AI Problem Coder

Bolt IoT - Artificial Intelligence Training

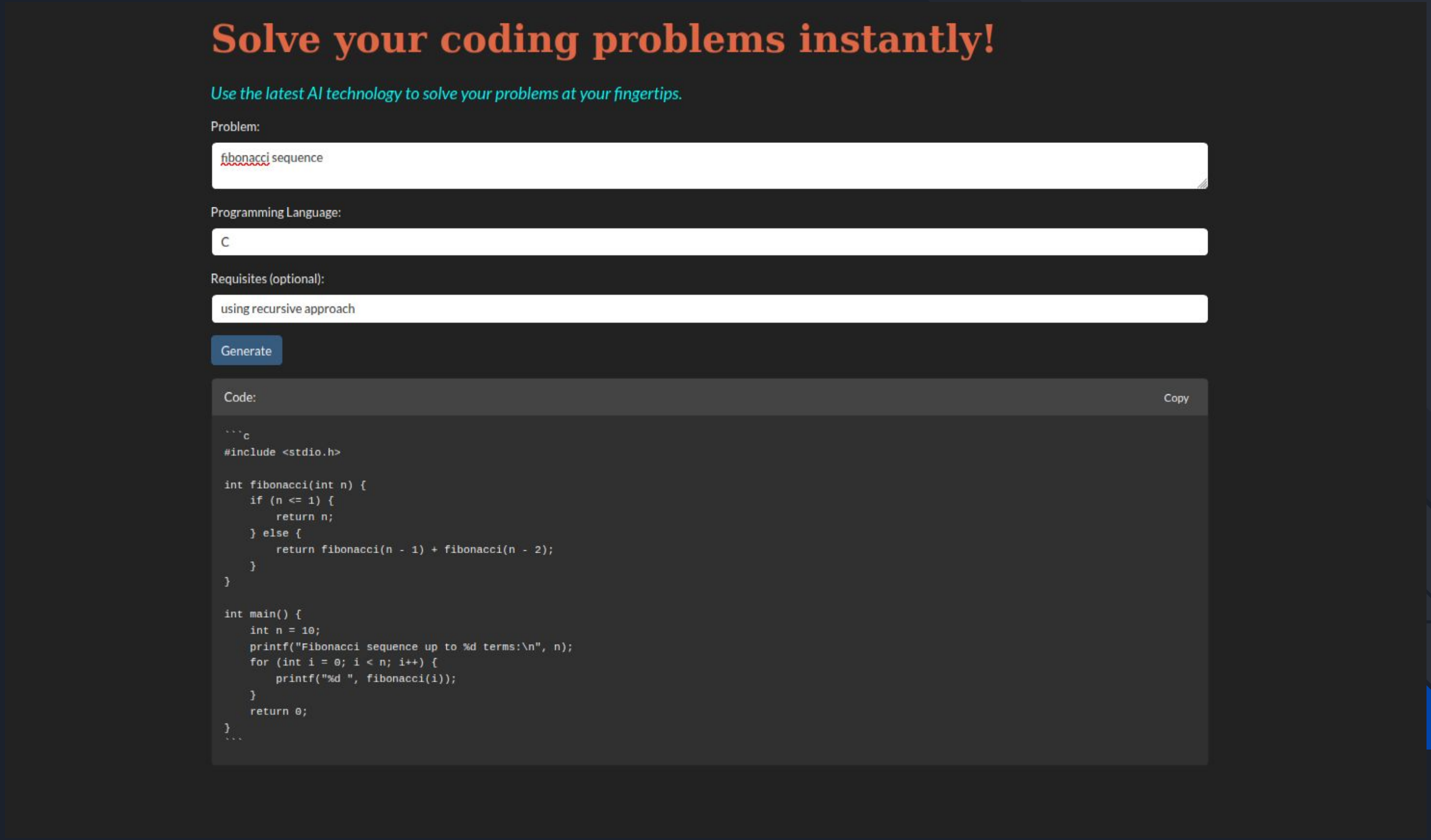
Yogesh Prashant Rane
yogeshrane.contact@gmail.com
+91 9527790023

Objective: Develop a web-based AI tool that generates programming code based on user-defined problems and programming languages using OpenAI's API. This tool should provide a user-friendly interface for students and developers to input their coding problems and requirements, and receive AI-generated code snippets as solutions.

Features:

1. **Input Form**:
   - Users can describe their coding problem.
   - Users can select the programming language they want the solution in.
   - Users can optionally specify any requisites or additional constraints for the code.
2. **AI Code Generation**:
   - The tool uses OpenAI's GPT-3.5-turbo model to generate code snippets.
   - Based on the provided problem description, programming language, and requisites, the AI generates appropriate code to solve the problem.
3. **Output Display**:
   - The generated code is displayed on the web page.
   - Users can easily copy the generated code to their clipboard.
4. **Error Handling**:
   - If the AI API call fails, the user receives a friendly error message asking them to try again later.
5. **AJAX-based Form Submission**:
   - The form uses AJAX to submit data without refreshing the page.
   - The generated code is updated dynamically on the page after form submission.

Project View


Solve your coding problems instantly!
Use the latest AI technology to solve your problems at your fingertips.
Generate
Copy

Problem:

fibonacci sequence

Programming Language:

C

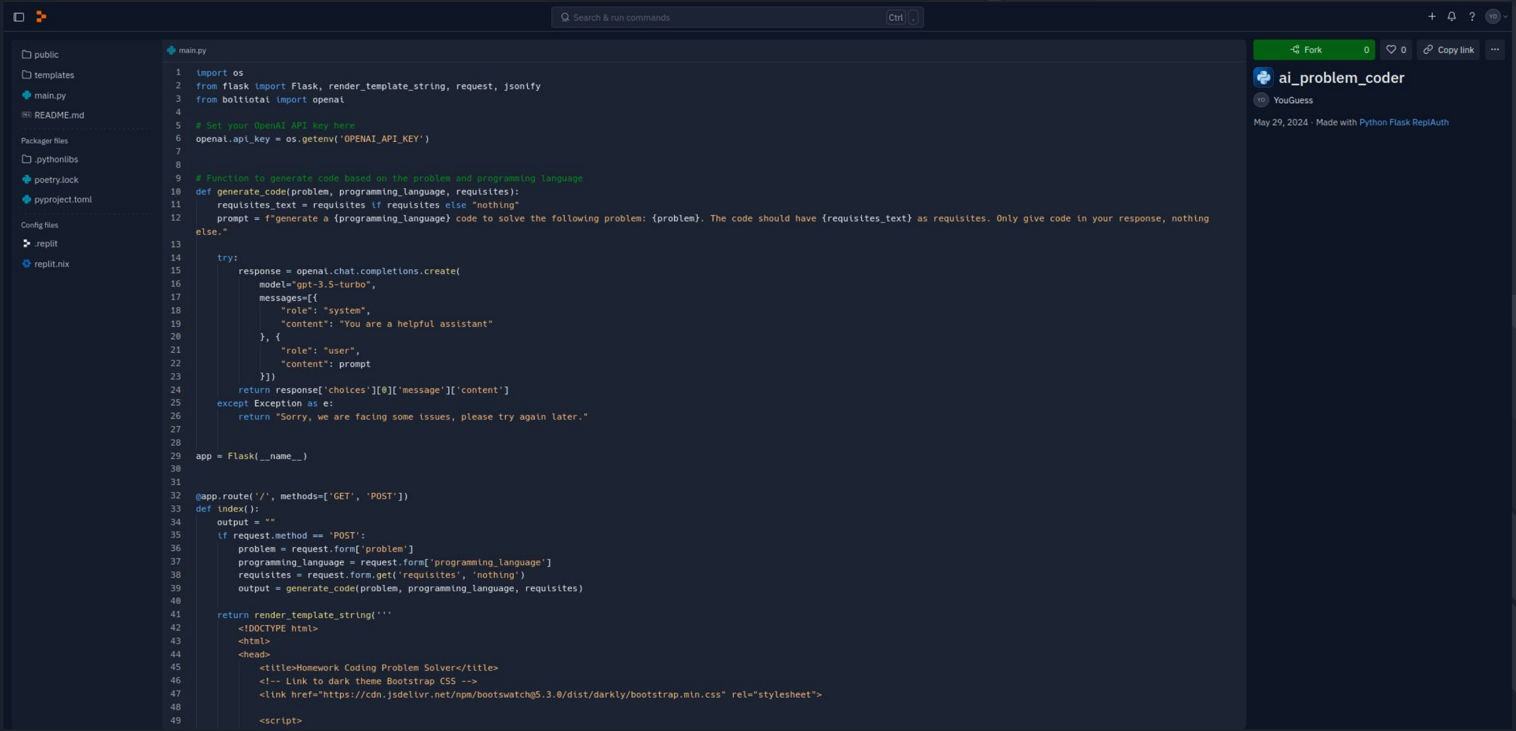
Requisites (optional):

using recursive approach

Code:

~~~
```c
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main() {
    int n = 10;
    printf("Fibonacci sequence up to %d terms:\n", n);
    for (int i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
    return 0;
}
```
~~~

Link to the Replit of project:
https://replit.com/@YouGuess/aiproblemcoder?v=1



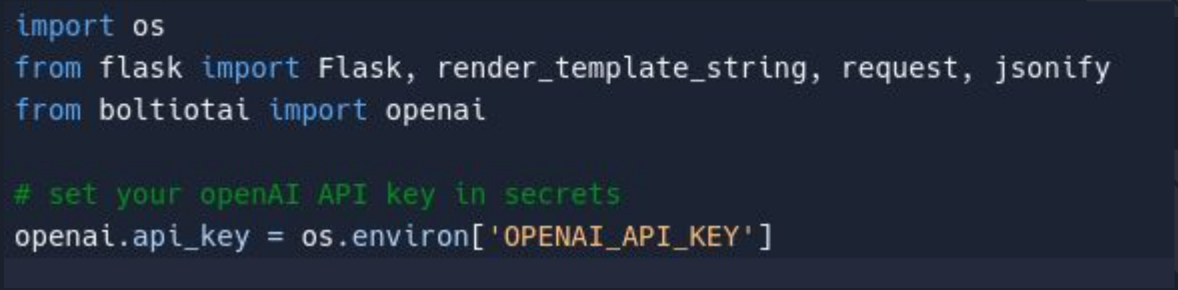
```python
import os
from flask import Flask, render_template_string, request, jsonify
from boltiotai import openai

# Set your OpenAI API key here
openai.api_key = os.getenv('OPENAI_API_KEY')


# Function to generate code based on the problem and programming language
def generate_code(problem, programming_language, requisites):
    requisites_text = requisites if requisites else "nothing"
    prompt = f"generate a {programming_language} code to solve the following problem: {problem}. The code should have {requisites_text} as requisites. Only give code in your response, nothing else."

    try:
        response = openai.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[{
                "role": "system",
                "content": "You are a helpful assistant"
            }, {
                "role": "user",
                "content": prompt
            }])
        return response['choices'][0]['message']['content']
    except Exception as e:
        return "Sorry, we are facing some issues, please try again later."


app = Flask(__name__)


@app.route('/', methods=['GET', 'POST'])
def index():
    output = ""
    if request.method == 'POST':
        problem = request.form['problem']
        programming_language = request.form['programming_language']
        requisites = request.form.get('requisites', 'nothing')
        output = generate_code(problem, programming_language, requisites)

    return render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>Homework Coding Problem Solver</title>
            <!-- Link to dark theme Bootstrap CSS -->
            <link href="https://cdn.jsdelivr.net/npm/bootswatch@5.3.0/dist/darkly/bootstrap.min.css" rel="stylesheet">

            <script>
```

How to use?

- Open the webpage.
- Type in the problem you want code solution for.
- Choose the programming language from drop-down list.
- Type in any requirements for the solution if necessary, otherwise leave blank.
- Press the “Generate” button and wait.
- You will get required code solution of the problem within a few seconds!

Writing the Code

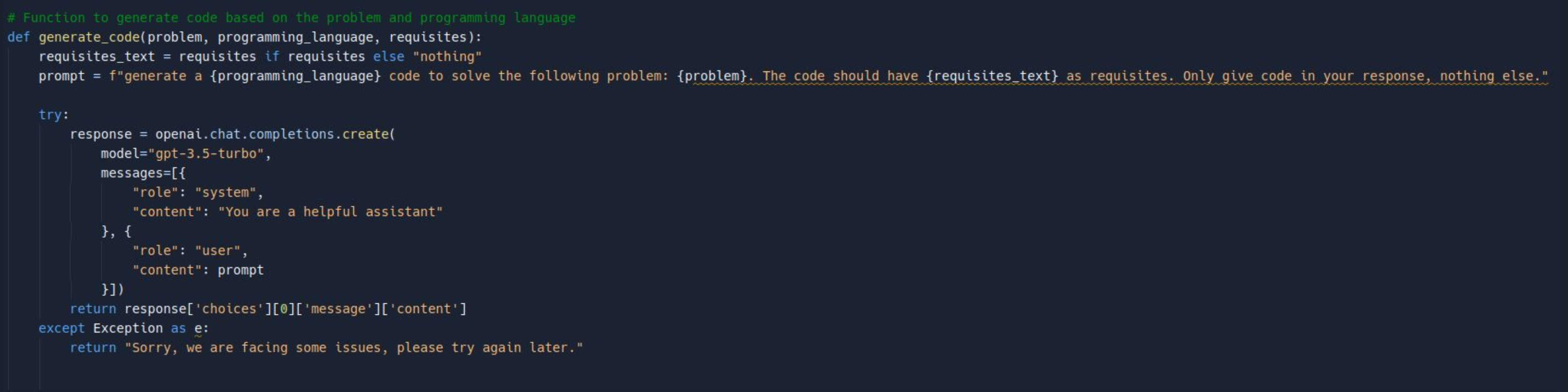
1. Importing the necessary Python modules and packages for building a web application that uses OpenAI's API



```python
import os
from flask import Flask, render_template_string, request, jsonify
from boltiotai import openai

# set your openAI API key in secrets
openai.api_key = os.environ['OPENAI_API_KEY']
```

- **'os'**: Interacts with the operating system to fetch environment variables.
- **'Flask'** components (Flask, render_template_string, request, jsonify): Builds and manages the web application.
- **'openai'** from **'boltiotai'** : Accesses OpenAI's API for generating content.

2. **‘generate_code’** to generate code solution

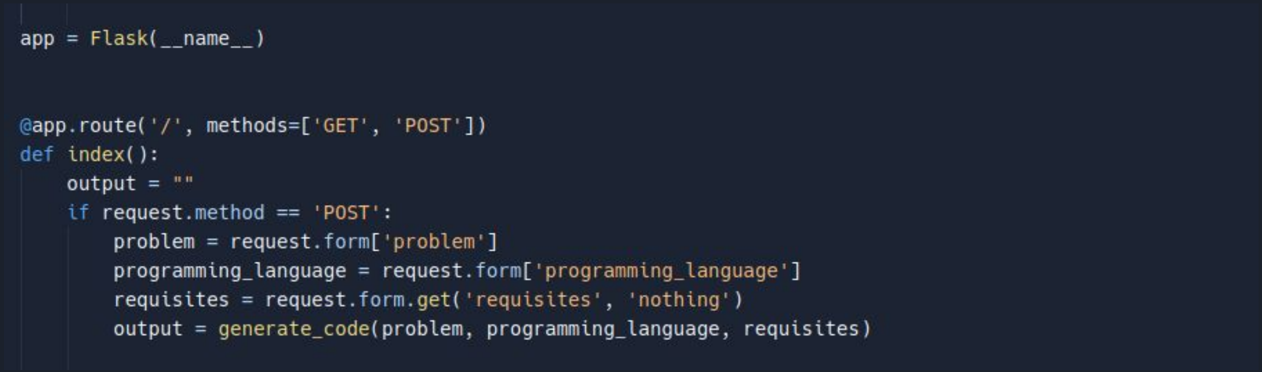


```python
# Function to generate code based on the problem and programming language
def generate_code(problem, programming_language, requisites):
    requisites_text = requisites if requisites else "nothing"
    prompt = f"generate a {programming_language} code to solve the following problem: {problem}. The code should have {requisites_text} as requisites. Only give code in your response, nothing else."

    try:
        response = openai.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[{
                "role": "system",
                "content": "You are a helpful assistant"
            }, {
                "role": "user",
                "content": prompt
            }])
        return response['choices'][0]['message']['content']
    except Exception as e:
        return "Sorry, we are facing some issues, please try again later."
```

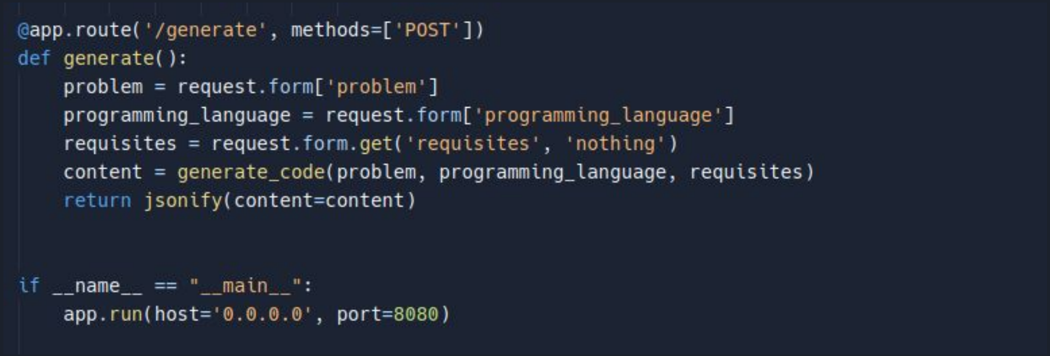
2. **‘generate_code’** to generate code solution

- The function calls the **‘OpenAI API’** to create a chat completion using the **‘GPT-3.5-turbo**’ model.
- It sends a ‘**prompt’** to the model, asking it to **generate code** to solve a **specified problem** using a specified **programming_language**, **problem** and **requisites_text** considering any additional requisites provided by the user.
- The prompt instructs the model to **only** generate code **without any additional explanations**.
- If the API call is **successful**, the function **extracts the generated code** from the API response and **returns** it.
- If an **error** occurs during the API call, the **function catches the exception** and returns a message indicating that there are issues, asking the user to **try again later**.

3. Using Python **‘Flask’**



```python
app = Flask(__name__)


@app.route('/', methods=['GET', 'POST'])
def index():
    output = ""
    if request.method == 'POST':
        problem = request.form['problem']
        programming_language = request.form['programming_language']
        requisites = request.form.get('requisites', 'nothing')
        output = generate_code(problem, programming_language, requisites)
```



```python
@app.route('/generate', methods=['POST'])
def generate():
    problem = request.form['problem']
    programming_language = request.form['programming_language']
    requisites = request.form.get('requisites', 'nothing')
    content = generate_code(problem, programming_language, requisites)
    return jsonify(content=content)


if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080)
```

3. Using Python **‘Flask’**

- **Create Flask App Instance**

```
app = Flask(__name__)
```

This line initializes a Flask web application by creating an instance of the Flask class.

- **Define Root Route ('/'):**

```
@app.route('/', methods=['GET', 'POST'])
def index():
    output = ""
    if request.method == 'POST':
    problem = request.form['problem']
    programming_language = request.form['programming_language']
    requisites = request.form.get('requisites', 'nothing')
    output = generate_code(problem, programming_language, requisites)
```

❏ This function handles the root URL (/) of the web application. It supports both GET and POST methods.
❏ If the request method is GET, it will display the initial form.
❏ If the request method is POST, it extracts the problem, programming_language, and requisites from the submitted form data and uses them to call the generate_code function. The generated code is then stored in the output variable.

3. Using Python **'Flask'**

- **Define Generate Route ('/generate')**:

```
        @app.route('/generate', methods=['POST'])
    def generate():
        problem = request.form['problem']
        programming_language = request.form['programming_language']
        requisites = request.form.get('requisites', 'nothing')
        content = generate_code(problem, programming_language, requisites)
        return jsonify(content=content)
```

  - ❑ This function handles the /generate route, it only accepts POST requests.
  - ❑ It retrieves the problem, programming_language, and requisites from the submitted form data.
  - ❑ It then calls the generate_code function with these parameters.
  - ❑ The function returns the generated code as a JSON response.

- **Run the Flask App**:

```
        if __name__ == "__main__":
        app.run(host='0.0.0.0', port=8080)
```

This block ensures that the Flask app runs when the script is executed directly. It starts the Flask web server, making the application accessible at port 8080.

3. Code for **‘Webpage’**

- **HTML and Content Structure**

```
<!DOCTYPE html>
<html>
<head>
<title>Homework Coding Problem Solver</title>
<!-- Link to dark theme Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootswatch@5.3.0/dist/darkly/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container">
<h1 class="my-4" style="font-family: monotone; font-weight: bold; color: #e06846;">Solve your coding problems instantly!</h1>
<p style="font-style: italic; font-size: 22px; color: aqua;">Use the latest AI technology to solve your problems at your fingertips.</p>

<form id="coding-form" onsubmit="event.preventDefault(); generateCode();" class="mb-3">
<div class="mb-3">
<label for="problem" class="form-label">Problem:</label>
<textarea class="form-control" id="problem" name="problem" required></textarea>
</div>
<div class="mb-3">
<label for="programming_language" class="form-label">Programming Language:</label>
<select class="form-control" id="programming_language" name="programming_language" required>
<option value="bash">Bash</option>
<option value="c">C</option>
<option value="c++">C++</option>
<option value="go">Go</option>
<option value="java">Java</option>
<option value="php">PHP</option>
<option value="python">Python</option>
<option value="ruby">Ruby</option>
</select>
```

3. Code for **‘Webpage’**

- **HTML and Content Structure**
  ...continued

```
</div>
<div class="mb-3">
<label for="requisites" class="form-label">Requisites (optional):</label>
<input type="text" class="form-control" id="requisites" name="requisites">
</div>
<button type="submit" class="btn btn-primary">Generate</button>
</form>
<div class="card">
<div class="card-header d-flex justify-content-between align-items-center">
Code:
<button class="btn btn-secondary btn-sm" onclick="copyToClipboard()">Copy</button>
</div>
<div class="card-body">
<pre id="output" class="mb-0" style="white-space: pre-wrap;">{{ output }}</pre>
</div>
</div>
</div>
</body>
</html>
```

3. Code for **'Webpage'**

- **HTML and Content Structure**
  - ★ The <head> section includes the title of the page and a link to the Bootstrap CSS for styling.
  - ★ The <body> section uses the container class from Bootstrap to center the content with some padding.
  - ★ The <h1> tag displays the page title with specific styling for font and color.
  - ★ A <p> tag provides a description of the page's purpose with italicized and colored text.
  - ★ The <form> element contains:
    - ○ A <textarea> input for the problem description.
    - ○ A <select> dropdown for choosing the programming language.
    - ○ An optional <input> field for additional requisites.
    - ○ A submit button that triggers the generateCode function without reloading the page.
  - ★ The card component is used to display the generated code:
    - ○ The card header includes a button to copy the generated content to the clipboard.
    - ○ The card body contains a <pre> element with the output ID to display the generated content, using white-space: pre-wrap to preserve formatting.

3. Code for **‘Webpage’**

- **JavaScript Functions**

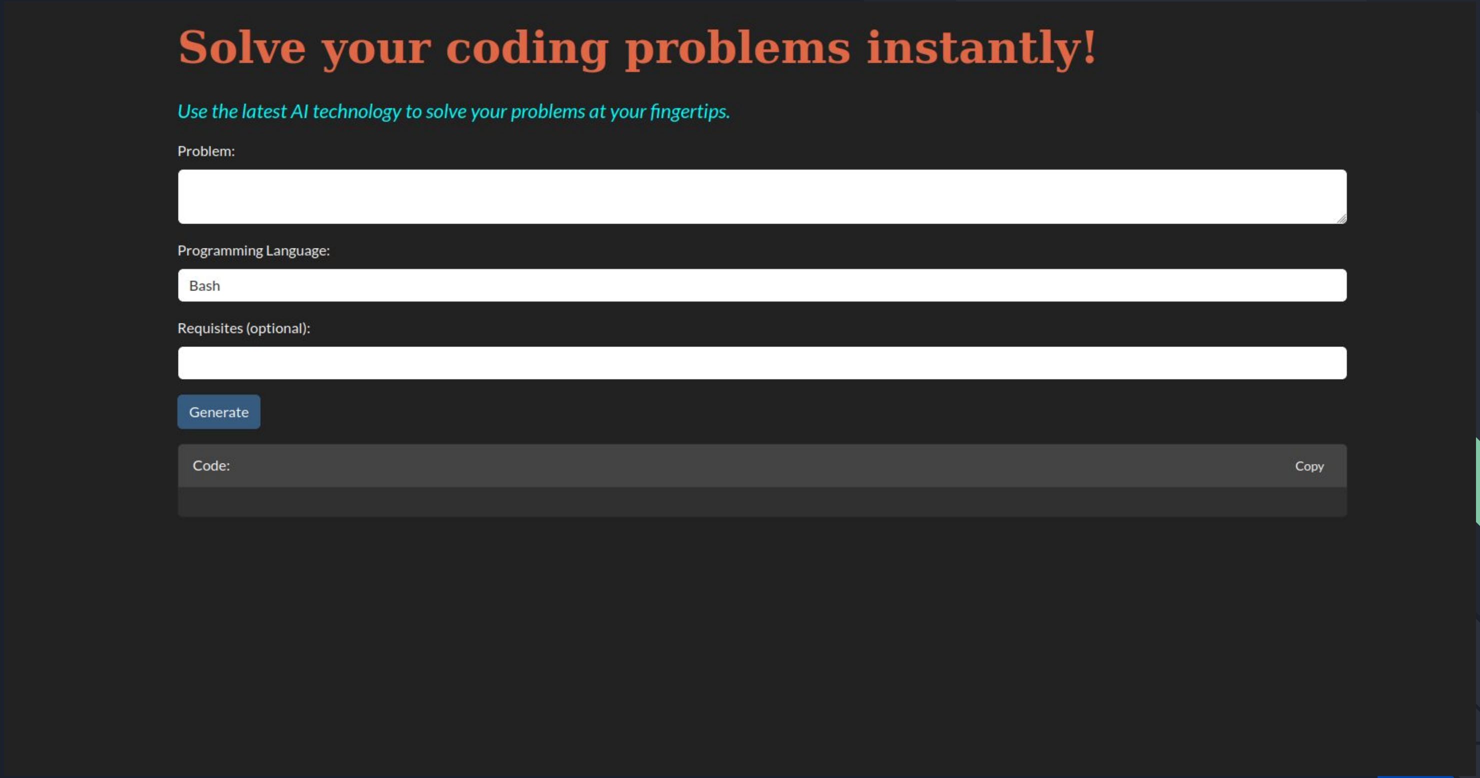
```
<script>
async function generateCode() {
const form = document.querySelector('#coding-form');
const output = document.querySelector('#output');
output.textContent = 'Generating code...';
try {           const response = await fetch('/generate', {
                method: 'POST',
                body: new FormData(form)});
if (!response.ok) {throw new Error('Network response was not ok ' + response.statusText);}
const result = await response.json();
output.textContent = result.content;
} catch (error) {          output.textContent = 'An error occurred: ' + error.message;
}}
function copyToClipboard() {
const output = document.querySelector('#output');
const textarea = document.createElement('textarea');
textarea.value = output.textContent;
document.body.appendChild(textarea);
textarea.select();
document.execCommand('copy');
document.body.removeChild(textarea);
alert('Copied to clipboard');
}
</script>
```

3. Code for **‘Webpage’**

- **JavaScript Functions**

★ **generateCode Function:**
  - This function is triggered when the form is submitted.
  - It fetches the value of the problem and programming_language entered by the user.
  - It sets the output element's text to "Generating code...".
  - It makes an asynchronous POST request to the /generate route with the form data.
  - It waits for the response, retrieves the text, and updates the output element with the generated content.
  - If an error occurs during the process, it catches the error and updates the output element with an error message.

★ **copyToClipboard Function**:
  - This function copies the content of the output element to the clipboard.
  - It creates a temporary textarea element, sets its value to the content of the output element, and appends it to the document body.
  - It selects the text in the textarea, copies it to the clipboard, and then removes the textarea element.
  - It shows an alert indicating the content has been copied.



Solve your coding problems instantly!

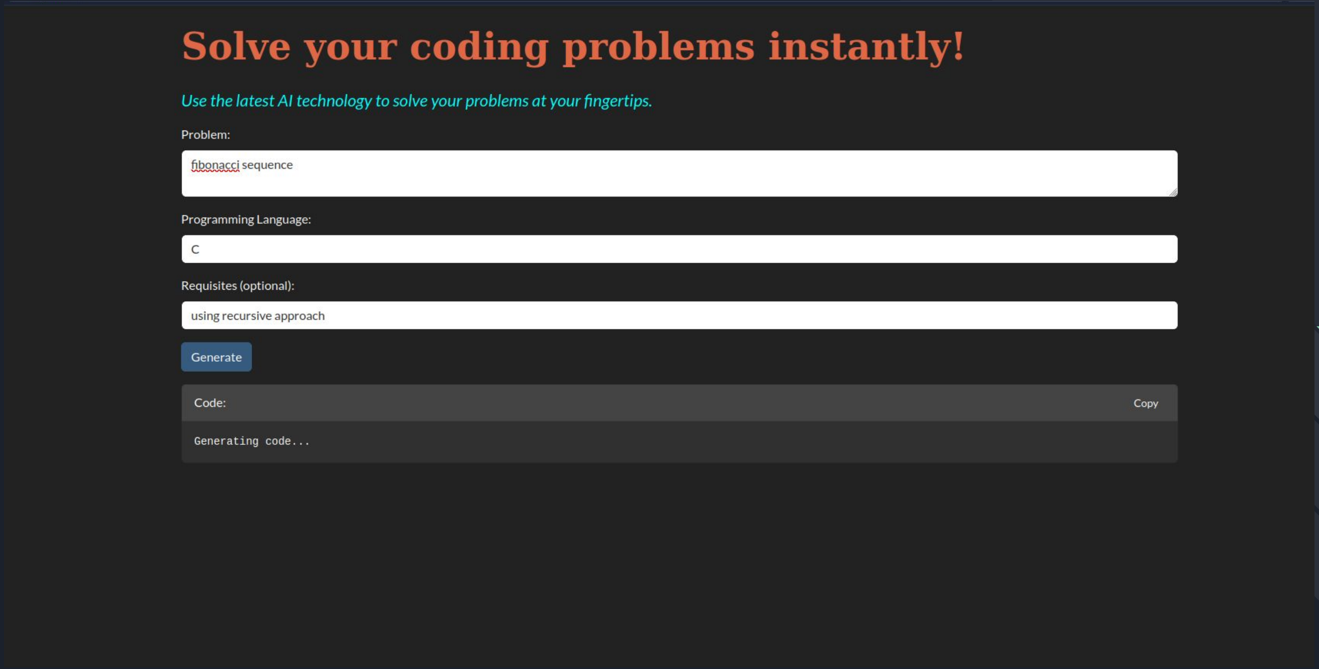
Use the latest AI technology to solve your problems at your fingertips.

Problem:

Programming Language: Bash

Requisites (optional):

Default webpage screen.



Solve your coding problems instantly!

Use the latest AI technology to solve your problems at your fingertips.

Problem:

fibonacci sequence

Programming Language:

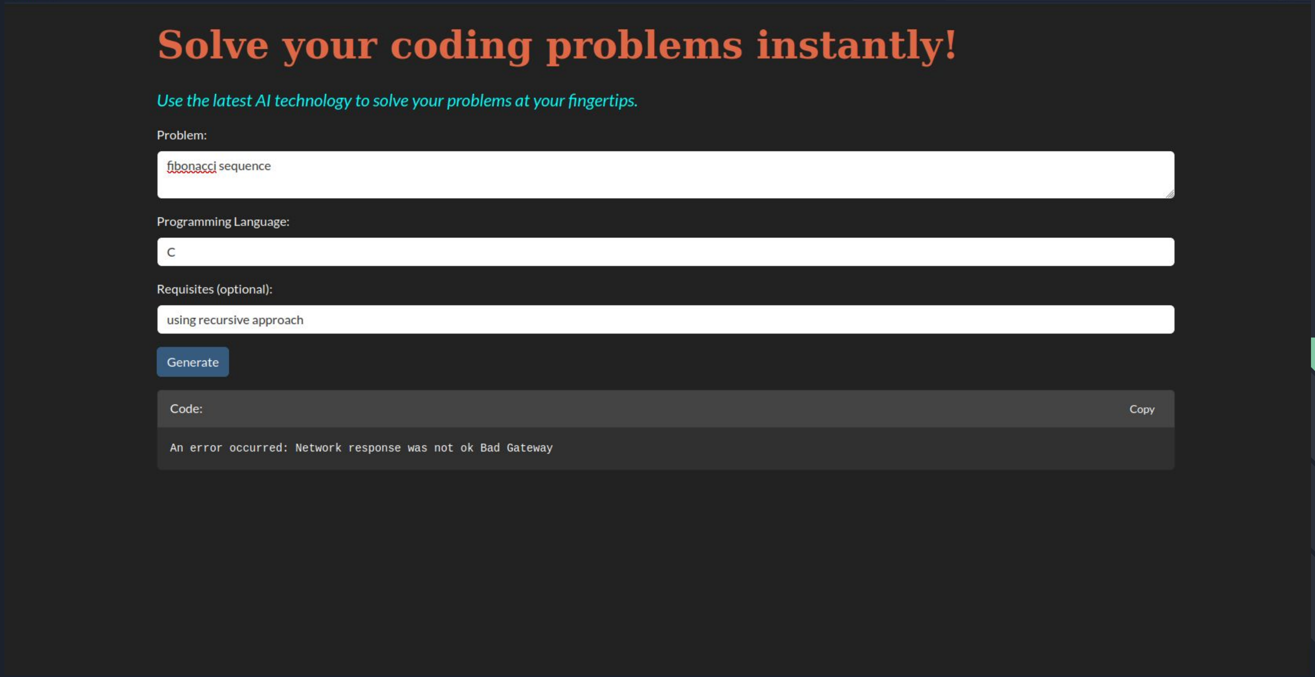
C

Requisites (optional):

using recursive approach

```
Generating code...
```

Generating code...



Solve your coding problems instantly!

Use the latest AI technology to solve your problems at your fingertips.

Problem:

fibonacci sequence

Programming Language:

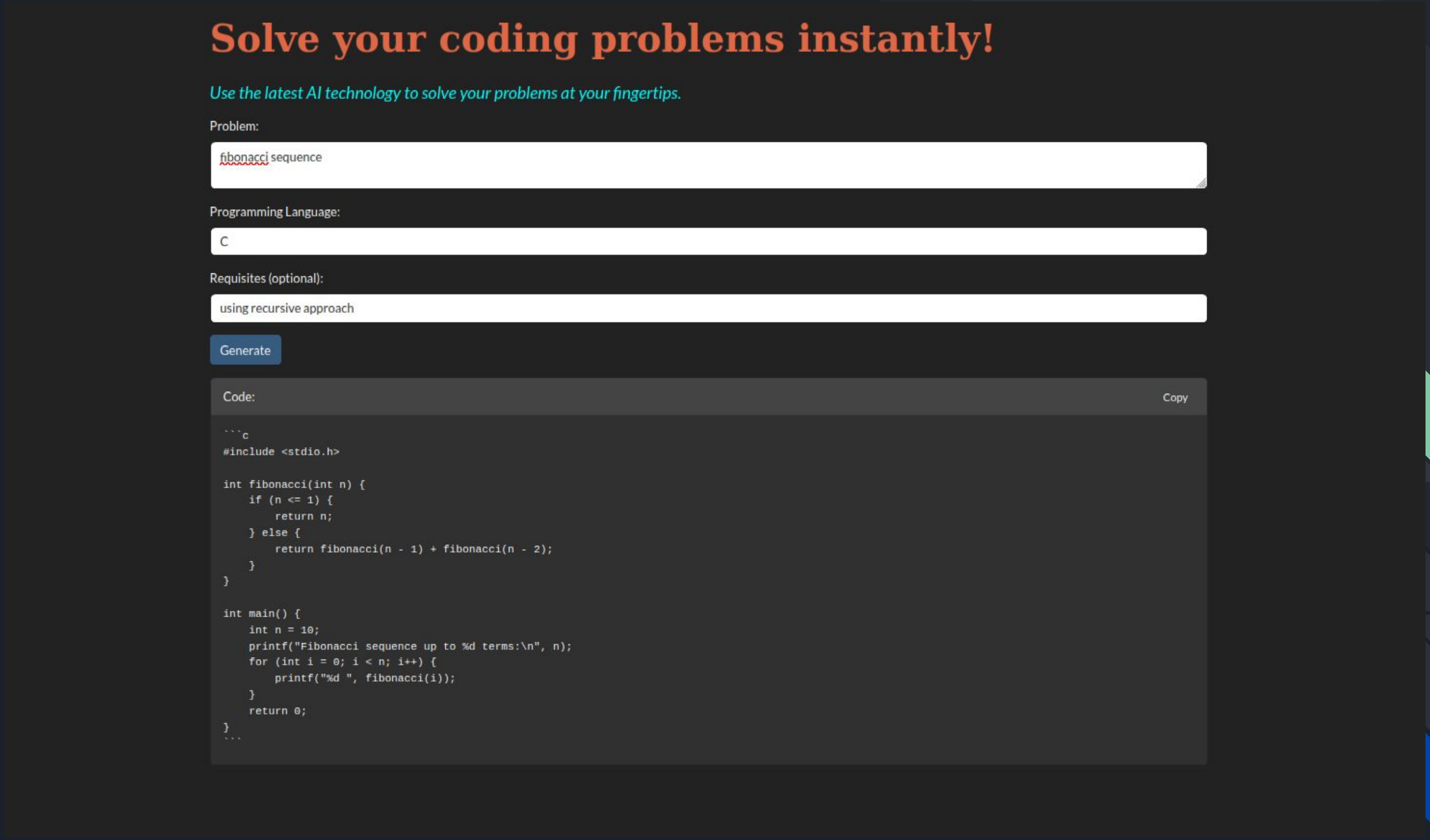
C

Requisites (optional):

using recursive approach

```
An error occurred: Network response was not ok Bad Gateway
```

Error message in case of network failure.



Solve your coding problems instantly!

Use the latest AI technology to solve your problems at your fingertips.

Problem:

fibonacci sequence

Programming Language:

C

Requisites (optional):

using recursive approach

~~~
```c
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main() {
    int n = 10;
    printf("Fibonacci sequence up to %d terms:\n", n);
    for (int i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
    return 0;
}
```
~~~

Webpage with required code solution of the problem.

Thank you.