Testing

Unit Testing

Database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Test case | Expected Results | Actual Results | Notes |
| 1 | parsePayload([]byte) | The byte array parsed using |  |  |
| 2 | IsDeviceStolen(string) |  |  |  |
| 3 | VerifyAccountInfo(string, string) |  |  |  |
| 4 | GetUserDevice(string) |  |  |  |

Website Blackbox Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Action | Expected Result | Actual Result | Notes |
| 1 | User enters URL | Home page | As expected | URL: localhost:8080/home |
| 2 | User fills Sign Up form and submit | New account and customer created on database | As expected |  |
| 3 | User access control: before log-in | Error page: not logged in | As expected |  |
| 4 | User fills login form and submit | Rerouted to user map page | As expected |  |
| 5 | User map receives updates | Markers on map updating | As expected |  |
| 6 | User logs out | User redirected to home page and logged out of session | As expected |  |
| 7 | User incompletely fills Sign Up form and submit | Login Page with error message to fill in expected values that were empty. | As expected |  |
| 8 | User incompletely fills Login form and submit | Routed to login page with error message to fill in expected values that were empty. | As expected |  |
| 9 | User gives username that does not exist | Routed to login page with error message that an account with that user name does not exist | As expected |  |
| 10 | User gives a password that doesn’t accompany the given username | Routed to login page with error message that incorrect password for that user name was given | As expected |  |
| 11 | User fills out signup form and gives and invalid email address | Routed to login page with error message that the email address is invalid | As expected |  |
| 12 | User fills out sign up form and gives an email address that is already linked to an account | Routed to login page with error message that the email address already exists in the system |  | Not yet implemented. |
| 13 | User fills out sign up form and gives a phone number that is already linked to an account | Routed to login page with error message that the phone number already exists in the system |  | Not yet implemented. |
| 14 | User fills out sign up form and gives a phone number that is too long or too short | Routed to login page with error message that the phone number is invalid |  | Not yet implemented. |
| 15 | User fills out sign up form and password and confirm password don’t match. | Routed to login page with error message that password and confirm password don’t match |  | Not yet implemented. |
| 16 | User is rerouted to Home Page | Home page shows up displaying the location of all the users registered devices on the map |  | Not fully implemented |
| 17 | User selects a specific device | User rerouted to the map page displaying the location of the device, time last updated, and additional view options |  | Not fully implemented |
| 18 | User changes laptop device status to stolen and hits submit | Device status updated in database. Service beings tracking device, and logging keys in database. |  | Not fully implemented |
| 19 | User changes geogram device status to stolen and hits submit | Device status updated in database. Service beings tracking device. |  | Not fully implemented |
| 20 | User changes laptop device status to not-tracking | Device status updated in the database. Service discontinues tracking the devices, and stops logging the keys. |  | Not fully implemented |
| 21 | User changes geogram device status to not-tracking | Device status updated in the database. Service discontinues tracking the devices. |  | Not fully implemented |
| 22 | User selects keylogger. | Keylogger is pulled from the database and displayed over the specific device page. |  | Not fully implemented |
| 23 | User selects clear keylogger | User prompted to confirm or cancel delete keylogger |  | Not fully implemented |
| 24 | User selects confirm clear keylogger | Keylogger information cleared in the database. User rerouted to specific device page |  | Not fully implemented |
| 25 | User selects keylogger close | Keylogger closes and the display returns to just the specific device page. | As expected |  |
| 26 | User selects rename | User is prompted for a new name. | As expected |  |
| 27 | User enters new name and hits submit. | Device name is changed in the database and user is returned to specific device page |  | Not yet implemented. |
| 28 | User selects delete | User is prompted whether they would like to delete the specific device | As expected |  |
| 29 | User selects confirms device delete | Device information is removed from the database. Rerouted back to home. |  | Not fully implemented |
| 30 | User selects toggle previous location | Previous device location information is displayed on the map. |  | Not fully implemented |
| 31 | User deselects toggle previous location | Previous device location information is removed from the map | As expected |  |
| 32 | User selects update location | Device is relocated, new information is entered into the database, and updated location is displayed on the map | As expected |  |
| 33 | User selects all devices | Rerouted to the home page displaying all the locations of every device associated with the account |  | Not yet implemented. |
| 34 | User selects add new device | User Rerouted to add new device form. |  | Not yet implemented. |
| 35 | User fills out new geogram device form validly and completely | New device added to database. User rerouted to specific device page. |  | Not yet implemented. |
| 36 | User fills out new device form and uses a name already used for one of their devices. | User routed to new device form with message indicating device name already exists with this account |  | Not yet implemented. |
| 37 | User fills out new device form and selects geogram device. Enters phone number that already exists as a geogram device. | User routed to new device form with message indicating device phone number already exists for a different geogram device |  | Not yet implemented. |
| 38 | User fills out new device form and selects laptop device. Enters IP address that already exists for a different laptop device. | User routed to new device form with message indicating device IP address already exists for a different laptop device |  | Not yet implemented. |
| 39 | User fills out new laptop device form validly and completely | New device added to database. User rerouted to specific device page. |  | Not yet implemented. |
| 40 | User selects account | User rerouted to account information page. |  | Not yet implemented. |
| 41 | User selects edit account info. | User rerouted to edit account information form |  | Not yet implemented. |
| 42 | User makes valid changes to their user info. Selects save | User information updated in database. User rerouted to account information page. |  | Not yet implemented. |
| 43 | User makes invalid changes to their user info. Selects save. | Appropriate error message displayed on edit account information form page. User information not updated. |  | Not yet implemented. |

Testing Screen Captures



Password: hello

Figure : Test #2 form filled before submit



Figure 1.1: Test #2 customer table after sign up submitted



Hashed password

Relation

Figure 1.2: Test #2 account table after sign up submitted



Figure : Test #3 Error page



Figure .1: Test #4 login form before submit



Figure 3.2: Test #4 login after submit, redirected to map page

Go server

3-tier architecture – Server, database, and client (Web)

* Server components –
  + Central Server
    - The central server is the entry point of the server. It initializes and starts the modules of the server, and initializes the communication channels used by the modules.
    - The central server connects all of the server modules together. All components send and received Requests through the central server. When a request is received, the central server redirects the request to the proper module.
  + RequestProtocol
    - The modules of the server communicate with a requesting protocol. Using a requesting protocol allows the server to be organized. It also allows for adding and removing server modules with less effort, because it allows each module to have a high cohesion.
    - The protocol contains opcodes for modules to use when creating requests.
    - A requests contains an id, the destination of the request, the source of the request, an opcode, a payload, and a response channel that the response to the request will be sent over.
  + Web Server
    - Http handler – handles http requests received and sends back an http response to the request. The response will contain all of the files needed for the client view. A typical response would contain files such as images, style sheets, scripts, and the html file.
    - Websocket handler – handles websocket requests made to the server. When a websocket request is received, the handler creates a new connection to the client and registers the connection in the Web hub.
    - WebClientConnection – Middleman between websocket and the web hub. Reads in messages from the websocket and pass it on to the hub. Also receives messages from the hub and passes it to the websocket.
    - Web hub – Maintains a list of connected clients as WebClientConnections; creates a channel for communication to and from those clients
    - Client Web Sessions – when a web client makes an http request to the server, a cookie-based session is created. A copy of the session is stored in the server. When a client logs in, the contents of the cookie are changed and represent a session for the client. The sessions are used for access control and to present the clients with relevant data.
    - RequestProtocol handler – handles incoming and outgoing requests made from the other server components.