AI App Documentation Chatbot Task Document

This is an AI/ML project divided into two distinct tasks. The ultimate goal is to build a solution that:

1. **Automatically generates help documents** for end users by analyzing Angular app code.
2. **Develops an intelligent chatbot** that can answer user queries about the app by leveraging the help documents and connecting to a database for user-specific and public data.

Task 1: Automated User Help Document Generation

Task Title:

Code-to-Help Documentation Generator

Task Overview:

The goal is to develop a system that reads **Angular app code** and generates **end-user help documents** automatically. These documents will describe the application's navigation, functionality, and UI elements in a way that is useful for end users. The solution must be capable of processing **new/upgraded versions** of the app seamlessly and regenerating updated help documents.

Requirements:

1. Input and Processing

- Input: **Angular app source code**.
- The system must analyze:

    - **Screen Components** (e.g., forms, buttons, navigation menus).
    - **Actions** (e.g., clicks, hovers, inputs).
    - **User Flow** (how users navigate between screens).
- The solution should handle any **upgraded version** of the app and regenerate documents automatically.

2. Output Specifications

The generated help documents should be in **two formats**:

- **Human Readable Help Documents**
    - Users can refer to this if the chatbot is malfunctioning
- **Machine Readable Data**
    - **Graph Data**:
        - Nodes: Represent UI elements (buttons, forms, screens, etc.) with actions, locations, and event types.
        - Edges: Represent navigation paths or transitions between nodes.

Example:
yaml
Copy code

```
Nodes:
- Node ID: settingsButton, Action: click, Event Type: navigation
- Node ID: firstNameInput, Action: input, Event Type: valueChange

Edges:
- From settingsButton to settingsForm, Action: click
```

    - **JSON Data**:
        - A structured format describing each screen and its elements, including:
            - **Element ID**
            - **Description** (plain language description for end users).
            - **Actions** (click, input, hover, etc.).
            - **Relative Screen Location** (element hierarchy).

Example Output:
json
Copy code

```
{
  "pages": [
    {
      "name": "SettingsPage",
```

```
      "elements": [
        {
          "id": "firstNameInput",
          "description": "An input field to update your first name.",
          "action": "input",
          "relativeScreenLocation": ["Settings form", "First name
input field"]
        },
        {
          "id": "saveButton",
          "description": "Click this button to save your changes.",
          "action": "click",
          "relativeScreenLocation": ["Settings form", "Save button"]
        }
      ]
    }
  ]
}
```

3. Output Accuracy Requirements

- **Graph Data**:
  - **Node Accuracy:** 95%
  - **Edge Accuracy:** 97%
  - **User Path Accuracy:** 95%
- **JSON Data**:
  - **Completeness:** 98% (all relevant content is captured).
  - **Fidelity:** 99% (content matches the app).
  - **Structure Accuracy:** 95% (logical organization of JSON).

4. Automation and Upgradability

- The system must accept **new or upgraded versions** of the Angular app code.
- Demonstrate that the help documents are automatically updated to reflect changes in the app.

5. Video Verification Requirements

- A video demonstrating:
  - Input (Angular app code).
  - Generated **Graph and JSON outputs**.

    - Comparison between help documents from two versions of the app.
    - Clear navigation instructions derived from the generated help docs.

Expected Results:

- A **fully automated solution** that generates up-to-date help documents from Angular app code.
- Output formats must be clean, accurate, and end-user-friendly.

Task 2: LLM-Powered Help and Data Search Chatbot

Task Title:

Multimodal Help and Data-Driven Chatbot

Task Overview:

Develop a chatbot that leverages the **help documents generated in Task 1** and connects to a **database** to answer user queries. The chatbot must be capable of understanding natural language questions and providing accurate, human-readable responses based on app navigation, functionality, and user/public data.

Requirements:

1. Input and Integration

- Inputs:
    - **Help Documents** (Graph and JSON data) from Task 1.
    - **Database Connection**:
        - A **NoSQL MongoDB** schema with tables, columns, and meaningful column descriptions.
        - Schema updates should be **automatically processed** to keep the chatbot updated.
- Queries can include:
    - App navigation questions.
    - App functionality questions.

    - Data-related questions (user-specific and public data).

2. Chatbot Output Specifications

The chatbot should:

- Provide **clear, step-by-step navigation instructions** derived from the help documents.
- Fetch and display **user-specific data** and **public data** using MongoDB queries.
- Example responses:

Navigation Question:
"How do I navigate to the settings and change my name?"
Response:
"Start at the HomePage, click on the 'settingsButton' in the navigation bar. On the SettingsPage, find the 'firstNameInput' field, update your name, and click the 'saveButton' to save changes."

Data Query:
"What is my first name?"
Response:
"Your current first name is Abdul."

3. MongoDB Integration

- The chatbot must dynamically generate MongoDB queries from natural language inputs.

Example Query:
js
Copy code
```
db.users.find({ firstName: "Abdul" })
```

-
- Demonstrate the ability to:
    - Retrieve accurate results.
    - Process user-specific and public data.

4. Continuous Learning Mechanism

- Integrate a feedback loop:
    - Allow users to provide feedback.
    - Demonstrate the chatbot's ability to adapt and improve its answers.

5. Accuracy Requirements

- **Navigation Questions:** 99% accuracy.
- **Data Retrieval:**
    - **One-to-One Queries:** 99% accuracy.

    - **One-to-Many Queries:** 99.99% accuracy.

6. PUT Endpoint Configuration

- Allow integration of a **PUT endpoint URL** to test MongoDB query flows on our end.

7. Video Verification Requirements

- A video demonstrating:
    - Input: Help Documents + Database Schema.
    - At least **10 natural language queries** answered.
    - MongoDB query generation and results retrieval.
    - Feedback mechanism for chatbot improvement.
    - Continuous updates to the chatbot based on schema changes.

Expected Results:

- A chatbot that accurately answers app navigation and functionality questions using Task 1 outputs.
- Integration with MongoDB to fetch and display user-specific and public data.
- A feedback mechanism for continuous improvement.

Notes:

- Both tasks must provide **end-to-end solutions**.
- Use **free open-source libraries** only.
- Maintain the required **video progress updates** and QA processes.
- All code will be reviewed via GitHub.

Each milestone includes clearly defined accuracy requirements and video deliverables for sign-off.

Task 1: Automated User Help Document Generation

Task Title:

Code-to-Help Documentation Generator

Task Overview:

Develop a system that reads **Angular app code** and generates up-to-date **end-user help documents** in two formats:

1. **Human-readable documents** (for users).
2. **Machine-readable Graph and JSON data** (for the chatbot in Task 2).

The solution must handle **new/upgraded app versions** automatically.

Milestones for Task 1

Milestone 1: Prototype (95% Accuracy)

Requirements:

- Input: Angular app code (Version 1).
- Generate help documents with the following accuracy:
    - **Graph Data:**
        - Node Accuracy: 95%
        - Edge Accuracy: 97%
        - Path Accuracy: 95%
    - **JSON Data:**
        - Completeness: 98%
        - Fidelity: 99%
        - Structure Accuracy: 95%
- Deliverables:
    - Human-readable and machine-readable outputs.
    - A **video demonstration** showing:
        - Input (app code).
        - Generated Graph and JSON outputs.
        - At least 5 navigation paths visually validated from code to output.

Milestone 4: Production (99.9% Accuracy)

Requirements:

- Input: Angular app code (Upgraded Version).

- Update the system to handle app changes with the following accuracy:
    - **Graph Data:**
        - Node Accuracy: 99.9%
        - Edge Accuracy: 99.9%
        - Path Accuracy: 99.9%
    - **JSON Data:**
        - Completeness: 99.9%
        - Fidelity: 99.9%
        - Structure Accuracy: 99.9%
- Deliverables:
    - Re-run the solution on an upgraded app version.
    - A **video demonstration** showing:
        - Comparison between old and updated help documents.
        - Verification of accuracy improvements.
        - Navigation instructions derived from regenerated outputs.

Expected Results:

- A fully automated solution generating accurate help documents from Angular app code.
- Outputs must be clean, user-friendly, and end-to-end testable.

Task 2: LLM-Powered Help and Data Search Chatbot

Task Title:

Multimodal Help and Data-Driven Chatbot

Task Overview:

Develop a chatbot that uses:

1. The **help documents** (Graph and JSON data) from Task 1.
2. A **MongoDB database connection** for answering user-specific and public data-related queries.

The chatbot must process natural language queries, dynamically generate MongoDB queries, and return accurate answers.

Milestones for Task 2

Milestone 2: Prototype (95% Accuracy)

Requirements:

- Input:
  - Help documents (Graph and JSON) from Task 1.
  - Database schema (mocked).
- Output:
  - Accurate responses to navigation and simple data queries (95% accuracy).
- Deliverables:
  - A **video demonstration** showing:
    - At least **5 navigation queries** answered using help documents.
    - At least **5 data queries** answered using MongoDB.
    - Generated MongoDB queries and database outputs.

Milestone 3: Production (99.9% Accuracy)

Requirements:

- Input: Updated help documents (from Milestone 4) and live or upgraded database schema.
- Output:
  - Achieve 99.9% accuracy for:
    - **Navigation Questions**.
    - **One-to-One Data Queries**.
    - **One-to-Many Data Queries**.
- Deliverables:
  - A **video demonstration** showing:
    - **10 navigation queries** (paths validated step-by-step).
    - **10 data queries** (MongoDB query → database result → human-readable output).
    - Continuous improvement using feedback mechanisms.
  - Demonstration of schema adaptation:
    - Show chatbot updating itself when a new table or column is added to the database schema.

Expected Results:

- A robust chatbot that:

- Answers app navigation and data queries with near-perfect accuracy.
- Dynamically adapts to schema changes.
- Incorporates a feedback loop for continuous improvement.

Project Deliverables

For each milestone, the following deliverables are mandatory before sign-off:

1. **Deployed Instance:**
   - A working, deployable instance of the solution for hands-on testing.
2. **Bundled Project:**
   - Complete project code (bundled for local testing).
3. **Video Demonstration:**
   - Step-by-step validation of outputs and accuracy requirements.

Final Notes:

- Use **free, open-source libraries** only.
- Maintain weekly video progress updates (via **Screenrec.com**).
- Code reviews will be performed via **GitHub PR** before approval.
- Stress testing will be conducted on our machines using the bundled project.

Milestone Summary

Milestone	Task	Accuracy Requirement	Key Outputs
Milestone 1	Task 1 - Prototype	95%	Graph + JSON Outputs, Video Demo
Milestone 2	Task 2 - Prototype	95%	Chatbot Demo, Navigation/Data Queries
Milestone 3	Task 2 - Production	99.9%	Advanced Accuracy, Feedback Adaptation
Milestone 4	Task 1 - Production	99.9%	Updated Graph/JSON Outputs

ADDITIONAL INFORMATION

This is an AI/ML project. The final solution you make should take Angular App code, JSON about the action elements on the page (routing elements.. etc.), screenshots, and a statement from the user like this:
"How do I navigate to the settings and change my name?"

Then, if the model was functioning correctly, the output/answer would look something like this: First, you will need to log in using the application credentials that you were given when you signed up. If you haven't signed up, do so by pressing the Register button on the login screen. Once you have signed up and logged in you should look at the top right corner of the screen while you are on the dashboard. You should see a settings icon. Click that and you will see a list. At the bottom of the list, there is a button called User Profile. Click that and you will see a name input. There you can enter a new name and click the button at the bottom labeled save to save this new User Profile Setting. We hope this answer was helpful. Let us know from where you would like to navigate in the app for more detailed information.

So it will analyze an Angular App, make in-memory notes/docs about it, and then use those docs to answer questions.

We have started the project for you.

Breaking it down:
This project can be separated into two tasks. The first part makes the data that is then used by the second part to turn it into the final output printed in the above project summary.

Task 1 **Task 2**
Angular App → Human readable App Description → Question about App → Answer about App

Prerequisites:

- We require an end-to-end solution, thus you provide your data, compute… etc.

- You make a Fiverr Custom Offer, no upfront money
- You are okay with a Prototype and Production Milestones
- Only use/fork from free open-source libs, no API keys
- Review process before milestone is accepted
    1. **30-minute video updates** of your progress must be submitted every Monday and upon request (use Screenrec.com)
    2. QA on a deployed instance
    3. Code review on GitHub
    4. Stress test

Task 1

Task Title: CODE SUMMARIZATION

Task Overview:

We are seeking a specialist with expertise in generating code summaries for Angular projects and integrating these summaries with a custom-developed LLM model. The format of the code summaries should probably be in Graph (for the routing structure between screens) and JSON (for screen descriptions) format. The ideal candidate should have substantial experience in LLM models, particularly in processing complex code structures. Key areas of focus include:

- **Code Summary Generation and LLM Model Adaptation:** How will you select and utilize tools for generating code summaries from an Angular project, and adapt the LLM model to process these summaries effectively?
- **LLM Model Efficiency and Angular Specifics:** What strategies will you implement to optimize the LLM model's efficiency, specifically tailored to Angular or JavaScript code architecture while ensuring data privacy?
- **Customization and Continuous Improvement:** Describe your approach to customizing the LLM model for interpreting Angular project structures and integrating continuous user feedback for progressive enhancement.

I/O: Input will be a javascript application, the output will be a structured Graph and JSON data

Prerequisite:

- We are in need of an end-to-end solution, thus you provide your own data.
- The ability of us to make you a Fiverr Custom Offer, no upfront money
- You are okay with a Prototype and Production Milestones, we will handle the deployment
- Only use/fork from free open-source libs, no API keys
- A 30-minute video update of your progress every Monday. This helps us quickly see who is serious about the project. Videos showing the details of your input and output of your

model processing with at least 5 samples. No code explanation is needed in the video. We will do code reviews of your code via GitHub PR before approval.
- Your solution should be hosted locally on your machine

Considerations:

- **Preparing the Description of an Angular Project:**
  - **Objective:** Identify and utilize the most effective tool for generating a detailed description of an Angular project, prioritizing compatibility with LLM model processing and focusing on Angular or JavaScript architecture. The description should facilitate documentation on navigation and functionality within the project. Your model should give the same result as a senior developer who was looking at code and detailing what the UI looks like and what it does. Documentation should be created to be consumed by the second part of this project.
  - **Reference Repositories:** Funcom, ICPC2020_GNN, Attn-to-FC, Compodoc.
  - **Expected Results:** A functional solution capable of generating a comprehensive code summary from a given project path.
- **Use In-House LLM Solution for Processing the Code Summary:**
  - **Objective:** Fine-tune an in-house LLM model based on the generated code summaries and Angular architecture. The model should be able to answer specific queries about the project, such as locating where profile details can be changed.
  - **Expected Results:** The model reliably interprets and responds to user queries regarding project navigation, with training guidelines for future modules or components.

Graph Accuracy Requirements

Node Accuracy: The model should correctly identify at least 95% of all nodes (activities, services, etc.) within the application.
Edge Accuracy: At least 97% of the edges (transitions between activities or services) identified by the model should correctly represent the actual possible transitions within the application.
Path Accuracy: The model should accurately represent at least 95% of all possible user flow paths within the application.

JSON Content Accuracy Requirements

Completeness: The JSON output should capture 98% of all relevant content (text, images, buttons, etc.) from the application.
Fidelity: The content within the JSON should have a 99% fidelity rate, meaning it should be an almost exact match to the original content in terms of text accuracy, image representation, etc.
Structure Accuracy: The JSON structure should logically organize the content in a way that reflects the app's structure with a 95% accuracy rate.

Overall Model Accuracy

- The model should meet all the above individual accuracy requirements while achieving an overall accuracy rate of 95% when considering both the graph and JSON content together.

Estimate Request:

We are seeking detailed estimates, including timelines for each milestone. Proposals for subdividing the project into more manageable phases are welcome for improved project management.

GRAPH Data Example OUTPUT

- Nodes:
  - Node ID: logo, Coordinates: (50, 20), Action: click, Event Type: navigation
  - Node ID: menu, Coordinates: (100, 20), Action: hover, Event Type: expand
  - Node ID: contactButton, Coordinates: (200, 150), Action: click, Event Type: navigation
- Edges:
  - From logo to contactButton, Action: click
  - From menu to contactButton, Action: hover

JSON Data Example OUTPUT


```
{
  "pages": [
    {
      "name": "HomePage",
      "elements": [
        {
          "id": "logo",
          "description": "The website's logo, situated at the top-left corner, serves as a clickable element that allows users to return to the homepage from any page within the site.",
          "action": "click",
          "coordinates": {"x": 50, "y": 20},
          "eventType": "navigation"
        },
        {
          "id": "menu",
          "description": "This is the main navigation menu, positioned at the top of the page. It expands upon hover to reveal additional navigation options to the user, facilitating easy access to various site sections.",
          "action": "hover",
          "coordinates": {"x": 100, "y": 20},
          "eventType": "expand"
        },
        {
          "id": "contactButton",
```

```
        "description": "A prominently placed button that prompts users to visit the contact page. This call-to-action button is designed to stand out and attract user attention, encouraging them to engage with the site.",
        "action": "click",
        "coordinates": {"x": 200, "y": 150},
        "eventType": "navigation"
      },
      {
        "id": "footer",
        "description": "The footer section contains supplementary information about the website, including copyright details, links to privacy policies, and potentially additional navigation links. It spans the bottom of the page and is designed for reference without interactive actions.",
        "action": null,
        "coordinates": {"x": 0, "y": 600},
        "eventType": null
      }
    ]
  },
  {
    "name": "ContactPage",
    "elements": [
      {
        "id": "contactForm",
        "description": "A form designed to facilitate user communication with the website administrators or company. Users can input their details and inquiries, which are then submitted for processing and response. This element is central to user interaction on the contact page.",
        "action": "submit",
        "coordinates": {"x": 120, "y": 300},
        "eventType": "formSubmission"
      },
      {
        "id": "backButton",
        "description": "A navigation element that allows users to return to the previously viewed page. It's typically used to enhance user navigation experience by providing a simple and intuitive means to go back without using the browser's back button.",
        "action": "click",
        "coordinates": {"x": 30, "y": 350},
        "eventType": "navigation"
      },
      {
        "id": "header",
        "description": "The header on the contact page provides context and branding continuity. It usually contains the logo, potentially a navigation menu, and could also feature a heading or title that indicates the user is on the contact page.",
```

```
        "action": null,
        "coordinates": {"x": 0, "y": 0},
        "eventType": null
      }
    ]
  },
  {
    "name": "MarketingPage",
    "elements": [
      {
        "id": "signupButton",
        "description": "This button is a call to action for the user to sign up, usually associated with a service or newsletter. Clicking this button typically directs the user to a form or another page where they can complete their registration or subscription.",
        "action": "click",
        "coordinates": {"x": 150, "y": 250},
        "eventType": "formSubmission"
      },
      {
        "id": "learnMore",
        "description": "A link or button that, when hovered over, provides the user with additional information about a product or service. It's intended to offer users an easy way to access more detailed information without cluttering the initial view.",
        "action": "hover",
        "coordinates": {"x": 220, "y": 200},
        "eventType": "expand"
      },
      {
        "id": "testimonial",
        "description": "This section features comments or endorsements from customers or clients, serving as social proof to potential new customers. The testimonials are displayed to build trust and credibility without requiring user interaction.",
        "action": null,
        "coordinates": {"x": 300, "y": 450},
        "eventType": null
      }
    ]
  }
 ]
}
```

Continuous Learning: Customize the LLM model for interpreting Angular project structures and integrating continuous user feedback for progressive enhancement. Feedback is coming from users so you could identify each part, collect feedback, annotate, retrain, and evaluate.

Task 2

Task Title: MULTIMODAL DATA-INTEGRATED CHATBOT

Task Overview:

The goal is to create a multimodal chatbot that functions as a custom virtual assistant for whichever app description you provide to it. The App description should be given in the form of Graph data, or JSON data. Your solution will need to be able to fetch database app data from NoSQL MongoDB APIs by reading an NL DB schema and constructing the query. Your solution should allow non-technical users to add data in these formats easily.

Input/Output (I/O) (when sending progress updates, show us this):

- **Input**: Application data in the form of Natural language queries and data in Graph, JSON, and MongoDB formats, and User documentation and MongoDB formats representing an application.
- **Output:** Natural language responses from the chatbot, demonstrating understanding of the application from the user's perspective.

Milestones:

1. Prototype: When testing chatbot accuracy using our own mocked Graph, JSON, and NoSQL DB app data we need 99 of 100 implicit directions to generalize to our dataset and give a correct answer about app functionality, navigation instructions, or data lookup and analysis
2. Production: 99.99%

Key Responsibilities:

- Develop an AI chatbot capable of processing natural language questions along with Graph, and JSON data as input. The output should consist of detailed answers regarding the website, including usage and navigation instructions.
- For queries that would require current user or public app data, the chatbot should be able to read the current MongoDB schema, send a MongoDB query to an API, retrieve user information, process it, and provide human-readable output.
    - Since you can directly connect to MongoDB in the Python script, you can execute the MongoDB query and produce the output. However, you should provide us with the ability to add the PUT endpoint URL for testing this flow on our end. The PUT endpoint will receive the MongoDB query in the request payload and return the output in JSON format.
    - You can assume the Field names to be meaningful so that you can search among them and find what the user might be looking for (example: usersFirstName)
- Ensure the chatbot can self-train and adapt to changes in MongoDB database schema or Graph and JSON data structures. **Provide a plan outlining how this self-training and adaptation process will be managed.**
- Integrate feedback mechanisms to continuously improve chatbot performance
- Using the output shown above, we will feed that JSON and GRAPH information into an LLM which will go on to answer direct questions. Make sure the above output has information that would help the LLM answer a question like this. This is how you would test that part. Again your output is the precursor to this, but it must have the relevant information.
    - Ask it: "How would I navigate to the Settings Screen and change my name if I started at the Home Screen?"
    - Depending on the app, it should answer something like this: "Hello, to navigate to the Settings Screen and change your name starting from the Home screen you will need to click on the User Menu in the top right corner. This will cause a dropdown to open where you will see a list of options. Click the Settings option. Once the Settings screen opens click the User Info button. You will see your first name in an input field, once you change it there it will automatically save and you will have a new first name. Hope that was helpful!"

Considerations:

- **Building the Chatbot:**
    - Develop a chatbot using Python and integrate it with an open source, in-house, non-API using NLP/LLM model.
    - Mock and label Graph, JSON, and MongoDB data for an application that you would like to test your solution with.

        - We have provided some example data below that you can refer to when making your dataset. For the MongoDB database, you can use the MongoDB Atlas Sample Data as a reference. It's advisable to select a complex structure from the beginning to support the processing of interdependent DB collections, as well as complex JSON and graph data.
    - Ensure the chatbot processes the user's questions and identifies whether they are related to Graph, JSON, or MongoDB.
        - For Graph or JSON queries, process and provide output using input data in a human-readable format.
        - For data-related queries, generate accurate results from MongoDB and provide output in a human-readable format.
    - Implement error-checking and validation mechanisms for user inputs.
    - **Expected Results:** Achieve 99% accuracy in handling one-to-one collection queries. Provide a comprehensive training video detailing the model development process.
- **Enhancing the ChatBot:**
    - Make your solution continuously learn from user feedback
    - **Expected Results:** Attain 99.99% accuracy in handling one-to-many collection queries. Include a feedback mechanism for ongoing model improvement.

INPUT EXAMPLE

INPUT NL Statement Examples

1. Can you tell me the first name I used in this application?
2. Can you tell me how to navigate to settings and change my first name?

INPUT MongoDB Queries Example

Intermediate (the user won't input this, your solution should read the DB schema, and once it reads the NL fields and structure it should be able to construct this query to help fetch up-to-date information from the server) MongoDB query example:

```
db.users.find({
  firstName: "Abdul",
  lastLogin: {
    $gte: ISODate("2023-05-20T00:00:00.000Z"),
    $lt: ISODate("2023-05-21T00:00:00.000Z")
  },
  "settings.theme": "dark",
  permissions: "admin",
  "socialLinks.platform": "twitter"
})
```

You can assume the Field Names in the DB will be meaningful

MongoDB response example (this is just a response, see below for an actual OUTPUT example for your model)

```
[
  {
    "_id": ObjectId("60a1b2c3d4e5f6g7h8i9j0"),
    "firstName": "Abdul",
    "lastName": "Mohammadi",
    "email": "abdul.mohammadi@example.com",
    "passwordHash": "$2b$10$Xn6LfzM9oW1Qr5D.e/kLxOc5y0t3q2w1",
    "createdAt": ISODate("2023-05-18T09:30:00.000Z"),
    "lastLogin": ISODate("2023-05-20T14:45:30.000Z"),
    "settings": {
      "theme": "dark",
      "language": "en",
      "notifications": true
    },
    "permissions": [
      "read",
      "write",
      "admin"
    ],
    "profilePicture": {
      "url": "https://example.com/img/abdul.jpg",
      "altText": "Profile picture of Abdul"
    },
    "socialLinks": [
      {
        "platform": "twitter",
        "url": "https://twitter.com/abdulmohammadi"
      },
      {
        "platform": "linkedin",
        "url": "https://linkedin.com/in/abdul-m"
      }
    ]
  }
]
```

GRAPH Data Example INPUT

Nodes:

- Node ID: logo, RelativeScreenLocation (HTML or JS element hierarchy), Action: click, Event Type: navigation
- Node ID: menu, RelativeScreenLocation (HTML or JS element hierarchy), Action: hover, Event Type: expand
- Node ID: contactButton, RelativeScreenLocation (In the Sidenav the 7th item down should be the contact button.), Action: click, Event Type: navigation
- Node ID: settingsButton, RelativeScreenLocation (In the Sidenav the 8th item down should be the settings button.), Action: click, Event Type: navigation
- Node ID: settingsForm, RelativeScreenLocation (The settings form should be located in the main content area of the settings page.), Action: submit, Event Type: formSubmission
- Node ID: firstNameInput, RelativeScreenLocation (The first name input field should be the first item within the settings form.), Action: input, Event Type: valueChange
- Node ID: saveButton, RelativeScreenLocation (The save button should be positioned at the bottom of the settings form.), Action: click, Event Type: formSubmission

Edges:

- From logo to contactButton, Action: click
- From menu to contactButton, Action: hover
- From logo to settingsButton, Action: click
- From menu to settingsButton, Action: hover
- From settingsButton to settingsForm, Action: click
- From settingsForm to firstNameInput, Action: submit
- From saveButton to settingsForm, Action: click

JSON Data Example INPUT


```
{
  "pages": [
    {
      "name": "HomePage",
      "elements": [
```

```
    {
      "id": "logo",
      "description": "The website's logo, situated at the top-left corner, serves as a clickable element that allows users to return to the homepage from any page within the site.",
      "action": "click",
      "relativeScreenLocation": [
        "Header",
        "Logo container",
        "Logo image"
      ],
      "eventType": "navigation"
    },
    {
      "id": "menu",
      "description": "This is the main navigation menu, positioned at the top of the page. It expands upon hover to reveal additional navigation options to the user, facilitating easy access to various site sections.",
      "action": "hover",
      "relativeScreenLocation": [
        "Header",
        "Navigation bar",
        "Menu list"
      ],
      "eventType": "expand"
    },
    {
      "id": "contactButton",
      "description": "A prominently placed button that prompts users to visit the contact page. This call-to-action button is designed to stand out and attract user attention, encouraging them to engage with the site.",
      "action": "click",
      "relativeScreenLocation": [
        "Header",
        "Navigation bar",
        "Menu list",
        "7th item in the list"
      ],
      "eventType": "navigation"
    },
    {
      "id": "settingsButton",
      "description": "A button that directs users to the settings page, where they can modify their account preferences and personal information.",
      "action": "click",
```

```
        "relativeScreenLocation": [
          "Header",
          "Navigation bar",
          "Menu list",
          "8th item in the list"
        ],
        "eventType": "navigation"
      },
      {
        "id": "footer",
        "description": "The footer section contains supplementary information about the website, including copyright details, links to privacy policies, and potentially additional navigation links. It spans the bottom of the page and is designed for reference without interactive actions.",
        "action": null,
        "relativeScreenLocation": [
          "Footer",
          "Footer container"
        ],
        "eventType": null
      }
    ]
  },
  {
    "name": "ContactPage",
    "elements": [
      {
        "id": "contactForm",
        "description": "A form designed to facilitate user communication with the website administrators or company. Users can input their details and inquiries, which are then submitted for processing and response. This element is central to user interaction on the contact page.",
        "action": "submit",
        "relativeScreenLocation": [
          "Main content",
          "Contact form container",
          "Contact form"
        ],
        "eventType": "formSubmission"
      },
      {
        "id": "backButton",
        "description": "A navigation element that allows users to return to the previously viewed page. It's typically used to enhance user navigation experience by providing a simple and intuitive means to go back without using the browser's back button.",
        "action": "click",
```

```
        "relativeScreenLocation": [
          "Header",
          "Navigation bar",
          "Back button"
        ],
        "eventType": "navigation"
      },
      {
        "id": "header",
        "description": "The header on the contact page provides context and branding continuity. It usually contains the logo, potentially a navigation menu, and could also feature a heading or title that indicates the user is on the contact page.",
        "action": null,
        "relativeScreenLocation": [
          "Header",
          "Header container"
        ],
        "eventType": null
      }
    ]
  },
  {
    "name": "MarketingPage",
    "elements": [
      {
        "id": "signupButton",
        "description": "This button is a call to action for the user to sign up, usually associated with a service or newsletter. Clicking this button typically directs the user to a form or another page where they can complete their registration or subscription.",
        "action": "click",
        "relativeScreenLocation": [
          "Main content",
          "Marketing section",
          "Signup button"
        ],
        "eventType": "formSubmission"
      },
      {
        "id": "learnMore",
        "description": "A link or button that, when hovered over, provides the user with additional information about a product or service. It's intended to offer users an easy way to access more detailed information without cluttering the initial view.",
        "action": "hover",
        "relativeScreenLocation": [
```

```
          "Main content",
          "Marketing section",
          "Learn more link"
        ],
        "eventType": "expand"
      },
      {
        "id": "testimonial",
        "description": "This section features comments or endorsements from customers or clients, serving as social proof to potential new customers. The testimonials are displayed to build trust and credibility without requiring user interaction.",
        "action": null,
        "relativeScreenLocation": [
          "Main content",
          "Testimonial section"
        ],
        "eventType": null
      }
    ]
  },
  {
    "name": "SettingsPage",
    "elements": [
      {
        "id": "logo",
        "description": "The website's logo, situated at the top-left corner, serves as a clickable element that allows users to return to the homepage from any page within the site.",
        "action": "click",
        "relativeScreenLocation": [
          "Header",
          "Logo container",
          "Logo image"
        ],
        "eventType": "navigation"
      },
      {
        "id": "menu",
        "description": "This is the main navigation menu, positioned at the top of the page. It expands upon hover to reveal additional navigation options to the user, facilitating easy access to various site sections.",
        "action": "hover",
        "relativeScreenLocation": [
          "Header",
          "Navigation bar",
```

```
        "Menu list"
      ],
      "eventType": "expand"
    },
    {
      "id": "settingsForm",
      "description": "A form that allows users to adjust their account settings. It includes fields for updating personal information, such as the user's first name (Abdul), email address, and password. Users can save their changes by submitting the form.",
      "action": "submit",
      "relativeScreenLocation": [
        "Main content",
        "Settings form container",
        "Settings form"
      ],
      "eventType": "formSubmission"
    },
    {
      "id": "firstNameInput",
      "description": "An input field within the settings form that allows the user to edit their first name. The current value is pre-populated with 'Abdul', which can be modified and saved by the user.",
      "action": "input",
      "relativeScreenLocation": [
        "Main content",
        "Settings form container",
        "Settings form",
        "First name input field"
      ],
      "eventType": "valueChange"
    },
    {
      "id": "saveButton",
      "description": "A button that, when clicked, submits the settings form and saves the user's updated information, including their edited first name.",
      "action": "click",
      "relativeScreenLocation": [
        "Main content",
        "Settings form container",
        "Settings form",
        "Save button"
      ],
      "eventType": "formSubmission"
    },
```

```
      {
        "id": "footer",
        "description": "The footer section contains supplementary information about the website, including copyright details, links to privacy policies, and potentially additional navigation links. It spans the bottom of the page and is designed for reference without interactive actions.",
        "action": null,
        "relativeScreenLocation": [
          "Footer",
          "Footer container"
        ],
        "eventType": null
      }
    ]
  }
 ]
}
```


OUTPUT EXAMPLE

1. In this application your current first name is Abdul.
2. To change your first name, navigate to the SettingsPage by clicking on the "settingsButton" located in the navigation menu on the HomePage. Once on the SettingsPage, find the "settingsForm" in the main content area. Within the form, locate the "firstNameInput" field, which should be pre-populated with your current first name (Abdul). Modify the value in this field to update your first name, then click on the "saveButton" at the bottom of the form to submit your changes and save your updated information.