Testing Document and Specification

Test Specification

Team Nova
CS_451R, 2025

Abdul Mohammed
Haley Nilsen
Sid Nsude
Kaleb Flores Lopez

Frontend:

Test Case ID:	1.1.1.1
Title:	Logging in as a Personal User
Feature/Subfeature:	User Functionality
Purpose:	To ensure that the user's account is tied to the correct permission set and that the correct credentials are stored for them to login with
Initial Conditions:	User must already have an account made as a "Personal User"
Test Data:	Test Data will include the following artificial data: **Personal User:** Username – sharpie@gmail.com Password – sharpie **Business Admin:** Username – smartinez@acmecorp.com Password – Leader2024! **Sub-Business User:** Username: alex.chen@acmecorp.com Password: Marketing2024!
Test Actions:	1. Go to the main website 2. User enters in the username and password 3. User clicks login 4. User is then taken to their home dashboard
Expected Results:	in, they are taken to their dashboard

Test Case ID:	1.1.2.1
Title:	Authenticating a message sent to the Chatbot
Feature/Subfeature:	User Functionality
Purpose:	To ensure that when a user sends a message to the Chatbot, it will take the message and respond.

Initial Conditions:	The user must navigate to the Chatbot page.
Test Data:	The test data will include clicking the following prompt: How can I save more money each month?
Test Actions:	1. Go to the main website 2. Navigate to the Chatbot page using the left Nav Bar 3. Enter a message into the chat box 4. The chatbot will then respond with a relevant answer
Expected Results:	Once the user enters in a message, the bot will load for a few seconds before spitting out an answer. This is so it can process the input and generate a correct output with the relevant data from the database

Test Case ID:	1.1.3.1
Title:	Setting a Goal
Feature/Subfeature:	User Functionality
Purpose:	To ensure that the user is able to create and edit their own goals
Initial Conditions:	The current user must have a monetary value and a name for their goal
Test Data:	The following test data will be used: Total amount -$10,000 Goal Name: New Car Down Payment
Test Actions:	1. Navigate to the main website 2. Then, select the goals page from the left Nav bar 3. Click New Goal 4. Enter in goal name 5. Enter in goal amount 6. Enter in any applicable savings toward goal 7. Click submit
Expected Results:	After step 7, the goal should now display on the page and dashboard with the entered information

Test Case ID:	1.1.3.2
Title:	Editing a Goal
Feature/Subfeature:	User Functionality
Purpose:	To ensure that the user is able to edit an existing goal

Initial Conditions:	The current user must have an existing goal on their page
Test Data:	The test data that will be used is: Amount total - $15000 New amount - $9000 Goal Name: Emergency Fund
Test Actions:	1. Navigate to the main website 2. Then, select the goals page from the left Nav bar 3. Click Edit Goal 4. Change monetary amount to match new savings 5. Click Save
Expected Results:	After step 5, the page should now show the updated goal amount for the selected goal

Test Case ID:	1.1.4.1
Title:	Changing Password
Feature/Subfeature:	User Functionality
Purpose:	To ensure that the user is able to change their password if they meet the requirements listed on the page
Initial Conditions:	A user must have an account created with ClariFi
Test Data:	The test data that will be used is: **Personal User:** Old Password: sharpie New Password: Sharps31! **Business User:** Old Password: Leader2024! New Password: Leading1589! **Sub-Business User:** Old password: Marketing2024! New Password: BusinessMarket1992!
Test Actions:	1. Navigate to the main website 2. Then, select the settings button in the bottom left corner of the Navigation Bar 3. Click “Change Password” 4. Enter in old password 5. Enter in the new password in both the “New” and “Confirm” password boxes

Expected Results:	After step 5, the website should display a message informing the user that the password was changed successfully, as long as the password meets the listed requirements on the page

Test Case ID:	1.1.4.2
Title:	Deleting Chats
Feature/Subfeature:	User Functionality
Purpose:	To ensure that users can delete Chatbot conversations whenever
Initial Conditions:	The user needs to have a ClariFi account already.
Test Data:	The data that will be used are previously stored chats within the database.
Test Actions:	1. Navigate to the main dashboard 2. Click on the settings icon in the bottom left corner of the navigation bar 3. Select the option in the settings labeled “Delete Chats” 4. Select confirm deletion
Expected Results:	After step 4, the screen should inform the user that the chats have been deleted. The user can now navigate to the Chatbot page and see no previous chat history.

Test Case ID:	1.1.4.3
Title:	Changing an email or name
Feature/Subfeature:	User Functionality
Purpose:	To ensure that the user is able to update their email or name when needed
Initial Conditions:	The user needs to have a ClariFi account.
Test Data:	The test data that will be used is: **Personal:** Name – James Sharp Email – sharpie@gmail.com **Business:** Name – Sarah Martinez Email – smartinez@acmecorp.com **Sub-Business User:** Name – Alex Chen Email – alex.chen@acmecorp.com
Test Actions:	1. Navigate to the main dashboard

	2. Select “settings” in the bottom left corner of the navigation bar 3. Edit the name and email 4. Click Save Changes
Expected Results:	After step 4, the user should see their updated name and email on the settings screen.

Backend:

Test Case ID:	1.1.5.1
Title:	Test login authentication
Feature/Subfeature:	System User Validation
Purpose:	To ensure user can login properly
Initial Conditions:	The user needs to have a ClariFi account.
Test Data:	The test data that will be used is: **Personal:** Name – James Sharp Email – sharpie@gmail.com **Business:** Name – Sarah Martinez Email – smartinez@acmecorp.com **Sub-Business User:** Name – Alex Chen Email – alex.chen@acmecorp.com
Test Actions:	1. User types in email 2. User types in password 3. User presses login
Expected Results:	After step 3, the user should then be routed to their dashboard

Test Case ID:	1.1.5.2
Title:	Get profile without authentication
Feature/Subfeature:	Authentication / User Profile Access
Purpose:	verify that accessing the user profile endpoint without authentication is blocked.
Initial Conditions:	• Backend server is running • No authentication token is provided • Client is not logged in
Test Data:	• Endpoint: /auth/profile • HTTP Method: GET • Headers: none
Test Actions:	Send a GET request to /auth/profile without including an authorization token.
Expected Results:	• Server returns HTTP status code 401 (Unauthorized) • User profile data is not returned

Test Case ID:	1.1.6.1
Title:	Get dashboard summary without authentication
Feature/Subfeature:	Dashboard
Purpose:	Verify that the dashboard summary endpoint cannot be accessed without authentication
Initial Conditions:	• Backend server is running • No authentication token is provided • Client is not logged in
Test Data:	• Endpoint: `/dashboard/summary` • HTTP Method: `GET` • Headers: none
Test Actions:	Send a GET request to /dashboard/summary without including an authorization token.

Expected Results:	• Server returns HTTP status code **401 (Unauthorized)** • Dashboard summary data is not returned

Test Case ID:	1.1.6.2
Title:	Get recent purchases without authentication
Feature/Subfeature:	Dashboard
Purpose:	Verify that the recent purchases dashboard endpoint cannot be accessed without authentication.
Initial Conditions:	• Backend server is running • No authentication token is provided • Client is not logged in
Test Data:	• Endpoint: `/dashboard/recent-purchases` • HTTP Method: `GET` • Headers: none
Test Actions:	Send a GET request to /dashboard/recent-purchases without including an authorization token..
Expected Results:	• Server returns HTTP status code 401 (Unauthorized) • Dashboard summary data is not returned

Test Case ID:	1.1.7.1
Title:	Get goals without authentication
Feature/Subfeature:	Goals
Purpose:	Verify that the goals endpoint cannot be accessed without authentication.
Initial Conditions:	• Backend server is running • No authentication token is provided • Client is not logged in
Test Data:	• Endpoint: `/goals/` • HTTP Method: `GET` • Headers: none
Test Actions:	Send a GET request to `/goals/` without including an authorization token.
Expected Results:	• Server returns HTTP status code 401 (Unauthorized) • Goals data is not returned

Test Case ID:	1.1.7.2
Title:	Get goals without authentication
Feature/Subfeature:	Goals
Purpose:	Verify that the goals endpoint cannot be accessed without authentication.
Initial Conditions:	• Backend server is running • No authentication token is provided • Client is not logged in •
Test Data:	• Endpoint: `/goals/` • HTTP Method: `GET` • Headers: none • Request Body: • name: Test Goal • type: personal • target_amount: 1000

	• current_amount: 0 • target_date: 2025-12-31 •
Test Actions:	Send a GET request to `/goals/` without including an authorization token.
Expected Results:	• Server returns HTTP status code 401 (Unauthorized) • Goal is not created • No data is written to the database

Test Case ID:	1.1.8.1
Title:	Verify database connection
Feature/Subfeature:	Database / Connectivity
Purpose:	Verify that the application can successfully connect to the database.
Initial Conditions:	• Database service is running • Application database configuration is valid
Test Data:	SQL Statement: `SELECT 1`
Test Actions:	• Establish a connection to the database using the configured engine. • Execute the SQL statement `SELECT 1`.
Expected Results:	• SQL query executes successfully • Query returns the value `1` • No connection or execution errors occur