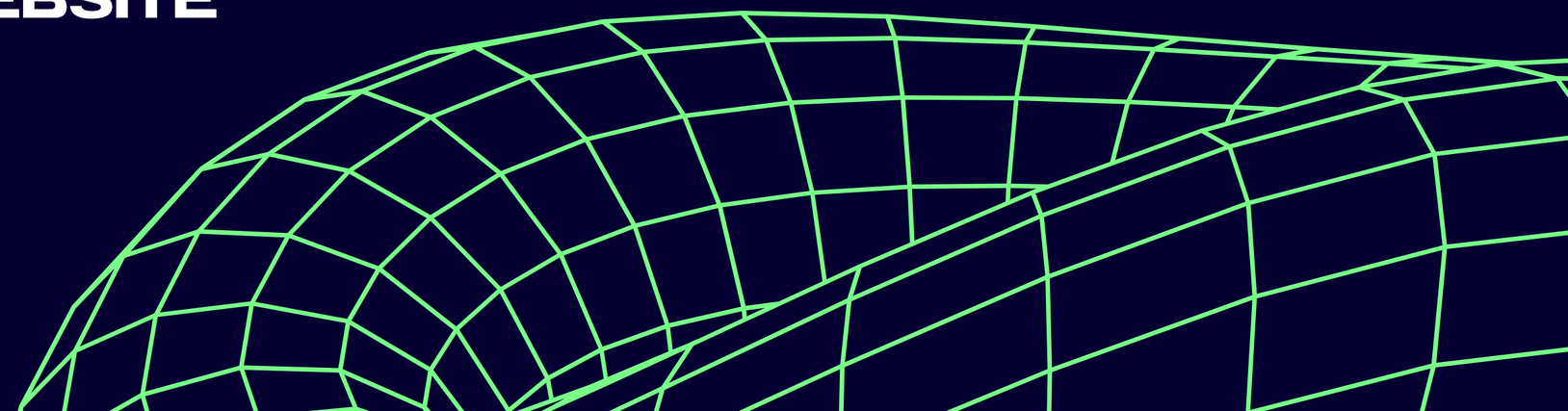
2023

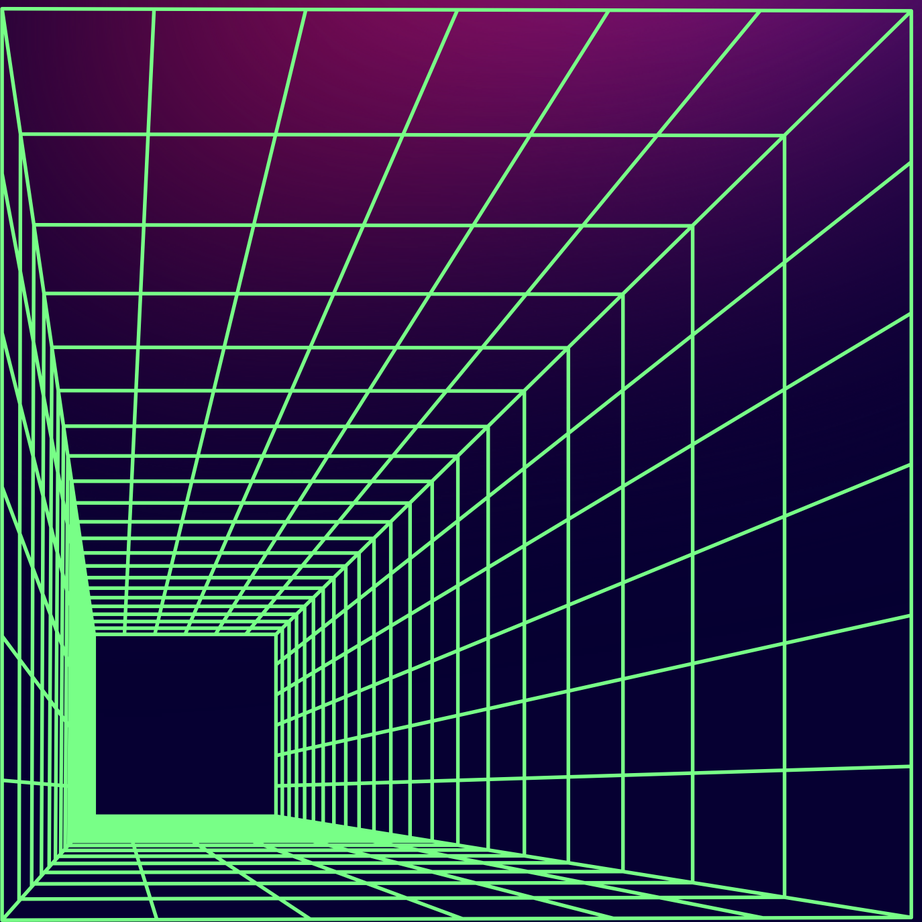
EXPLAINIQ

A STEP TOWARDS SIMPLICITY

CONTENT



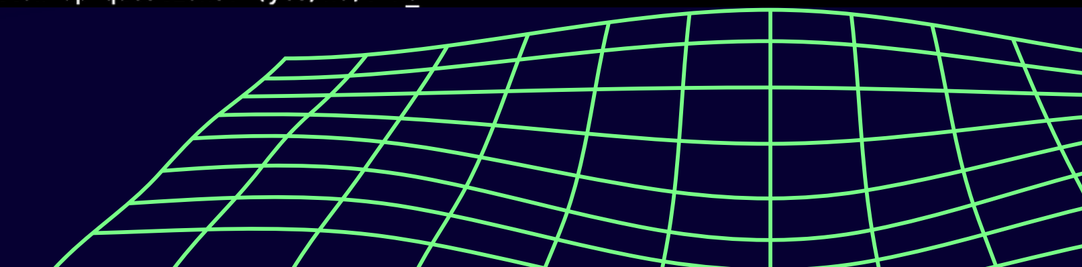
TOOLS USED



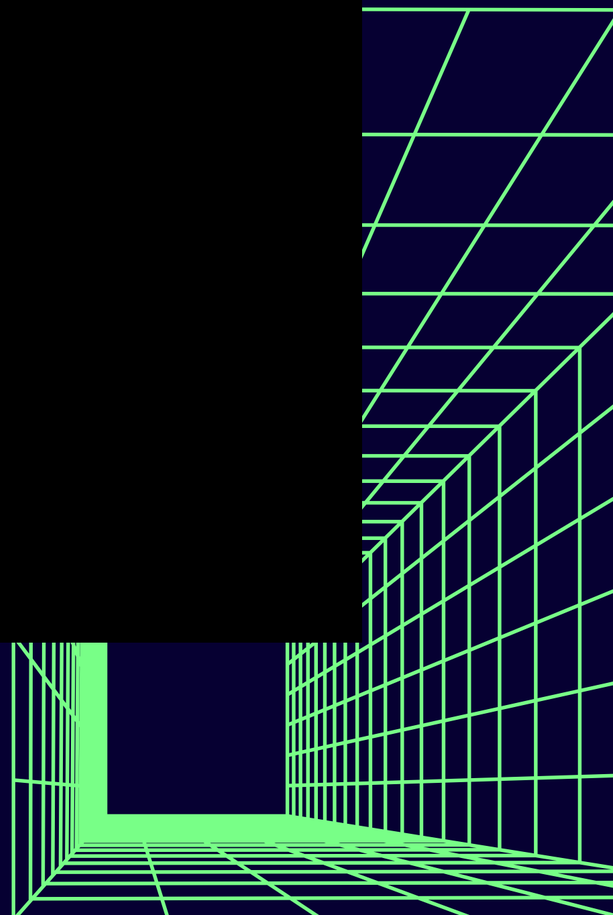
-HTML
-CSS
-React Js
-Node.Js
-Express.Js
-GPT API

API CALL

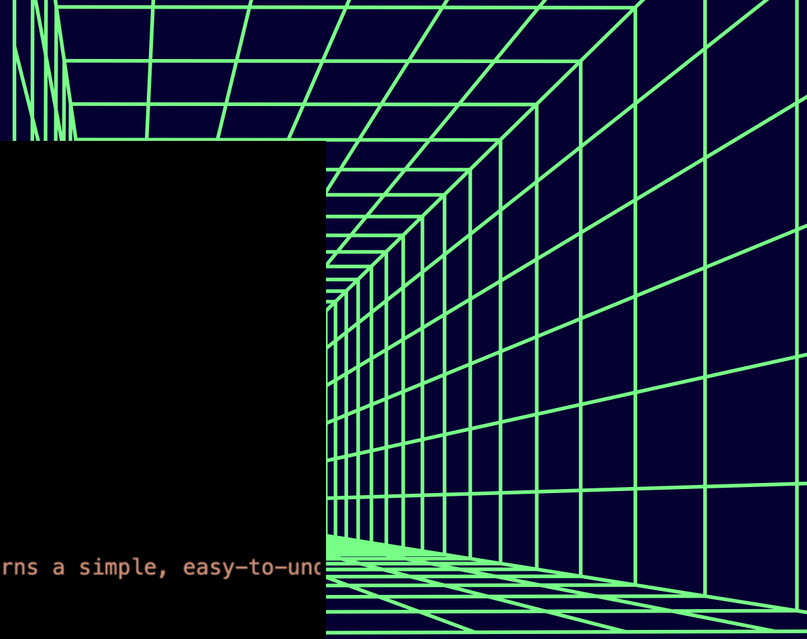
```
Explanation:
This chapter discusses the logic of compound statements. It focuses on logical forms and logical equivalence.

One specific argument form discussed is modus ponens, which has a valid form. An example is given to illustrate this:

"If p, then q. If you are in this class, then you are a USF student.
p: You are in this class.
Therefore, q: You are a USF student."

This argument is considered sound because the form used is valid and each premise is true. Being in the class implies being a USF student.

It's important to note that even if someone not in this class were to hear this argument, it remains valid, but just not sound. Making an unsound argument d
oes not provide any insight into the truth of the conclusion.
Do you have any follow-up questions? (yes/no): yes
What is your follow-up question? can you explain modus ponens
Explanation:
Certainly! Modus Ponens is a logical rule that allows us to draw a conclusion from two statements, known as the "if-then" form. It is often used in deductiv
e reasoning.

Here's how it works:

1. We start with a premise or an assumption in the form of an "if-then" statement. This is called the antecedent or the "if" part. For example, "If it is ra
ining, then the ground is wet."

2. Then, we have another statement that affirms the antecedent or the "if" part. This is called the first premise. For example, "It is raining."

3. By applying modus ponens, we can draw a logical conclusion or an affirmation based on the two statements. In this case, the conclusion is that the conseq
uent or the "then" part must also hold true. So, the conclusion inferred from the given statements is, "The ground is wet."

In summary, modus ponens allows us to affirm the consequent or the "then" part of an "if-then" statement when the antecedent or the "if" part is proven to b
e true. It is a simple yet powerful rule in logic that helps us make valid deductions.
Do you have any follow-up questions? (yes/no): no
```



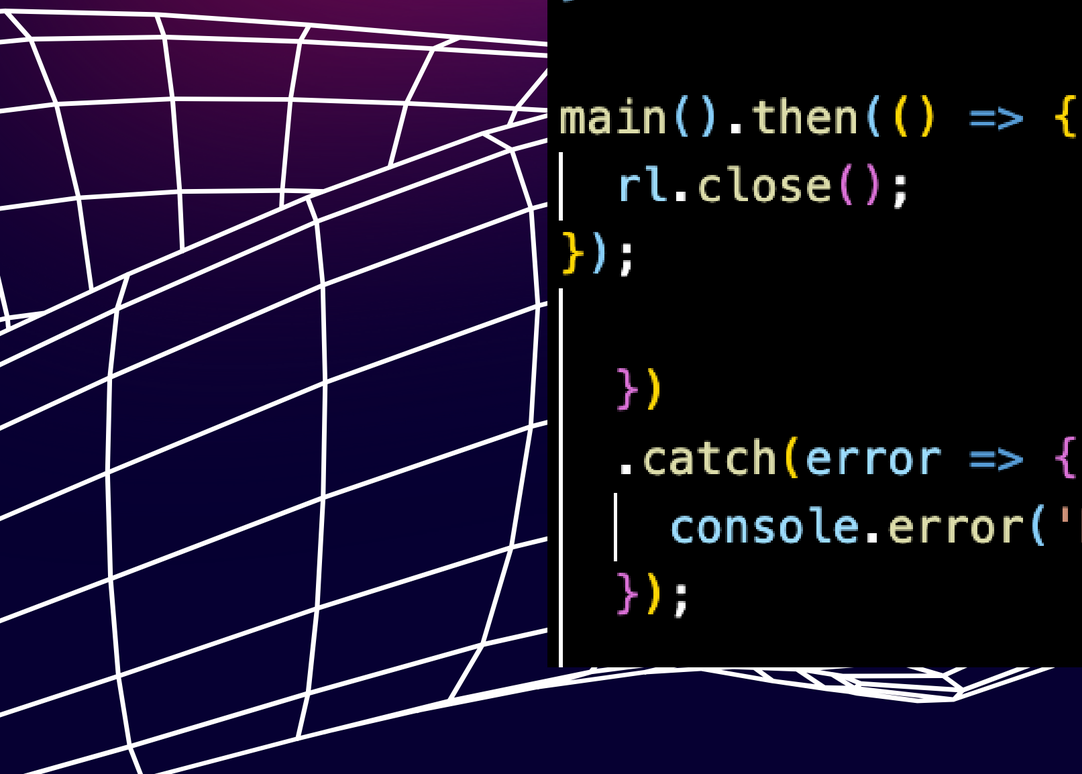
```
(base) harshitsuri@eduroam-226-24-117 HACKJAM % node openai-test.js


Chapter 2: The Logic of Compound StatementsSection 1: Logical Forms and Logical Equivalence
One of the first argument forms that we will look at in §2.3 ismodus ponens. We will
see that it has the following valid form:
Example:
If p, then q.If you are in this class, then you are a USF student.
pYou are in this class.
)q.)You are a USF student.
IThis is a sound argument because the form used is valid and each premise is
true: you are in this class and being in this class implies that you are a USF
student.
IEven if I were talking to someone not in this class, the argument remains valid.
It’s just not sound.
IObserve that making an unsound argument gives no insight into the truth of the
conclusion: someone not in this class might be a USF student or they might not
be.
John Theado, PhDIntro to Discrete StructuresOctober 10, 202314 / 246



Explanation:
This chapter discusses the logic of compound statements. It focuses on logical forms and logical equivalence.

One specific argument form discussed is modus ponens, which has a valid form. An example is given to illustrate this:

"If p, then q. If you are in this class, then you are a USF student.
p: You are in this class.
Therefore, q: You are a USF student."
```



```
//CODE FOR PDF READING AND API CALL


const fs = require('fs');
const pdf = require('pdf-parse');

let s; // Declare s outside the Promise

function extractTextFromPDF(pdfPath) {
  return new Promise((resolve, reject) => {
    const dataBuffer = fs.readFileSync(pdfPath);

    pdf(dataBuffer)
      .then(data => {
        const text = data.text;
        s = text; // Assign the extracted text to the s variable
        resolve(text);
      })
      .catch(err => {
        reject(err);
      });
  });
}

// Usage:
const pdfPath = 'test.pdf';

extractTextFromPDF(pdfPath)
  .then(extractedText => {
    // You can use the extracted text in the 'extractedText' variable, and 's' is also available here
    console.log(s); // Print the extracted text stored in 's'
    console.log("\n");
    console.log("\n");

const OpenAI = require("openai");
const readline = require("readline");

const openai = new OpenAI();
```

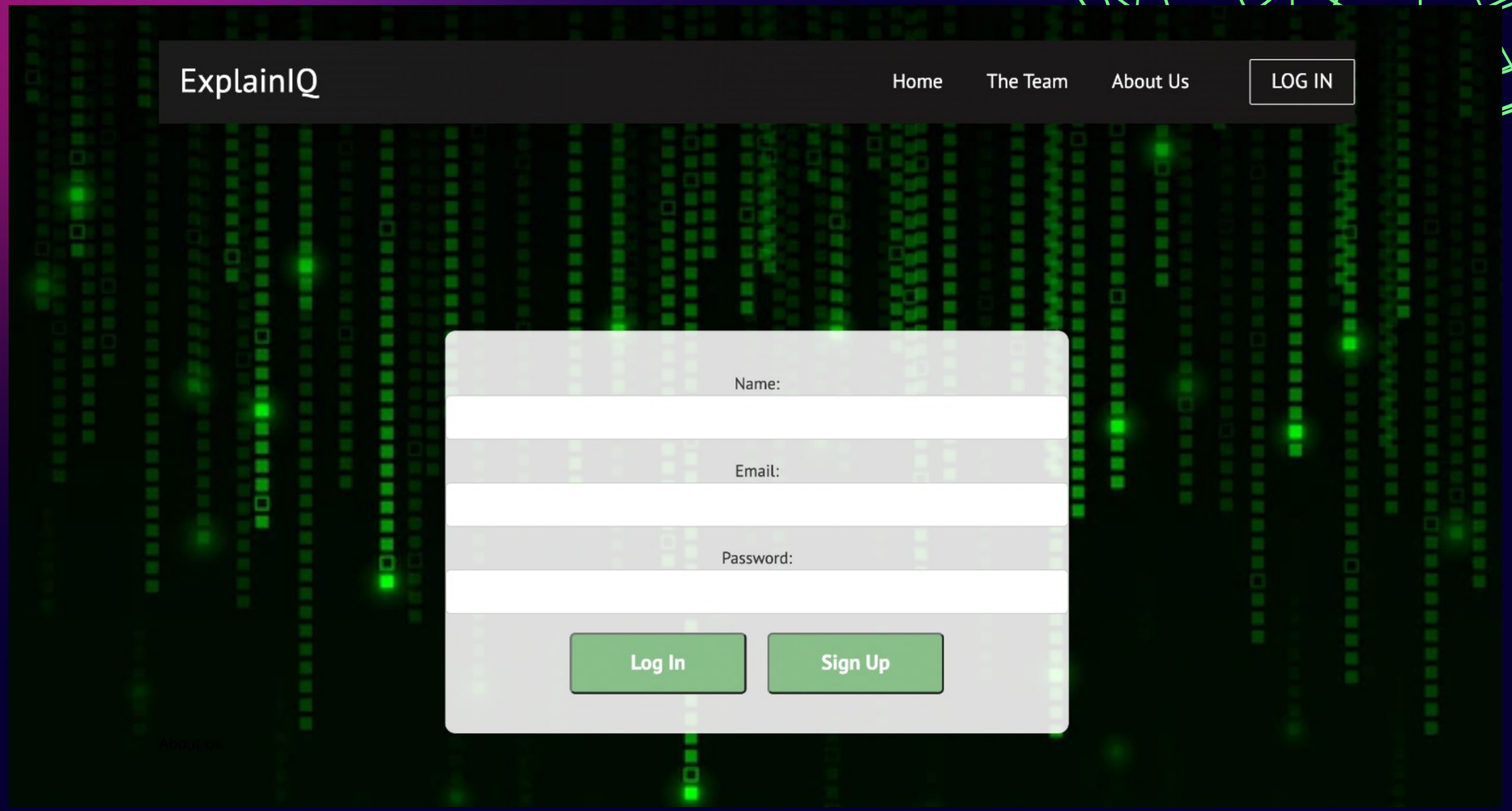


```
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

async function main() {
  let userMessage = s;

  while (true) {
    // Generate an explanation using OpenAI's API
    const completion = await openai.chat.completions.create({
      messages: [
        { role: "system", content: "Create a program that accepts user input in the form of text and returns a simple, easy-to-unc
        { role: "user", content: userMessage },
      ],
      model: "gpt-3.5-turbo",
    });

    // Extract and format the explanation
    let explanation = completion.choices[0].message.content;
    explanation = explanation.replace(/§/g, "\n§");

    // Display the explanation to the user
    console.log("Explanation:");
    console.log(explanation);

    // Ask the user if they want to ask follow-up questions
    const response = await promptUser("Do you have any follow-up questions? (yes/no): ");

    if (response.toLowerCase() === "no") {
      break; // Exit the loop if the user says "no"
    }

    // Ask the user for their follow-up question
    userMessage = await promptUser("What is your follow-up question? ");
  }
}
```

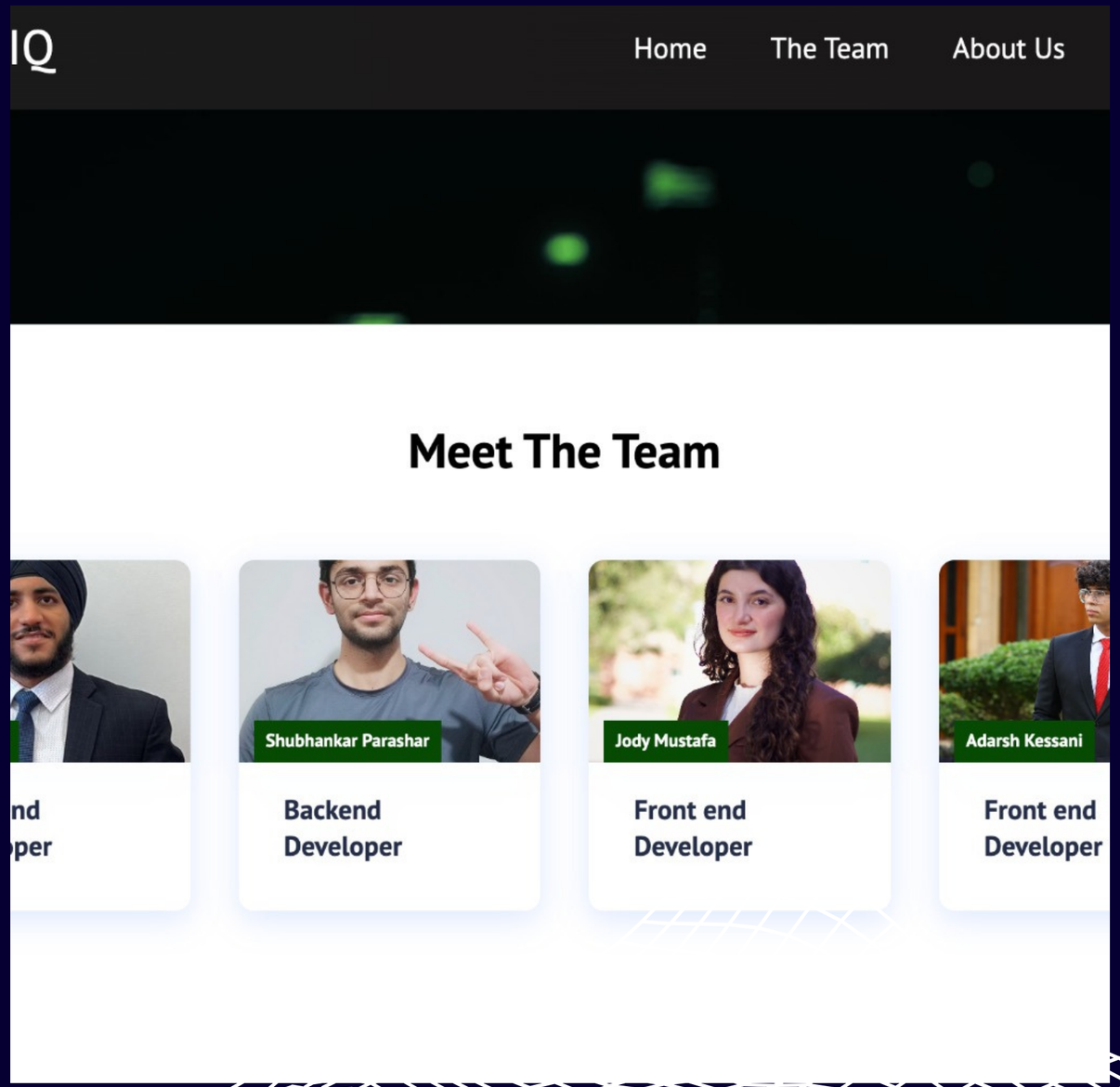


```
function promptUser(question) {
  return new Promise((resolve) => {
    // Prompt the user for input
    rl.question(question, (userInput) => {
      resolve(userInput);
    });
  });
}

main().then(() => {
  rl.close();
});

  })
  .catch(error => {
    console.error('Error:', error);
  });
```

LOG IN PAGE


ExplainIQ
Home
The Team
About Us
LOG IN
Name:
Email:
Password:
Log In
Sign Up

HOME PAGE


IQ
Home
The Team
About Us
Meet The Team
Shubhankar Parashar
Backend Developer
Jody Mustafa
Front end Developer
Adarsh Kessani
Front end Developer

HAPPY JUDGING