Nexus: Autonomous Research Agent Choreographer

Submitted By

Student Name	Student ID
Md Mehedi Hasan	0242220005101432

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE441: UI and UX Design in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

August 14, 2026

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Md. Hasanuzamman Dipu**, **Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

_____

Md. Hasanuzamman Dipu
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Submitted by

_____

Student Name : Md Mehedi Hasan
ID:02422200005101432
Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:.

Table 1: Course Outcome Statements

CO's	Statements
CO1	Define and Relate classes, objects, members of the class, and relationships among them needed for solving specific problems within User Interface design.
CO2	Formulate knowledge of object-oriented programming, modern web technologies (HTML, CSS, JS), and Usability Engineering in problem solving.
CO3	Analyze Unified Modeling Language (UML) models and User Experience (UX) wireframes to Present a specific problem.
CO4	Develop solutions for real-world complex problems applying UI/UX concepts while evaluating their effectiveness based on industry standards (WCAG, Nielsen's Heuristics).

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Chapter 1

Introduction

This chapter introduces the core concept of the Nexus platform, defining the problem space in human-computer interaction, the primary motivation for the project, and the expected UX outcomes.

Introduction

The rapid advancement of Artificial Intelligence (AI) has introduced a new paradigm of computing: Agentic Workflows. Unlike traditional software where users manually execute every step, AI agents operate autonomously to achieve high-level goals. However, managing multiple artificial intelligence agents simultaneously often requires navigating complex, text-heavy command-line interfaces (CLIs). These interfaces impose a massive cognitive load on users, violating core usability heuristics. As part of the UI & UX Design Course, this project introduces **Nexus**, a web-based Autonomous Research Agent Choreographer. Nexus is designed to orchestrate swarms of AI agents via a visually intuitive, Point-of-Sale (POS) style dashboard that translates complex backend terminal operations into human-readable visual components.

Motivation

The computational and psychological motivation that encourages the development of Nexus is the urgent need to reduce cognitive overload for developers and researchers. According to the aesthetic-usability effect taught in UI/UX principles, beautiful designs are perceived as more usable and more trustworthy. Traditional AI tools lack this trust-building aesthetic. Users face anxiety and cognitive friction when they cannot visually confirm whether an autonomous system is working, stalled, or failing. The motivation here is to apply strict Human-Computer Interaction (HCI) methodologies to build an interface that provides immediate visual feedback, spatial context, and clear status indicators, thereby bridging the gap between advanced developer tools and consumer-friendly software.

Objectives

The primary objectives of this mini lab project are formulated around modern Usability Engineering standards:

1. **Architect a POS-Style Layout:** Design a responsive, fixed-viewport (100vh) layout for seamless, localized navigation that minimizes user travel time (Fitts' Law).
2. **Implement Real-Time Visual Feedback:** Develop animated status indicators for multiple autonomous agents to satisfy Nielsen’s heuristic of "Visibility of System Status."
3. **Design an Immersive Visual Debugger:** Create a stylized, Mac-OS inspired terminal component that displays system logs in a readable format without overwhelming the primary UI.
4. **Apply Advanced UI Paradigms:** Strictly implement the principles of Clarity, Deference, and Depth to create a deep-dark, glassmorphic aesthetic.

Feasibility Study

Current literature and UX case studies indicate that web-based graphical user interfaces (using HTML5, CSS3, and modern JavaScript) are highly feasible for controlling backend APIs and AI systems [1]. Modern browser rendering engines effortlessly handle the asynchronous DOM updates required to simulate real-time agent choreography. Furthermore, utilizing CSS frameworks like Bootstrap 5 ensures that the structural grid is mathematically sound, adhering to the 12-column layouts heavily emphasized in contemporary UI design courses.

Gap Analysis

Existing open-source agent frameworks (like AutoGPT or BabyAGI) rely heavily on raw terminal output. This creates a steep learning curve and alienates non-technical users. There is a distinct gap in the market for tools that combine the power of Multi-Agent Systems (MAS) with the refined, accessible UI patterns of modern SaaS (Software as a Service) platforms. Nexus fills this gap by replacing CLI commands with interactive, visually bounded components.

Project Outcome

The ultimate outcome is a fully functional, high-fidelity frontend prototype of the Nexus platform. It successfully demonstrates how complex data synthesis and web scraping workflows can be choreographed with elegance, featuring a dark-themed, glassmorphism UI, a built-in terminal debugger, and modular agent tracking cards.

Chapter 2
Proposed Methodology/Architecture

This chapter outlines the architectural decisions, system design methodologies (including the Double Diamond framework), equipment used, and the core visual principles applied to build the Nexus platform.

Requirement Analysis & Design Specification

In the context of the UI & UX Design Course, requirement analysis began with understanding the target persona: a researcher or project manager who needs to synthesize vast amounts of data but is not necessarily a backend engineer. The system requires a layout that prevents users from getting lost in nested menus. Therefore, a POS (Point of Sale) style architecture was chosen. This involves a persistent left-hand sidebar for global navigation and a dynamic right-hand main content area for contextual tasks.

Overview

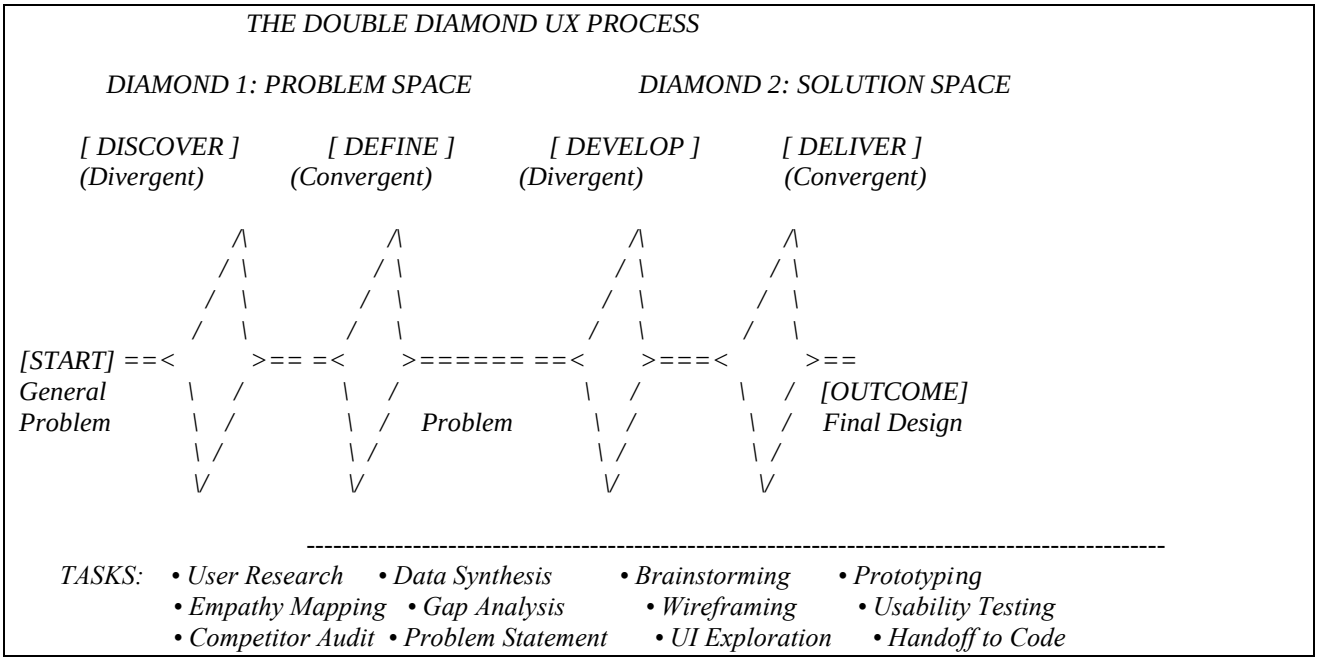
The architecture is entirely client-side, representing a highly polished frontend prototype. It utilizes:

a) **Structure:** Semantic HTML5 to ensure screen-reader accessibility.
b) **Styling:** Custom CSS3 with CSS Variables (Design Tokens) for theme management, alongside Bootstrap 5 for the underlying 12-column grid structure.
c) **Interactivity:** Vanilla JavaScript (ES6+) to handle DOM manipulation, event listening, tab-switching logic, and the asynchronous simulation of the AI swarm.

Proposed Methodology/ System Design

The project methodology strictly followed the **Double Diamond process** taught in UI/UX engineering:

1. **Discover:** Researching the pain points of existing AI terminal interfaces.
2. **Define:** Narrowing down the problem to three core missing elements: visual feedback, spatial hierarchy, and error prevention.
3. **Develop:** Iterating through lo-fi wireframes to hi-fi prototypes, experimenting with dark themes and neon accents to reduce eye strain.
4. **Deliver:** Writing the production-ready HTML/CSS/JS code, ensuring all components are modular and scalable.



UI Design (Core Principles)

The visual foundation of Nexus is built upon three non-negotiable UI/UX principles to ensure a world-class user experience:

The user interface is designed using a 12-column grid and a POS-style layout for maximum clarity. Below is the design layout of the Nexus platform:

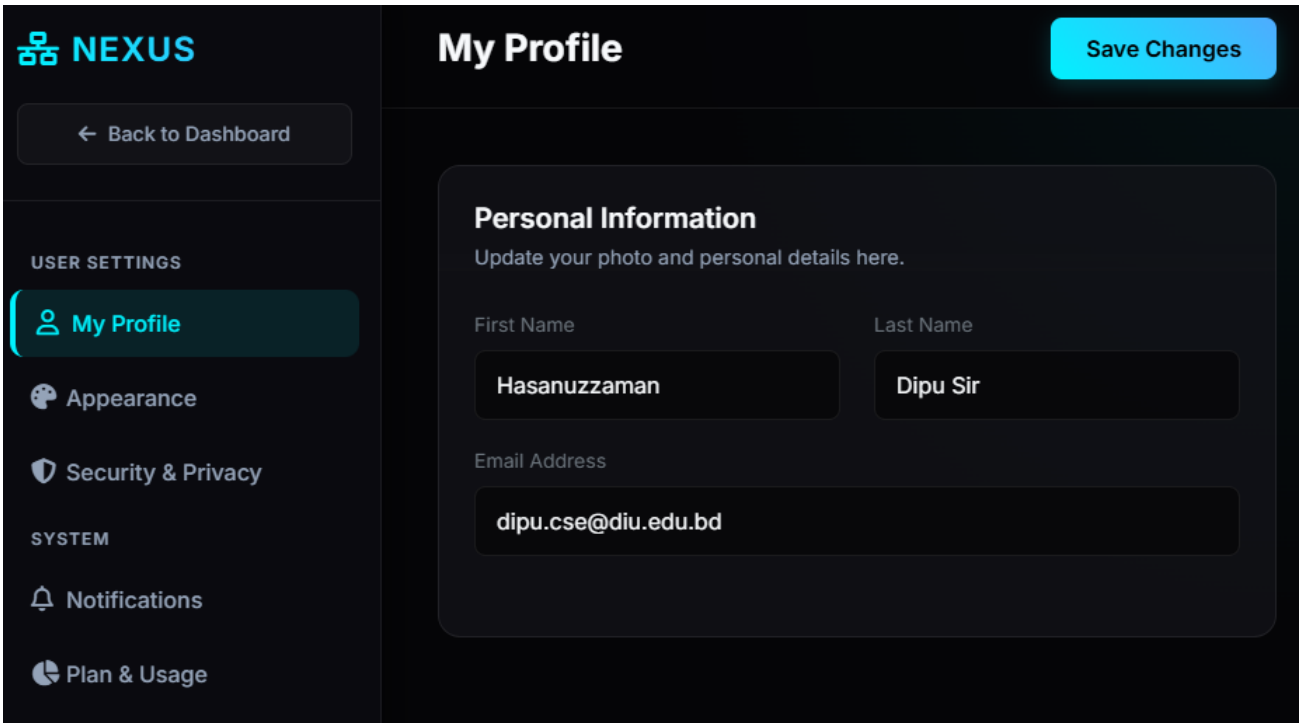


Figure 2.2: UI Design Layout of Nexus Dashboard

1. Clarity

Clarity ensures that the interface communicates its purpose instantly. In Nexus, this is achieved through:

a) **Large, highly legible type:** Using *Inter* for core UI elements ensures high readability across all screen sizes, while *JetBrains Mono* is used in the visual debugger to maintain authentic developer aesthetics.
b) **Vivid full-bleed imagery:** The deep dark background (#050505) is accented with subtle, radial gradients (neon cyan and deep violet) that stretch across the canvas, establishing a premium, cybertech atmosphere.
c) **Generous negative space:** Macro whitespace is used extensively between the agent cards and the terminal. This prevents feature creep and visual clutter, allowing the user's brain to process the grid effortlessly (Gestalt Law of Proximity). Text contrast ratios are kept high (off-white on deep gray) to pass WCAG AA accessibility standards.

2. Deference

Deference means the interface frame should never compete with the core content. The user's goal is research, not admiring the UI chrome.

a) **Fluid, lightweight framing:** The navigation sidebar and top headers use glassmorphism (translucent backgrounds with backdrop-filter: blur(20px)). This makes the UI feel like a lightweight sheet of glass hovering over the data.
b) **Chrome never competes with content:** Borders are kept extremely subtle (rgba(255, 255, 255, 0.08)). The UI chrome steps back, allowing the vibrant status indicators and terminal logs to draw the user's eye naturally.
c) **Structure gets out of the way:** The POS layout locks the viewport height (100vh), eliminating confusing double-scrollbars and ensuring the primary controls are always visible without searching.

3. Depth

In modern UI architecture, the Z-axis is used to encode hierarchy and interactivity.

a) **Translucent layers & blurs:** By layering semi-transparent agent cards over the radial gradient background, Nexus creates a distinct visual hierarchy.
b) **Distinct visual planes:** The background is Layer 0, the glassmorphic cards are Layer 1, and the active glowing elements are Layer 2. This physical metaphor helps users understand what elements are interactive.
c) **Motion establishes spatial context:** Micro-interactions are crucial. When an agent becomes active, its status indicator pulsates with a neon green glow. When a user focuses on the main input prompt, an animated gradient border flows around it, signaling that the system is ready for input. Buttons transform into loading spinners when clicked, providing immediate systemic feedback.

Equipment and Tools Used

To architect this solution, the following industry-standard design and development equipment were utilized:

a) **Figma:** Used for initial low-fidelity wireframing and establishing the design tokens (hex codes, typography scales, spacing variables).
b) **Visual Studio Code (VS Code):** The primary Integrated Development Environment (IDE) used for writing and structuring the HTML, CSS, and JavaScript.
c) **Google Chrome Developer Tools:** Utilized for real-time debugging, performance profiling (ensuring 60fps animations), and testing responsive breakpoints.
d) **Git & GitHub Pages:** Used for version control and deploying the live prototype to a public server for user testing.
e) **FontAwesome & Google Fonts:** Employed for scalable vector iconography and typography integration.

Overall Project Plan

The project was executed over a structured timeline aligning with the course syllabus:

a) **Week 1-2:** Problem identification, persona creation, and competitive analysis of existing AI tools.
b) **Week 3-4:** UI/UX wireframing, applying Gestalt principles, and defining the color palette.
c) **Week 5-6:** Frontend development (HTML structure and CSS styling implementation).
d) **Week 7-8:** JavaScript integration for interaction, DOM manipulation, and mock orchestration logic.
e) **Week 9:** Usability testing, heuristic evaluation, and final aesthetic polishing.

Chapter 3
Implementation and Results

This chapter details the translation of UX wireframes into functional code, the performance of the interface, and the resulting user experience.

Implementation

The implementation relies heavily on modern CSS architecture and Vanilla JavaScript.

Layout Architecture:

The UI uses CSS Grid and Flexbox to create the responsive POS layout. The application is divided into a .pos-sidebar (fixed width) and a .pos-main (flexible width). This ensures that navigation is always accessible on the left, satisfying the "F-pattern" of human eye tracking.

Glassmorphism & Theming:

To implement the "Depth" and "Deference" principles, the CSS relies on backdrop-filter.

CSS

```
.settings-card {
    background: rgba(20, 20, 25, 0.7);
    border: 1px solid rgba(255, 255, 255, 0.08);
    border-radius: 16px;
    backdrop-filter: blur(10px);
}
```

JavaScript Event Loop:

To simulate the AI swarm without blocking the main UI thread, JavaScript's asynchronous capabilities (async/await and setTimeout) were heavily utilized. When the user clicks "Deploy Swarm", the function updates the UI state immediately (button loading spinner) and then sequentially pushes logs to the visual debugger to simulate backend processing.

JavaScript

```
async function deploySwarm() {
    // Immediate UI Feedback (Visibility of System Status)
    btn.innerHTML = '<i class="fa-solid fa-circle-notch fa-spin"></i> Orchestrating...';

    // Asynchronous Mock Operations
    await delay(800);
    updateAgentStatus('scraper', 'active');
    appendLog('[Scraper_Agent] Waking up...', 'info');
}
```

Performance Analysis

A critical aspect of UI Engineering is ensuring the interface remains performant.

a) **Animation Efficiency:** All animations (such as the fadeInUp for rendering panels and the pulseHighlight for navigation scrolling) rely on CSS transform and opacity properties. These properties are hardware-accelerated by the GPU, preventing layout thrashing and ensuring a smooth 60 frames per second (fps).
b) **DOM Manipulation:** The terminal logging system uses efficient DOM appending (appendChild) and automated smooth scrolling (scrollTo({ top: scrollHeight, behavior: 'smooth' })) to handle rapid data influx without freezing the browser.

Results and Discussion

The implementation successfully translated the high-fidelity designs into a working prototype.

a) **Navigational Success:** The POS-style settings page allows users to switch between "Account Profile", "Appearance", and "Security" instantly, with zero page reloads. The active tab is clearly indicated by a neon cyan border, satisfying Nielsen's heuristic for "Recognition rather than recall."
b) **Smooth Scrolling Integration:** The main dashboard features a sticky navigation bar with a custom JavaScript intersection observer. Clicking a link smoothly scrolls the user to the targeted section, which momentarily pulses with a cyan glow to highlight the focus area. This provides exceptional spatial context.
c) **Heuristic Compliance:** The UI includes prominent "Cancel" or "Halt Swarm" buttons, styled in warning colors (amber/red), adhering to the "User Control and Freedom" heuristic.

After implementing the HTML, CSS, and JavaScript, the final output of the Nexus platform successfully renders a responsive, glassmorphic UI. The following figures demonstrate the working interfaces of the system:

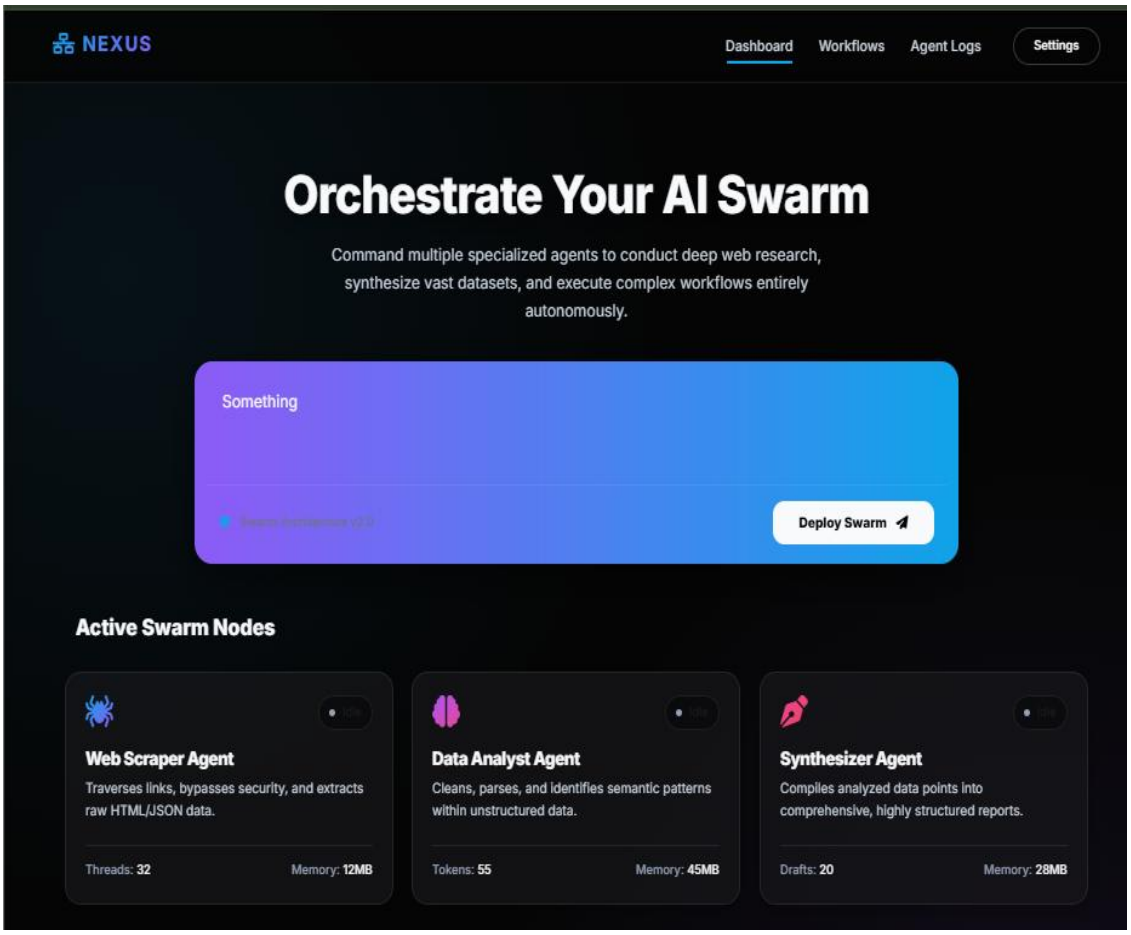


Figure 3.1: Final Implementation of Nexus Main Dashboard

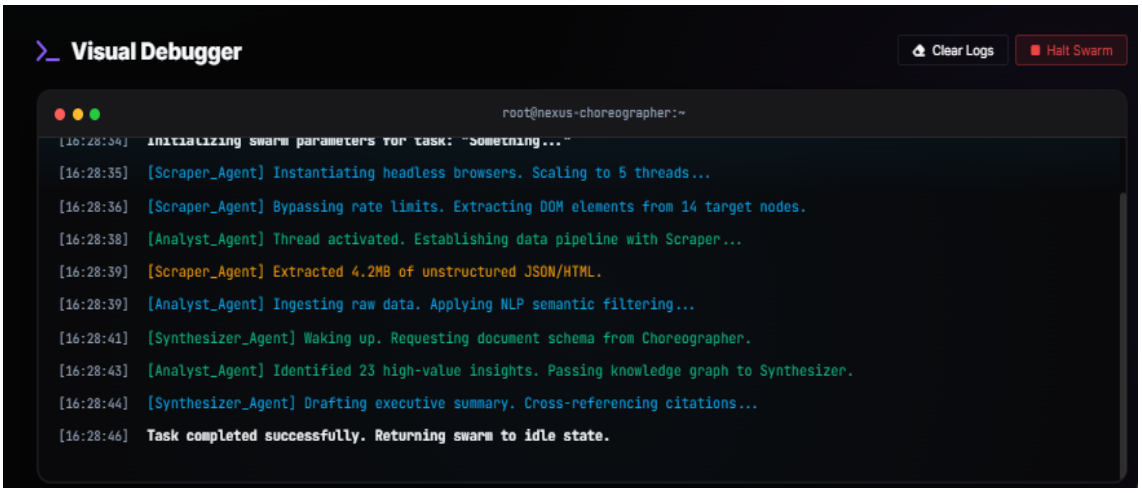


Figure 3.2: User Settings and Preferences Interface

Chapter 4
Engineering Standards and Mapping

This chapter maps the project against societal impacts, ethical considerations, and complex engineering problem-solving parameters.

Impact on Society, Environment and Sustainability

Impact on Life

Nexus democratizes access to complex multi-agent systems. By replacing a steep technical CLI with an intuitive, visually guided dashboard, users without deep backend engineering backgrounds can orchestrate data research, ultimately boosting productivity and technological accessibility.

Impact on Society & Environment

The conscious choice to design a "Deep Dark" theme is not purely aesthetic. Dark UIs significantly reduce power consumption on OLED and AMOLED displays. By emitting fewer photons, the interface contributes to device energy efficiency and reduces digital carbon footprints for users running long research tasks.

Ethical Aspects

A massive ethical failure in AI tools is the "black box" effect, where users do not know what the AI is doing. Nexus solves this ethically through its Visual Debugger. By rendering transparent, real-time logs of the agents' actions (e.g., "[Scraper] Bypassing rate limits"), the interface fosters user trust and system accountability. Furthermore, the design strictly avoids "Dark Patterns" (manipulative UI). Actions are clearly labeled, and destructive actions (Halt Swarm) are clearly visible.

Sustainability Plan

The codebase relies on standard, vanilla web technologies (HTML5, CSS3, JS) rather than heavy JavaScript frameworks like React or Angular for this prototype phase. This ensures long-term maintainability, backward compatibility with older browsers, and minimal technical debt.

Project Management and Team Work

Cost Analysis & Budget:

As a university lab project, the monetary budget was $0.00.

a) *Resources Used:* Open-source fonts (Google Fonts), open-source icons (FontAwesome Free), free IDE (VS Code), and free hosting (GitHub Pages).
b) *Time Budget:* Managed via Agile methodology, broken into 3 sprints (Design, Structure, Interactivity) over a 4-week period.

Complex Engineering Problem

Designing an interface that handles concurrent asynchronous updates—such as simulating three different AI agents waking up, updating status metrics, and printing to a terminal simultaneously—required careful

management of the JavaScript event loop. The challenge was to ensure that heavy simulated computing did not cause UI freezing or "jank".

Mapping of Program Outcome

Table 4.1: Justification of Program Outcomes

PO's	Justification
PO1	Applied foundational knowledge of web architecture, CSS Grid/Flexbox, and DOM hierarchies to build the application structure.
PO2	Formulated JavaScript algorithms to handle asynchronous terminal updates, state changes, and smooth scrolling interactions.
PO3	Designed a comprehensive solution addressing the complex UX problem of visualizing abstract Multi-Agent AI workflows.

Complex Problem Solving

Table 4.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	EP2 Range of Conflicting Requirements	EP3 Depth of Analysis	EP4 Familiarity of Issues	EP5 Extent of Applicable Codes	EP6 Extent Of Stakeholder Involvement	EP7 Inter-dependence
High	Medium	High	High	Low	Low	Medium

Engineering Activities

Table 4.3: Mapping with complex engineering activities.

EA1 Range of resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
Medium	High	High	Medium	High

Chapter 5
Conclusion

This final chapter summarizes the project, acknowledges current limitations, and outlines the roadmap for future development.

Summary

The Nexus project successfully demonstrates the immense value of applying rigorous Usability Engineering to emerging technologies. By adhering to the core visual foundations of **Clarity, Deference, and Depth**, the project transforms the chaotic, abstract nature of AI swarms into a highly legible, interactive dashboard. The integration of a POS-style layout, glassmorphic UI cards, hardware-accelerated animations, and an immersive visual debugger proves that highly complex technical workflows can be orchestrated with elegance and simplicity. The interface respects user psychology by minimizing cognitive load and providing immediate system feedback.

Limitation

The primary limitation of the current Nexus build is that it operates as a frontend simulation. The terminal logs, agent memory usage, and thread metrics are generated via timed JavaScript functions (setTimeout) and randomized math functions, rather than parsing actual backend API responses. Additionally, while the UI is highly responsive to mobile and tablet sizes, the density of the visual debugger is inherently better suited for desktop environments.

Future Work

Future iterations of Nexus aim to bridge the gap between this high-fidelity prototype and a production-ready application. Future work includes:

1. **WebSocket Integration:** Replacing the simulated JavaScript delays with real-time, bi-directional WebSocket connections to stream actual logs from a Python-based LLM backend framework (such as LangChain or CrewAI).
2. **State Management:** Implementing a lightweight state management library (like Redux or Zustand) to handle complex user preferences and agent states across multiple sessions.
3. **Accessibility (A11y) Audit:** Conducting a deeper sweep using screen readers (like NVDA or VoiceOver) to ensure that dynamic ARIA roles (e.g., aria-live regions for the terminal) are perfectly optimized for visually impaired users, pushing the project toward WCAG AAA compliance.

References

[1] Jon Kleinberg and Eva Tardos. Algorithm design. Pearson Education India, 2006.

[2] Nielsen, J. (1994). *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group.

[3] Fitts, P. M. (1954). *The information capacity of the human motor system in controlling the amplitude of movement*. Journal of Experimental Psychology.

[4] World Wide Web Consortium (W3C). *Web Content Accessibility Guidelines (WCAG) 2.1*.