Test Plan and Report

SlugQuest by SlugQuestTeam

Ananya Batra , Sneha De , Alex Asch , Thomas Dillow , Sulayman Asif , Ethan Nel

March 10, 2024

System test scenarios:

User Story 1: As a user, I want to be able to securely login so that my information isn't leaked.

Scenario 1a: user login (Pass)

1. Start at SlugQuest frontpage
2. Click "Login" to redirect to the Auth0 login page.
    a. Assumes the user already has an account, otherwise they go to the "Signup" link to create new credentials.
3. Using valid credentials to login, the user should be redirected back to their dashboard of tasks.

Scenario 1b: user logout (Pass)

1. Repeat Scenario 1a steps to be logged in.
2. Click "Logout" to redirect to the Auth0 logout page.
3. If logout is successful, the user should see the SlugQuest frontpage.

Scenario 1c: validating access (Fail*)

1. Be in the logged out state.
2. Attempt to access to an internal application URL (such as the dashboard located at /loggedin)
3. Without being logged in, these internal routes should not be accessible.
    a. Partially fails since the route is accessible and viewable, but no actions can be taken such as adding tasks, and no data is retrieved from the backend.
    b. The backend server correctly returns 401 to any requests sent.

User Story 2: As a user, I want to be able to keep track of and update all my tasks. (Adding, editing, and removing tasks).

Scenario 2a: adding a task (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click “Add task”, the task adding modal should pop up
3. User fills out the fields as they wish
4. Click “Add task” (within the modal)
5. Should return the user back to their dashboard with the new task added

Scenario 2b: adding a recurring task (Fail*)

1. Start at main task dashboard (assuming logged in)
2. Click “Add task”, the task adding modal should pop up
3. User fills out the fields as they wish, including the checkbox “is recurring”
4. User selects how often task should recur on new popup (Daily weekly monthly)
5. Click “Add task” (within the modal)
6. Should return the user back to their dashboard with the new task added
7. User should be able to validate the task recurring as specified when navigating to the calendar view.
    a. Partially fails, this works correctly when initialized, however, relaunching the backend server creates duplicates of recurring tasks.

Scenario 2c: editing a non-recurring task (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click on any task card, the detailed task modal should pop up
3. Click on “Edit task”, the editing task modal should pop up
4. User edits task fields
5. Click on “Save task”, should return back to the detailed task modal with the correct updates reflected

Scenario 2d: editing a recurring task (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click on “Calendar View”
3. Click on any event (blue box) to open the edit modal
4. User edits task fields (for one specific instance)
5. Click on “Update”, should return back to the calendar view with the edits reflected on each recurring instance as well

Scenario 2e: deleting a task (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click on any task card, the detailed task modal should pop up

3. Click on “Delete task”, should return back to the task dashboard with the task card gone from view
    a. If it was the only task in a specific category, the category should also disappear from the sidebar

Scenario 2f: Actions from calendar view (Fail)

1. Repeat all above cases and verify for calendar view
    a. Calendar view needs to be refreshed to propagate changes made

Scenario 2g: Drag and drop edits from calendar view (pass)

1. Navigate to calendar view
2. Select a non recurring task and drag it to move the time it is scheduled for
3. Verify the start and end time move correctly

User Story 3: As a busy college student, I want a way to track and organize my tasks so that I stay on top of my course schedule.

Scenario 3a: Creating a task in a category (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click “Add task”, the task adding modal should pop up
3. User fills out the category field with the respective category they want
4. Click “Add task” (within the modal)
5. Should return the user back to their dashboard with a new category section for that created category

Scenario 3b: Filtering tasks by category (Pass)

1. Start at main task dashboard (assuming logged in)
2. Run Scenario 3a to create a task in a category
3. Click on the corresponding category filter.
4. There should only be the tasks of that category visible.

Scenario 3b: Filtering tasks by Completion (Pass)

1. Start at main task dashboard (assuming logged in)
2. Create a task as in Scenario 2a
3. Click the task and mark it as completed
4. Click on the completed filter on the sidebar
5. The newly completed task should be shown in the view.

Scenario 3c: Filtering tasks by Recurrence (Pass)

1. Start at main task dashboard (assuming logged in)
2. Create a recurring and non-recurring task, as in Scenario 2b and 2a
3. Users should be able to see only recurring tasks under the recurring filter
4. User should only be able to see non recurring tasks under the non recurring filter

Scenario 4c: Viewing tasks in calendar view (Pass)

1. Start at main task dashboard (assuming logged in)
2. Navigate to calendar view
3. Users should be able to view all tasks arranged by date and time, including recurring tasks.

User Story 4: As a social person, I want to be able to search for and add friends so I can work with them to be productive.

Scenario 4a: searching for other users (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click on “Social Page”
3. Click on “Add Friend”
4. Search a user’s username or social code
5. A list of users with similar usernames (or exact social code match) should be displayed with the option to add them as friends

Scenario 4b: adding friends (Pass)

1. Repeat Scenario 4a to see list of names
2. Click on “Add” next to the specific user to add as a friend
3. Should return to the social page view, with the new friend seen in the “Friends” column (in the format Username#SocialCode)

Scenario 4c: seeing own social code (Pass)

1. Starting at main dashboard (assuming logged in)
2. Click on “Social Page”
3. User should see their own social code at the top left of social page

Scenario 4d: seeing own leaderboard (Pass)

1. Starting at the Social Page (assuming logged in)
2. Users should see a leaderboard of themselves compared to their friends, by points achieved.

Scenario 4e: removing a friend (Pass)

1. Starting at the social page (assuming logged in)
2. Add a friend as outlined in Scenario 4b
3. Click the red x next to the friend username
4. Verify that the friend is in fact deleted

User Story 5: As a team member, I want to easily collaborate with my team on our project so that we can easily divide responsibilities and stay on track for deadlines.

Scenario 5a: creating teams (Pass)

1. Start at main task dashboard (assuming logged in)
2. Click on “Social Page”
3. Click on “Create Team”, the team creation modal to enter a name should pop up
4. User enters the team’s name
    a. If the user inputs a blank name, the user is prompted to enter again
5. Click on “Create Team” (inside the modal)
6. Should return back to the Social Page with the new team shown under “Teams” and with a single user (themselves)

Scenario 5b: adding team members (Pass)

1. Repeat Scenario 5a steps to create a team
2. Next to the last team member, click “Add Friend to Team”
3. Should display popup with a dropbox showing user’s friends, select one
4. Click “Add to team”, new team member should be placed below the last one

Scenario 5c: removing team members (Pass)

1. Repeat Scenario 5a steps to create a team
2. Next to the newly added team member, click the red x icon
3. The team member should no longer show up on the team view

Scenario 5d: Creating team tasks (Pass)

1. Repeat Scenario 5a to create a team.
2. Navigate back to the task view using the go back button

3. Select Add task
4. Create a task under a team from the team dropdown

Scenario 5e: Getting team tasks (Fail)

1. Repeat Scenario 5d to ensure create a task on a team
2. Navigate to Team tasks filter from left sidebar
3. User should be able to see all tasks on the team, including ones by other users
    a. Users cannot see tasks created by other team members

User story 6: As a lazy person, I want a way to treat completing my tasks like a game, so that I stay motivated and also have fun.

Scenario 6a: Marking a task as completed (Pass)

1. Repeat Scenario 2 to add a task
2. Once a task is created, click on the specific task to open the task-card view
3. On that modal, click the green 'Complete Task' button on the bottom
4. On the bottom of the dashboard, boss health should decrease by either 1,2, or 3 points (corresponding to the difficulty self-assigned to the task - easy, medium, or hard).

Scenario 6a1: leaderboard (Pass)

5. Click on the "Social Page"
6. The users own points on the leaderboard should be incremented by the appropriate amount

Scenario 6b: Marking a task a failed (Pass)

1. Repeat Scenario 2 to add a task
2. Once a task is created, click on the specific task to open the task-card view
3. On that modal, click the 'Fail Task' button on the bottom
4. On the bottom of the dashboard, boss health should increase by either 1,2, or 3 points (corresponding to the difficulty self-assigned to the task - easy, medium, or hard).

Scenario 6a1: leaderboard (Pass)

5. Click on the "Social Page"

6. The users own points on the leaderboard should be decremented by the appropriate amount

User Story 7: As a social person, I want to be to collaborate with and play games with my friends so that I can stay in touch with them (cluster and manage groups of friends to share tasks.)

Scenario 7a: Seeing the leaderboard (Pass)

1. Repeat Scenario 4 to search for and add friends
2. Once friends are added, on the right hand side of the social page, a leaderboard appears. It is ranked by user points

Scenario 7b: Seeing the boss (Pass)

1. Assuming logged in on main page
2. Boss health can be seen on the bottom taskbar
3. Clicking the view boss button shows a boss image

Unit tests

Unit tests for the backend were implemented with the Go unit test framework and covered the basic components of the backend system: users, tasks, teams, and other smaller features. These are located in `backend/unit_tests`.

No unit tests fail in the release branch.

Running unit tests

A full set of instructions can be found in the README in the unit tests folder in the backend repo.

Assuming Go is installed and are currently in the root of the backend, all unit tests can be run via:

```
go test ./unit_tests
```

Running the tests individually via the command line leads to issues as the initial test suite setup is not done (such as the in-memory database used as a test stub), so it is recommended to use VSCode's Go extension that allows for running unit tests individually with initial setup.