System Features and Requirements

*** and *** signify critical requirements, **orange** means they have not been adequately tested by the end of the CW, **green** means they have been tested in some manner. Tutorial page is not critical yet as it is a WIP.

1. Functional Requirements

1.1 User Authentication - Signup

***FR1.1.1 Users can create new accounts with a unique username, email and password once all of the restrictions below are met.

***FR1.1.2 Users will be required to confirm their password another time during the Signup process, and if the two passwords do not match, the user will be prompted and the account not created.

FR1.1.3 Passwords may only contain uppercase letters (A-Z), lowercase letters (a-z), digits (0-9), and special characters (!, @, #, $, %, ^, &, *, _, -), If the password contains any other characters, the user will be prompted and the account will not be created.

FR1.1.4 The email provided must be in the format of an email address. It must contain an @ symbol and text before and after the symbol. If this is not met, the account will not be created and the user will be prompted.

FR1.1.5 The username must only contain alphanumeric characters, and be between 5 to 20 characters in length, if these conditions are not met then the user will be prompted and their account will not be created.

***FR1.1.6 The sign up button will not sign up the user if the above is not met.

***FR1.1.7 If the above is met, and the sign-up button is pressed, the user's account will be created and they will be prompted of the success.

FR1.1.8 User credentials must be securely transmitted and handled during the signup process.

FR1.1.9 A new user account will not be created for a username that already exists.

1.2 User Authentication - Signin

****FR1.2.1 User Login: The system shall allow users to log in using a correct username and a password pair, upon which they become authorized. If the pair is incorrect(not present in the database), an error message is presented to the user and they remain unauthorized.

****FR1.2.2 Unauthorized users may not access any pages except home and login, an attempt to access another page will redirect them to the login page.

****FR1.2.3 Unauthorized users may not perform any actions except for logging in or registering. They will not have access to any pages other than the Login Page and the Home Page and will not be redirected upon login failure.

****FR1.2.4 Authorized users will get access to every webpage, but will first be redirected to the dashboard.

****FR1.2.5 Authorized users are able to make any request to the backend, all requests from unauthorized users are rejected

FR1.2.6 There should be a session timeout mechanic, i.e. a user must be logged out after a certain period of time to verify that they still know their credentials.

FR1.2.7 Users should be able to continue their session assuming that they are still within the active time session limit.

1.3 Dashboard

***FR1.3.1 An authorized user has access to the dashboard.

***FR1.3.2 A user will have three choices from the dashboard : Logout, Interactive Mode and Links Tutorial Mode.

***FR1.3.3 If the user choses log out, their session becomes invalidated, they get redirected to the login page and they return to an unauthorized user discussed in 1.2.

***FR1.2.4 If the user choses Interactive mode, they get redirected to the interactive mode page.

***FR1.2.5 If the user choses Links Tutorial Mode, they get redirected to the tutorial page.

1.4 Interactive Mode Page

***FR1.4.1 Upon opening the interactive mode page, the user will be prompted to wait until the LINKS REPL shell gets started.

***FR1.4.2 Once the REPL shell starts, the user will be presented with a description of what it does.

***FR1.4.3 The user will have an area to enter links code, and upon pressing enter, they will receive a response of the evaluated code.

FR1.4.4 If the Links code is incorrect, it will still get evaluated and the relevant error will be displayed.

***FR1.4.5 Alternatively, the user will be able to go through a short series of tutorials outlining the basic syntax of Links. The first tutorial will be automatically printed “First try type just a literal like 52;” and the users can move on to the next tutorials by typing next tip; into the REPL shell.

FR1.4.6 The user will be able to skip the whole tutorial series by typing “skip intro;” into the REPL shell.

***FR1.4.7 The phrases “next tip;” and “skip intro;” will not be treated as Links code and evaluated when entered into the REPL shell.

1.4 Tutorial Page

FR1.4.1 Upon entering the tutorial page, the user will be greeted with a code editor, result window, tutorial explanation pane and tutorial selection sidebar.

FR1.4.2 When an existing user enters the page, they will be returned to the spot before they last exited the application e.g. Tutorial 4. Everything they have since written to the code editor will remain there.

FR1.4.2 The user will be able to move between the tutorials using the side panel and when a tutorial is clicked, the explanation and code editor panes will switch to the ones for that tutorial. (The explanation for that tutorial, and the user's saved work for that tutorial)

FR1.4.3 The user can follow the tutorial, by completing the tasks written in the tutorial explanation, they will be able to write the corresponding code in the code editor.

FR1.4.4 After the user is happy with their code, they can click the save & compile button, after this, the user will get a loading popup that will continue until the link's code is compiled.

FR1.4.5 Once compiled, the user will be able to see their compiled webpage on a panel on the page.

1.5 Miscellaneous

***FR1.5.1 If there is an issue connecting to the backend, the user must be informed that the web app is down instead of trying to interact with a front end that does not function as expected.

2. Non-Functional Requirements

2.1 User Authentication

2.1.1 During user authentication, informative error messages must be provided but should not disclose sensitive information that could aid an attacker. For example, the system SHOULD NOT specify if the username or password was incorrect, but SHOULD specify if the register attempt failed, and why (is it the user's fault, maybe the password they chose is wrong or is the server down).

2.1.2 After submitting correct login details, the user should be allowed into the website within 3 seconds of hitting the login button.

2.2 Dashboard

2.2.1 The dashboard should be intuitive and easily navigable with clear labels and accessible design.

2.3 Interactive Mode Page

2.3.1 Once the user is prompted to wait for the REPL shell to be loaded, it should take no longer than 10 seconds for this to happen.

2.3.2 Even under heavy load, the user must receive a response to their REPL shell commands up to 3 seconds after sending them.

2.3.3 The user must not be able to execute commands on the REPL shell that will allow for arbitrary reading of files from the server, for example the database that stores passwords.

2.4 Miscellaneous

2.4.1 The system should be deployed in a way that is scalable, so that it can keep up with a growing user base if required, but also remain cost effective if the web app is not popular.

2.4.2 The system should be easy to maintain, it should handle this by having an organized Github Repo with a CICD pipeline, standardized commits and feature branches as well as a develop and production branch which are each deployed automatically when changes get pushed.

2.4.3 The web-application must achieve a minimum of 99.9% availability as a monthly average, this includes all the functionality of the web application.

2.4.5 The database of the system should be backed up regularly to prevent data loss, and the source code securely help in a version management application like github.

2.5 Robustness Requirements

2.5.1 The system must have mechanisms to recover from common failures, such as database unavailability or network interruptions, without losing user data or requiring a system restart.

2.5.2 In case of high traffic that slows down the application, the user should be informed in a meaningful manner, including the estimated wait time (whether for login, compile etc).

2.5.3 Every field that contains user input must be validated and sanitized to prevent common exploits like SQL injection, XSS, etc.

2.5.4 The system should not be broken by nonsensical data being entered into the fields, instead it should provide detailed error messages explaining what the user did incorrectly.

2.6 Security Requirements

S.R1.1 Usernames and Passwords must be hashed and salted before stored in the database

S.R1.2 Secure HTTPS protocols must be used for all data transmissions to protect against eavesdropping and man-in-the-middle attacks.