**Textual Description of System’s Structure**

For our website structure, we've opted for a JavaScript React frontend framework paired with a Python Flask backend framework, running locally on http://localhost:5000. Upon local website initiation, users are directed to the login page, where their login email and password are sent as parameters via Axios post request in the handleSubmit function to the authentication\_service.py file. This file is linked to a MongoDB database, storing users’ names, email addresses, and account passwords, utilizing the Pymongo service and MongoDB API. These parameters are then utilized within the login route to invoke the verify\_login method in the authentication service instance. Within this method, the MongoDB database is queried to confirm the existence of the provided email account and validate the password. A successful match results in a JSON successful message response with an HTTP 200 code, while a failed attempt triggers a JSON error message response with an HTTP 401 code. Users without an account are prompted to register by clicking the sign-up button, redirecting them to the signup page.

On the signup page, users are required to input their name, email, and password, along with checking a verification box before submitting the form. Upon submission, the data is sent in the register route of authentication\_service.py through an Axios post request in the handleSubmit function. The route then passes the parameters to the register method in the authentication service instance. Within this method, the password is salted and hashed before being stored along with the other provided inputs in the database. A success message is returned and sent to the showModal component in the signup page that redirects successfully registered users to the login page.

A computer screen shot of text

Description automatically generated

Upon successful login, users are redirected to the home page that hosts the project generation form. Here, users select a programming language, enter desired skills and technologies, and submit the form. The form data is then sent to the handleSubmit function, which forwards it to the generation page. This page connects to the project\_generation\_service.py file via an Axios post request with the forwarded parameters. The generate route then passes the parameters to the generate\_project\_idea method in the project service instance. Within the generate\_project\_idea method, the parameters are inserted into the create README prompt for a post request to the OpenAI API and the appropriate headers. Afterward, the README response is posted to the generation page alongside a form prompting users to input the project name and their GitHub access token for repository creation. Upon submission, the handleRepoCreation function validates the form data before initiating an Axios post request to the create-repo route. The form data is then sent as parameters to the create\_repository method in the project\_service instance. The create\_repository method utilizes the GitHub API and access token to establish a working path and directory, creating the project repository and adding the README file. Upon successful creation, the repository URL is returned to the generation page, enabling users to commence work on their project.

A text on a white background

Description automatically generated

A screenshot of a computer code

Description automatically generated