AUST

UNIVERSITY

AUST

NAME: Waqas Ahmed

Subject: DSA

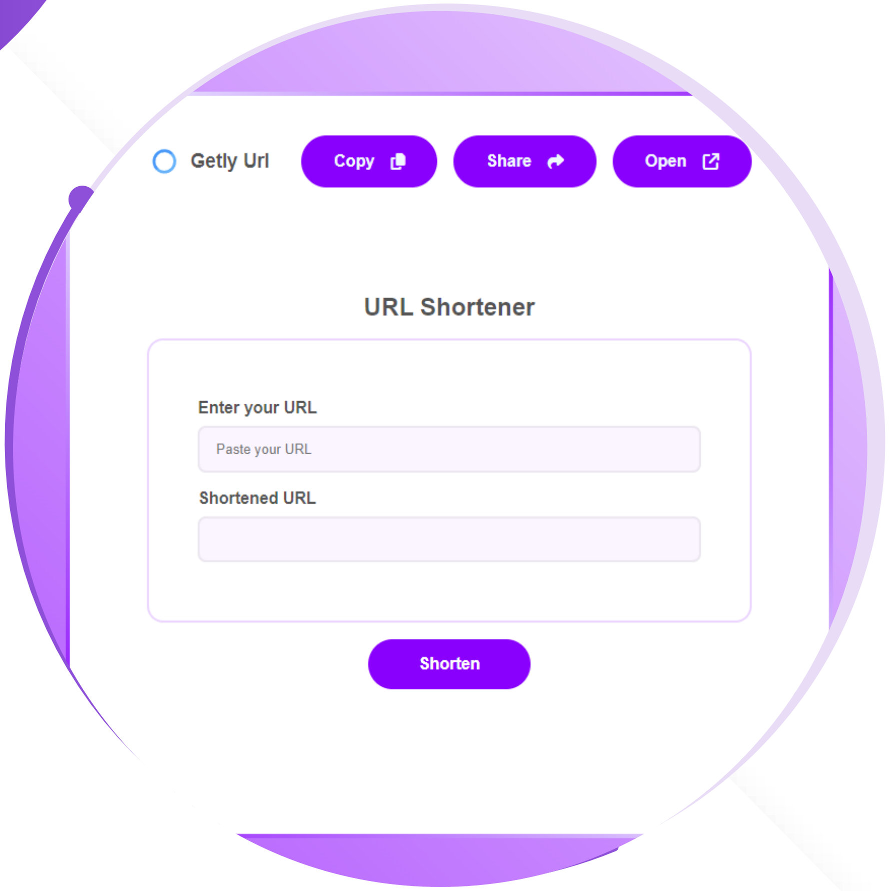
Project: URL Shorter

Submitted To: Mr. Jamal Abdul Ahad

Class: 3rd BScs

Roll no: 14832

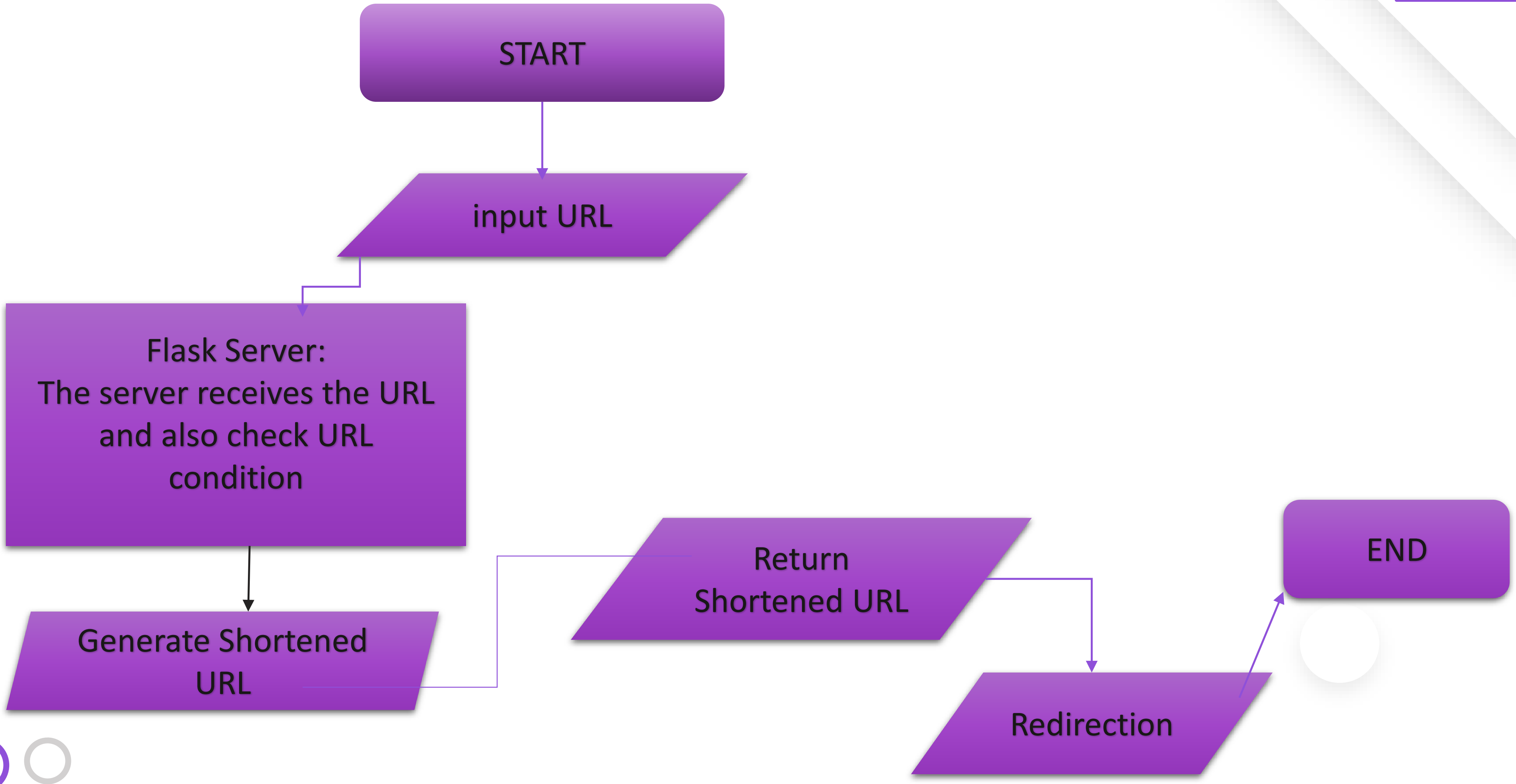
Link GitHub: https://github.com/waaqas969/URL-Shortner.git



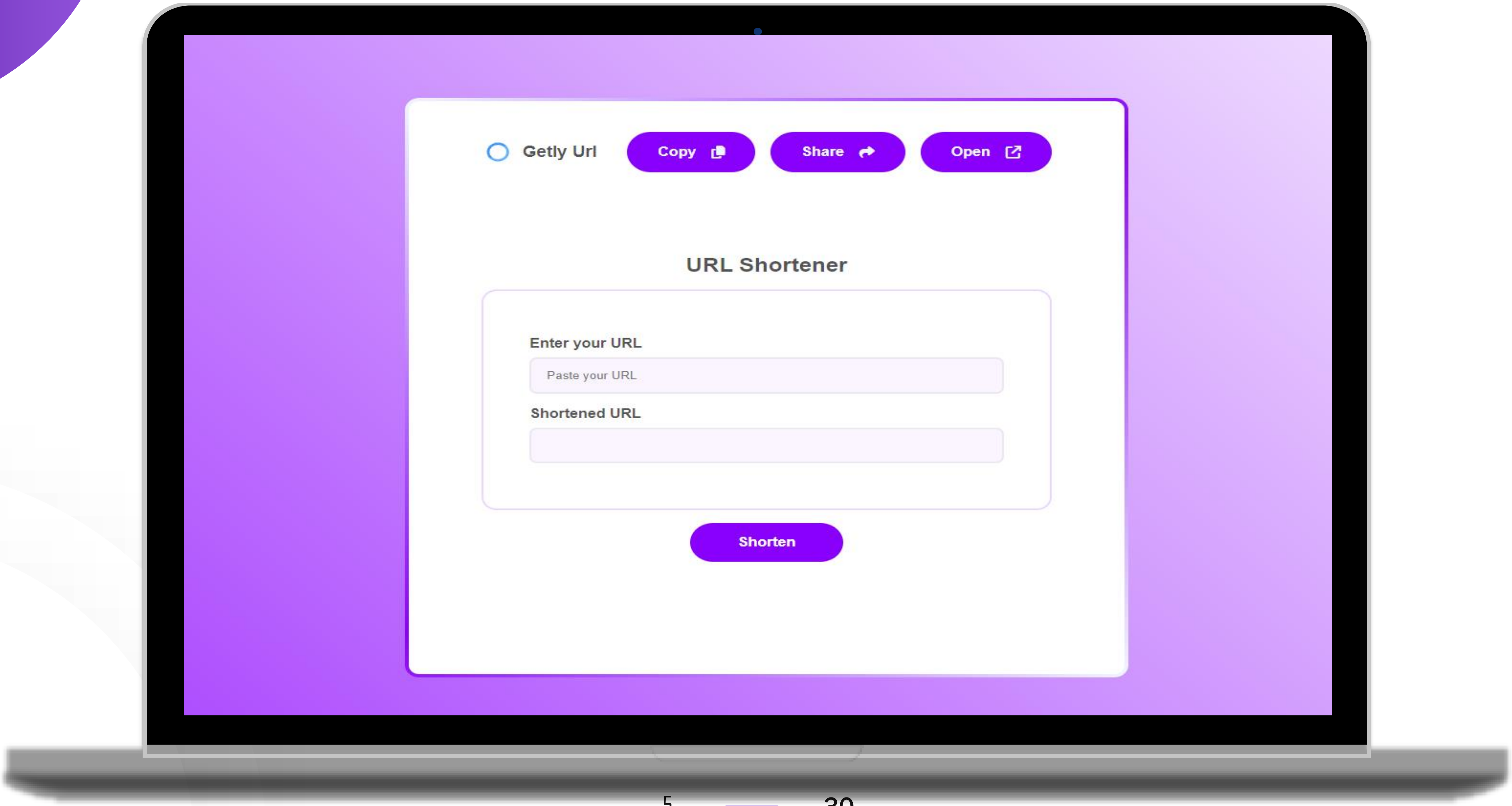
overview of Project

This project is a straightforward URL shortener created with Flask for the backend and HTML/CSS for the frontend. Users can enter a URL, which is then shortened using an MD5 hash and saved in a dictionary. The shortened URL is shown to the user, who can copy, share, or open it. The Flask app takes care of redirecting from the shortened URL back to the original. JavaScript is used to manage user interactions such as shortening, copying, and sharing.

flowchart of the project



UI Design


Getly Url
Copy
Share
Open
URL Shortener
Enter your URL
Paste your URL
Shortened URL
Shorten

Introduction of project

Purpose: To build a URL shortener application that converts long URLs into shorter, manageable links.

Features:

Generate short URLs for user-provided links.

Copy, share, or open the shortened link directly.

Redirect to the original URL using the shortened link.

Technologies Used

Frontend

HTML, CSS for layout and styling.
JavaScript for interactivity and API calls.
FontAwesome for icons.
LordIcon for animated icons.

Backend:

Flask for handling requests and generating shortened URLs.
Python libraries : hashlib for URL hashing
Hosting: Local development server (Flask).

Frontend Details

HTML Structure
•Organized into sections for navigation, input form, and actions.
•Input fields for the original URL and generated short URL.

CSS Highlights
•Responsive design using media queries.
•Gradient backgrounds and modern button styles.
•Flexible layouts using flexbox

JavaScript Functionalities:

Shorten Button:
Sends POST request to the backend with the URL.
Displays the shortened URL in the input field.

Copy Button : Copies the shortened URL to the clipboard.
Alerts the user upon successful copy
Share Button:
Use the navigator. shareAPI for easy sharing
Open Button :
Opens the shortened URL in a new tab

Backend Implementation of project

Step:1

Flask: A basic library for developing web applications.

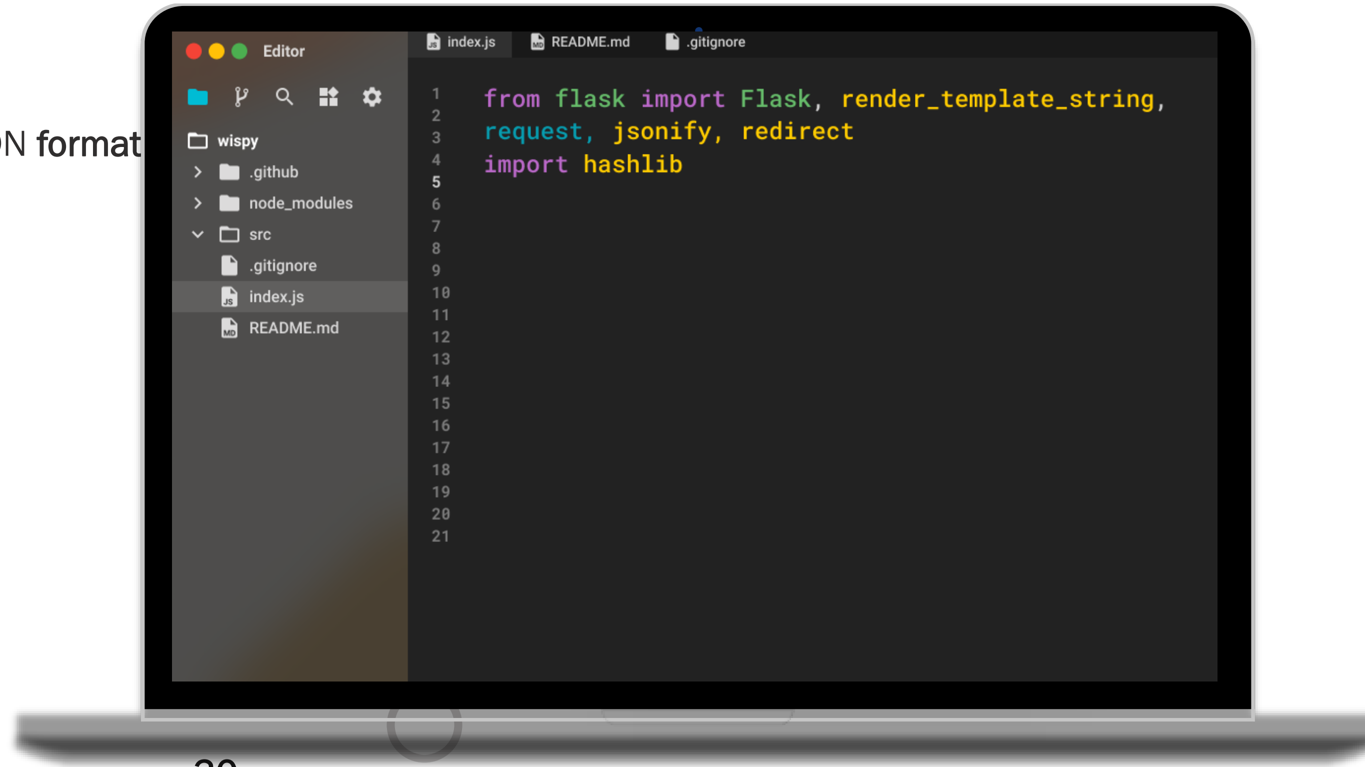
render_template_string: Renders an HTML template directly from a string, used for front-end.

Query: Retrieves incoming data (such as a URL) from the user.

jsonify: Converts Python data structures (such as dictionaries) to JSON **format** which will be used to return data to the client side.

redirect: Redirects the user to the specified URL.

hashlib: Used to create a unique hash for the original URL..



```
from flask import Flask, render_template_string,
request, jsonify, redirect
import hashlib
```

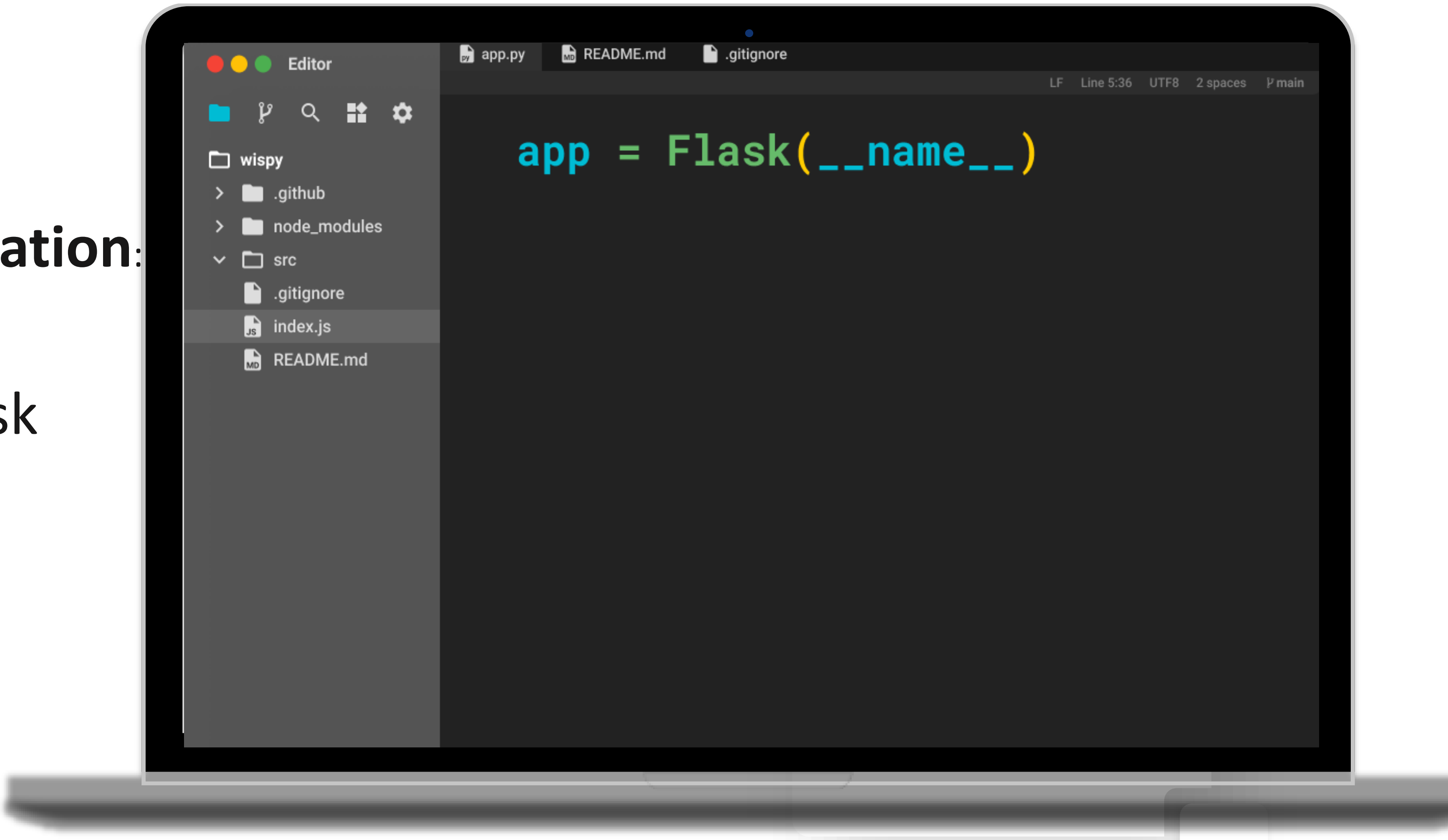
Steps of project

Step:2

Setting Up Flask Application:

This line initializes the Flask

application



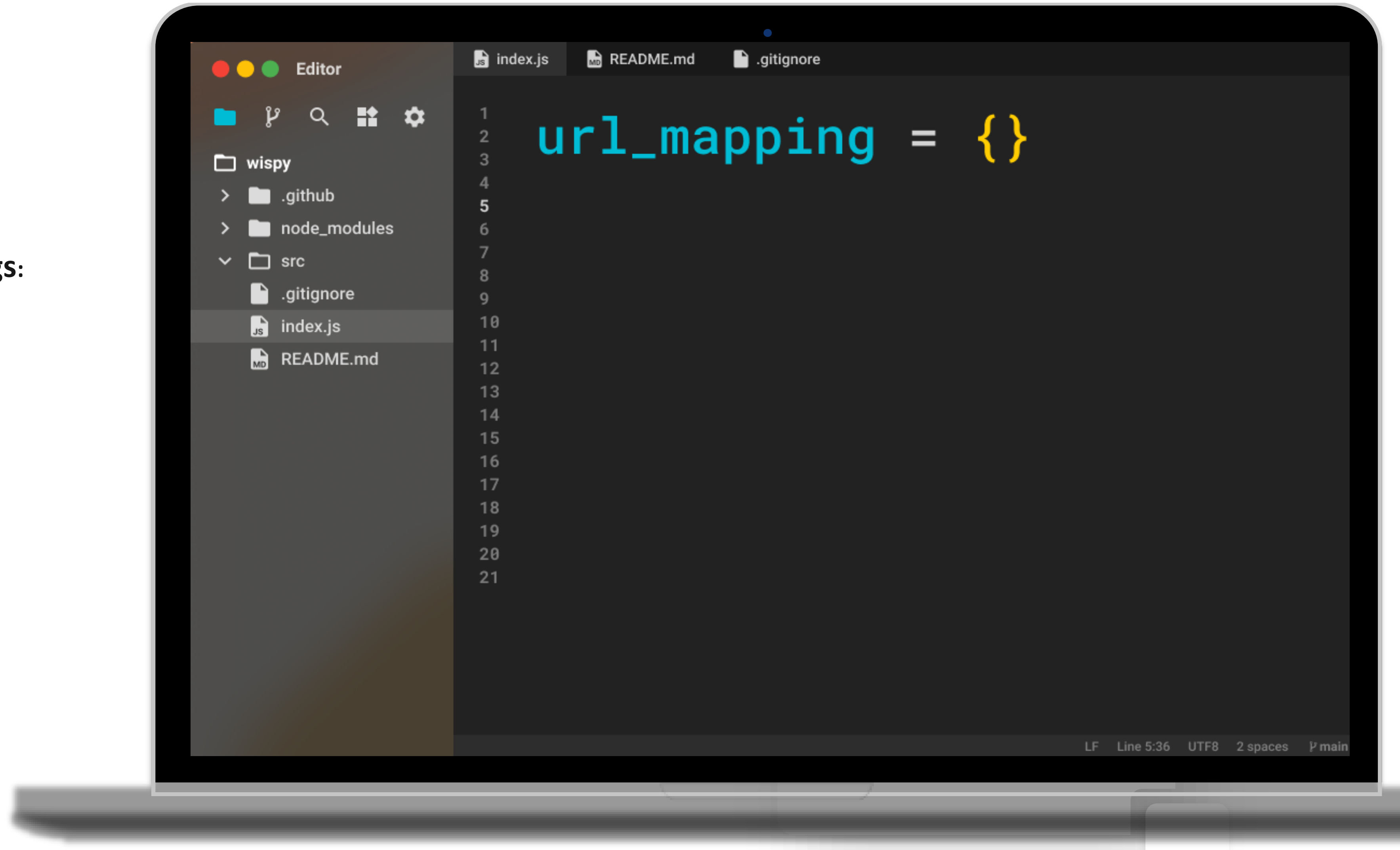
```
app = Flask(__name__)
```

Steps of project

Step:3

Global Variable to Store URL Mappings:

This dictionary will map the original URL to the corresponding short code.



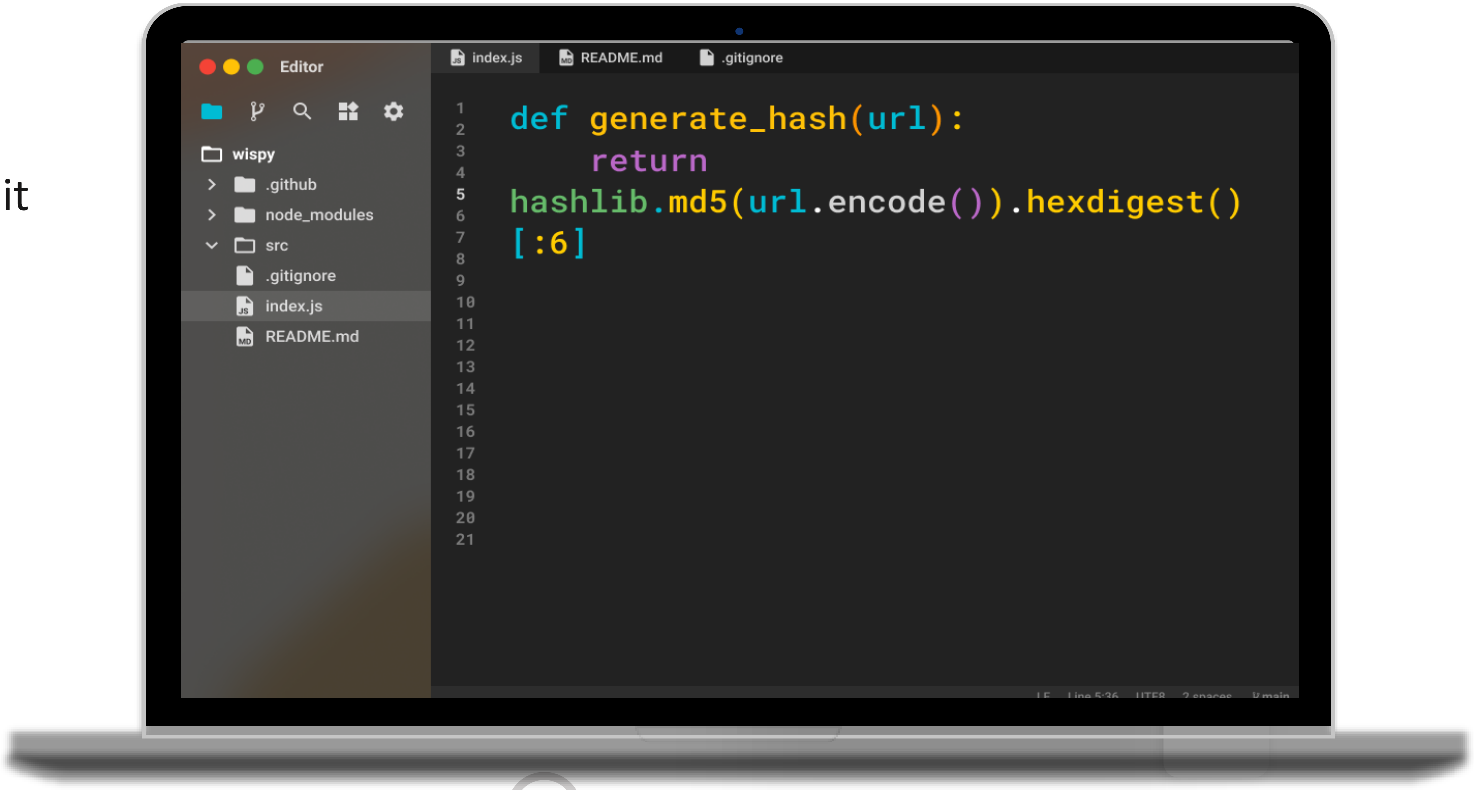
```
url_mapping = {}
```

Steps of project

Step:4

Function to Generate Hashes for URLs:

Creates an MD5 hash for the input URL
Returns the first 6 characters of the hash, making it a short, unique identifier for the URL.



```python
def generate_hash(url):
    return hashlib.md5(url.encode()).hexdigest()[:6]
```

Steps of project

Step:5

HTML and CSS Template for the Front-End:

The HTML_template variable holds the HTML and CSS needed for the user interface. It includes:

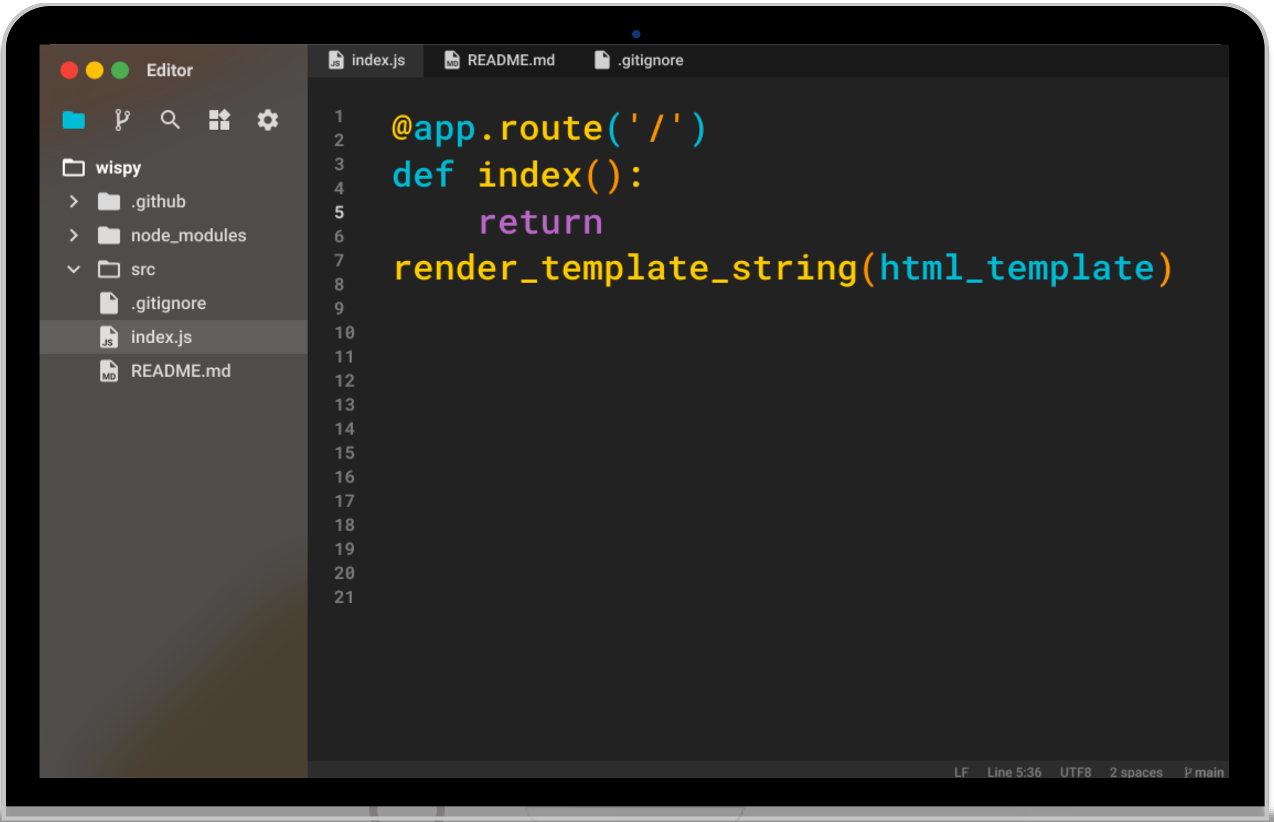
An input field to paste the original URL
A disabled input field to display the shortened URL
Buttons for shortening, copying, sharing, and opening the shortened UR

Steps of project

Step:6

Index Route:

When a user visits the root {/} this route renders the HTML template, displaying the input fields and buttons.



```
@app.route('/')
def index():
    return
render_template_string(html_template)
```

Steps of project

Step:7

This route listens for POST requests on / shorten

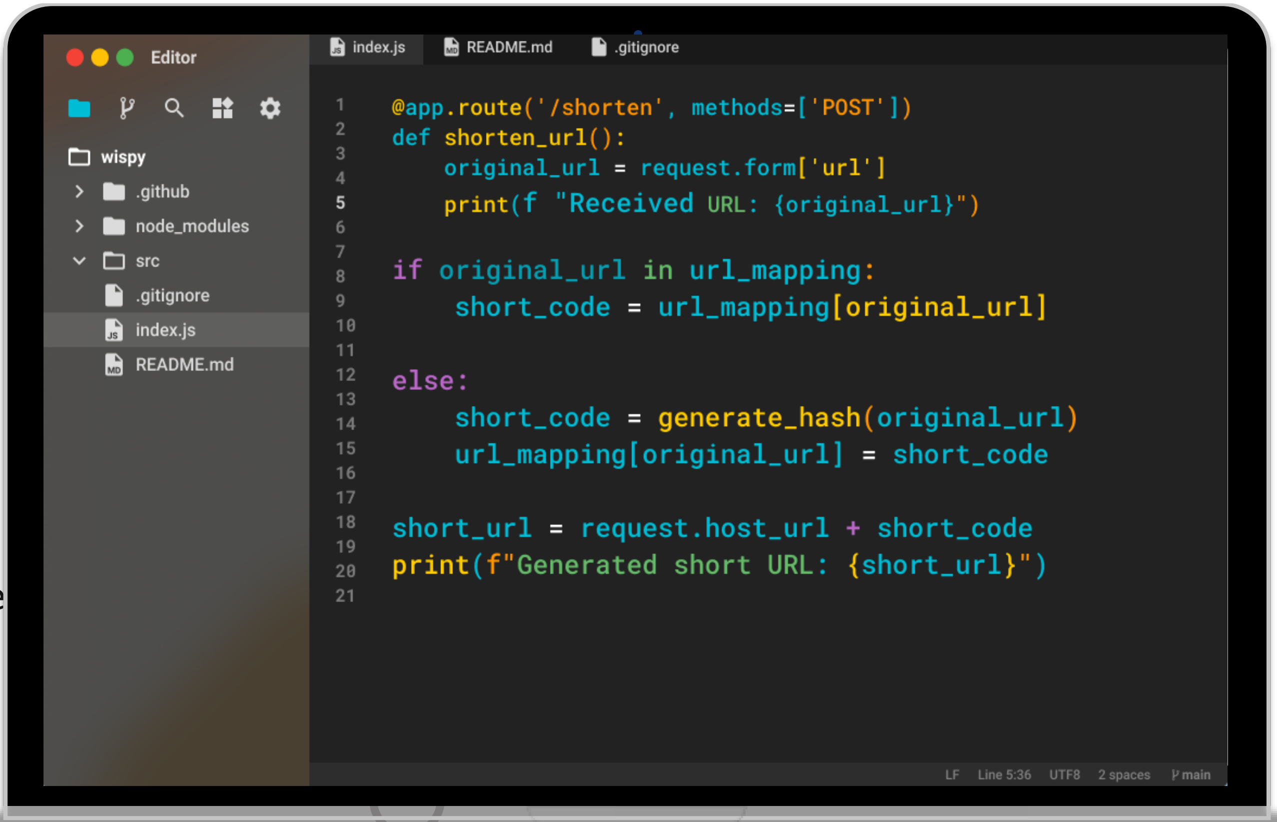
request. Form['url']

.If the original URL is already shortened, it retrieves the short code from the url_mapping dictionary.

If the URL hasn’t been shortened before, it generates a new short code using the **generate hash** function and stores the mapping in url_mapping

The short URL is created by appending the short code to the base URL (host URL of the server).

This short URL is then returned as JSON to the client.



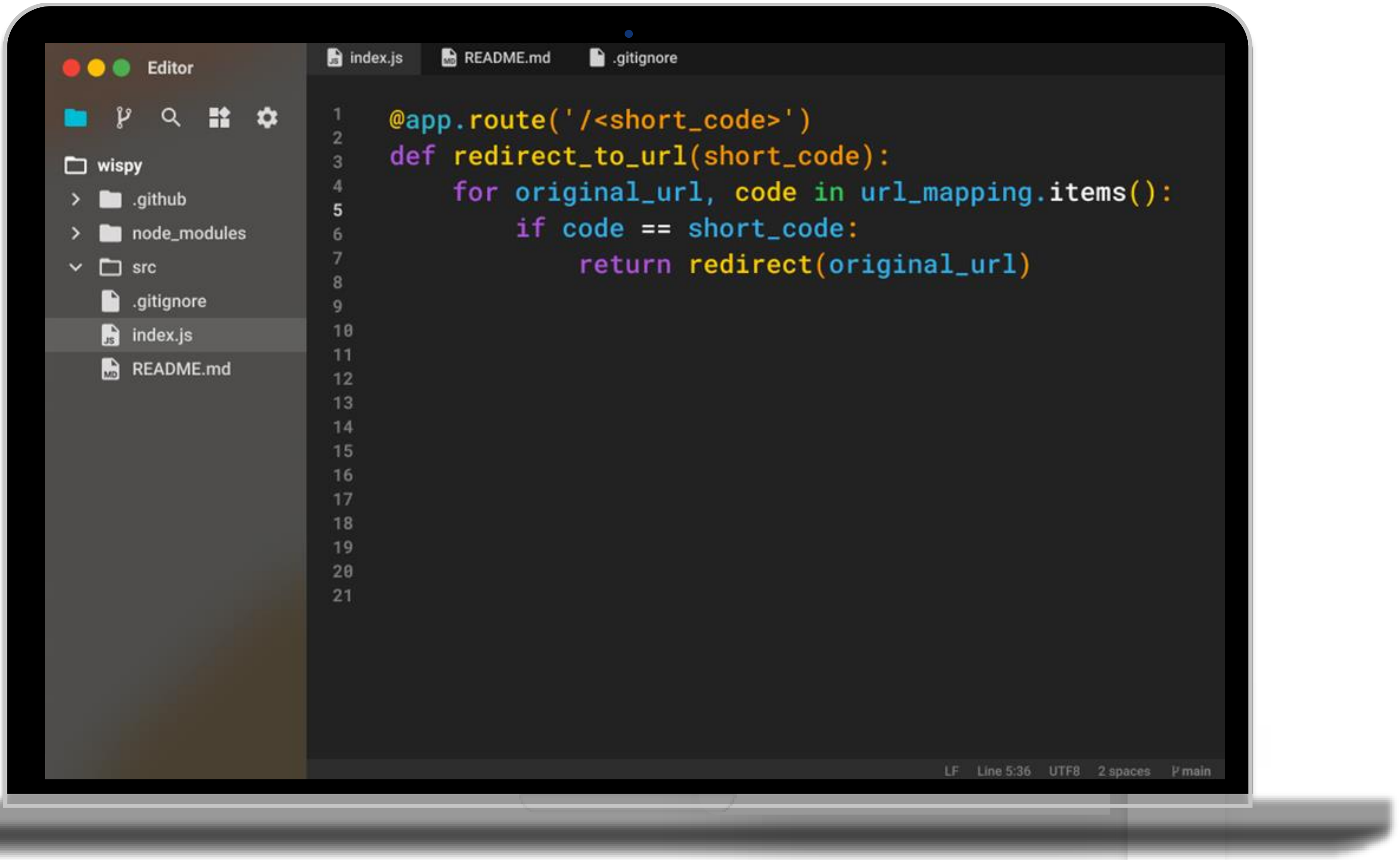
```
@app.route('/shorten', methods=['POST'])
def shorten_url():
    original_url = request.form['url']
    print(f "Received URL: {original_url}")

if original_url in url_mapping:
     short_code = url_mapping[original_url]

else:
     short_code = generate_hash(original_url)
     url_mapping[original_url] = short_code

short_url = request.host_url + short_code
print(f"Generated short URL: {short_url}")
```

Steps of project

Step:8

Redirection to Original URL:

This route captures the short code from the URL.
It looks up the short code in the url_mapping dictionary and redirects the user to the corresponding original URL.



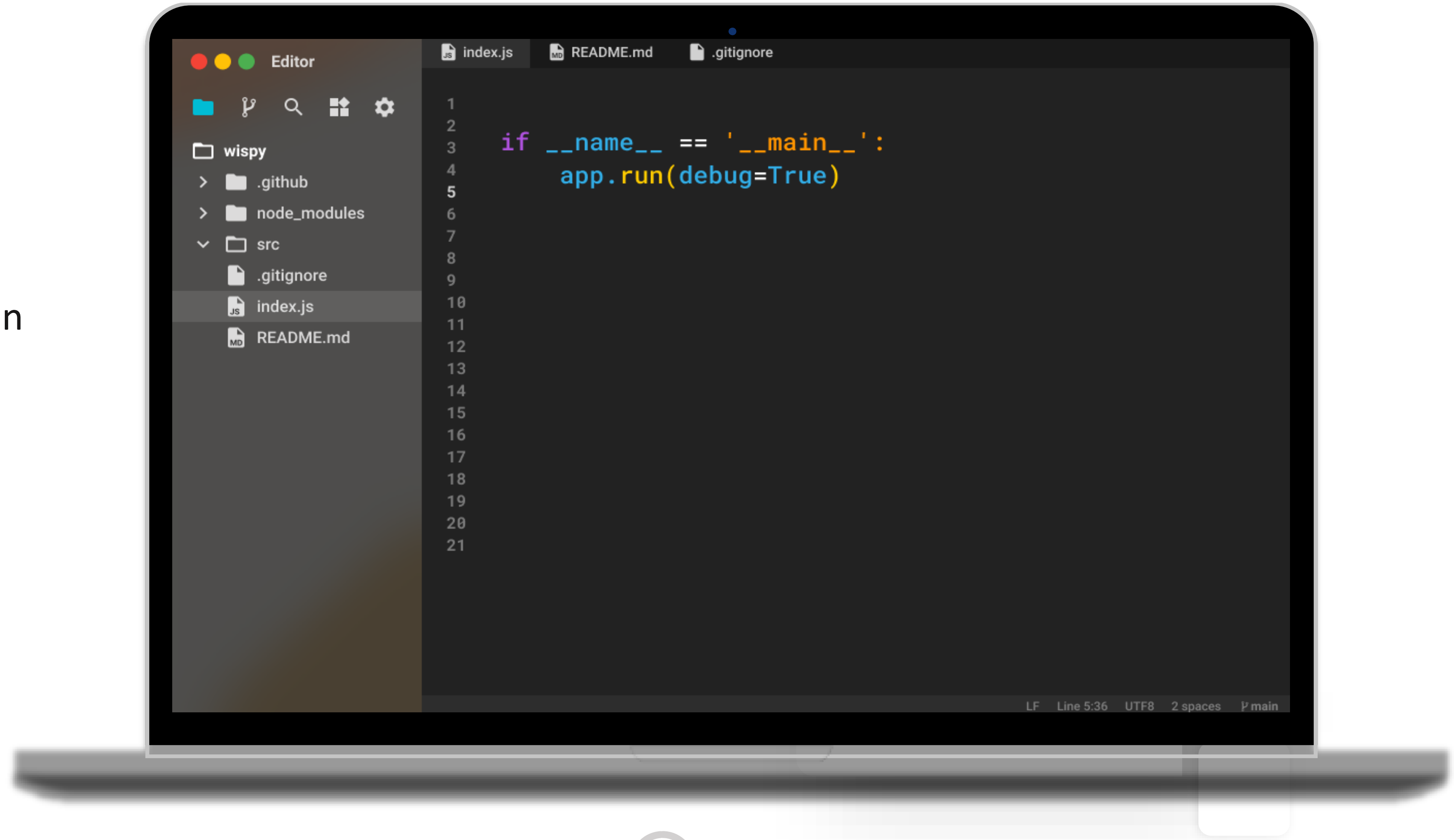
```
@app.route('/<short_code>')
def redirect_to_url(short_code):
    for original_url, code in url_mapping.items():
        if code == short_code:
            return redirect(original_url)
```

Steps of project

Step:9

***Running the Application*:**

This starts the Flask development server in debug mode.



```
if __name__ == '__main__':
    app.run(debug=True)
```

AUST

THANK YOU

I hope you like my presentation
Next time I will do better