**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. Checkout Process |

[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 |  |  |  |  |  |  |  |

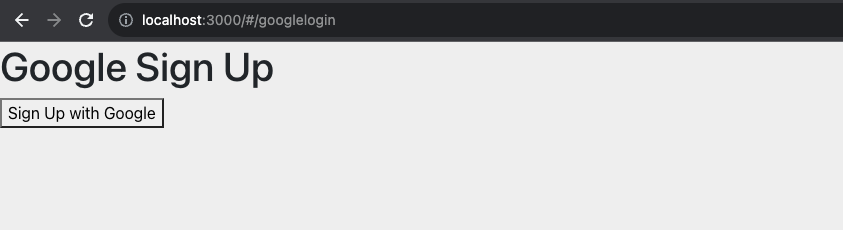


Fig1: Google Sign-up on local host

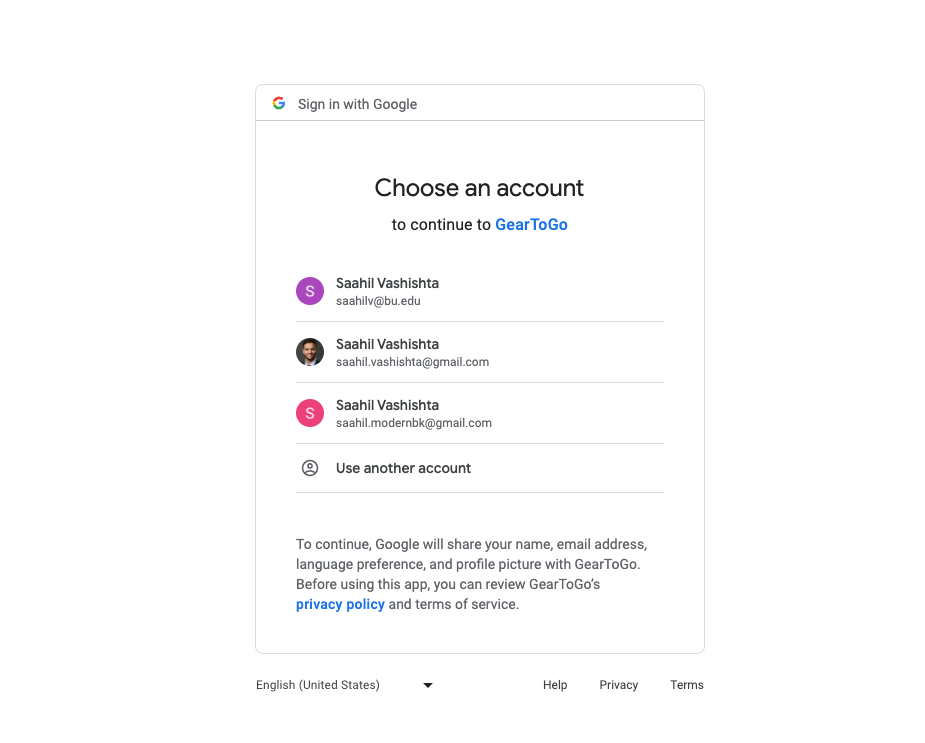
****

Fig2: Gmail account options

****

Fig3: Successful login with session information and logout option

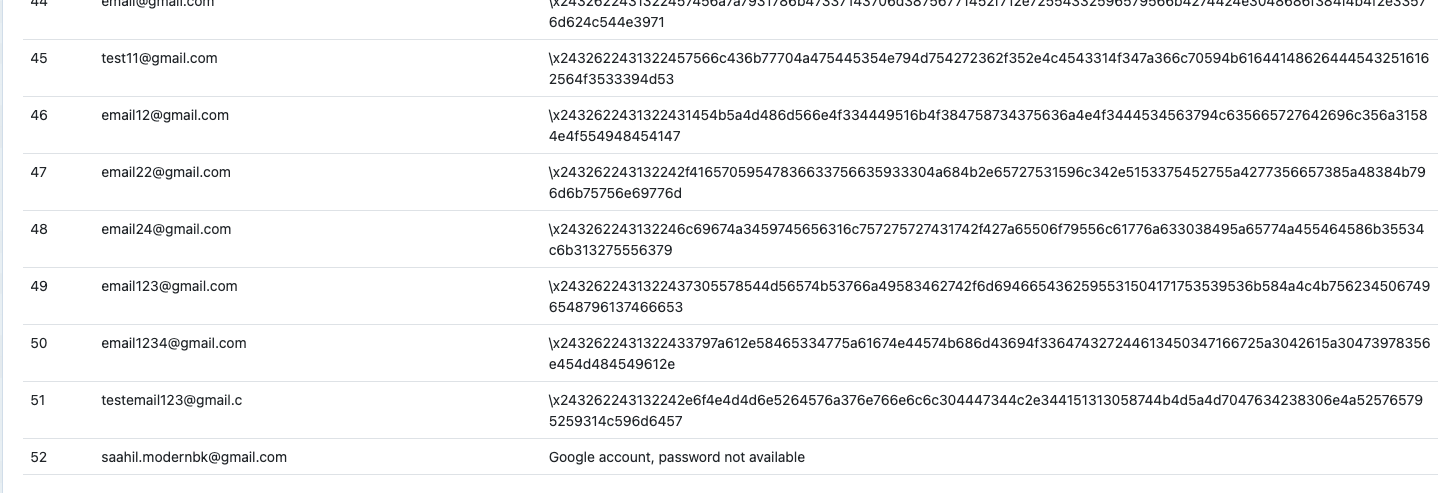


Fig4: Entry of google email chosen to database on Elephant SQL

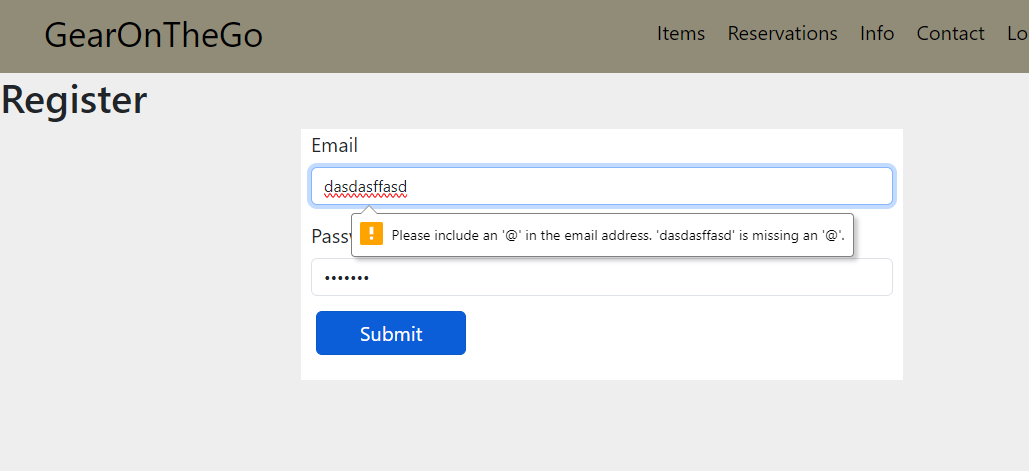


Fig6: Register page on React site. Incorrect email format entered warning

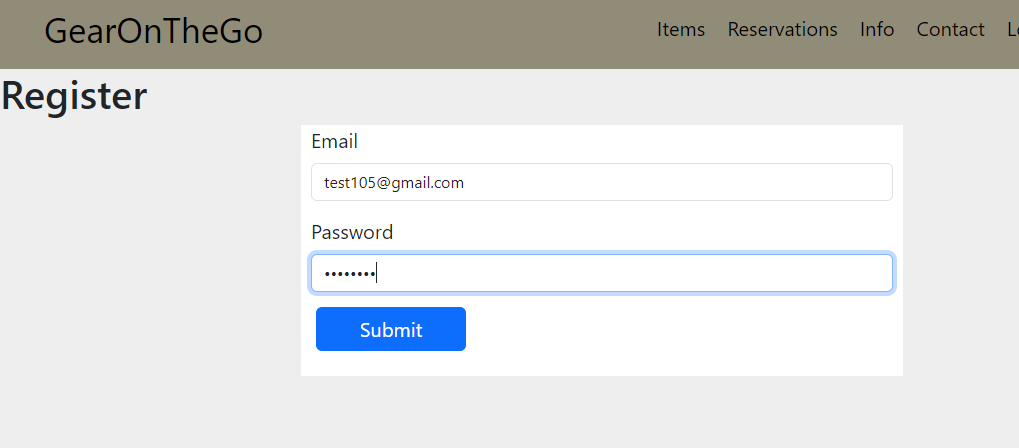


Fig7: Entering appropriate credentials

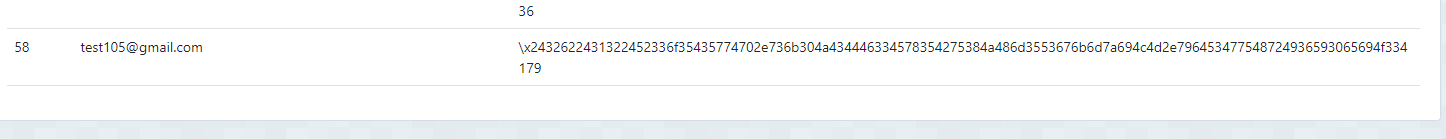


Fig8: User credentials stored in database with hashed password

**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 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 Login page and enter 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 |

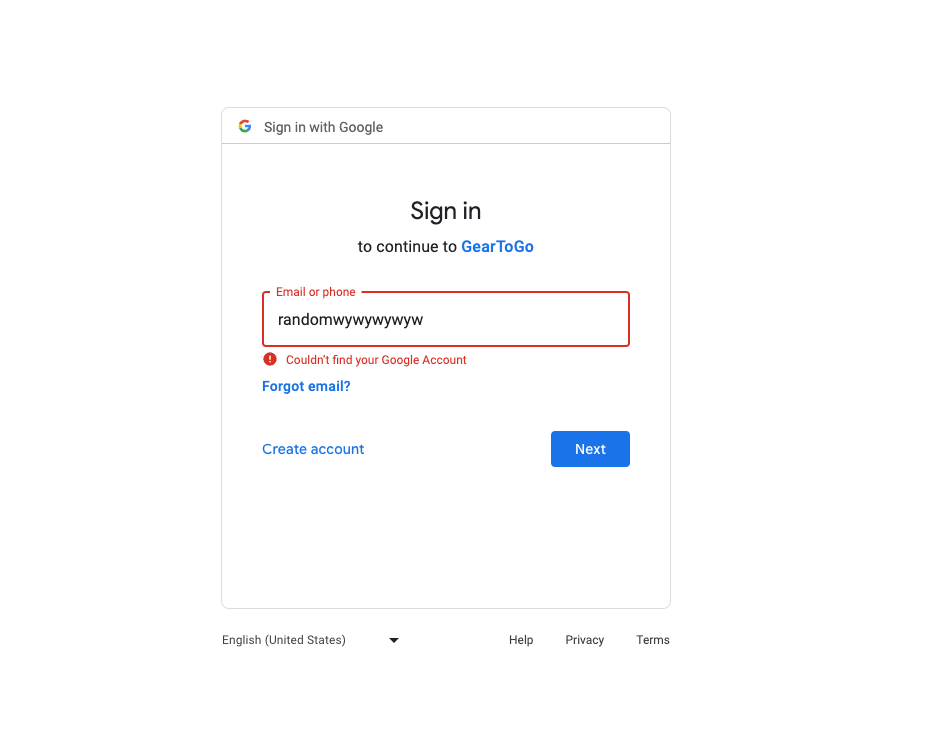


Fig8: Incorrect email error (Google sign-in)

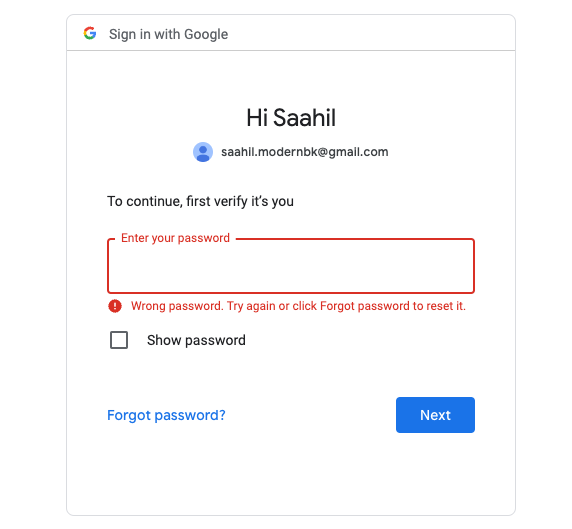


Fig9: Incorrect Password sign-in Google

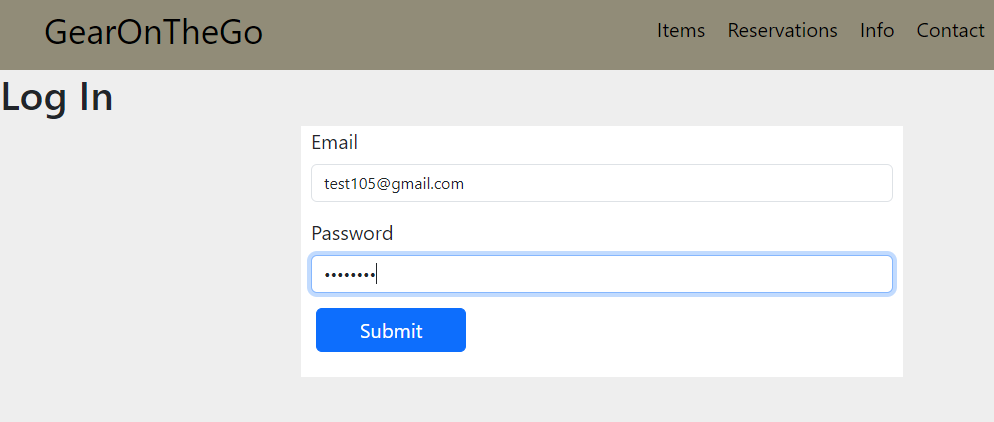


Fig10: Logging in with valid credentials

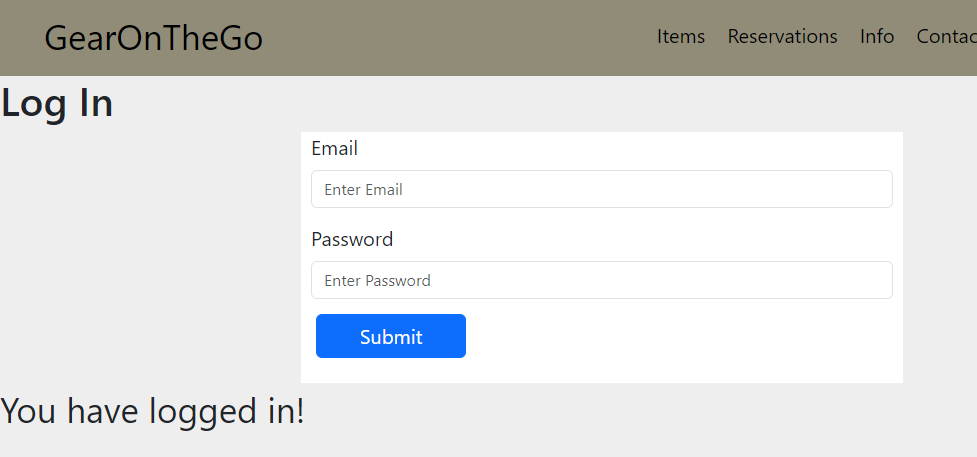
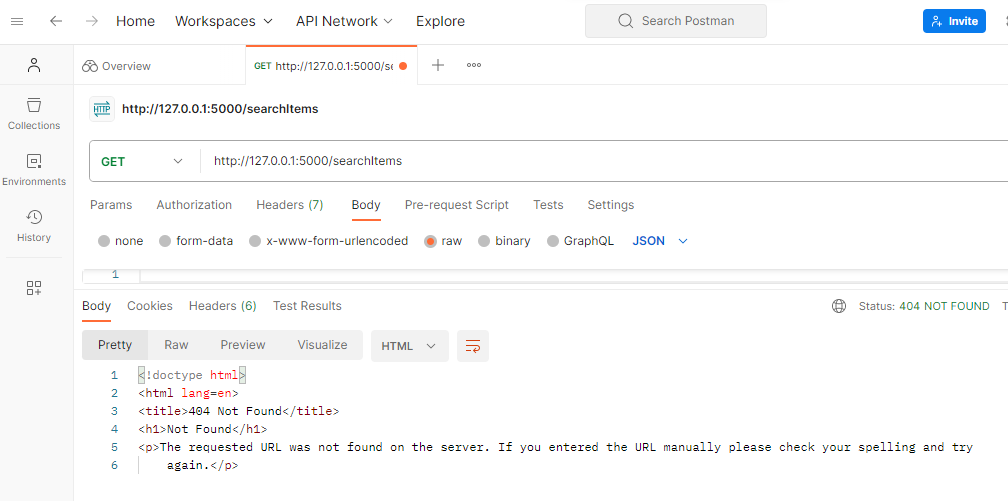
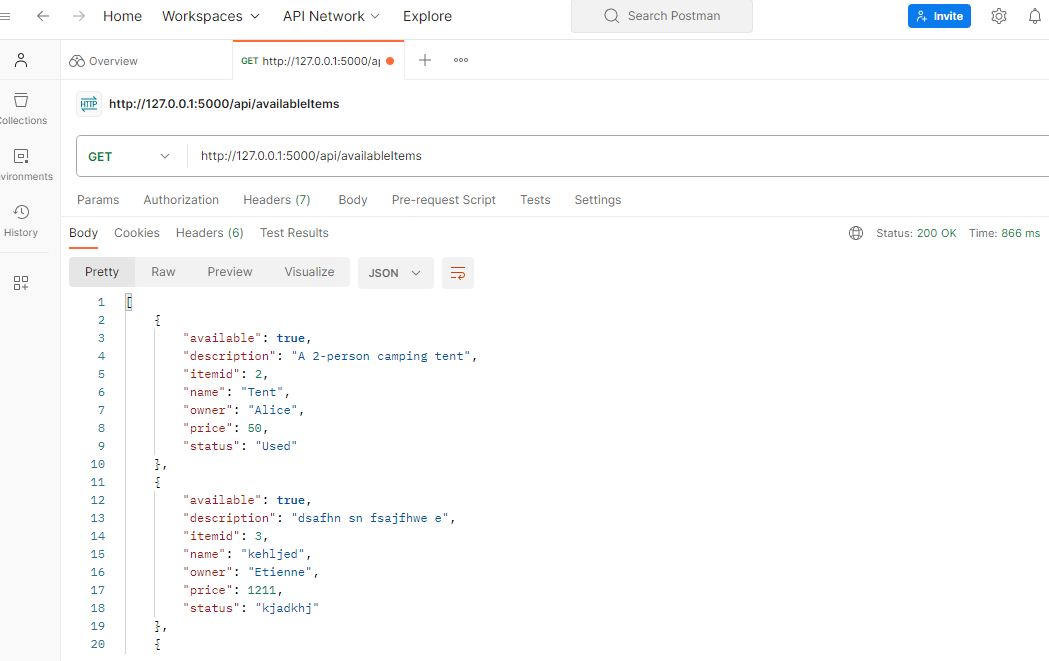


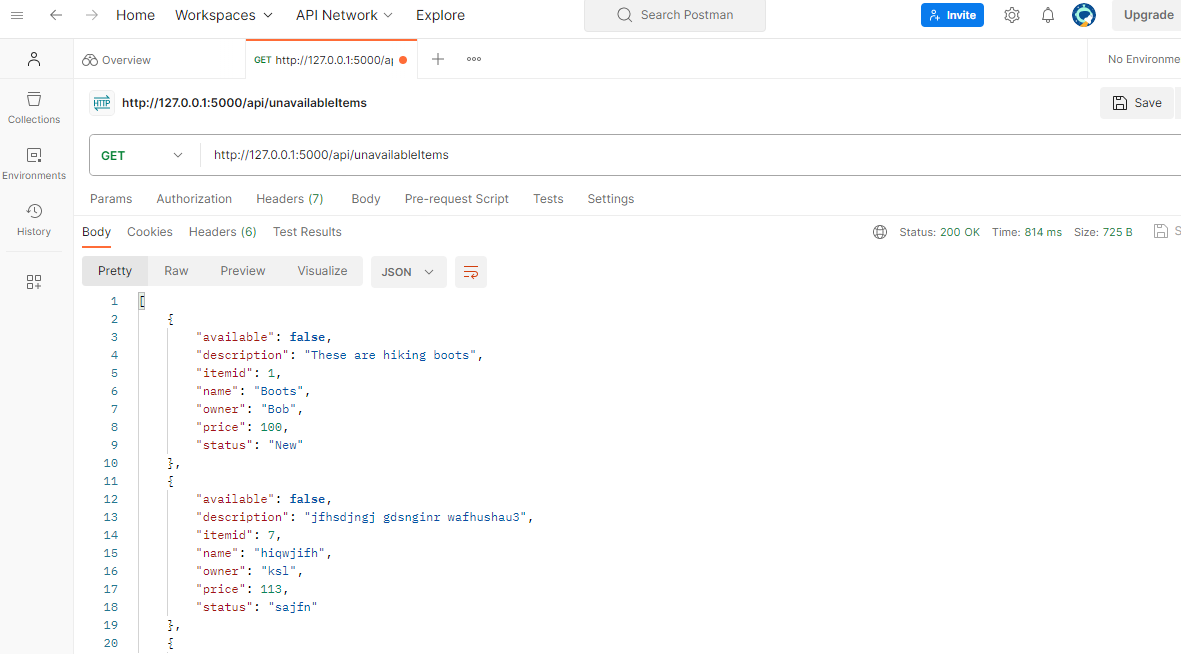
Fig11: Displays logged in message

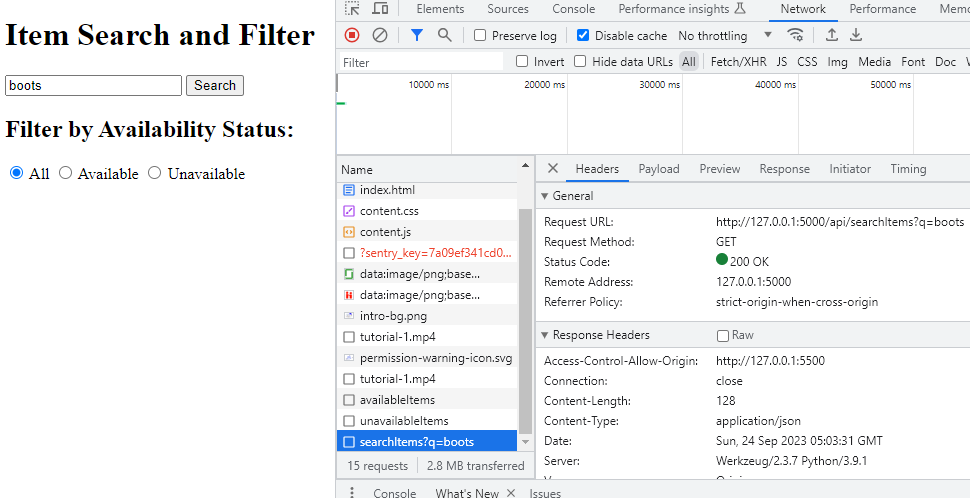
**Search Reservation Availability:**

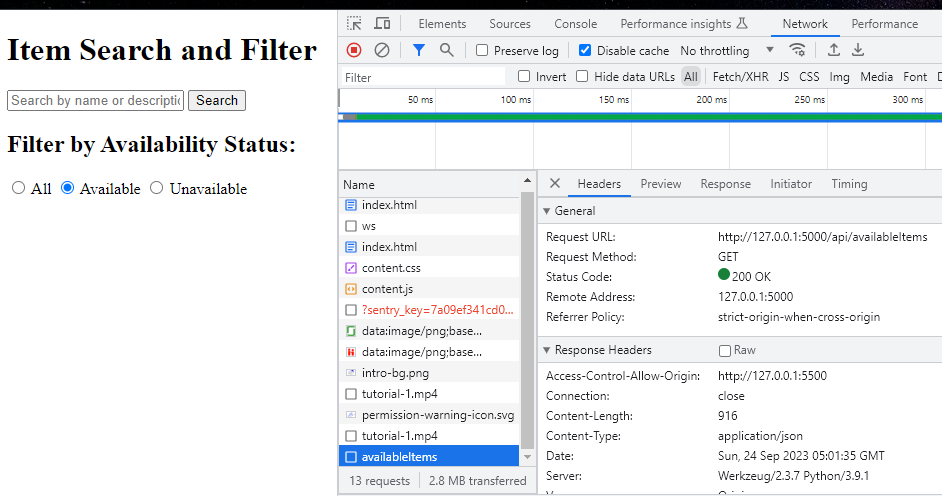
| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0013 | Search Items API Testing in Postman | Database in ElephantSQL attached with Pyscopg2 | Items that are listed in the Item tables | 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 |

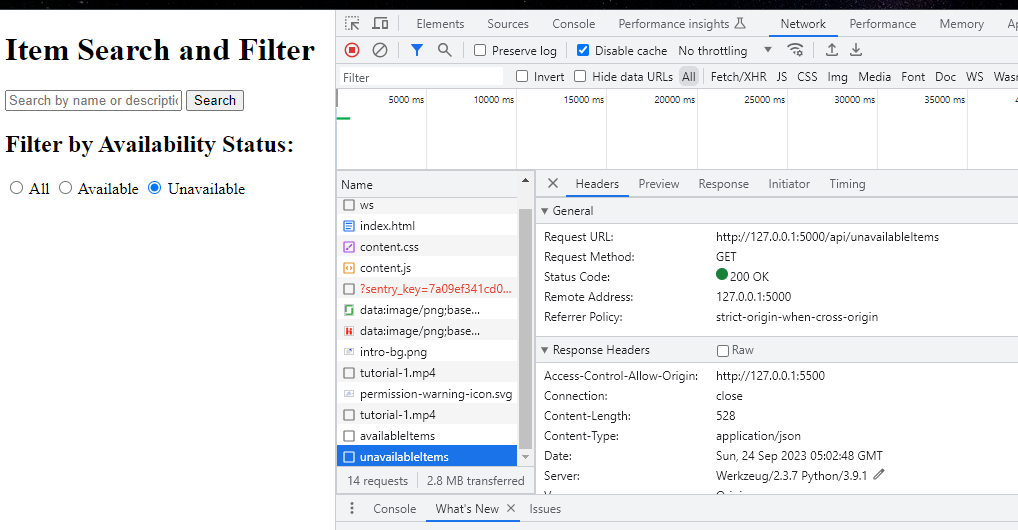
****

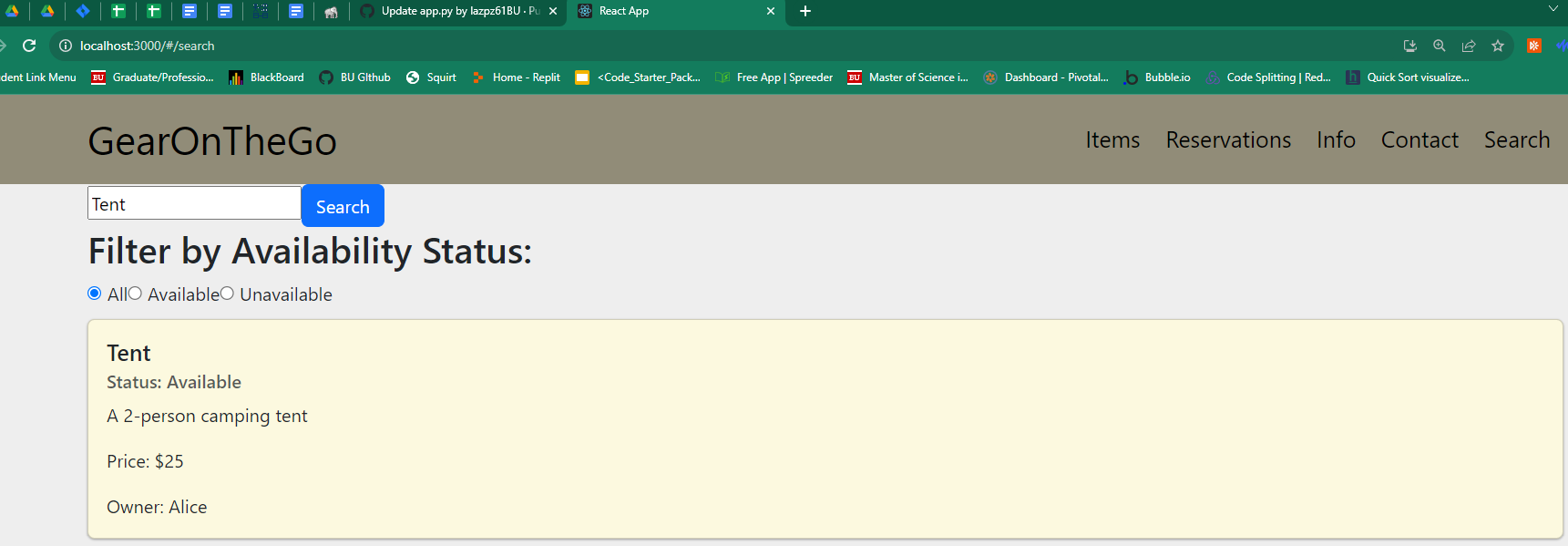
****

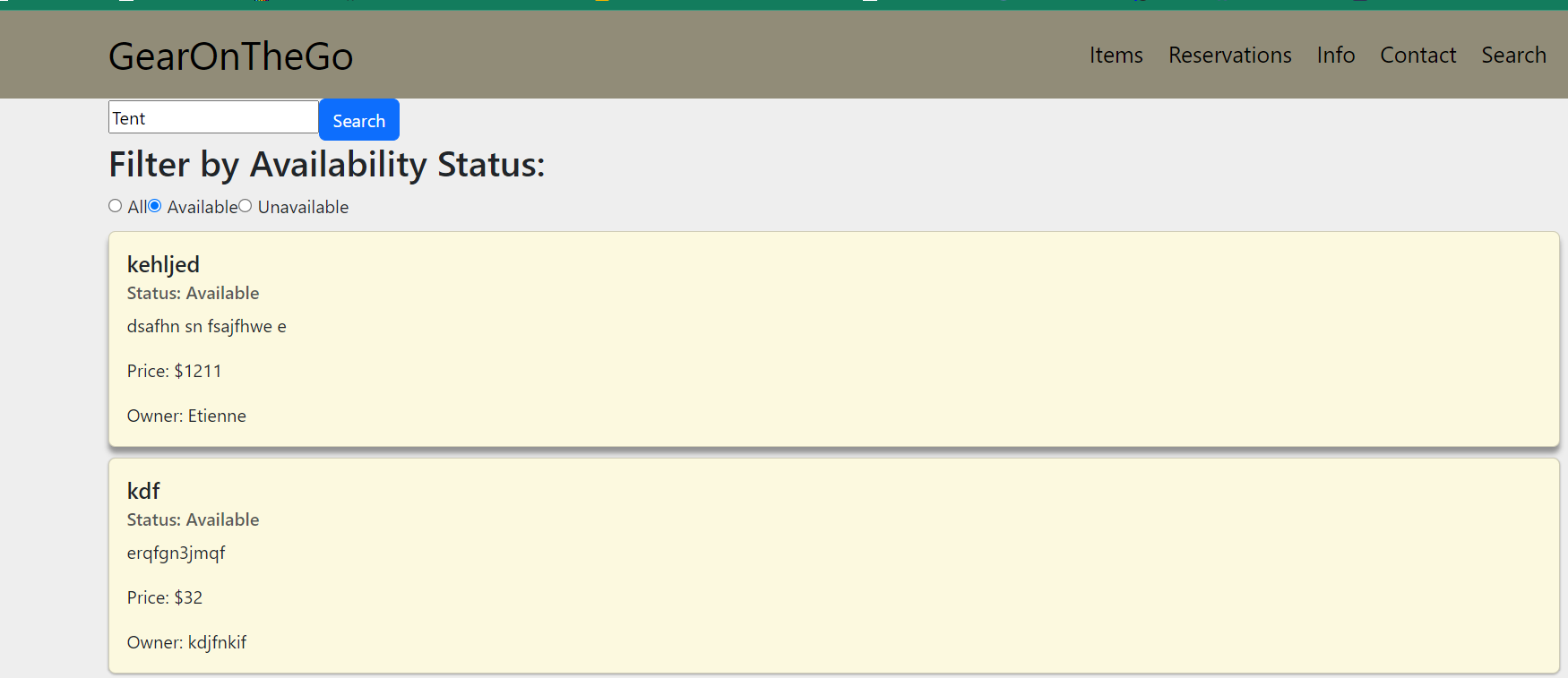
****

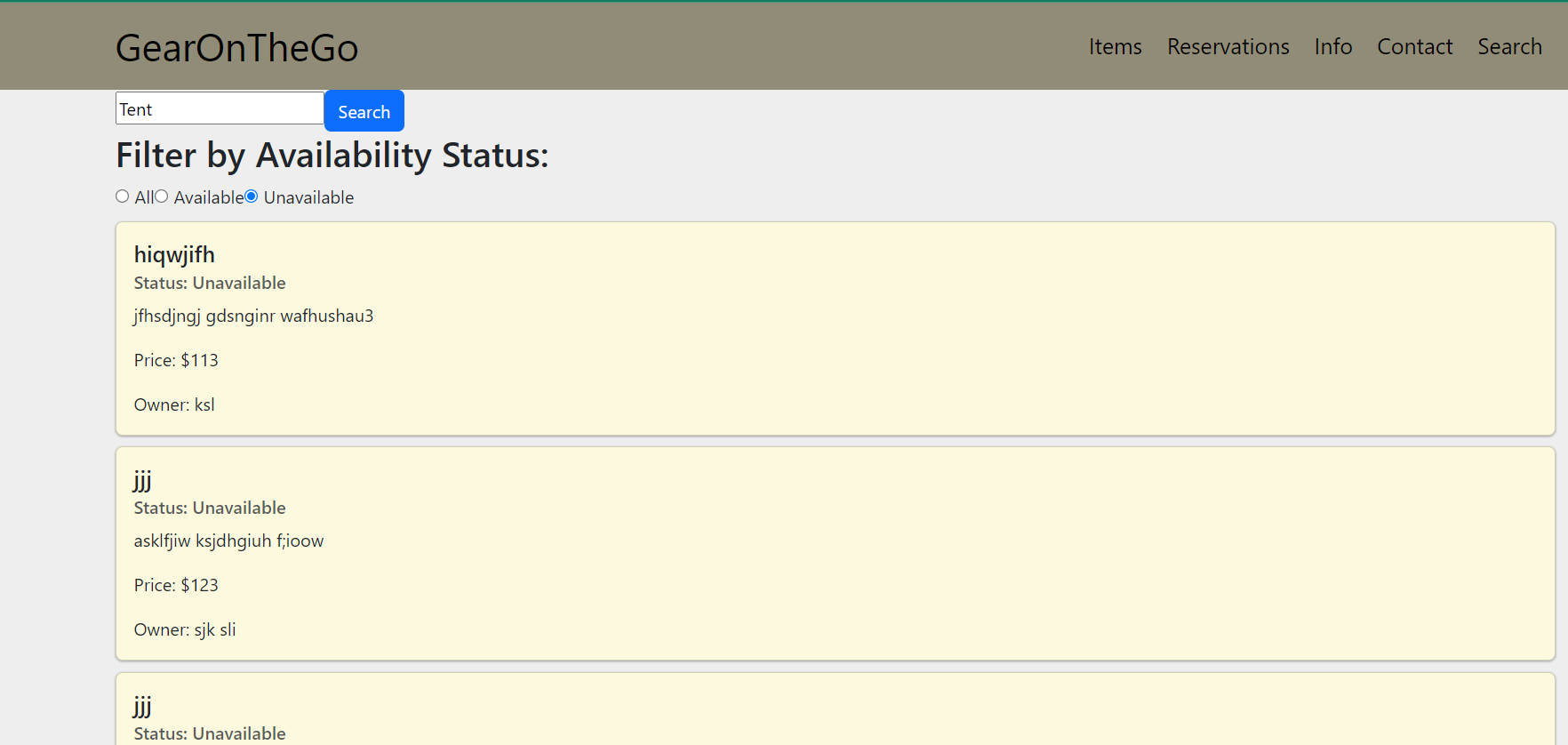
****

****

****

****

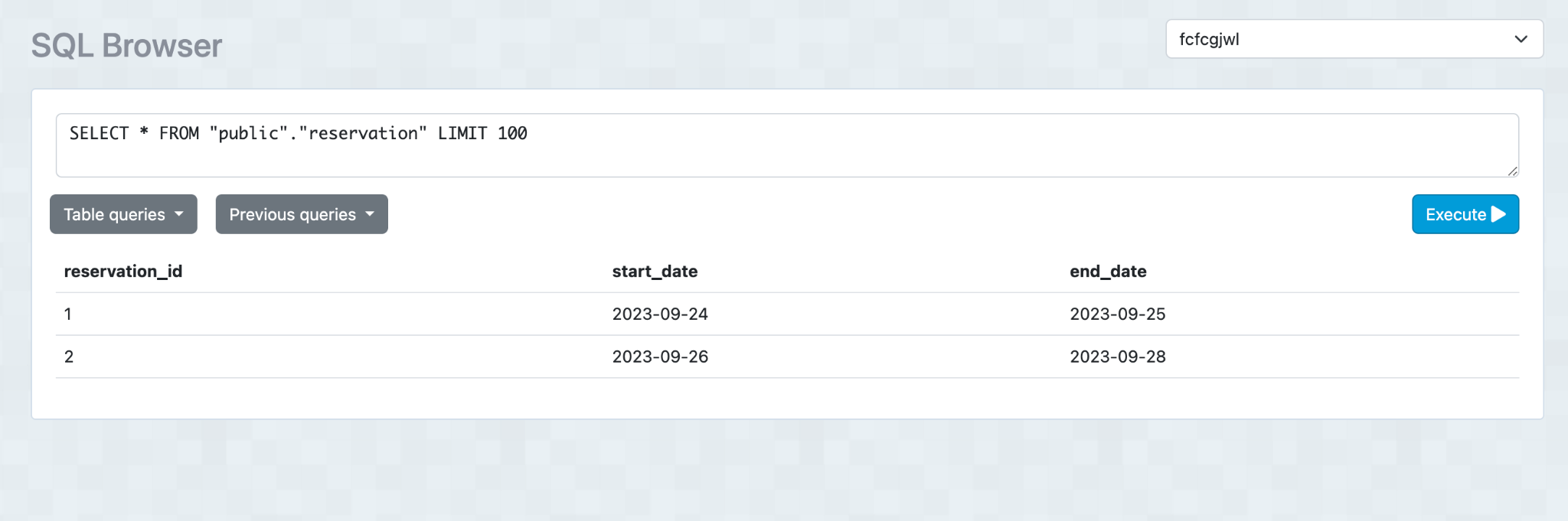
****

****

**Make Reservation**

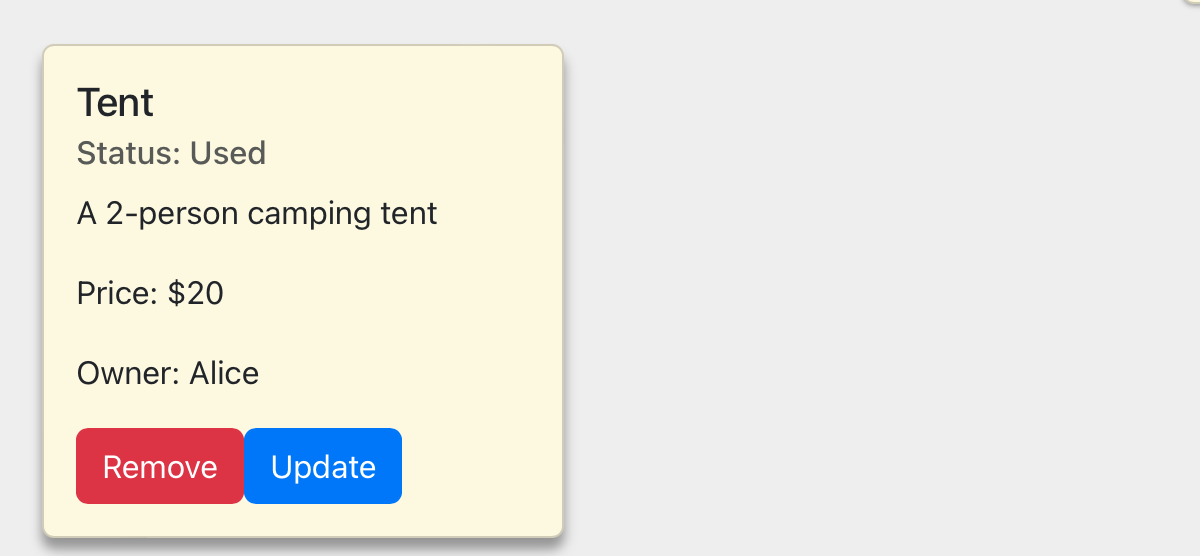
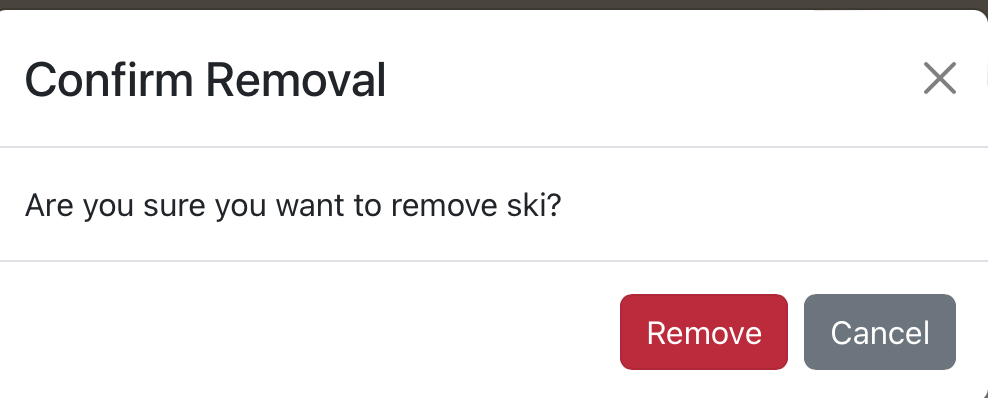
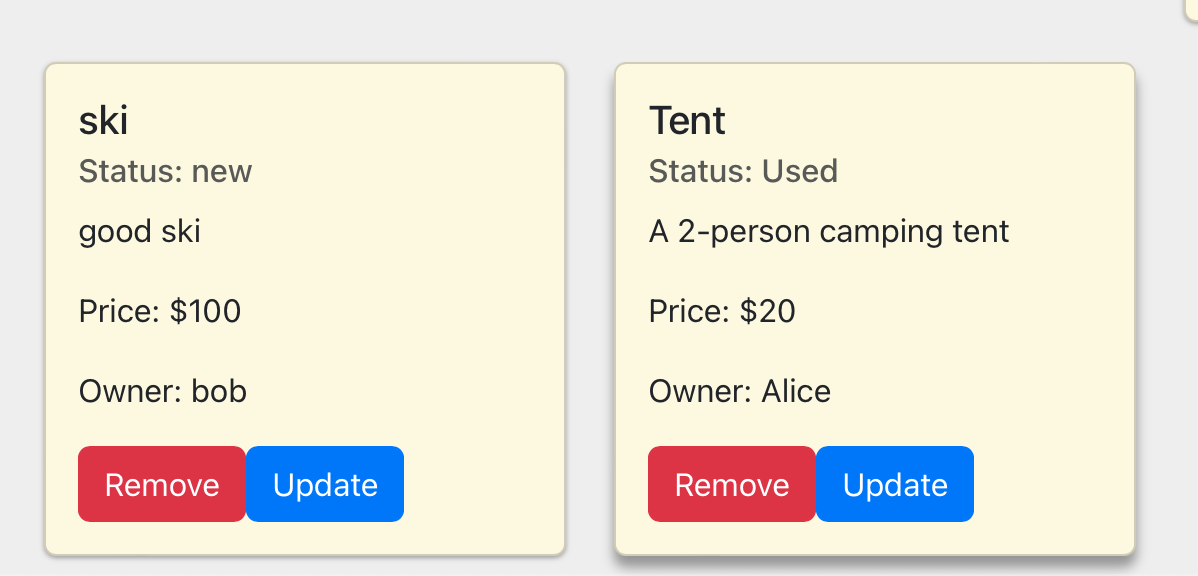
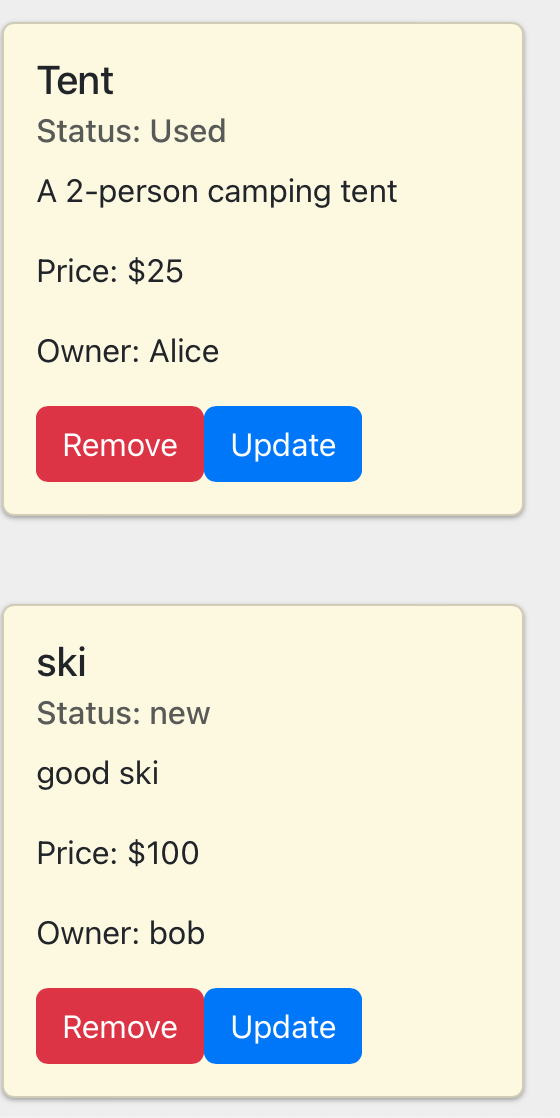
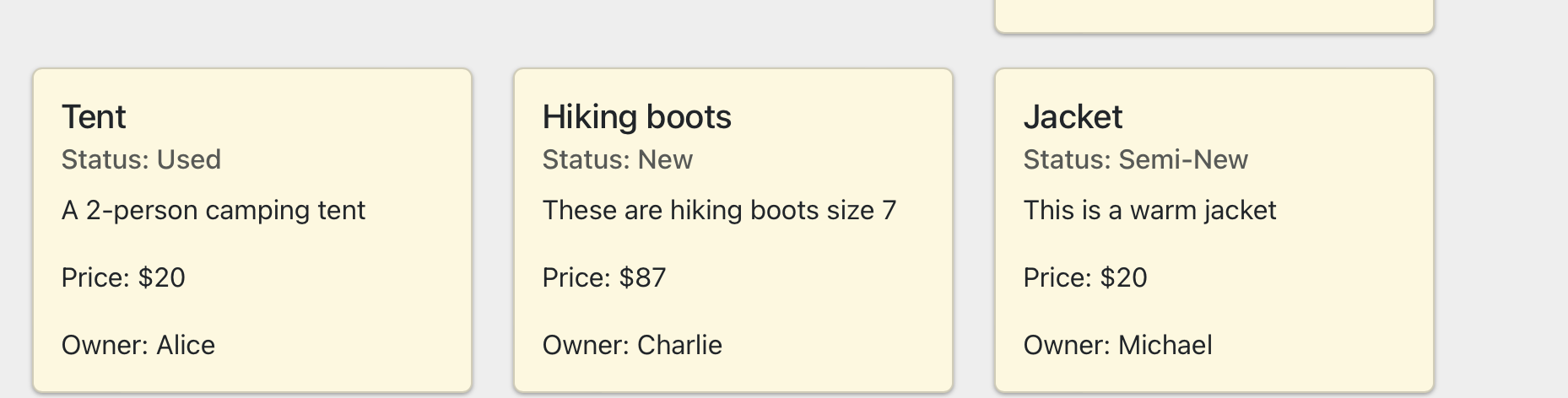
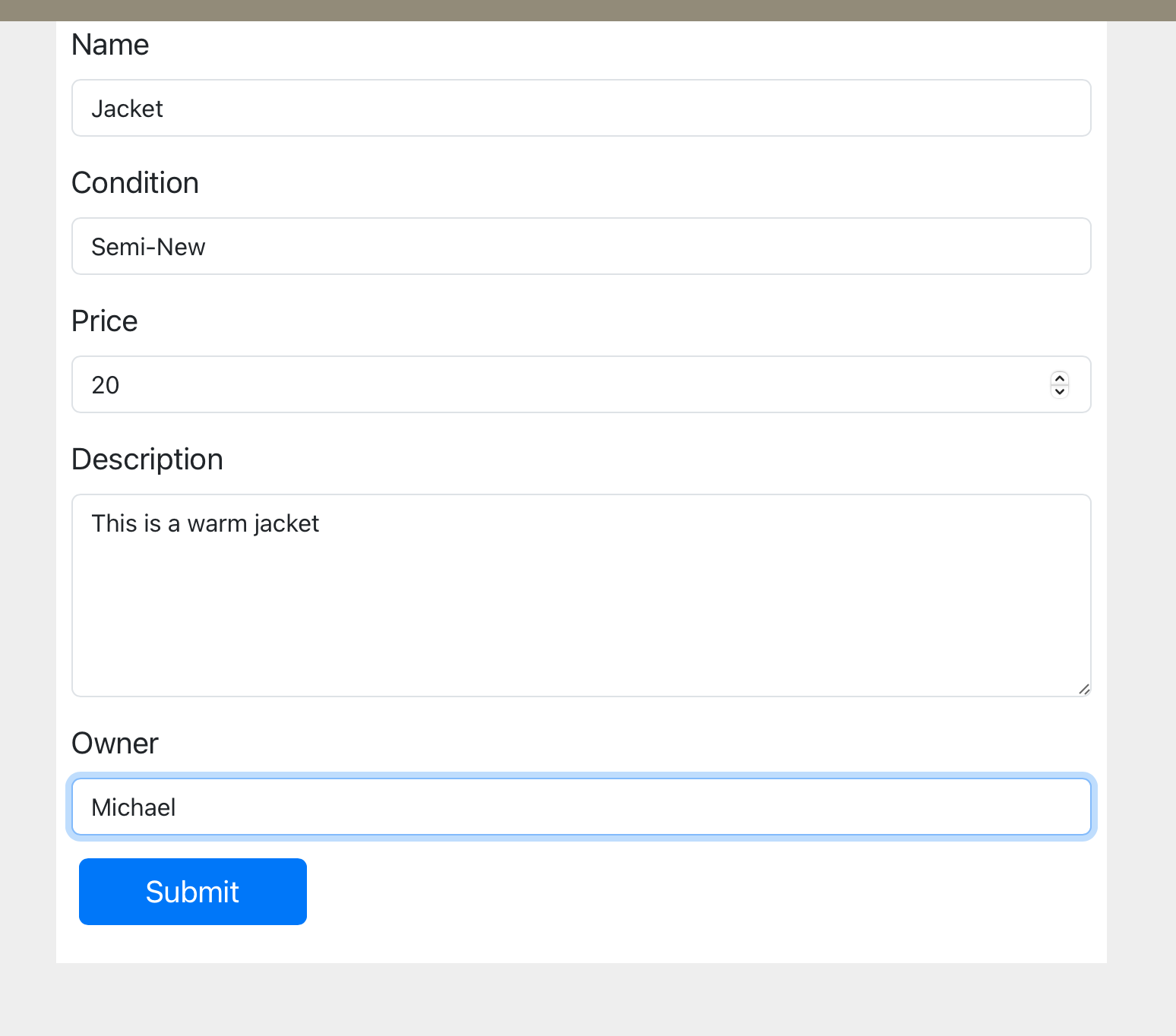
| Test ID | Test Description | Dependencies | Input Data | Test Steps | Expected result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0019 | Save reservation start date and end date to database | Database in ElephantSQL connected | Start date, end date | Go to Reservation page, Enter start date, Enter end date, Click Submit | Confirmation message display on the page; Entered start date and end date is saved to database | Confirmation message display on the page; Entered start date and end date is saved to database | 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 |

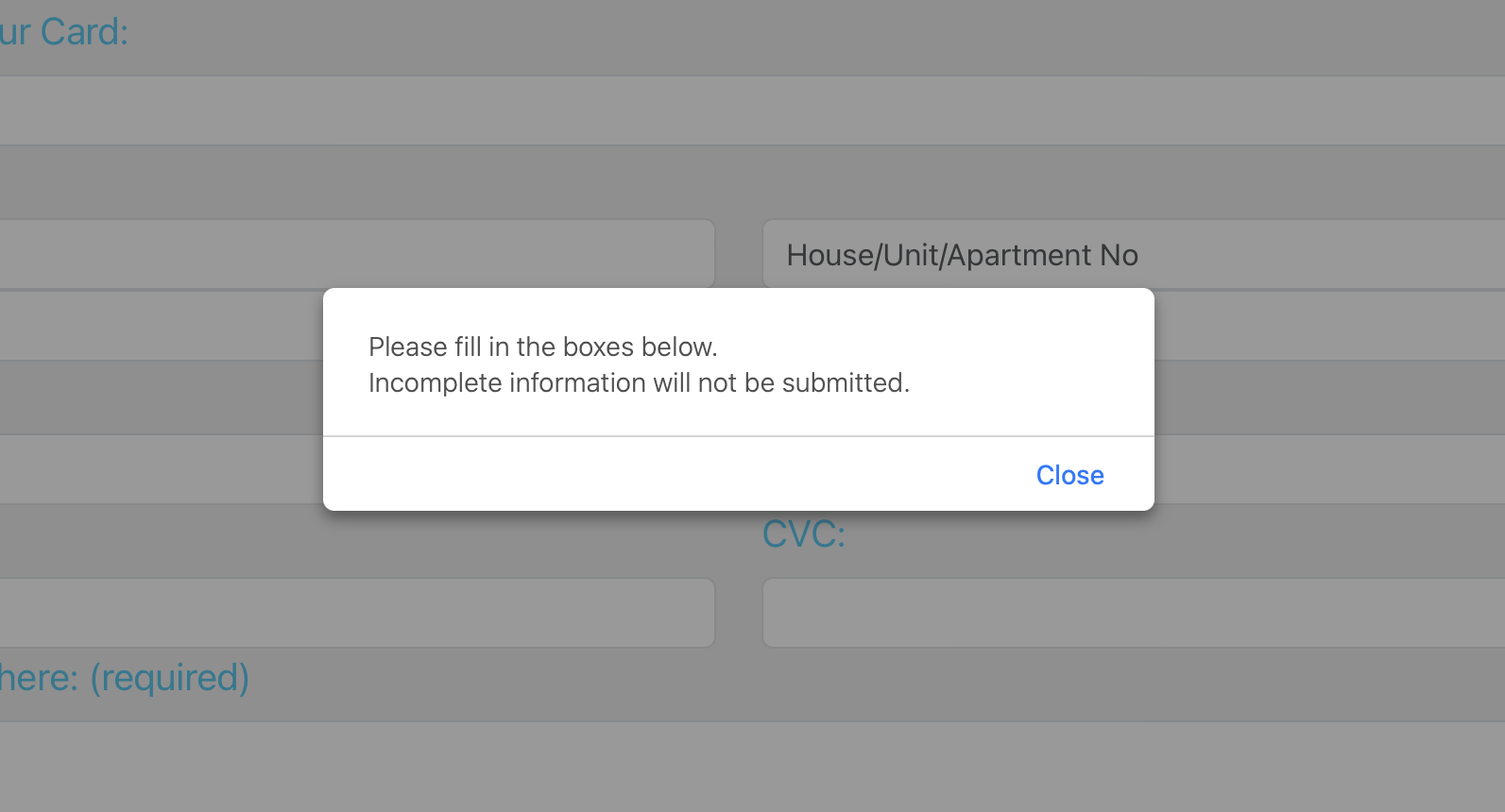
****

**Checkout Process Testing**

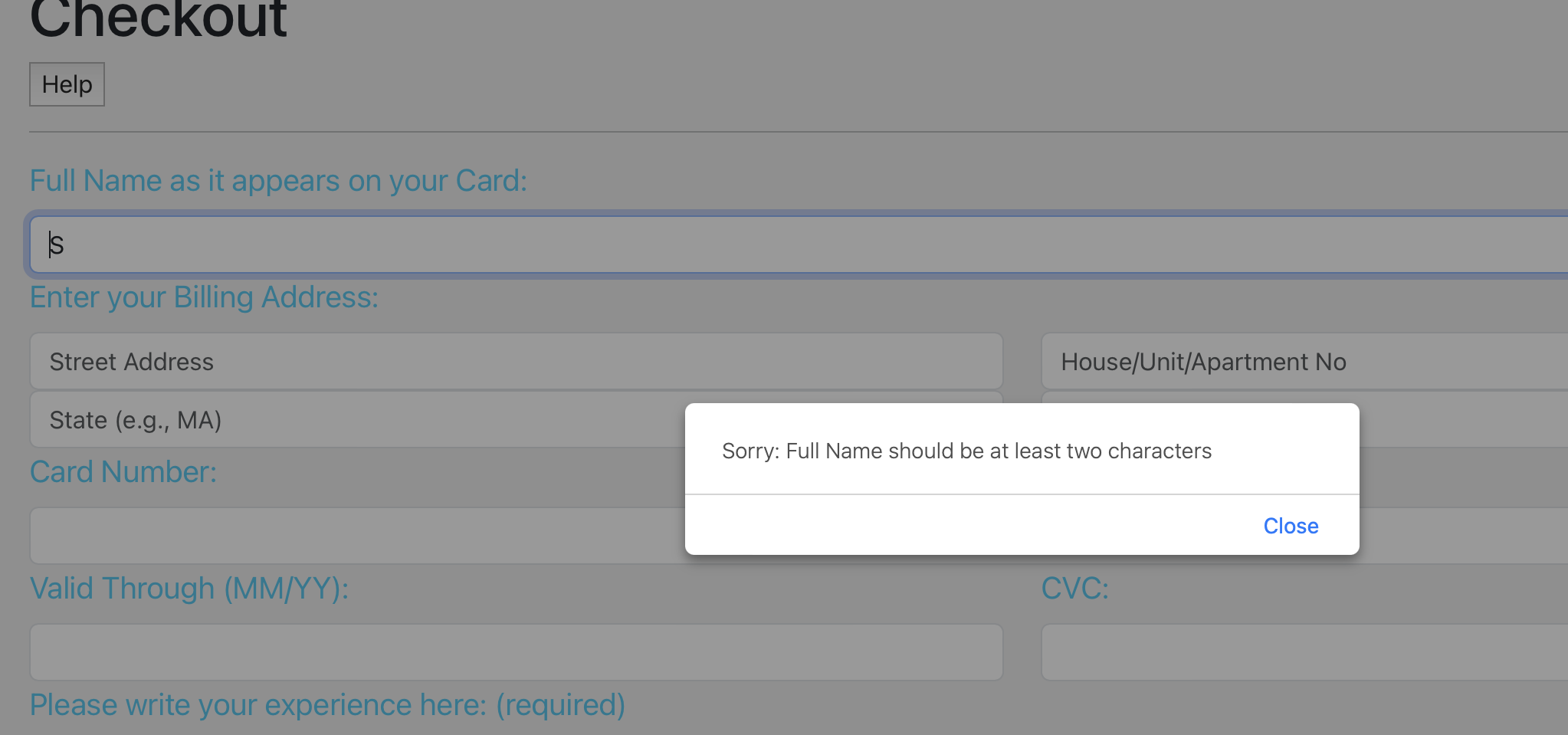
The Checkout Process is reliant on the Renting User’s Name on Card, Billing Address, Card Information etc. For now while, it is not processing a payment - this will be implemented in a later iteration - it is still essential to have a base template Checkout page where the Checkout process does not allow the payment to be submitted until the Renting User has inputted the appropriate information and has inputted 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:

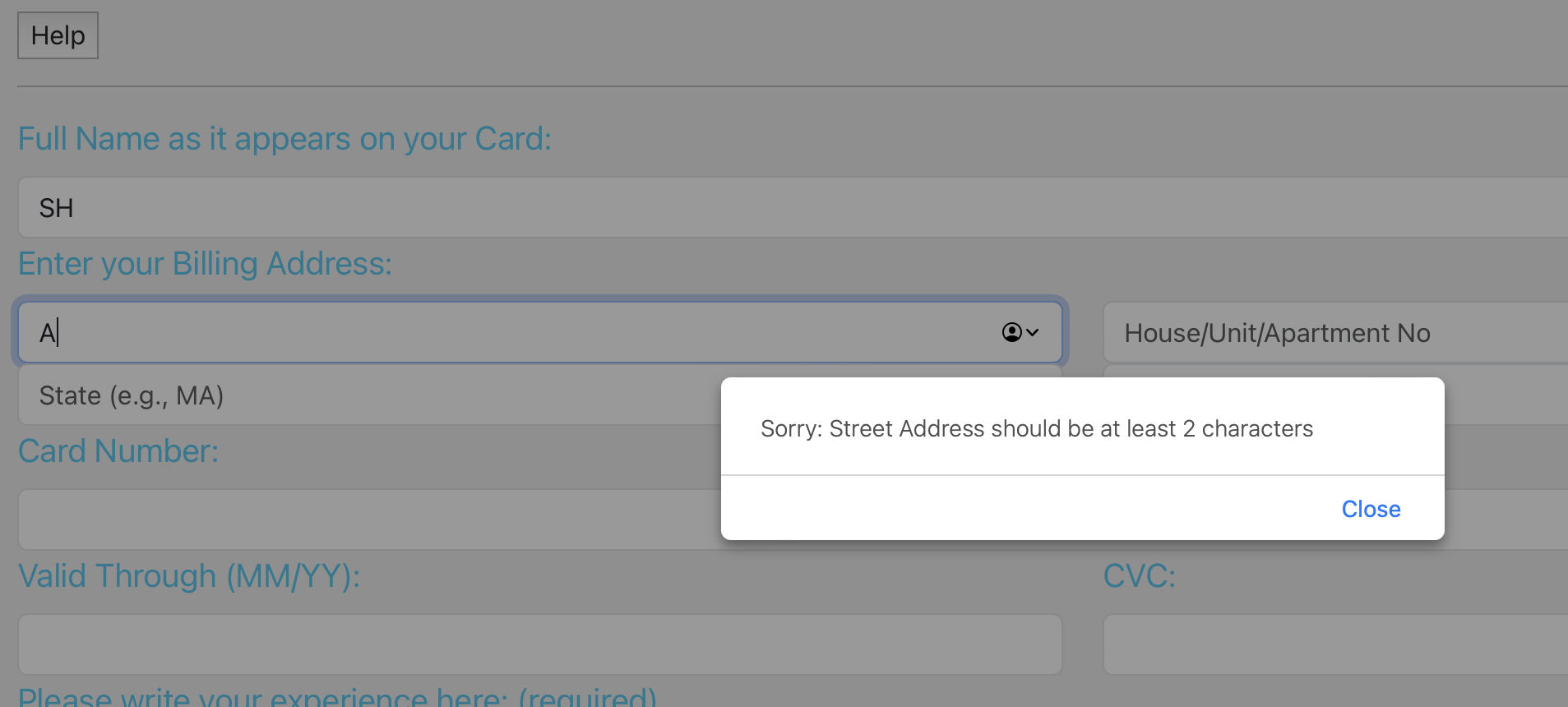
**Testing the help button:**

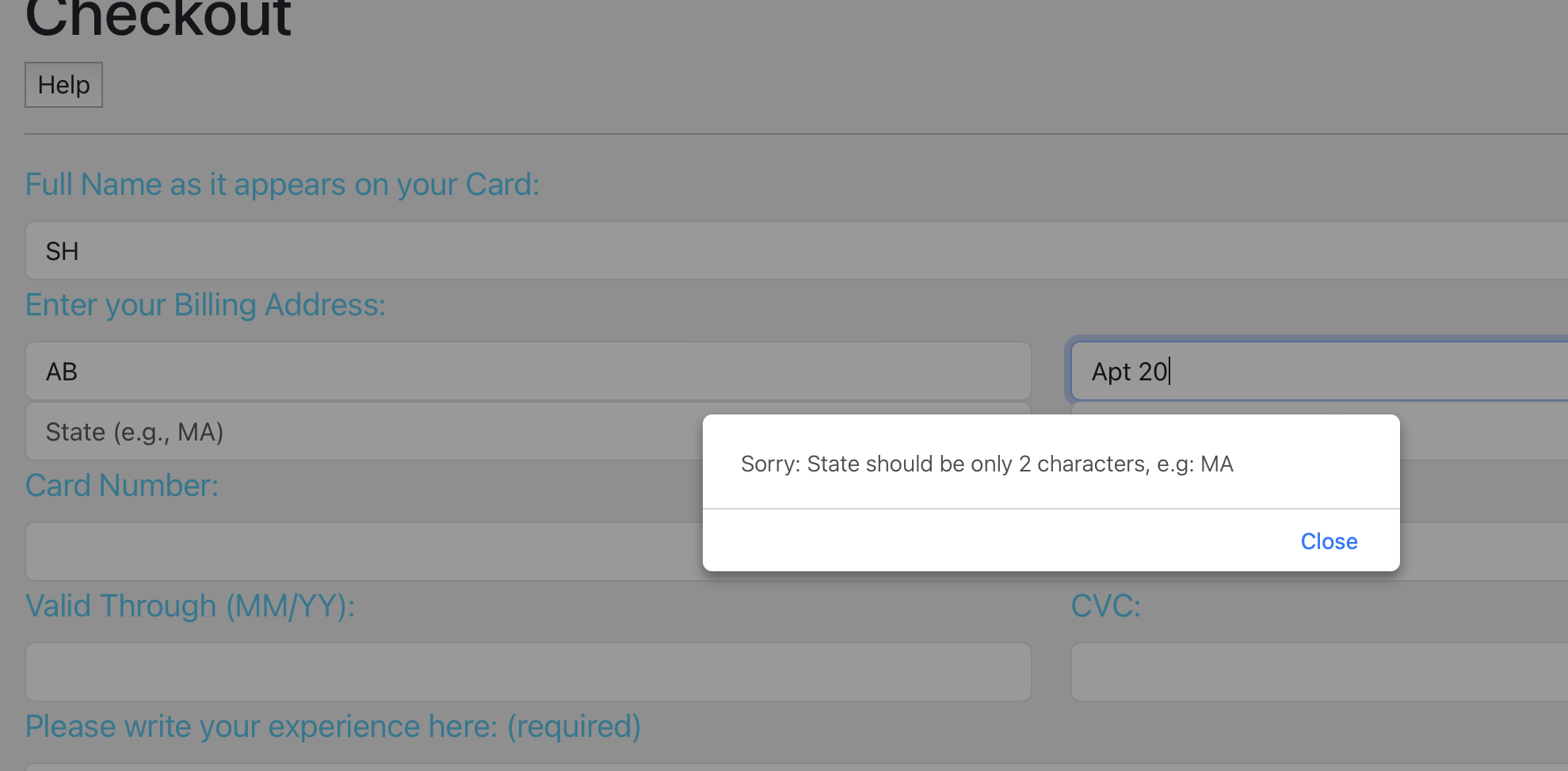
****

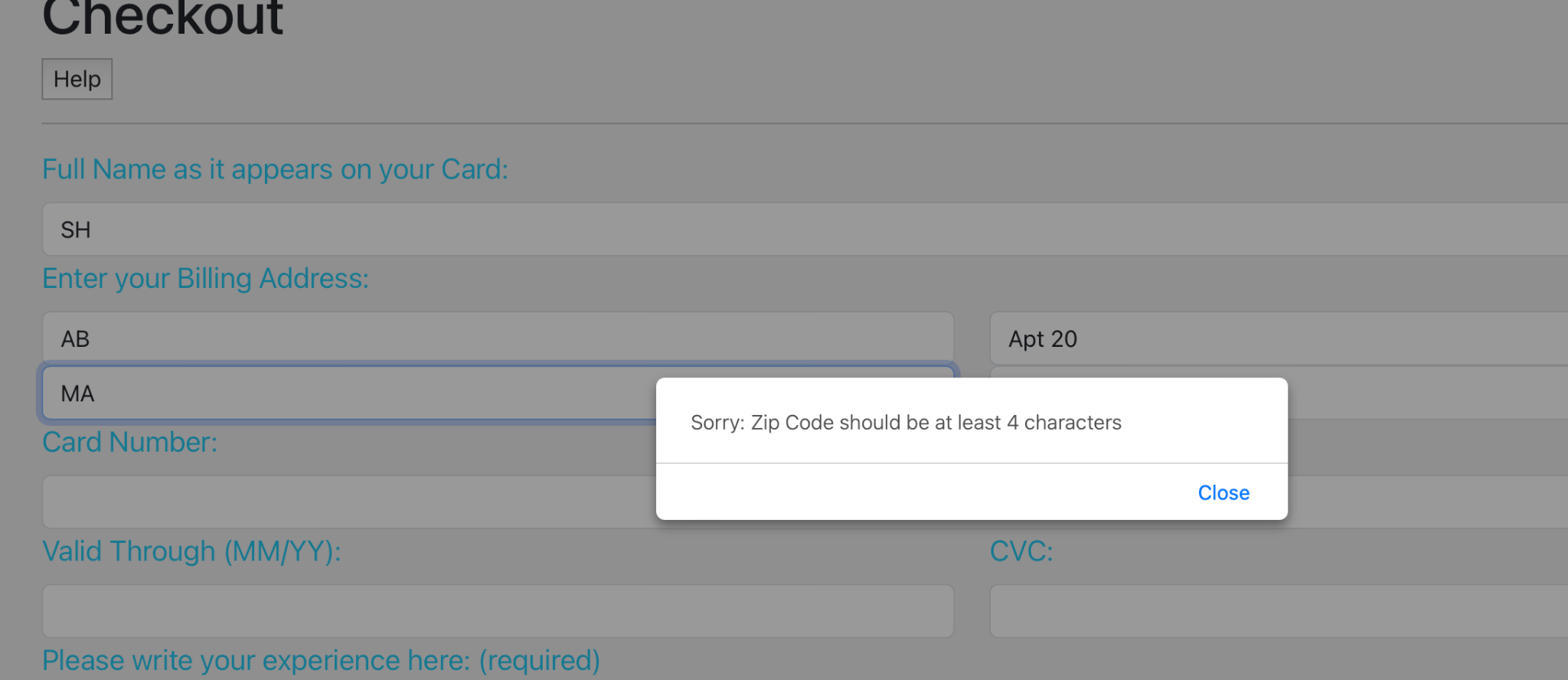
**Testing the Full Name on Card field:**

****

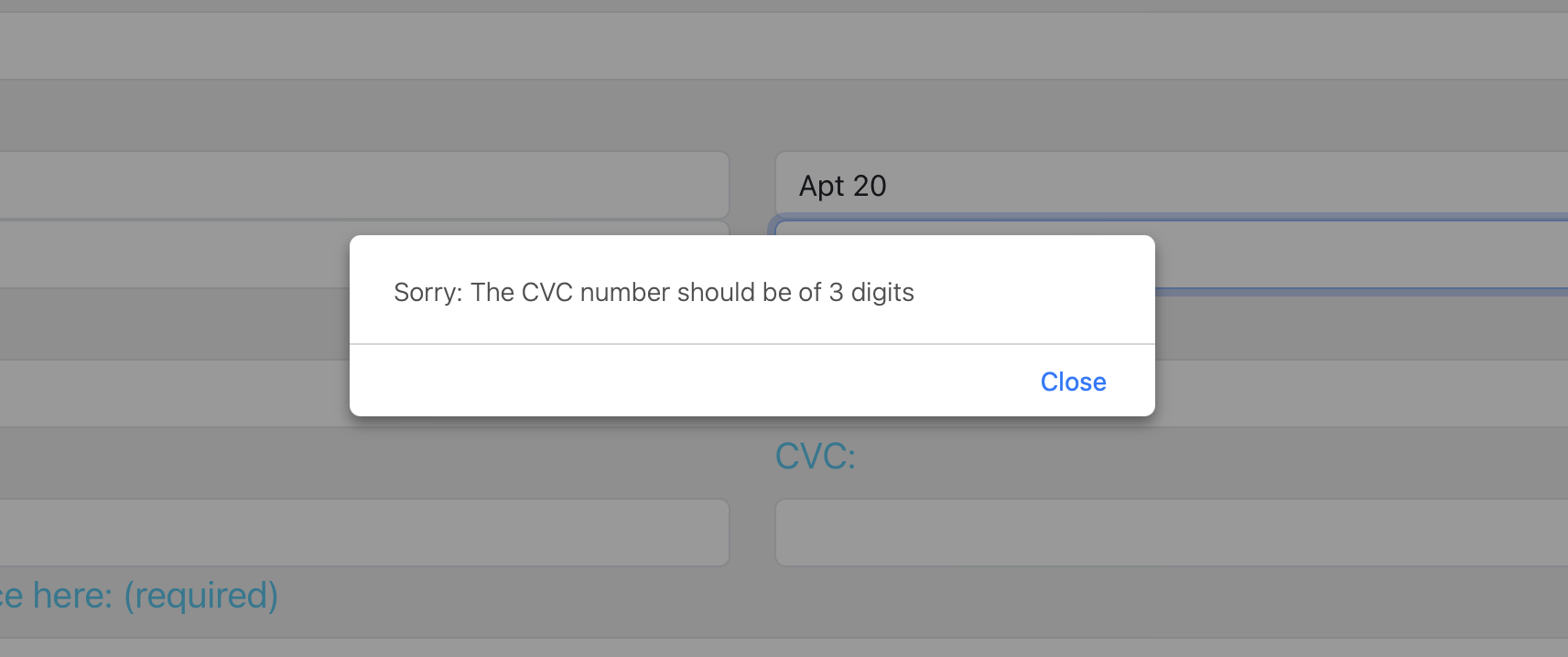
**Testing the different Billing Address fields:**

****

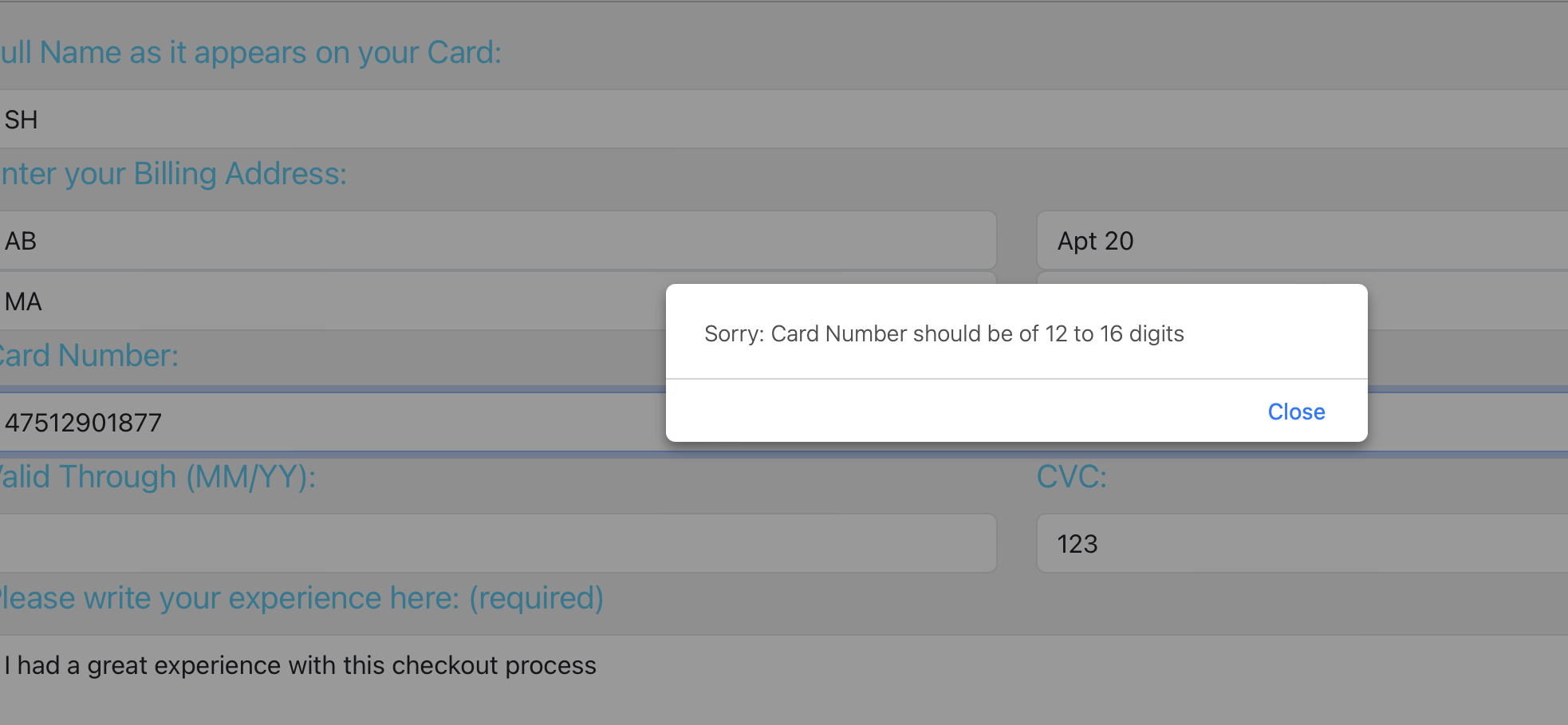
****

****

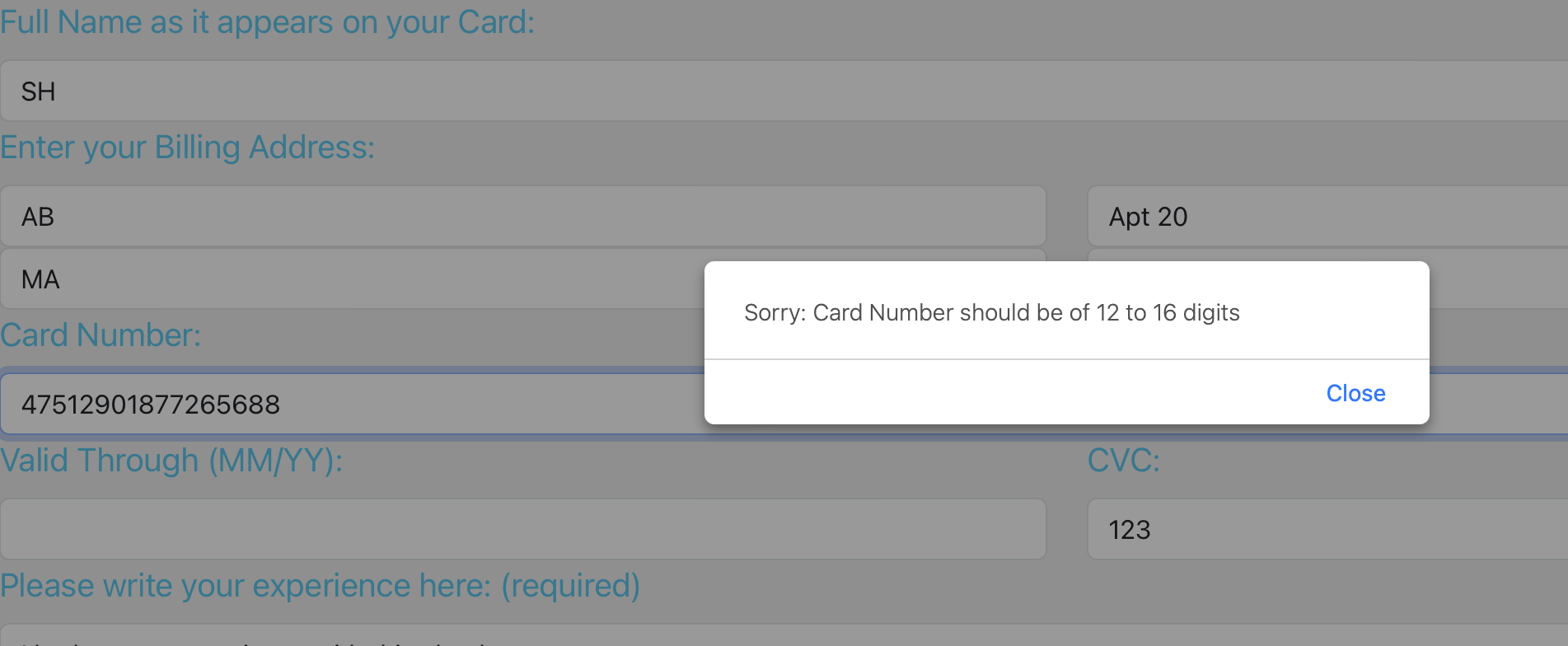
**Testing the Card Information fields:**

****

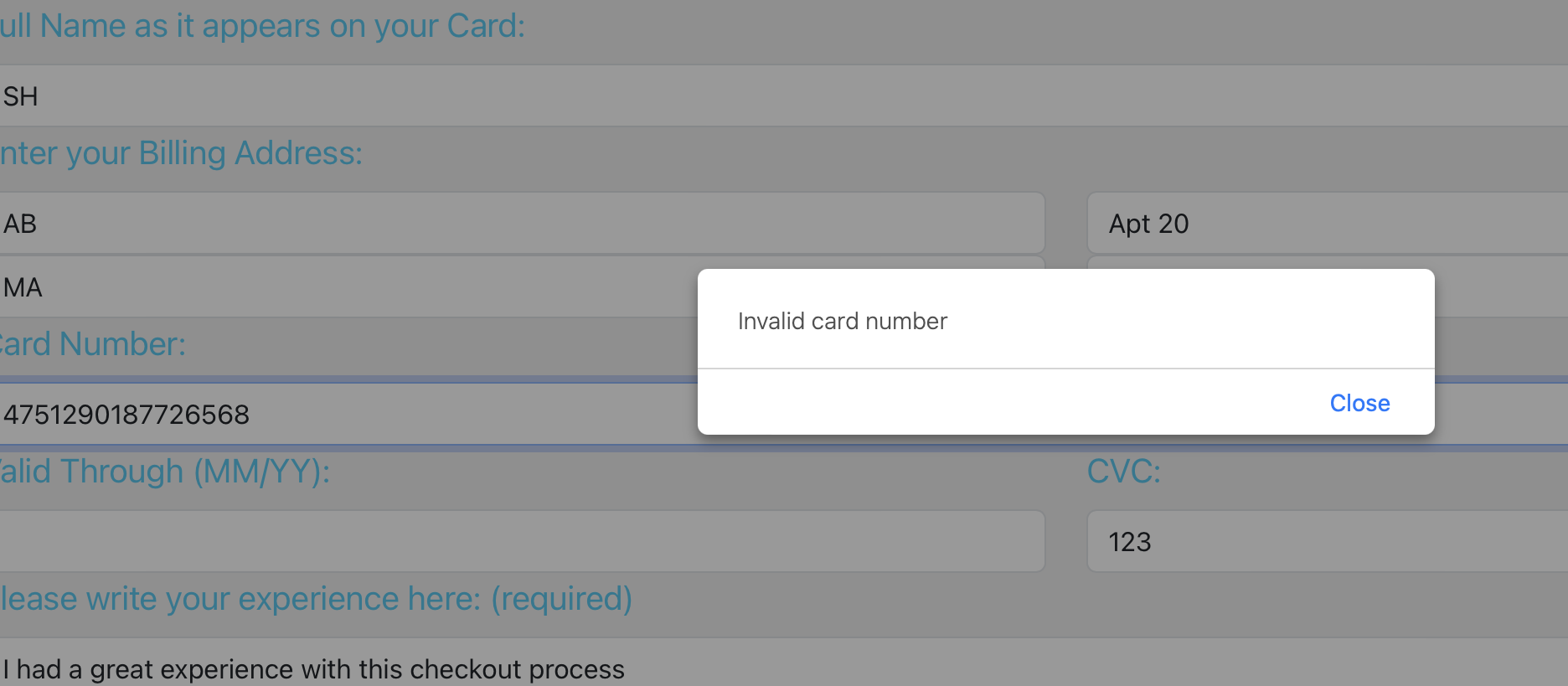
**Using less than 12 digits:**

****

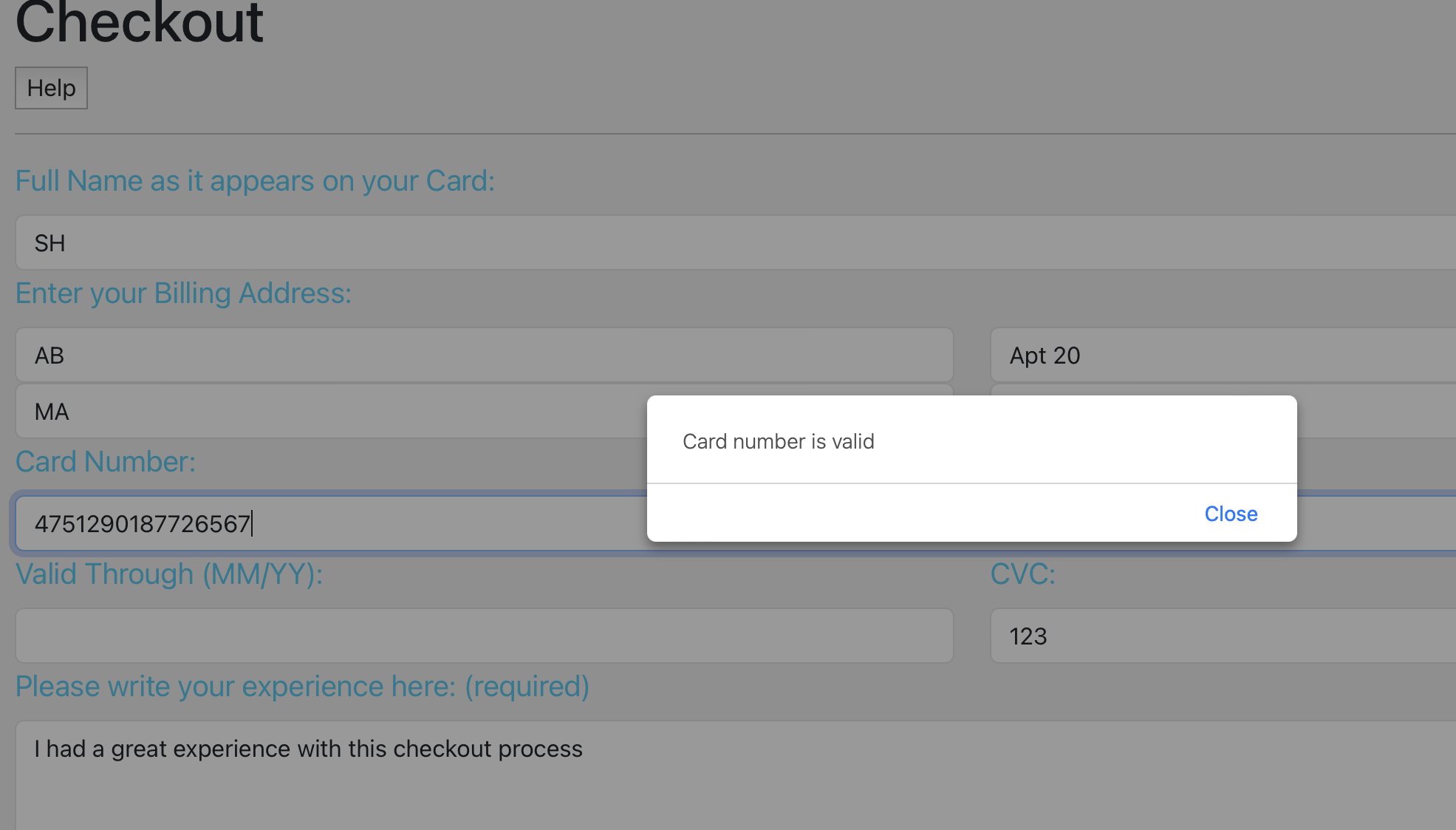
**Using more than 16 digits:**

****

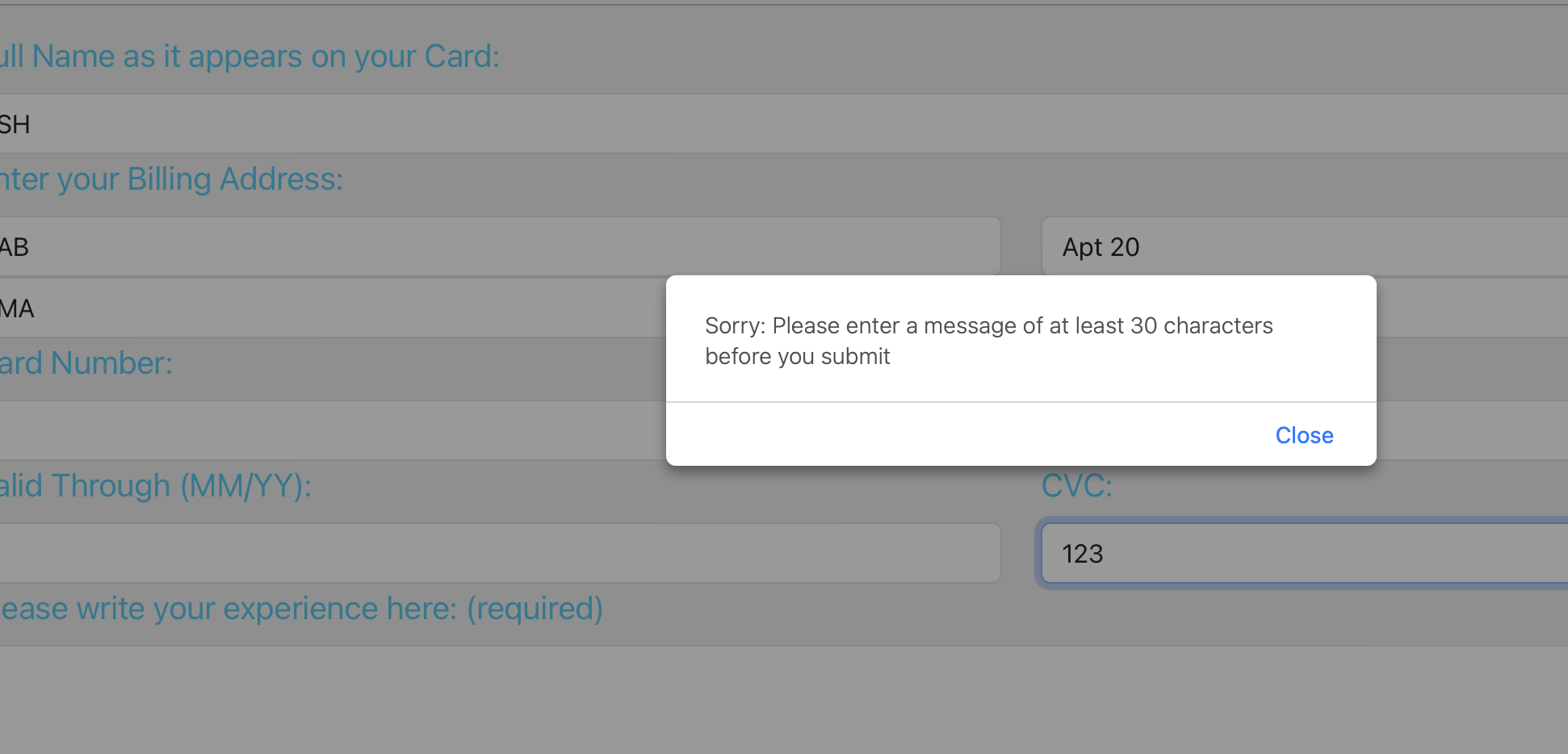
**Correct digits but invalid card number:**

****

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

****

**Testing the short message text box:**

****

| 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 characters | 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. |

# Automated Testing Report

**For this iteration, the team will not be conducting automated testing, in the future iterations, we plan on using selenium for automated testing.**

# 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 |
| 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