CSC490 Assignment 1: OpenBrowser-AI

General-Purpose AI Agent with Agentic Browser Capabilities

Muhammad Enrizky Brillian (1009713712), Madhav Kanna Thenappan (1007841659),
Rohit Shetty (1007979020), Joseph Yu (1009196521)
University of Toronto

January 17, 2026

1 Interest Statement

1.1 Problem Statement

Our team is working on **OpenBrowser-AI**—a general-purpose AI agent platform where the agentic browser is one of many tools. Unlike single-purpose browser automation tools, OpenBrowser-AI can orchestrate complex workflows that combine web research, data extraction, and content creation. For example: "Research competitor pricing and create a PowerPoint presentation" or "Analyze this website's frontend using React Fiber and build a similar UI." The agentic browser becomes a tool for gathering context, while the agent handles the end-to-end task.

1.2 Why We Care

The browser automation market is experiencing explosive growth. In December 2025, Meta acquired Manus AI for $2.5 billion, and in September 2025, Atlassian acquired The Browser Company for $610 million. These acquisitions signal that autonomous AI agents represent a multi-billion dollar opportunity.

We care about this problem because:

- **Real-world impact**: General-purpose agents that combine browsing, file creation, and app integration save thousands of hours
- **Technical depth**: Combines LLMs, computer vision, DOM manipulation, reinforcement learning, and multi-tool orchestration
- **Open-source opportunity**: Most solutions are proprietary; we're building an open, extensible alternative
- **Easy integration**: Seamless integration with existing tools (Slack, Discord) via simple @mentions
- **Course alignment**: Directly applies fine-tuning (Week 5, 8), RL (Week 9), and model serving (Week 5, 12)

1.3 Team Member Contributions

- **Muhammad Enrizky Brillian**: Fine-tuning smaller models (QLoRA/DoRA) and reinforcement learning (RLHF/PPO/GRPO) for multi-tool orchestration
- **Madhav Kanna Thenappan**: Cloud infrastructure, app integration layer (Slack/Discord), and visual workflow builder
- **Rohit Shetty**: Systems optimization, model serving, and Agent vs CodeAgent mode selection optimization

- **Joseph Yu**: Evaluation framework and benchmark pipeline for Agent vs CodeAgent performance comparison

2 Landscape Analysis

Relevant Item	Description	Commentary
Anthropic Computer Use	Claude's native ability to control desktop applications via screenshots and mouse/keyboard actions.	Pioneering multimodal agent control. Limitation: Screenshot-based (slower than DOM), no open-source implementation.
Manus AI (Meta, $2.5B)	AI agent startup focused on autonomous task execution. Acquired December 2025.	Validates market opportunity. Proprietary technology now inside Meta. Shows enterprise demand.
The Browser Company (Atlassian, $610M)	AI-native browser with built-in automation. Acquired September 2025.	Browser-first approach. Integration with Atlassian suggests enterprise workflow automation is key.
Playwright (Microsoft)	Cross-browser automation library. Industry standard for web testing.	Foundation technology we build on. Excellent reliability but no AI/LLM integration.
Selenium (Open Source)	Legacy browser automation framework. Widely used in enterprise.	Mature but dated. No native AI support. Large user base represents migration opportunity.
AgentGPT / AutoGPT	General-purpose autonomous AI agents using GPT-4.	Broader scope than browser-specific. Key gaps: Not optimized for browser tasks, no DOM understanding.
WebVoyager (Research, 2024)	"End-to-End Web Agent with Large Multimodal Models" benchmark.	Important evaluation framework. We use their benchmark for testing.
SeeAct (Research, 2024)	"GPT-4V(ision) is a Generalist Web Agent" - multimodal web navigation.	Demonstrates vision-based approach. Our hybrid DOM+vision approach outperforms pure vision.
n8n / Zapier	Visual workflow automation platforms connecting 5000+ apps.	Different approach: pre-built integrations vs. autonomous browsing. Complementary.
OpenAI Atlas (Oct 2025)	AI-native browser with ChatGPT deeply integrated. Agent Mode for multi-step tasks.	Major validation from industry leader. Closed-source, requires Plus/Pro subscription.
Perplexity Comet (Jul 2025)	AI-powered browser with Comet Assistant for agentic automation.	Strong enterprise features (SOC 2, GDPR, HIPAA). Known hallucination issues in complex workflows.

3 Project Outline

3.1 Problem Statement

Current AI assistants are either (1) chat-only systems that cannot take action, or (2) narrow automation tools that only work within a single domain. Real-world tasks often require combining multiple capabilities: researching information online, creating documents, and integrating with workplace tools.

The gap: There is no open-source, general-purpose AI agent that can use browser automation as one tool among many, while seamlessly integrating with existing workplace applications through simple interfaces like @mentions.

3.2 Proposed Solution

OpenBrowser-AI: A general-purpose AI agent platform where the agentic browser is a powerful tool, not the entire system:

- **Browser as a Tool**: Use web browsing for research, data extraction, and DOM analysis
- **Multi-Tool Orchestration**: Combine browsing with file creation (PowerPoint, CSV, PDF), code generation, and API calls
- **App Integrations**: Invoke the agent from Slack (@openbrowser), Discord, or any platform via webhooks
- **12+ LLM Providers**: OpenAI, Anthropic, Google, Groq, AWS Bedrock, Ollama, etc.
- **Two Agent Modes**: Tool-based Agent for reliability, CodeAgent (Jupyter-like) for flexibility

3.3 Architecture

OpenBrowser-AI supports two agent modes optimized for different use cases:

- **Agent (Tool-Based)**: LLM outputs structured JSON tool calls executed by a Tools Registry. More reliable and easier to debug. Best for UI interactions.
- **CodeAgent (Python-Based)**: LLM generates and executes Python code directly (Jupyter-like). More flexible for complex data processing. 3x faster for extraction tasks.

See Appendix A for the general-purpose agent vision and Appendix C for detailed architecture diagrams.

3.4 Milestones

1. **General-Purpose Tool Registry**: Extend beyond browser to file creation, code execution, API calls
2. **App Integration Layer**: Build Slack, Discord, webhook integrations with @mention invocation
3. **Fine-tuning Pipeline**: Implement QLoRA/DoRA fine-tuning for smaller models
4. **Reinforcement Learning**: Implement RLHF/PPO/GRPO to optimize tool selection
5. **Evaluation Framework**: Build benchmarks for end-to-end multi-tool workflows
6. **Visual Workflow Builder**: Drag-and-drop interface for non-technical users

3.5 Unknowns to Investigate

- **Tool selection optimization**: How does the agent decide when to use browser vs. other tools?
- **Context propagation**: How to efficiently pass context from Slack threads to downstream tools?
- **Multi-tool RL rewards**: How to define rewards for complex workflows spanning multiple tools?
- **DOM-to-code generation**: Can we reliably extract React Fiber structures and generate components?
- **Integration security**: How to safely handle credentials across multiple apps?

4 Project Press Release

OpenBrowser-AI: The AI Agent That Does Your Work, Not Just Answers Questions

Open-source general-purpose AI agent platform combines autonomous browser automation with multi-tool orchestration and seamless Slack/Discord integration

Target Launch Date: April 2026 (Public Beta) **Location**: Toronto, ON

OpenBrowser-AI is an open-source AI agent platform for knowledge workers and engineering teams. Unlike chatbots that only answer questions, OpenBrowser-AI agents autonomously execute multi-step workflows: browsing websites, extracting data, creating presentations, and integrating with any social media platform or workplace tool like Slack or Discord—all from a single command.

The Problem

1. **Fragmented workflows**: Research requires switching between browser, spreadsheets, and presentation tools.
2. **Chatbots don't act**: Current AI assistants answer questions but cannot perform tasks.
3. **Brittle automation**: Traditional browser automation breaks when websites change.

The Solution

- **End-to-end execution**: Browses, extracts, and generates documents automatically.
- **App integration**: Type `@openbrowser` in Slack/Discord.
- **Intent-based**: LLMs adapt to website changes automatically.
- **Two agent modes**: Tool-based for UI, CodeAgent for data.

Why We Built This: *"Browser automation alone isn't enough. Real tasks require combining web research with document creation and integration with tools teams already use."* — OpenBrowser-AI Team

How It Works

Option 1: Cloud (Zero Setup)

1. Go to `https://openbrowser.me`
2. Sign in with Google or GitHub
3. Start typing tasks—no API keys required

Option 2: Local (Full Control)

1. `pip install openbrowser-ai`
2. Set your LLM API key (12+ providers)
3. Run via Python API, CLI, or Slack/Discord

Customer Quote

"What took 2 days now takes 20 minutes. We just type @openbrowser in Slack and it handles everything—from web research to the final PowerPoint."

— **Sarah**, Data Operations Lead, TechCorp

Get Started Today

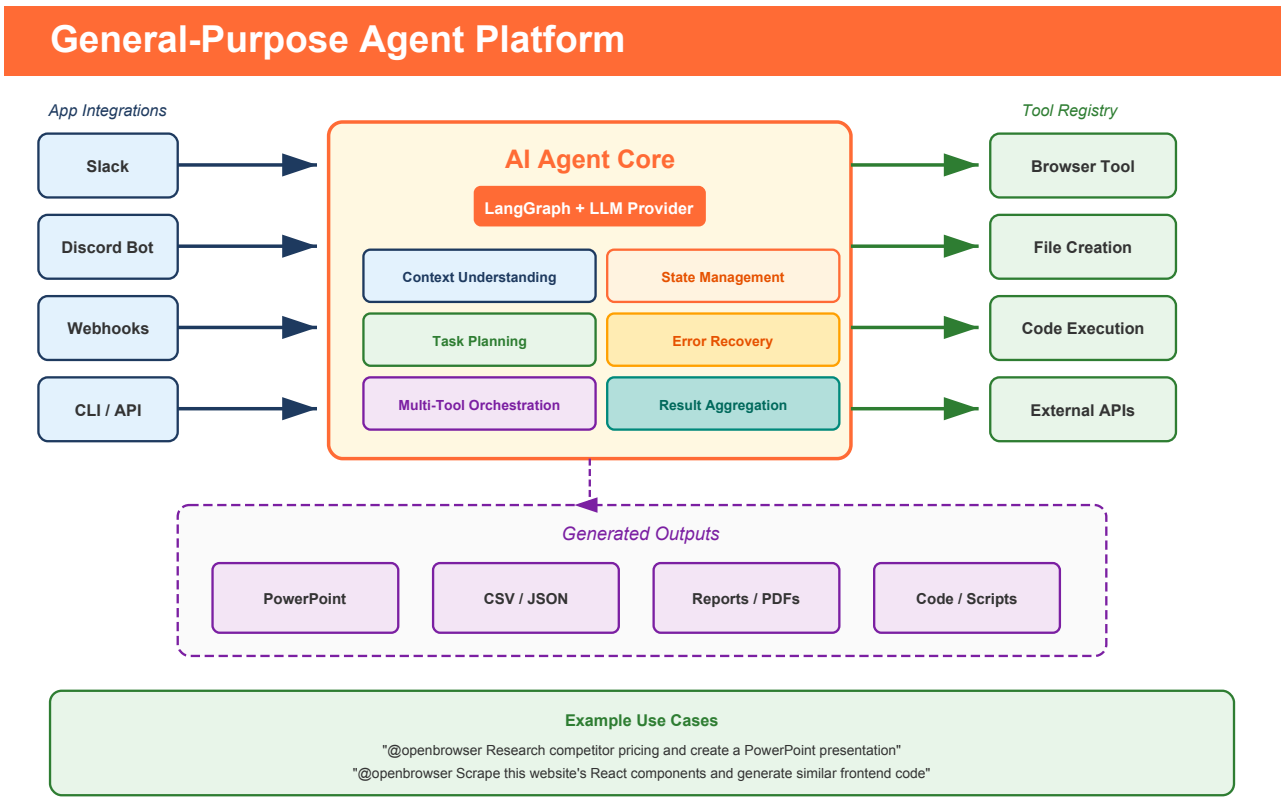
Cloud: `https://openbrowser.me`
PyPI: `pip install openbrowser-ai`
GitHub: `github.com/billy-enrizky/openbrowser-ai`

Contact: billy.suharno@mail.utoronto.ca — madhav.thenappan@mail.utoronto.ca — rohit.shetty@mail.utoronto.ca — yujoseph.yu@mail.utoronto.ca

For frequently asked questions, see Appendix E.

A Appendix A: General-Purpose Agent Vision

The following diagram illustrates the broader vision for OpenBrowser-AI as a general-purpose agent platform with multiple input sources, tools, and output types.



B Appendix B: Performance Benchmarks

Performance benchmarks comparing OpenBrowser Agent vs CodeAgent modes.

Test Configuration:

- **LLM**: Google Gemini (gemini-2.5-flash and gemini-3-flash-preview)
- **Temperature**: 0 (deterministic outputs)
- **Browser**: Headless Chrome via Playwright
- **Agent Settings**: max_failures=5, max_actions_per_step=10
- **CodeAgent Settings**: max_steps=50, max_failures=5

Task Descriptions:

1. Product Configurator: Navigate to product page, select MacBook Pro with Space Gray color, 1TB storage, 32GB RAM, then click Add to Cart. Tests dropdown selection and button clicks.

2. Flight Booking (6-step): Complete multi-step booking flow—search flights (NYC to London, dates, passengers, class), select flight, enter passenger details (name, DOB, passport), select seat, enter payment info (card number, expiry, CVV), complete booking. Tests complex sequential workflows.

3. HN Data Extraction + CSV: Navigate to Hacker News, extract top 10 story titles, URLs, and points, save to CSV file. Tests data extraction and file I/O.

Task Type	Agent Time	Agent Steps	CodeAgent Time	CodeAgent Steps	Winner
Product Configurator	16.50s	3	40.64s	6	Agent 2.5x
Flight Booking (6-step)	149.19s	19	46.87s	8	CodeAgent 3.2x
HN Data Extraction + CSV	48.10s	5	14.72s	3	CodeAgent 3.3x

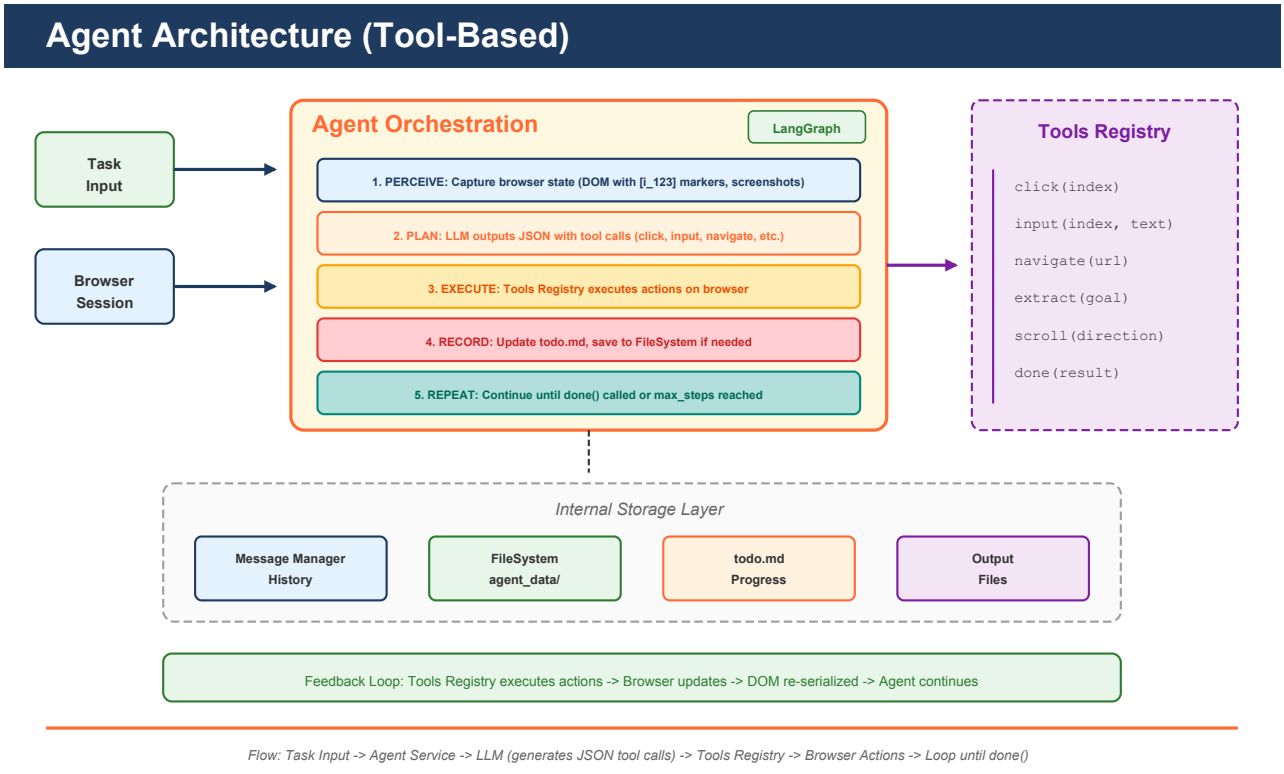
Key Findings:

- **CodeAgent** is 3x faster for data extraction and research tasks (uses JavaScript evaluate() + Python csv/pandas)
- **Agent** is 2.5x faster for simple UI interactions (batches multiple tool calls in single LLM response)
- CodeAgent uses fewer steps by combining operations in Python code blocks
- Agent more consistent across model versions due to structured JSON tool calls
- CodeAgent performance highly dependent on LLM code generation quality

C Appendix C: Agent Architecture Diagrams

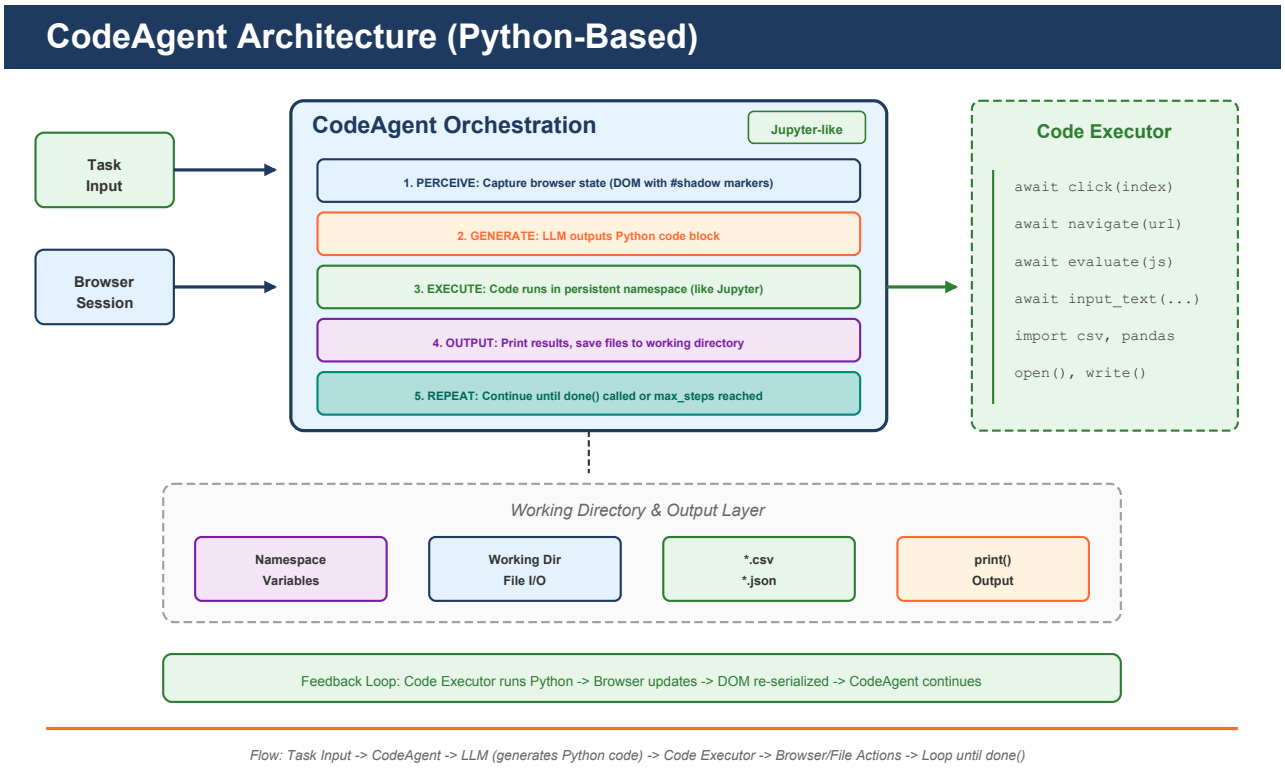
Agent Architecture (Tool-Based)

The tool-based Agent uses a structured approach where the LLM outputs JSON tool calls that are executed by a Tools Registry.



CodeAgent Architecture (Python-Based)

The CodeAgent generates and executes Python code directly, similar to a Jupyter notebook.



D Appendix D: Press Release Iterations

Iteration 1 (Technical Focus): Initial draft emphasized technical architecture (LangGraph, CDP, DOM serialization). Feedback: Too technical for general audience, buried the value proposition.

Iteration 2 (Feature List): Rewrote to lead with feature list. Feedback: Reads like documentation, not compelling narrative. Missing the "why should I care" hook.

Iteration 3 (Problem-Solution): Restructured around the pain point (brittle scripts) and solution (intent-based agents). Added customer quotes. Feedback: Much stronger, but opening paragraph was weak.

Final Version: Refined opening to immediately contrast with chatbots ("actually does work"), added concrete examples, strengthened customer testimonials with specific metrics, emphasized the general-purpose agent vision with app integrations.

E Appendix E: Frequently Asked Questions (FAQ)

Customer FAQs

Q: How is OpenBrowser-AI different from ChatGPT or Claude?
A: ChatGPT/Claude answer questions but cannot take actions. OpenBrowser-AI is an *agent* that autonomously executes tasks: browsing, filling forms, extracting data, creating files. You describe what you want done, and it does it.

Q: What LLM providers are supported?
A: 12+ providers: OpenAI (GPT-4o), Anthropic (Claude), Google (Gemini), Groq, AWS Bedrock, Azure OpenAI, Ollama (local), DeepSeek, Cerebras, OpenRouter. Switch providers with one config change.

Q: Is my data secure?
A: Yes. Runs locally—data never passes through our servers. Browser sessions are isolated, sensitive data can be masked in logs.

Q: Can I use this for web scraping?
A: Yes. CodeAgent mode is optimized for extraction using JavaScript + Python (pandas, csv). 3x faster than tool-based Agent.

Q: What tasks can it handle?
A: Competitive research, lead generation, price monitoring, form filling, report generation, social media monitoring, automated testing.

Internal/Technical FAQs

Q: Why two agent modes?
A: Tool-based Agent uses JSON tool calls—reliable for UI interactions. CodeAgent generates Python code—faster for data extraction. Agent is 2.5x faster for UI tasks; CodeAgent is 3x faster for extraction.

Q: How does it handle website changes?
A: Uses LLMs to understand page structure semantically, not brittle CSS selectors. DOM serialized with element indices; LLM decides based on intent.

Q: What's the architecture?
A: LangGraph for workflow orchestration (perceive-plan-execute loop). Browser via Chrome DevTools Protocol (CDP)/Playwright. Maintains conversation history and browser state.

Q: How to add custom tools?
A: Decorator-based API: define Python function with type hints, add `@tool` decorator. For apps, use webhook endpoints for Slack/Discord.

Q: System requirements?
A: Python 3.10+, 8GB RAM (16GB recommended), Chrome/Chromium. LLM API access required.