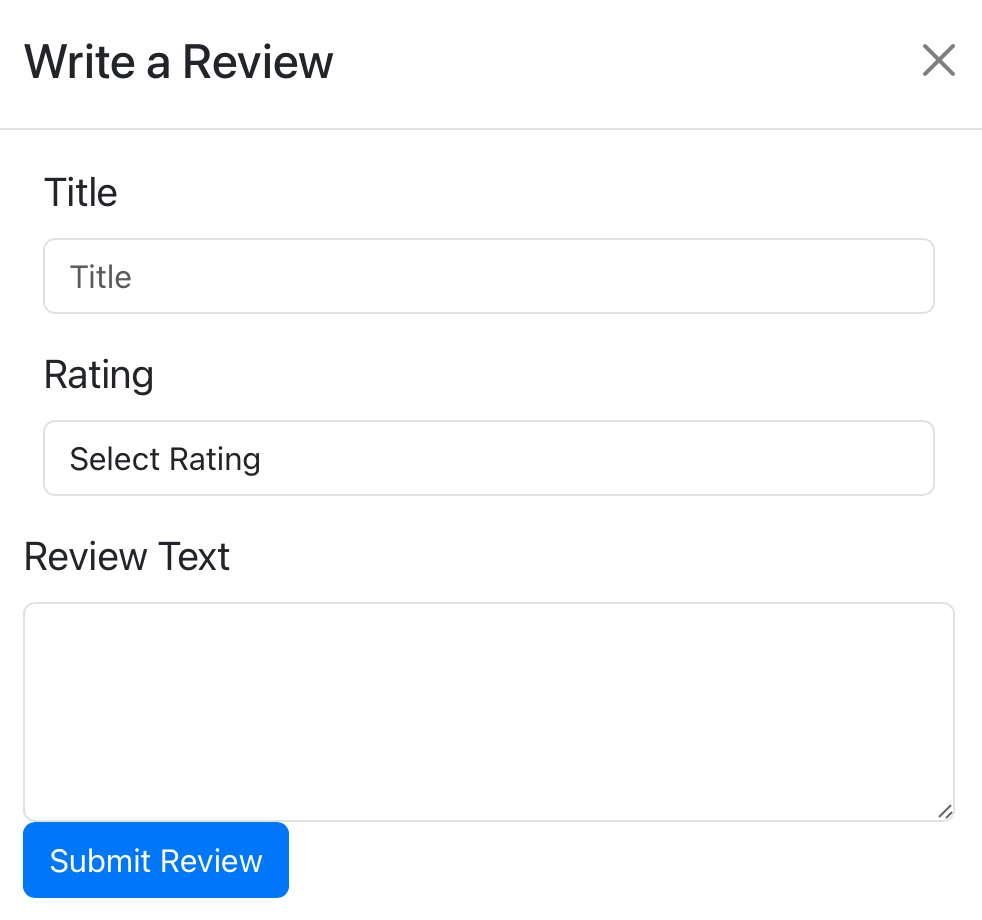
Once the user has appropriately inputted the correct information AND a valid card number, upon clicking the ‘Pay’ button, the user is redirected to the ‘Payment Successful’ page. This page has a link that allows the user to redirect to the Contact Us page, which upon clicking should execute the desired task.

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0023 | Help Button |  | On Click | Clicked | Display alert | Displayed Desired Alert | Pass |
| 0024 | Full Name |  | Text Input | Less than 2 characters | Alert Message prompts at least 2 character | Displayed Desired Alert | Pass |
| 0025 | Street Address |  | Text Input | Less than 2 characters | Alert Message prompts at least 2 character | Displayed Desired Alert | Pass |
| 0026 | State |  | Text Input | Less or more than 2 characters | Alert Message prompts exactly 2 character | Displayed Desired Alert | Pass |
| 0027 | Zip Code |  | Text Input | Less than 4 characters | Alert Message prompts at least 4 character | Displayed Desired Alert | Pass |
| 0028 | Experience Textbox |  | Text Input | Less than 30 characters | Alert Message prompts at least 30 characters | Displayed Desired Alert | Pass |
| 0029 | CVC Number |  | Text Input | Less or more than 3 digits | Alert Message prompts exactly 3 digits | Displayed Desired Alert | Pass |
| 0030 | Card Number Length |  | Text Input | Less than 12 or greater than 16 digits | Alert Message prompts 12 to 16 digits | Displayed Desired Alert | Pass |
| 0031 | Card Number Invalid |  | Text Input | Input random card number | Alert message notifies invalid card number | Displayed Desired Alert | Pass |
| 0032 | Card Number Valid |  | Text Input | Input known card number | Alert message notifies card number is valid | Displayed Desired Message | Pass |
| 0033 | Pay Button will always ask for Card Number valid |  | On Click | Try different variations of all previous tests until Card Number is valid | Alert message notifies Card Number is valid | Displayed Desired Message | Pass: The button correctly relies on Card number being valid as True in order to submit. |
| 0034 | Pay Button redirects to ‘Payment Successful’ page |  | On Click | Does not redirect until Card number is valid | Checkout Page redirects to Payment Successful page | Displayed Desired Page | Pass |

**Profile Pages and Reviews**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0035 | Profile Page can show equipment that they are hosting | Database in ElephantSQL connected  User is logged in | None | View that the items that they are hosting are displayed | Correct information stored as an entry in the database | All items under that user are displayed | Pass |
| 0036 | Profile Page can show reviews that are attached to that user | Database in ElephantSQL connected  User is logged in | None | View that the reviews that they have are displayed | Correct information stored as an entry in the database | All reviews under that user are displayed | Pass |
| 0037 | View another person profile | Database in ElephantSQL connected  User is logged in | Title, Rating, Description on the user | Click the View other profiles button, then click on a user then click on write a review | See all the items that they host as well as their reviews. See your new review join the review column | The New review has been added to the other user’s profile | Pass |

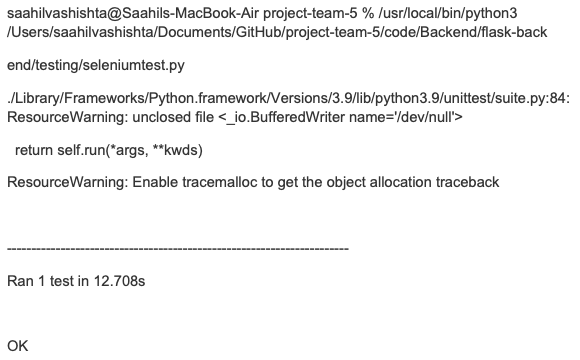
 

# Automated Testing Report

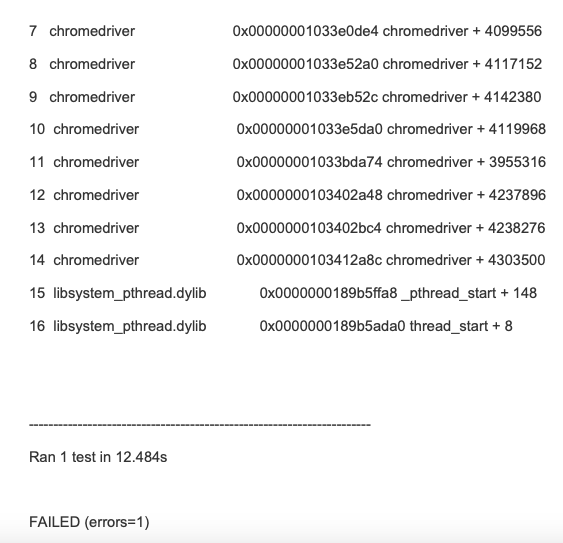
In this section, we will provide a detailed description of each automated test case performed using Selenium.

1. **User Registration**

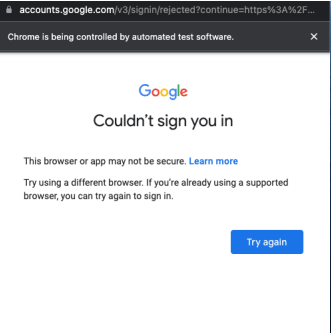
| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A-001 | User Registration with Google Auth - correct e-mail | 1.Google account  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script | Simulated:  1.Navigate to the user registration page  2.Click on Sign-up with Google  3.Navigate to the Pop-up window  4. Add email ID  5.Enter | Email ID is valid so the result would return PASS | The Email ID was correct so result returned PASS would Google Security detected non-secure activity so it didn’t allow entering a password | PASS |
| A-002 | User Registration with Google Auth - incorrect e-mail | 1.Google account  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script | Simulated:  1.Navigate to the user registration page  2.Click on Sign-up with Google  3.Navigate to the Pop-up window  4. Add incorrect email ID  5.Enter | Email ID is invalid so the result would return FAIL | The result returned FAIL | PASS |
| A-003 | User Registration with Google Auth - correct e-mail and password | 1.Google account  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script | Simulated:  1.Navigate to the user registration page  2.Click on Sign-up with Google  3.Navigate to the Pop-up window  4. Add email ID  5.Enter  6. Add password  7.Press Enter | Email ID and password would return PASS | Due to the vulnerability detection from Google, the test didn’t go beyond entering User email. | FAIL |
| A-004 | User Registration with Google Auth - correct e-mail and password with timing gap changed to trick Google OAuth | 1.Google account  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script | Simulated:  1.Navigate to the user registration page  2.Click on Sign-up with Google  3.Navigate to the Pop-up window  4. Add email ID  5.Enter  6. Add password  7.Press Enter | Email ID and password would return PASS | Due to the vulnerability detection from Google, the test didn’t go beyond entering User email, the introduced lag didn’t work. | FAIL |



*Passed Selenium test for Google Email ID*



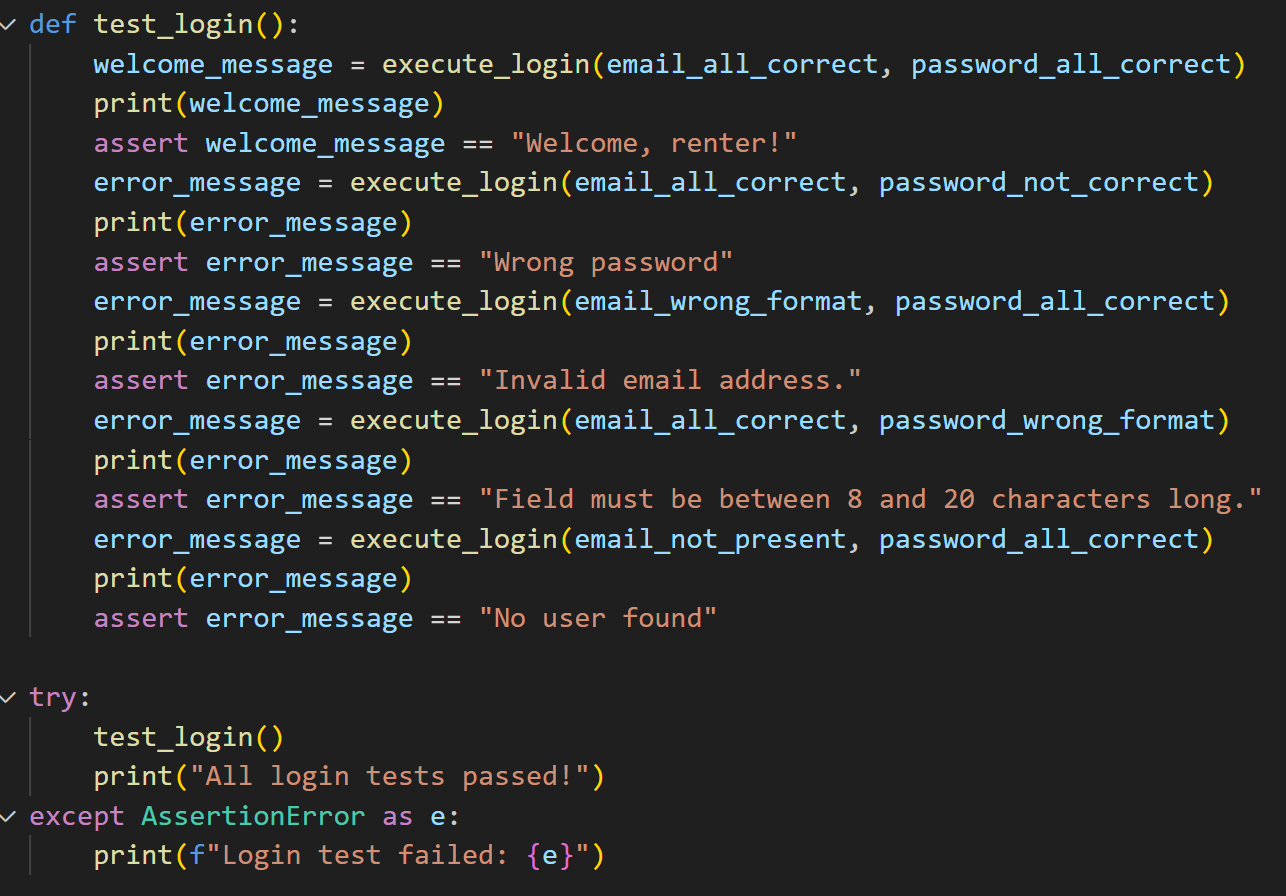
*Passed Selenium test for Google Email ID with incorrect Email*



*Google OAuth detecting the selenium test as not secure preventing further attempts and thus failing the test*

1. **User Login**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A-005 | User Login with correct email and password | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Correct email and password | Simulated:  1.Navigate to the user login page  2.Enter correct email and password combination 3. Click Login | Email and password is valid, so User is logged in and redirected to homepage | Email and password is valid, so User is logged in and redirected to homepage | PASS |
| A-006 | User Login with correct email and incorrect password but correct format | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Correct email and incorrect password but correct format | Simulated:  1.Navigate to the user login page  2.Enter correct email and incorrect password 3. Click Login | Email and password is not valid, so User is not logged in. Incorrect password error message displayed. | Email and password is not valid, so User is not logged in. Incorrect password error message displayed | PASS |
| A-007 | User Login with incorrect email format and correct password | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Incorrect email format and correct password | Simulated:  1.Navigate to the user login page  2.Enter incorrect email format and correct password combination 3. Click Login | Email and password is not valid, so User is not logged in. Invalid email format error message displayed. | Email and password is not valid, so User is not logged in. Invalid email format error message displayed. | PASS |
| A-008 | User Login with correct email and incorrect password format | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Correct email and incorrect password format | Simulated:  1.Navigate to the user login page  2.Enter correct email and incorrect password 3. Click Login | Email and password is not valid, so User is not logged in. Incorrect password format error message displayed. | Email and password is not valid, so User is not logged in. Incorrect password format error message displayed. | PASS |
| A-009 | User Login with incorrect email and password | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Incorrect email and password | Simulated:  1.Navigate to the user login page  2.Enter incorrect email and password combination 3. Click Login | Email doesn’t exist, so User is not logged in. User not found error message displayed. | Email doesn’t exist, so User is not logged in. User not found error message displayed. | PASS |



*Selenium log in test code with all test combinations*

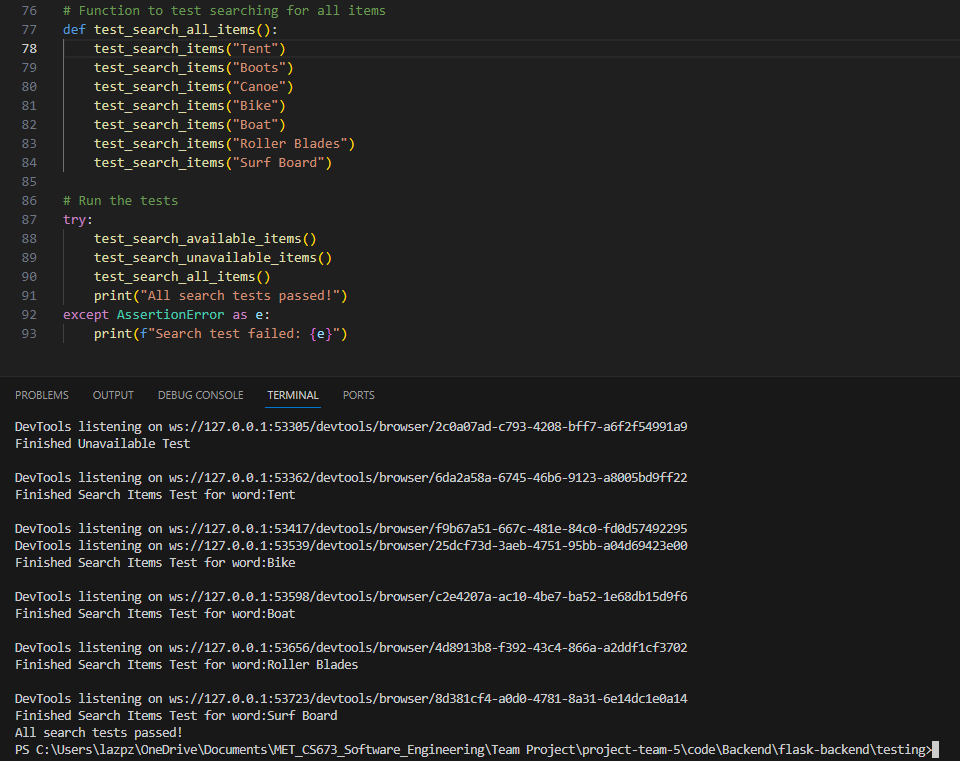
1. **Items**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A-010 | User Login and add items to the database to hose | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Correct email and password and Information about the item to add | Simulated:  1.Navigate to the user login page  2.Enter correct email and password combination 3. Click Login  4. Navigat to Add Item page  5. Enter in information  6. Click submit | Email and password is valid, so User is logged in and redirected to homepage where you can click the view page of the user logged in and see the new item | Email and password is valid, so User is logged in and redirected to homepage where you can click the profile page of the user logged in and see the new item | PASS |
| A-011 | User Login and remove items to the database to hose | 1.Registered User  2.Selenium installed  3.Database in Elephant SQL connected | Input Automated with Selenium test script  Correct email and password and remove the item chosen | Simulated:  1.Navigate to the user login page  2.Enter correct email and password combination 3. Click Login  4. Navigtet to view page  5. Click remove button  6. Click confirm remove button | Email and password is valid, so User is logged in and redirected to view page where an item is then removed | Email and password is valid, so User is logged in and redirected to view page where you can see the item selected is no longer there | PASS |

****

1. **Search**

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A-011 | Available Items | get\_Items function in app.py app route api/items, Selenium WebDriver | Clicking the Available Radio Button to run results | Login into Account, and Press Search, then press Available Radio Button, and then Search | Rendering of Items on Screen that are available | Rendering of Items on Screen that are available | Pass |
| A-012 | Unavailable Items | get\_Items function in app.py app route api/items Selenium WebDriver | Clicking the Unavailable Radio Button to run results | Login into Account, and Press Search, then press Unavailable Radio Button, and then Search | Rendering of Items on Screen that are unavailable | Rendering of Items on Screen that are specific to the search term | Pass |
| A-013 | Search Term Items | Search\_items functions in app.py route for api/searchItems. Selenium WebDriver | Clicking the All radio button injecting a search term and running the search button | Login into Account, and Press Search, Clicking the All radio button injecting a search term and running the search butto | Rendering of Items on Screen that are specific to the search term | Rendering of Items on Screen that are specific to the search term | Pass |



# Functional Testing:

# In this section, we will provide a detailed description of functional tests (if conducted) for features that require it.

1. User Registration for Google (Using Pytest)

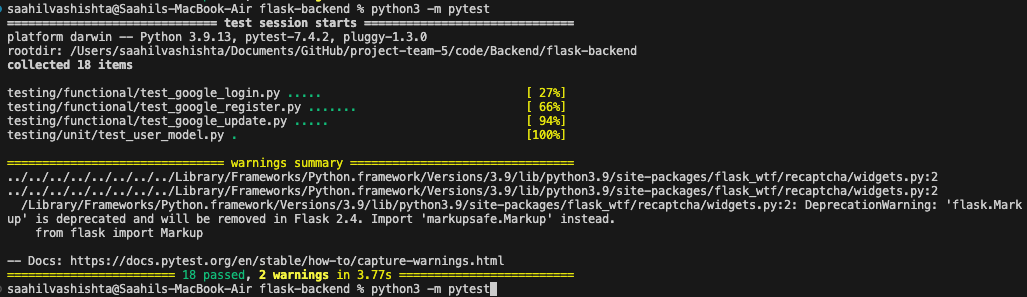
| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F-001 | Options Request for Google Registration | 1.App route api/register-google exists | 1.Test client (app.test\_client) | 1.Run python3 -m pytest | Status code - 200, message - Success | Status code - 200, message - Success | PASS |
| F-002 | Testing a pre registered user for registration | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2.google data  3.valid credential  4. Registered user token | 1.Run python3 -m pytest | Status code - 200, message - User validated successfully | Status code - 201, message - User validated successfully | PASS |
| F-003 | Testing an unregistered user for registration | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2.google data  3.valid credential  4. Unregistered user token | 1.Run python3 -m pytest | Status code - 201,  message - User added successfully | Status code - 201,  message - User added successfully | PASS |
| F-004 | Testing an invalid credential | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2.google data  3.Valid credential  4. registered user token | 1.Run python3 -m pytest | Status code - 500, message - Error validating user | Status code - 500, message - Error validating user | PASS |
| F-005 | Testing an invalid google data | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2.bad request format  3.Valid credential  4. registered user token | 1.Run python3 -m pytest | Status code - 500, message - Error validating user | Status code - 500, message - Error validating user | PASS |
| F-006 | Testing a long e-mail | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2.bad request format  3.Valid credential  4. Long email user token | 1.Run python3 -m pytest | Status code - 500, message - Error validating user | Status code - 500, message - Error validating user | PASS |
| F-007 | Testing unsupported request method | 1.App route api/register-google exists | 1.Test client (app.test\_client)  2. Use of .get method input | 1.Run python3 -m pytest | Status code - 405, ‘method not allowed’ | Status code - 405, ‘method not allowed’ | PASS |

1. User Login for Google (Using Pytest)

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F-008 | Options Request for Google Login | 1.App route api/login-google exists | 1.Test client (app.test\_client) | 1.Run python3 -m pytest | Status code - 200, message - Success | Status code - 200, message - Success | PASS |
| F-009 | Testing a pre registered user for login | 1.App route api/login-google exists | 1.Test client (app.test\_client)  2.google data  3.valid credential  4. Registered user token | 1.Run python3 -m pytest | Status code - 201, message - User validated successfully | Status code - 201, message - User validated successfully | PASS |
| F-010 | Testing an unregistered user for login | 1.App route api/login-google exists | 1.Test client (app.test\_client)  2.google data  3.valid credential  4. Unregistered user token | 1.Run python3 -m pytest | Status code - 404, message - Please register user first | Status code - 404, message - Please register user first | PASS |
| F-011 | Testing an invalid request | 1.App route api/login-google exists | 1.Test client (app.test\_client)  2.bad request format  3.invalid credential  4. registered user token | 1.Run python3 -m pytest | Status code - 500, message - Error validating user | Status code - 500, message - Error validating user | PASS |
| F-012 | Testing unsupported request method | 1.App route api/login-google exists | 1.Test client (app.test\_client)  2. Use of .get method input | 1.Run python3 -m pytest | Status code - 405, ‘method not allowed’ | Status code - 405, ‘method not allowed’ | PASS |

1. User Update for Google (Using Pytest)

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F-013 | Options Request for Google Update | 1.App route api/update-google exists | 1.Test client (app.test\_client) | 1.Run python3 -m pytest | Status code - 200, message - Success | Status code - 200, message - Success | PASS |
| F-014 | Update, change user type for registered user | 1.App route api/update-google exists | 1.Test client (app.test\_client)  2. Valid email  3. Type: host | 1.Run python3 -m pytest | Status code - 201, message - User updated successfully | Status code - 201, message - User updated successfully | PASS |
| F-015 | Update, change user type for unregistered user | 1.App route api/update-google exists | 1.Test client (app.test\_client)  2. Invalid email  3. Type: host | 1.Run python3 -m pytest | Status code - 404, message - User not found | Status code - 404, message - User not found | PASS |
| F-016 | Update with invalid user type | 1.App route api/update-google exists | 1.Test client (app.test\_client)  2. Invalid email  3. Type:invalid user type | 1.Run python3 -m pytest | Status code - 500, message - Error validating user | Status code - 500, message - Error validating user | PASS |
| F-018 | Testing unsupported request method | 1.App route api/update-google exists | 1.Test client (app.test\_client)  2. Use of .get method input | 1.Run python3 -m pytest | Status code - 405, ‘method not allowed’ | Status code - 405, ‘method not allowed’ | PASS |



*Functional Tests- screenshot- User Registration, Login and Update*

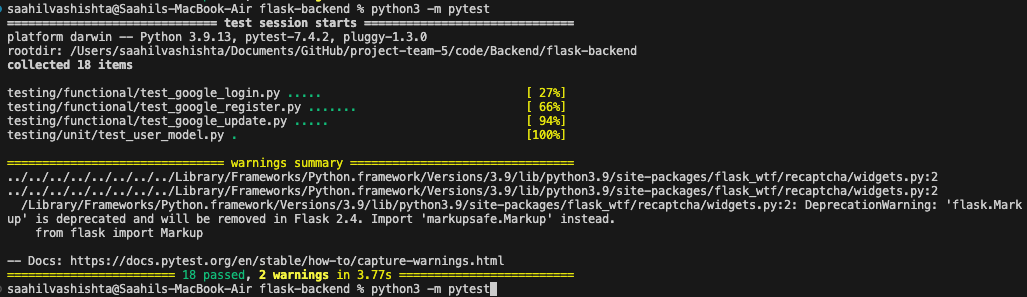
**

*Functional Tests- screenshot- sample code*

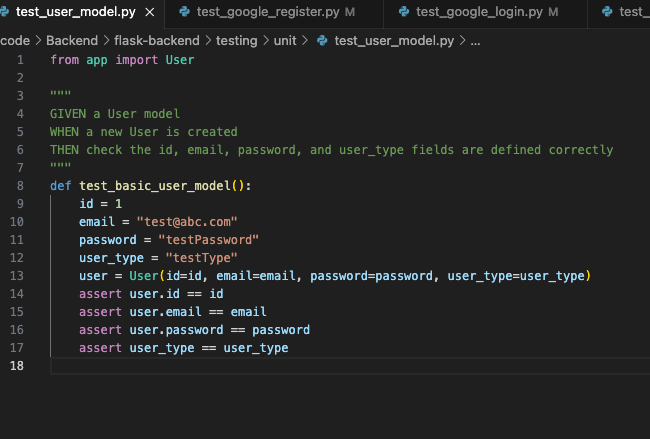
# Unit Test

# In this section, we will provide a detailed description of unit tests (if conducted) for features that require it.

| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| U-001 | Testing user database model | 1.db with SQLALchemy exists | 1.id  2.email  3.password  4.user type | 1.Run python3 -m pytest | passed | passed | PASS |



*Unit Tests- screenshot- database table*

**

*Unit Tests- screenshot- code snippet*

# Testing Metrics

| DB Tables | Database table structures that include the necessary attributes for each intended table. |
| --- | --- |
|  | * Equipment: Name, Description, Status, Price, Owner, Available * User: email, password * Payment: Itemid, is\_paid, price * Reservation: start\_date, end\_date, item\_id, user\_name * Reviews: id,target\_username,origin\_username,name, rating, description |
| API Connectivity | This metric is used to ensure that the API’s created in the backend are functioning and thoroughly tested for each use case and user story made |
|  | * We implemented the API connectivity as we needed the endpoints. All the endpoints are tested and used for the features |
| Code Coverage | How much of the code can be tested to ensure that bugs do not persist within the production of the software |
|  | * Each Feature was manually tested and displayed in this document |
| Code Duplication | Metric to measure the use of redundant code in the project. |
|  | * Multiple versions to grab the equipment as flask endpoint |
| Test Pass Rate | This metric would be used to measure the amount of successful passing test cases while testing the Database as well as the functionality of the front end UI |
|  | * User Registration had a 100% test pass rate * User Login had a 100% test pass rate * Search Reservation Availability had a 90% pass rate * Make Reservation had a 100% pass rate. * Hosting Equipment had a 100% pass rate. * Checkout had a 100% pass rate. |

In this section, you shall report any metrics used for the evaluation, e.g. # of test cases, test coverage, defects rate, etc.

# References

# Glossary