Testing the System is done with Postman. Once the server is running, API requests are called, and the responses are validated. The document outlines the different system tests / API calls. Refer to the development manual for details about the API.

Testing:

GET /

This test should return the home webpage where the user can interact with the server.

Testing:

GET /logout

This should remove the "username" attribute from the session object.

Testing:

POST /login?username=username&password=password

1. This, if valid, should add a "username" attribute to the current session, and it will redirect to GET /
2. If it fails, it will return a JSON object about the failure.
3. Usually, you'll want to use POST /user to create a user to test login.)

Testing: POST /query

1. To test query for user's game history
    a. {"type":"hist", "username":"the username"}
    b. This requires that the requested user exists; otherwise, it will return an error.
2. To test query for a game object
    a. {"type":"game", "gameId":"thegameId"}
    b. This requires that the game object exists; otherwise, it will return an error.
3. To test query for a game record
    a. {"type":"record", "gameId":"thegameId"}
    b. This requires that the game object exists; otherwise, it will return an error.
4. To test query for a board obejct
    a. {"type":"board", "gameId":"thegameId"}
    b. This requires that the game object exists; otherwise, it will return an error.

Testing: POST /user

1. To test rejecting an invitation:
    a. {"type":"replno", "inviteId":"the invite id"}
    b. This requires that the invite exists; otherwise, it will return an error.
2. To test accepting an invitation:
    a. {"type": "repl", "inviteId":"the invite id"}
    b. This requires that the invite exists; otherwise, it will return an error.
3. To test sending an inviation:
    a. {"type": "inv", "toUser":"tousername"}
    b. This requires that the user is logged in, and that the invited user exists. This will create an invitation object associated with the specified users in the system.
4. To test creating a user:
    a. {"type":"user", "username":"username", "email":"the email", "password":"password"}
    b. This will create a new user in the system.

Testing: POST /game

1. To test simulating a move:
    a. {"type":"move", "gameId":"thegameid", "fromCell":"thefromcellid", "toCell":"the tocell id"}
    b. This requires that the game has been created. To create a game, create two users, send an invite, accept the invite, and then the game will be created.