PROJECT DOCUMENTATION: EitherAssistant

Domain: Accessibility & Digital Inclusion
Team Name: Vanguard
Date: 29th December, 2025

---

1. Executive Summary

EitherAssistant is a desktop tool designed to help people with motor disabilities control their computers using only their voice. While big names like Siri or Alexa are great for checking the weather, they can’t actually "use" a computer for you, meaning they can't click buttons on a random website or fill out a form. We built EitherAssistant to fix that. It uses a custom C# interface (Avalonia) for speed and a Python backend to handle the heavy lifting. The result is a system that lets users navigate the web and control their PC completely hands-free, even if they have a slow internet connection.

---

2. Problem Statement & Motivation

- **The Problem:** Most computers are designed for a mouse and keyboard. If you can't use your hands, you are effectively locked out of the digital world.
- **Scope:** We are focusing on controlling the Windows OS and automating web browsers.
- **Why current tools aren't enough:**
    - **They break easily:** Most simple bots just click at specific screen coordinates (like x=500, y=200). If a popup appears or the window moves, the bot clicks the wrong thing.
    - **They need good internet:** Most voice assistants send your audio to the cloud. If your internet is spotty, they stop working.
    - **They are heavy:** Many modern apps use Electron, which eats up RAM. We wanted something that runs smoothly on older, cheaper laptops.
- **Who is this for:** People with limited mobility (like ALS or Parkinson’s) and anyone in areas with poor internet connectivity.

---

3. Background & Related Work

Our Take: We realized that to make a truly useful tool, we needed to move the "brain" to the local device (Edge computing) and stop relying on coordinates. Instead, our bot reads the code of the website to find buttons by name.

Work/Tool	Technique	The Good	The Bad
Siri/Alexa	Cloud Processing	Very good at understanding speech.	Can't fill out forms or navigate complex websites. Needs fast internet.
Windows Voice Access	OS Integration	Built-in and fast.	Struggles with websites that don't have standard coding.
Standard Macros	Recording clicks	Easy to make.	Extremely brittle. One pixel off, and the script fails.

---

4. Our Solution

- **Core Idea:** A bridge that turns what you say ("Open Gmail and compose") into what the computer does (clicks the 'Compose' button).
- **Key Components:**
  - **Voice Engine:** Listens to commands locally on your computer.
  - **Browser Controller:** The logic that finds and clicks elements on a webpage.
  - **Dashboard:** A simple, high-contrast app where you can see what the bot is doing.
- **Why it works:** It respects user privacy (no audio leaves your room) and it’s "self-healing", meaning if a website updates its layout, our bot usually still finds the right button because it looks for the button's name, not its location.

---

5. Technical Approach & Architecture

Data & Algorithms

- **How it finds things:** We use a keyword-matching system. If you say "Send," the system scans the webpage for buttons labeled "Send," "Submit," or icons that look like a paper plane.
- **Speed:** Searching the code of a webpage is much faster than analyzing a screenshot of the page, so it feels instant.

Architecture

- **Frontend (C# / Avalonia):** We chose C# because it's native to Windows and very fast.
- **Backend (Python):** We used Python for the logic because it has the best libraries for automation (Selenium) and speech recognition (Vosk).
- **How they talk:** The C# app sends a message to the Python script over a local socket connection. It's simple, secure, and fast.

---

6. Gap Analysis

There is a big gap between "Information Assistants" (Siri) and "Automation Tools" (Macros).

- **Information Assistants** can tell you the weather but can't book a flight.
- **Macros** can book a flight but break if the website changes.
- **EitherAssistant** sits in the middle: it understands intent and can robustly interact with the web.

Feature	Regular Assistants	EitherAssistant
Internet	Needs constant Cloud connection.	**Works Offline (Edge-First).**
Web Tasks	Can only open pages.	**Can click, type, and navigate.**
Tech	Often web wrappers (heavy).	**Native Code (Lightweight).**

Efficiency of Approach

- **Why Avalonia UI?** We avoided Electron because it uses too much memory. Avalonia lets us build a modern-looking app that runs on low-end hardware without lagging.
- **Why Python?** It allows us to easily swap out the speech model or the automation library if something better comes along.
- **The "Either" Logic:** The name comes from our design philosophy: the system works either online or offline. Accessibility shouldn't depend on your Wi-Fi signal.

---

7. Impact, Feasibility & Ethics

- **Impact:** We estimate this could speed up computer tasks by 50-70% for someone who currently uses an on-screen keyboard.
- **Feasibility:** We only used open-source libraries (MIT License), so anyone can use or modify this project for free.
- **Ethics:** We included an "Emergency Stop" feature. If the bot starts doing something you didn't ask for, a single voice command kills the process immediately.

---

8. Phase 2 Strategy

The next phase will focus on improving intelligence, reach, and system level control. Planned features include Context Memory for handling follow up commands, Multilingual Support with Hindi, and Custom Commands for personalized workflows.
The assistant will move beyond browser-based automation by integrating directly with the Operating System's Accessibility API, enabling background control of any application. A Lightweight Edge Mode is also planned to reduce CPU usage and improve offline performance.