Project Plan: AI Operator – Local Browser Agent

1. Project Objectives

The AI Operator project seeks to develop a privacy-focused, intelligent browser agent capable of automating routine web-based tasks entirely on a user's local machine. The primary goal is to create a seamless, desktop-based assistant that can handle complex online actions—such as ordering groceries, booking tickets, filling out forms, and navigating websites—through natural language commands, without sending any data to external servers or cloud platforms. This local-first approach ensures user data, credentials, and browsing activity remain completely private and secure.

Unlike traditional automation tools that rely on cloud services or external processing, our agent will function entirely on the user's device. It will integrate a chatbot interface for intuitive communication, web scrapers to analyse open browser tabs, and a locally-hosted large language model (LLM) that interprets user instructions. At the heart of this process lies Microsoft's Semantic Kernel, which will act as the cognitive layer responsible for decomposing high-level instructions into logical and actionable web interactions. This semantic reasoning engine transforms vague or complex user goals into step-by-step sequences that can be executed with precision through browser automation tools like Selenium.

The project aims to combine multiple sophisticated components into a single modular system: a desktop UI for interaction, an LLM for interpreting intent, a task planner for generating execution steps, and browser automation scripts for performing actions. The modular architecture allows for continuous development, testing, and iteration within an agile framework. Each feature will be developed in sprints, enabling rapid feedback loops and measurable progress toward a fully functional minimum viable product (MVP).

Beyond functionality, our objective is to demonstrate that intelligent automation can be both practical and ethical. With rising concerns about data misuse, surveillance, and AI transparency, this project promotes a responsible AI vision where users retain control of their digital interactions. By running entirely on local infrastructure, the AI Operator protects against leaks of sensitive personal data and mitigates risks associated with third-party data handling.

The broader ambition of this project is not just to build a smart assistant, but to redefine what intelligent web automation looks like in the era of AI. It bridges natural language processing, web scraping, real-time planning, and local execution in a coherent system that reflects strong software engineering principles and privacy-by-design values. By leveraging open-source technologies and a clear ethical stance, the AI Operator will offer a powerful alternative to cloud-dependent automation services, tailored for privacy-conscious users and developers alike.

In essence, the AI Operator is more than a tool—it's a statement. A statement that useful, personalised AI can exist without compromising on security or privacy. Through this project, we aim to show that local-first, AI-powered agents are not only feasible but also preferable in a growing landscape of decentralised digital intelligence.

2. Project Plan (8-Week Sprint Plan)

Week	Sprint Focus	Deliverables
1	UI Setup + Environment Configuration	Desktop chatbot prototype, GitHub repo setup
2	Web Scraping Module	Scraper to read content from active browser tabs
3	LLM Integration	Basic prompt-response loop with parsed HTML
4	Semantic Kernel Integration	Convert intent to structured actions
5	Browser Automation	Selenium: Clicks, inputs, navigation
6	Agent Pipeline Assembly	Full pipeline: Chatbot > LLM > Action > Feedback
7	Testing & Edge Case Handling	Local execution tests + security validation
8	Final Refinements + Demo Prep	MVP demo, documentation, report, code freeze

3. Team Roles

- **Sudhanshu Ghuge – LLM Prompt Engineer and Backend Developer**
  Designs effective prompts and integrates the LLM with the semantic kernel.
- **Deepak Shelke – Full stack Developer**
  Builds scraping and automation features using Selenium and APIs.
- **Ayush Poojari – Frontend and AI Developer**
  Frontend Developer and ML Engineer skilled in building responsive UIs, integrating machine learning models, and developing LLM systems.
- **Tarun Kumar – Scrum Master and Backend Developer**
  Leads sprint planning and oversees backend architecture and integration.
- **Soham Deo – AI Engineer and Developer**
  Builds the chatbot UI and connects it to the backend with logging support.
- **Preet Raut –Cloud Engineer and Risk Analyst**
  Manages cloud data flow and structures LLM outputs with risk analysis.

4. Architecture Setup

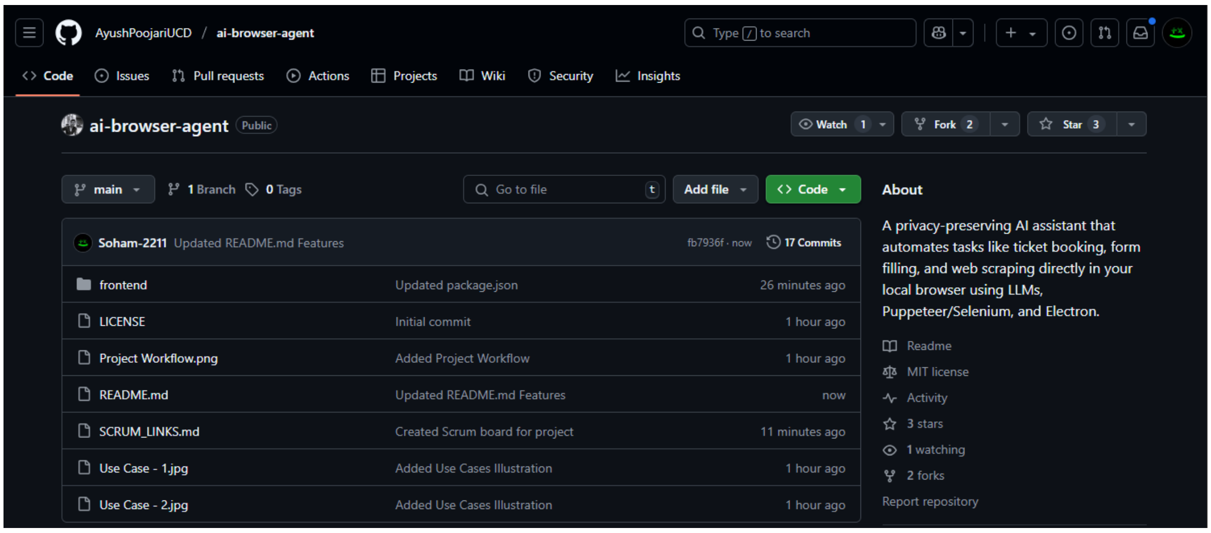
- **Cloud Platform**: Azure
- **Desktop App**: Built with Electron/React + Python backend
- **LLM**: Locally hosted or Azure OpenAI model (for testing)
- **Automation**: Selenium
- **Semantic Kernel**: MS Semantic Kernel for step breakdown

5. Data Plan

- **Input Data**: Live content from browser tabs (HTML/DOM)
- **Generated Data**: Structured web actions from LLM, logs of agent decisions
- **Evaluation Data**: User test cases (e.g., shopping task), success/failure logs, task completion time

6. GitHub Evidence

- All team members contribute via Git branching and pull requests
- Repo Link: Repository Link
- Repo includes:
    - /code: UI, backend, kernel, LLM, testing modules
    - /sprints: weekly goals, completed tasks
    - GitHub Insights shows individual commit history (non-trivial)



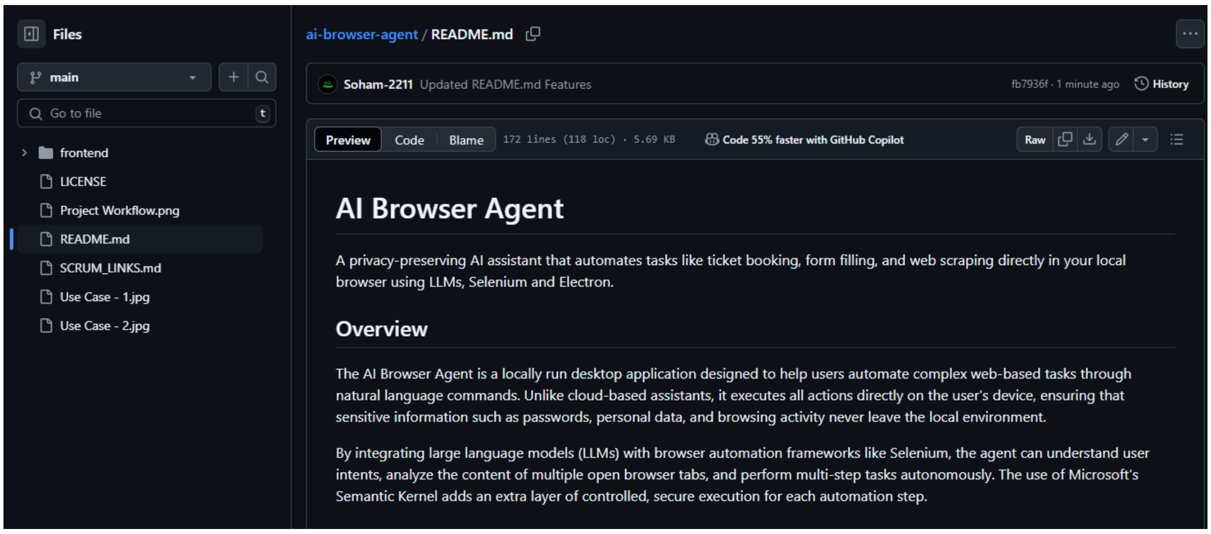
Soham-2211 Updated README.md Features fb7936f · now 17 Commits

frontend	Updated package.json	26 minutes ago
LICENSE	Initial commit	1 hour ago
Project Workflow.png	Added Project Workflow	1 hour ago
README.md	Updated README.md Features	now
SCRUM_LINKS.md	Created Scrum board for project	11 minutes ago
Use Case - 1.jpg	Added Use Cases Illustration	1 hour ago
Use Case - 2.jpg	Added Use Cases Illustration	1 hour ago

About

A privacy-preserving AI assistant that automates tasks like ticket booking, form filling, and web scraping directly in your local browser using LLMs, Puppeteer/Selenium, and Electron.

Readme
MIT license
Activity
3 stars
1 watching
2 forks
Report repository



Files

main

Go to file

frontend
LICENSE
Project Workflow.png
README.md
SCRUM_LINKS.md
Use Case - 1.jpg
Use Case - 2.jpg

ai-browser-agent / README.md

Soham-2211 Updated README.md Features fb7936f · 1 minute ago History

Preview Code Blame 172 lines (118 loc) · 5.69 KB Code 55% faster with GitHub Copilot Raw

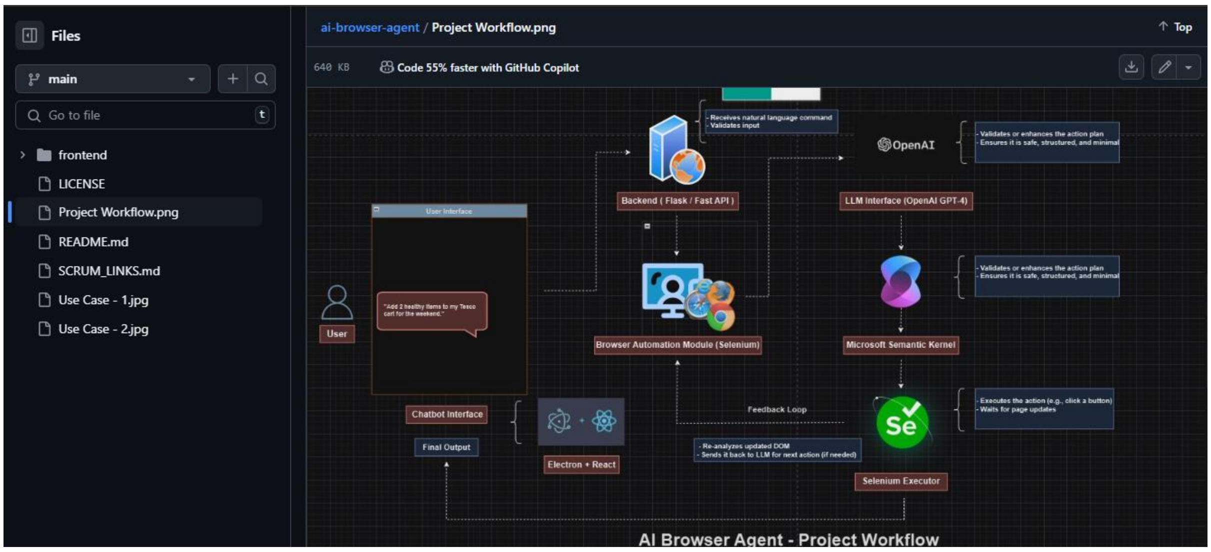
AI Browser Agent

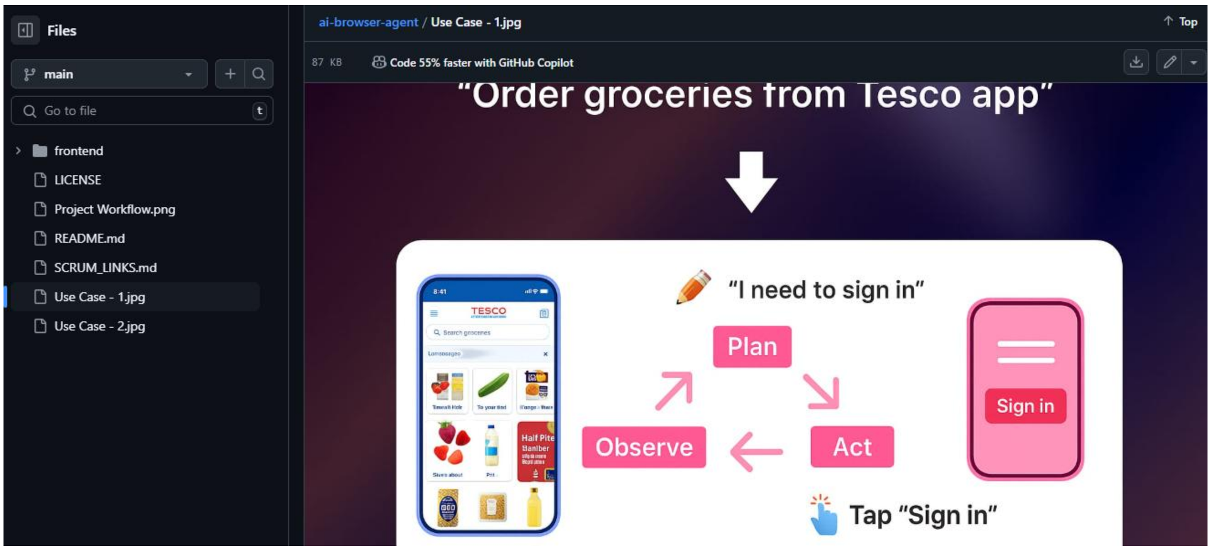
A privacy-preserving AI assistant that automates tasks like ticket booking, form filling, and web scraping directly in your local browser using LLMs, Selenium and Electron.

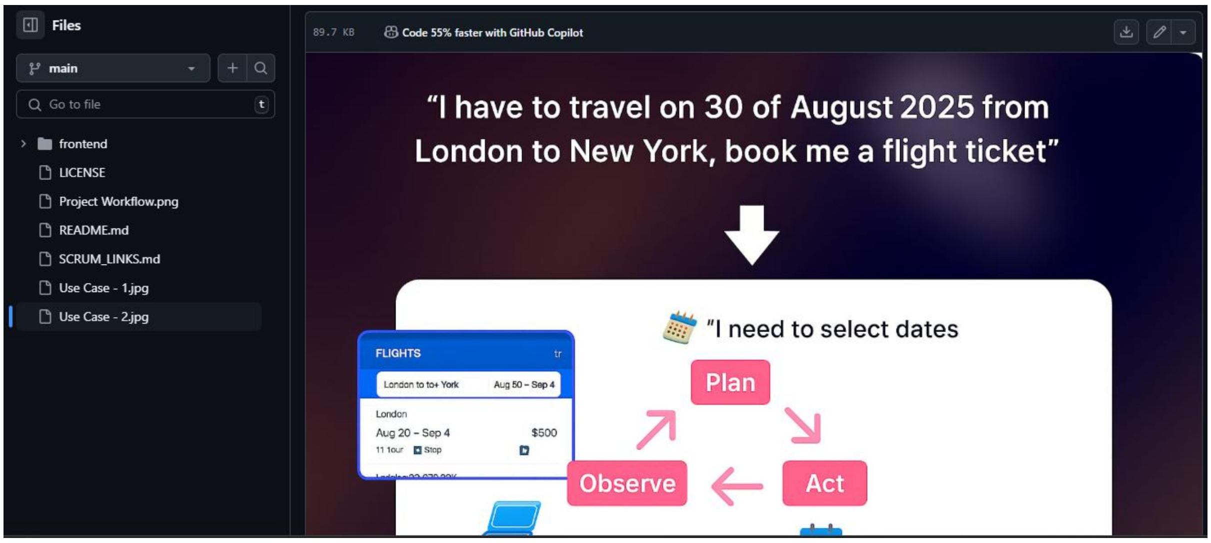
Overview

The AI Browser Agent is a locally run desktop application designed to help users automate complex web-based tasks through natural language commands. Unlike cloud-based assistants, it executes all actions directly on the user's device, ensuring that sensitive information such as passwords, personal data, and browsing activity never leave the local environment.

By integrating large language models (LLMs) with browser automation frameworks like Selenium, the agent can understand user intents, analyze the content of multiple open browser tabs, and perform multi-step tasks autonomously. The use of Microsoft's Semantic Kernel adds an extra layer of controlled, secure execution for each automation step.

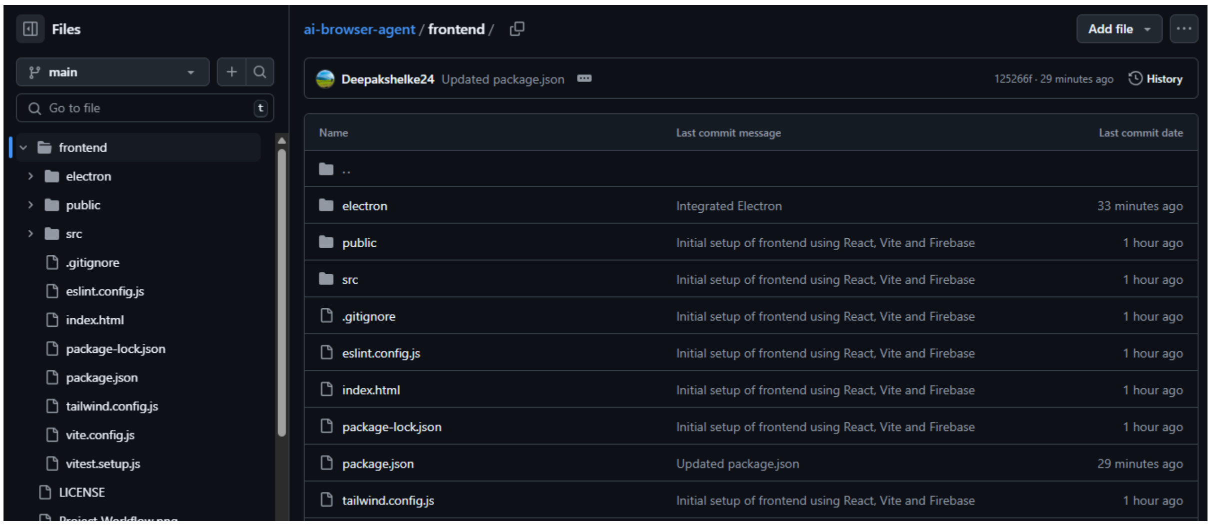

Files
main
Go to file
frontend
LICENSE
Project Workflow.png
README.md
SCRUM_LINKS.md
Use Case - 1.jpg
Use Case - 2.jpg
ai-browser-agent / Project Workflow.png
Top
640 KB
Code 55% faster with GitHub Copilot
- Receives natural language command
- Validates input
Backend (Flask / Fast API)
OpenAI
- Validates or enhances the action plan
- Ensures it is safe, structured, and minimal
LLM Interface (OpenAI GPT-4)
User Interface
User
"Add 2 healthy items to my Tesco cart for the weekend."
Browser Automation Module (Selenium)
- Validates or enhances the action plan
- Ensures it is safe, structured, and minimal
Microsoft Semantic Kernel
Chatbot Interface
Final Output
Electron + React
Feedback Loop
- Re-analyzes updated DOM
- Sends it back to LLM for next action (if needed)
- Executes the action (e.g., click a button)
- Waits for page updates
Selenium Executor
AI Browser Agent - Project Workflow





"I have to travel on 30 of August 2025 from London to New York, book me a flight ticket"

"I need to select dates

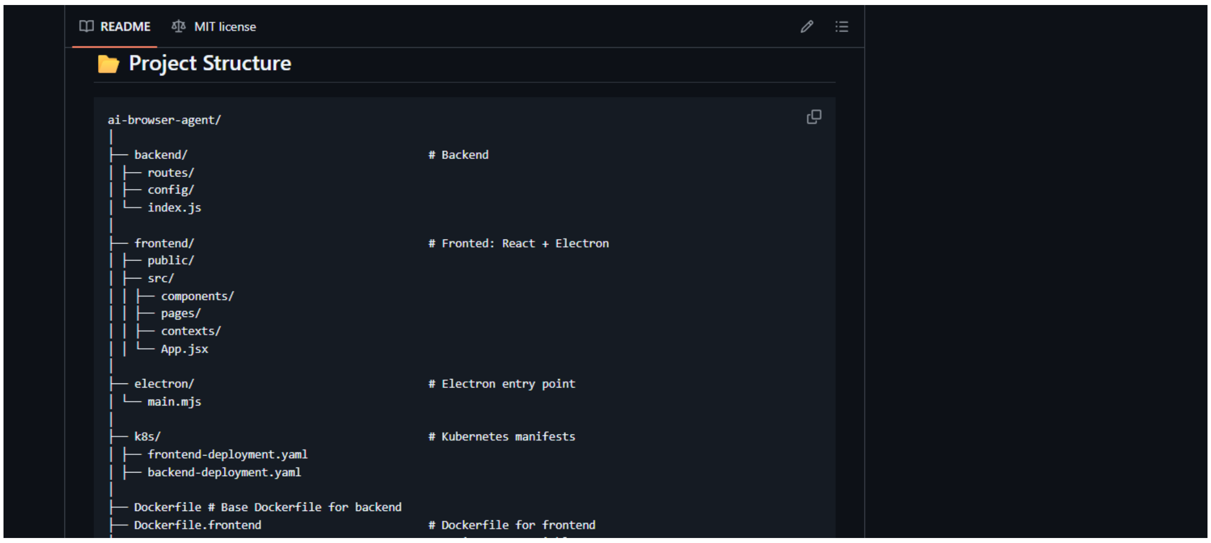


ai-browser-agent / frontend /

Deepakshelke24 Updated package.json — 125266f · 29 minutes ago

Name	Last commit message	Last commit date
..		
electron	Integrated Electron	33 minutes ago
public	Initial setup of frontend using React, Vite and Firebase	1 hour ago
src	Initial setup of frontend using React, Vite and Firebase	1 hour ago
.gitignore	Initial setup of frontend using React, Vite and Firebase	1 hour ago
eslint.config.js	Initial setup of frontend using React, Vite and Firebase	1 hour ago
index.html	Initial setup of frontend using React, Vite and Firebase	1 hour ago
package-lock.json	Initial setup of frontend using React, Vite and Firebase	1 hour ago
package.json	Updated package.json	29 minutes ago
tailwind.config.js	Initial setup of frontend using React, Vite and Firebase	1 hour ago

```
frontend
  electron
  public
  src
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  tailwind.config.js
  vite.config.js
  vitest.setup.js
LICENSE
```



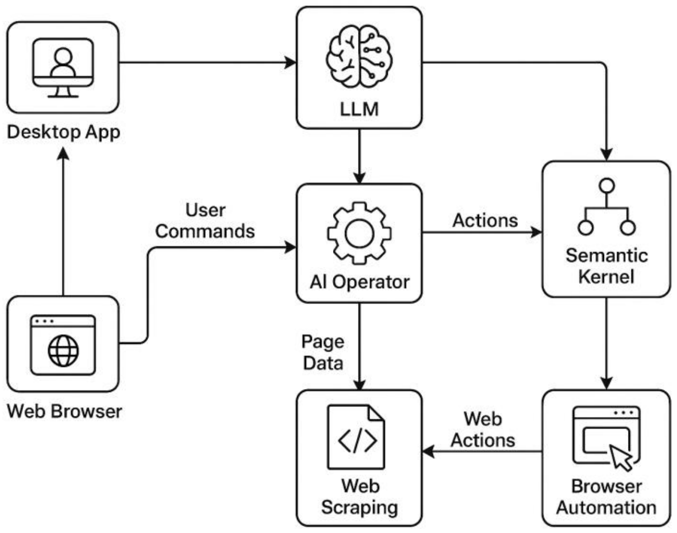
📂 Project Structure

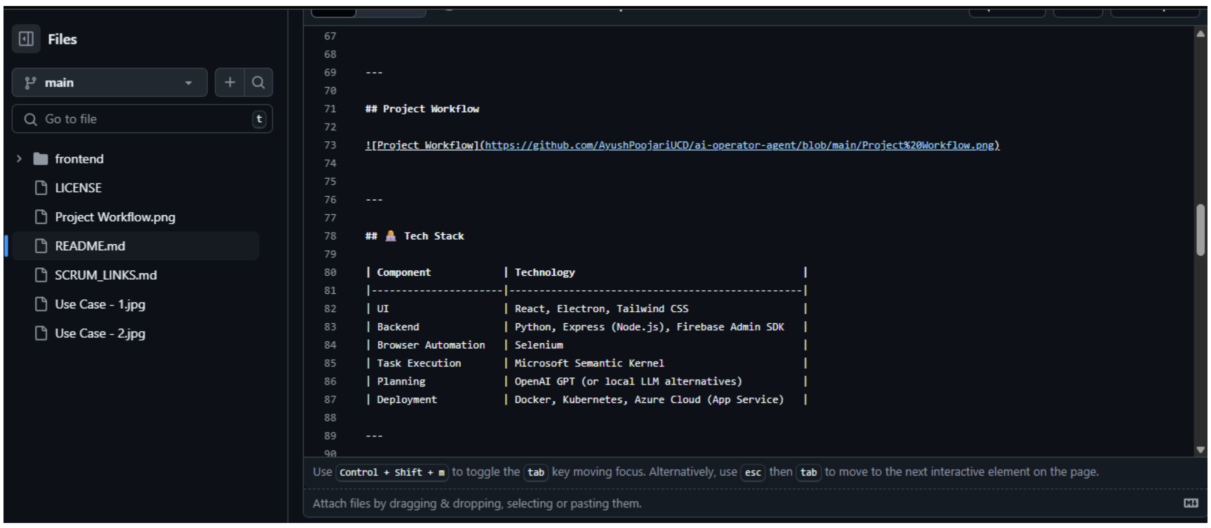
```
ai-browser-agent/
│
├── backend/                            # Backend
│  ├── routes/
│  ├── config/
│  └── index.js
│
├── frontend/                           # Fronted: React + Electron
│  ├── public/
│  ├── src/
│  │  ├── components/
│  │  ├── pages/
│  │  ├── contexts/
│  │  └── App.jsx
│
├── electron/                           # Electron entry point
│  └── main.mjs
│
├── k8s/                                # Kubernetes manifests
│  ├── frontend-deployment.yaml
│  ├── backend-deployment.yaml
│
├── Dockerfile # Base Dockerfile for backend
├── Dockerfile.frontend                 # Dockerfile for frontend
```

7. Team Management

- **Tool Used**: Jira (for sprint planning & task assignment), GitHub Projects
- **Meeting Format**: Twice weekly (Tuesday on MS Teams)
- **Minutes Structure**:
    - Attendance
    - Sprint progress
    - Bottlenecks
    - Assigned tasks with deadlines

8. Architecture Diagram and Tech Stack





```


---

## Project Workflow

![Project Workflow](https://github.com/AyushPoojariUCD/ai-operator-agent/blob/main/Project%20Workflow.png)



---

## 🧑‍💻 Tech Stack

| Component           | Technology                                      |
|---------------------|-------------------------------------------------|
| UI                  | React, Electron, Tailwind CSS                   |
| Backend             | Python, Express (Node.js), Firebase Admin SDK   |
| Browser Automation  | Selenium                                        |
| Task Execution      | Microsoft Semantic Kernel                       |
| Planning            | OpenAI GPT (or local LLM alternatives)          |
| Deployment          | Docker, Kubernetes, Azure Cloud (App Service)   |

---
```

Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.

Attach files by dragging & dropping, selecting or pasting them.