Features That Will Be Tested

1. Login/Logout/Register:

   Test Cases:

   - Users should be able to utilize a mix of uppercase and lowercase characters in their username and password. (Case sensitivity)
   - When the user leaves a required field blank, they should receive a notification that the field they left blank is required. (Blank test/Mandatory Fields case)
   - Account registration fails when the user tries to create an account where the password entered does not contain a capital letter, at least 8 characters, at least 2 numbers, and at least 2 special characters.
   - Account registration fails when the user tries to create an account with a username that is already used. (Unique username)
   - Users should be redirected to the login page after registration. (Login -> Registration)
   - Users should receive a warning when attempting to login with an incorrect username or password. (Incorrect credentials)
   - When a user presses the login/registration hyperlinks, it should bring them to the corresponding page. (Hyperlinks)
   - When a user presses the logout hyperlink, they should be redirected to the login page with a message informing them that they logged out. (Logout)

   Test Data:

   - Blank data for all fields. Alphanumeric Test data and only characters for the password field

   Test Environment:

   - Login and Register pages. Logout link on the Navbar.
   - Mocha for testing NodeJS.

   Test Results:

   - When a user follows the restrictions for registering an account, they should be redirected to the login page, and their credentials stored in the database.
   - When a user fails to complete the specification for account registration, they should receive a popup warning, and their credentials should not be added to the database.
   - When a user attempts to log in and their credentials are in the database, they will be redirected to the joke generator page.
   - When a user attempts to log in and their credentials are not in the database, they should not be redirected, and a warning should appear.

   User Acceptance Testers:

- Wide range of students, family, friends, and volunteers with differing backgrounds in technology. This not only makes sure each feature works as intended, but is also user-friendly.

2. Joke Generating:
   Test Cases:
   - The user must specify at least one “type” of joke (at least one API to pull from).
   - The user should fill the filter fields with opinions to give the best joke based on the filters (API testing)
   - The program must be able to handle both multiple or no keywords.
   - Turning off a filter/removing a keyword should refresh jokes displayed so that specified jokes no longer appear.
   - The jokes will be displayed after filling the filter fields form.

   Test Data:
   - Data where no filters are specified.
   - Data where no keywords are specified.
   - Data where multiple keywords are specified.
   - Data that combines these edge cases (no filters or keywords; filters and no keywords; vice versa; etc.)
   - Data where filters and keywords are toggled on and off repeatedly.

   Test Environment:
   - Loading up the page and messing with it.
   - Mocha for testing NodeJS.

   Test Results:
   - When no filters are specified, an error should be thrown.
   - When no keywords are specified, there should NOT be an error.
   - When there are no jokes that match keywords or filters, a “No Jokes Found” message should be shown.
   - Selecting multiple filters and/or keywords should display jokes that fill *all* of the specifications, not just one.
   - Deselecting filters or keywords should actively make jokes leave the view in real time.

   User Acceptance Testers:
   - Wide range of students, family, friends, and volunteers with differing backgrounds in technology. This not only makes sure each feature works as intended, but is also user-friendly.

3. Joke Saving/Recall:
   Test Cases:

- A saved joke can be recalled after a logout, additional jokes saved, changes to profile, etc.
- A saved joke that is removed cannot be recalled until saved again.
- A saved joke cannot be saved again (not case-sensitive).
- A user can only save a maximum of 25 jokes.
- Every user has their own data for the jokes the user prefers.

Test Data:

- Large amounts of jokes with which to test saving and recall.
- Groups of identical jokes (including jokes with different capitalizations to test case sensitivity).
- Groups of jokes that have exactly or more than 25 jokes to test upper limit.

Test Environment:

- We will also be forcing our parents to test this too.
- Mocha for testing NodeJS.

Test Results:

- Saved jokes should be able to be recalled when requested, and requesting unsaved jokes should return an error.
- Deleting a joke that isn't saved should cause an error.
- Trying to recall a joke that has been deleted should cause an error.
- Trying to submit a joke that has already been saved should cause an error (not case-sensitive).
- Trying to save a 26th joke without first deleting a previous joke should cause an error.

User Acceptance Testers:

- Wide range of students, family, friends, and volunteers with differing backgrounds in technology. This not only makes sure each feature works as intended, but is also user-friendly.