AI Agents

- Zed (not going to use but keep an eye for the windows release date)
    - Coming to windows eventually
    - Designed for collaboration with the human programmer and AI
    - Entire Zed code editor is open source under GPL version 3, and scratch-built in Rust down to handcrafted GPU shaders and OS graphics API calls
    - Zed's AI capabilities are also open-source, just like the rest of the editor
    - Can use multiple ai applications (Claude 3.7, GPT, openAI, etc)
    - Secure, no data used for training
    - Paywall for AI however
- Cursor
    - Available for download on microsoft
    - Another ai code editor
    - 3 categories
        - Agent
            - Complete tasks end to end. Does this quickly while keeping programmers in the loop.
            - Finds context using custom retrieval models, cursor can understand a codebase. This reduces the need to manually add context
            - Runs commands by automatically write and run terminal commands. By default, you'll be asked to confirm all commands
            - Loops on errors - cursor can detect lint errors automatically and apply fixes, reducing the need for manual debugging
        - Tab
            - Cursor includes a powerful autocomplete that predicts your next edit. Once enabled, it is always on and will suggest edits to your code across multiple lines, taking into account recent changes
                - Multi-line Edits - Cursor can suggest multiple edits at once, saving time
                - Smart Rewrites - type carelessly, and Cursor will fix your mistakes
                - Cursor Prediction - Cursor predicts your next cursor position so you can seamlessly navigate your code
        - Chat
            - Lets you talks with an AI that sees your codebase.
            - Chat can always see current file and cursor, so you can ask it things like "Is there a bug here?"
            - Can add particular blocks of code to the context with Ctrl+Shift+L or "@"
            - Instant apply
                - Apply the code suggestions from chat back into your codebase by clicking the play button on top of any chat codeblock

- Codebase Answers
    - Use @Codebase or Ctrl Enter to ask questions about codebase
    - Cursor will search codebase to find relevant code to your query
- Reference your code
    - Reference code with @ symbols to be used as context for the AI. Just type @ to see a list of all the files and code symbols in your folder
- Use images
    - Hit the image button under chat to drag an image into the input box to include visual context into chat
- Ask the web
    - Get up-to-date information from the internet with @ Web
    - Cursor will search the web for you and use the latest information to answer your question
- Use documentation
    - Reference popular libraries using @LibraryName, or add your own using @Docs -> Add new doc

- Ctrl K
    - Lets you edit and write code with the AI
    - To edit, select some code, click “Edit” and describe how the code should be changed
    - To generate completely new code, just type Ctrl K without selecting anything
    - Terminate Ctrl K
        - Write terminal commands in plain english.
        - Cursor will convert them into the terminal command you need
    - Quick questions
        - Quick questions can be asked by clicking “quick question” to get your answer right away
- Security Concerns
    - Cursor depends on subprocessors for infrastructure security
        - AWS - sees the code data. Primary host for the infrastructure. Most servers in US, with some in latency critical servers located in AWS regions like Tokyo and London
        - Microsoft Azure - sees the code data. Secondary infrastructure hosted on Azure. All servers in US
        - Google Cloud Platform (GCP) - sees code data. Some more secondary infrastructure on GCP. All servers in US
        - Fireworks - sees code data. Custom models hosted with Fireworks, servers in US, Asia, Europe. May store code data if privacy mode is disabled to speed up inference for models. Zero data retention agreement with Fireworks

- OpenAI - gives AI response. Requests may be sent to OpenAI even if you have Anthropic (or someone else's) model selected in chat (e.g. for summarization) Zero data retention agreement
- Anthropic - Sees code data. Rely on many of anthropic's models to give AI responses. Requests may be sent to Anthropic even if you have an OpenAI (or something else's) model selected in chat. Zero data retention agreement with Anthropic
- Google Cloud Vertex API - sees code data. Rely on some Gemini models offered over GCP Vertex API to give Ai responses. Requests might be sent to GCP API even if you have another model selected in chat. Zero data retention agreement with vertex
- xAi - sees code data. Rely on some grok models offered over xAi API to give AI responses. Zero data retention agreement with xAI
- Turbopuffer - Stores obfuscated code data. Embeddings of indexed codebases, as well as metadata associated with embeddings (obfuscated file names), are stored with Turbopuffer on Google Cloud's server servers in the US. Users can disable codebase indexing
- Exa and SerpApi - Sees search requests (potentially derived from code data) Used for web search functionality. Search requests potentially derived from code data (e.g. when using @ web in the chat, a separate language model will look at the message, conversation history and current file to determine what to search for, and Exa/SerpApi will see resulting search query)
- Rest of subprocessors see no code data

- Payment Plans
  - Free
    - Pro two-week trial
    - 2000 Completions
    - 50 slow requests
  - Pro
    - 20/month
    - 500 requests per month
    - Unlimitied Slow requests
    - Max mode (users turn on maximum content)
  - Business
    - 40/user/month
    - Enfore privacy mode org-wide
    - Centralized team billing
    - Admin dashboard with usage stats
    - SAML/OIDC SSO
      - SAML - Security Assertion Markup Language (authentication)

            - OIDC - OpenID Connect used to implement SSO (Single Sign-On)
    - Personal test
        - Had the AI make frontend design changes to match the changes made by Windsurf
        - Code ran in the application with no issues
        - Calendar in the add activity page ended up being shrunk, had to prompt it a couple times to fix that
        - Had the ai commit the changes with a message and push it to Github
- Github Copilot
    - Integrates with leading editors, including Visual Studio Code, Visual Studio, JetBrains IDEs, and Neovim, and, unlike other AI coding assistant, is natively built into GitHub
    - Deals with issues - when assigned issues, GitHub Copilot plans, writes, tests and iterates-using GitHub Actions to run code and deliver ready-to-review pull requests
    - Codes like an insider - GitHub Copilot hooks into MCP servers to draw on data from your repositories and external resources-working like an onboarded team member frond ay one
    - Human and Agent in the loop - Comment to guide Github Copilot, polish your code for merge, or take over locally in your IDE
    - Agent mode helps make sweeping changes by analyzing code, proposing edits, running tests, and validating results across multiple files
    - Can swap between models like Claude 3.7 Sonnet, OpenAI o1, and Google Gemini 2.0 Flash to crush coding tasks fast or go deep when it counts
    - Plans/Pricing
        - Free
            - 50 agent mode or chat requests per month
            - 2,000 completions per month
            - Access to Claude 3.5 Sonnet, GPT-4.1, and more
        - Pro
            - Unlimited completions and chats with access to more models
        - Pro+
            - Maximum Flexibility and model choice
            - Access to all models, including GPT-4.5
            - 30x more premium requests than Copilot free to use latest models, with option to buy more
            - Coding agent (preview)
        - Individual
            - Designed for individual developers, probably not best for this internship
        - Business
            - Features copilot in the coding environment
                - IDE, CLI, and GitHub Mobile

        - Enterprise
            - Includes everything in GitHub Copilot
            - Added layer of customization for organizations and integrates into GitHub.com as a chat interface to allow developers to converse with copilot throughout the platform
    - Personal experience
        - Needs very very specific prompts to even try and get close to what Cursor and Windsurf were doing on my personal project
        - However, very helpful with autocompletion based on you're file
        - Inline chats can help with explanation of what might potentially be wrong with code
- Ollama
    - Stands for Omni-Layer Learning Language Acquisition Model
    - Democratizes access to Large Language Models by enabling users to run them locally on their machines
    - Developed with a vision to empower individuals and organizations, Ollama provides user-friendly interface and seamless integration capabilities, maing it easier than ever to leverage power of LLMs for various applications and use cases
    - Key Features
        - Local Execution: One of the distinguishing features of Ollama is its ability to run LLMs locally, mitigating privacy concerns associated with cloud based solutions. Bringing AI models directly to users' devices, Ollama ensures greater control and security over data while providing faster processing speeds and reduced reliance on external servers
        - Extensive Model Library: offers access to an extensive library of pre-trained LLMs, including popular models like Llama 3. Users can choose from a range of models to different tasks, domains, and hardware capabilities, ensuring flexibility and versatility in their AI projects
        - Seamless Integration: Ollama seamlessly integrates with a variety of tools, frameworks, and programming languages, making it easier for developers to incorporate LLMs into their workflows.
        - Customization and Fine-Tuning: With Ollama, users have the ability to customize and fine-tune LLMs to suit their needs and preferences. From prompt engineering to few-shot learning and fine-tuning processes, Ollama empowers users to shape the behavior and outputs of LLMs, ensuring they align with desired objectives.
    - Personal test
        - Only in the command prompt, so unavailable within an IDE
        - Cloud based, so not going to be very helpful in terms of coding
        - Maybe not our most ideal option
- Windsurf (formerly Codeium)
    - Built for the way AI is meant to work with humans
    - Flows

        - Combination of Agents and Copilots
        - Windsurf editor collaborates like a copilot and tackle complex tasks independently like an agent
    - Cascade
        - Combines deep codebase understanding, a breadth of advanced tools, and a real-time awareness of your actions into a powerful, seamless, and collaborative flow. It is most powerful way to code with AI
        - Mult-file, mult-edit capability
        - Deep contextual awareness
        - Terminal command suggestions
        - LLM-based search tools that outperform embeddings
        - Implicit reasoning of your actions in the text editor
    - Other features
        - Linter integration - if code generated doesn't pass linter, Cascade automatically fixes errors
        - Model Context Protocol (MCP) - Enhance AI workflows by connecting to custom tools and services
        - Tab to jump - predicts next location of your cursor to seamlessly navigate through the file
        - Supercomplete - analyzes what your next action might be, beyond just inserting next code snippet
        - In-line Command + Follow Ups - Press Cmd + I in editor to generate or refactor in-line code using natural language
        - Command in Terminal - Press Cmd + I in terminal and type in terminal instructions in natural languages
        - Codelense - available next to breadcrumbs, codelenses let you understand or refactor code with one click
        - Highlighted code actions - you can directly mention highlighted code in the Cascade panel or refactor it using Command
        - @ mentions in Cascade - refer to your functions, classes, files, or entire directories to guide Cascade to relevant context
    - Pricing
        - Free
            - 2 week pro trial
            - 25 prompt credits per month
            - All premium models
            - Optional zero data retention
            - Unlimited fast tab
            - Unlimited SWE-1 Lite
            - Unlimited Command
            - Previews
            - 1 App Deploy / day
        - Pro
            - 15/month

                - 500 prompt credits/month
                - SWE-1 model
                - Add-on credits at $10/250 credits
                - 5 app deploys / day
            - Teams
                - 30 per user per month
                - 500 prompt credits for each user per month
                - Add on credits at $40/1000 credits
                - Windsurf reviews
                - Centralized billing
                - Admin dashboard with analytics
                - Priority support
                - Automated zero data retention
                - SSO available for +$10/user/month (coming soon)
            - Enterprise
                - Starting at 60 per user per month
                - Each member gets 1000 prompt credits per month
                - Add on credits at $40/1000 credits
                - Role-Based Access Control (RBAC)
                - SSO + Access control features
    - Personal Test
        - Had the AI make some minor changes to the frontend design on one of my side projects
        - Tried running in the microsoft window, it did not work
        - I ran the changed code in VS Code and it did work
        - Design (for the one part) looks much cleaner, a little bit more editing will make it look nice.
        - Used the AI to then commit my changes and push it to my repository on my personal Github account
- Continue
    - Open VS Code Extension
    - Features
        - Chat
            - Makes it easy to ask for help from an LLM without actually leaving the IDE
            - Send a task, include relevant information, and the ai will reply with text/code most likely to complete task. Keep doing this until task is completed
            - Can highlight portions of code to send to LLM
            - Reference context with the @ symbol, you can include info from codebase, documentation, IDE or other tools you want to include as context
        - Autocomplete
            - Provides inline code suggestions as you type

        - Edit
            - Convenient way to make quick changes to specific code and files
            - Select code, describe code changes, and a diff will be streamed inline to your file which can be accepted/rejected
            - Recommened for small, targeted changes like writing comments, generating unit tests, or refactoring functions/methods
        - Agent
            - Agent equips chat model with the tools needed to handle a wide range of coding tasks, allowin the model to make decisions and save you the work of manually finding context and performing actions
            - Agent lives within same interface as chat, so the same input is used to send messages and you can still use same manual methods of providing context (such as @ context providers)
- Sourcegraph Cody
- Research from Brian's Documents
    - LangChain
        - Open-source framework designed to simplify creation of applications using Large Language Models
        - Provides standard interface for chains, many integrations with other tools, and end-to-end chains for common applications
        - Allows AI developers to develop applications based on the combined Large Language Models (such as GPT-4) with external sources of computation and data
        - Framework comes with a package for both Python and JavaScript
        - Helps manage complex workflows, making it easier to integrate LLMs into various applications like chatbots and document analysis
            - Modular Workflow: Simplifies chaining LLMs together for reusable and efficient workflows
            - Prompt management: Offers tools for effective prompt engineering and memory handling
            - Ease of integration: Streamlines the process of building LLM-powered applications
        - Key Components
            - Chains
                - Define sequences of actions, where each step can involve querying an LLM, manipulating data, or interacting with external tools
                - Simple Chains: Single LLM invocation
                - Multi-step Chains: Multiple LLMs or actions combined, where each step can take output from previous step
            - Prompt management
                - LangChain facilitates managing and customizing prompts passed to the LLM

    - Developers can use PromptTemplates to define how inputs and outputs are formatted before being passed to the model. Also simplifies tasks like handling dynamic variables and prompt engineering, making it easier to control LLM's behavior
- Agents
    - Agents are autonomous systems within LangChain that take actions based on input data
    - Call external APIs or query databases dynamically, making decisions based on the situation
    - Agents leverage LLMs for decision-making, allowing them to respond intelligently to changing input
- Vector Database
    - Integrated with vector database, used to store high-dimensional vector representations of data
    - Helps perform similarity searches, where LLM converts a query into a vector and compares it against vectors in the database to retrieve relevant information
- Models
    - LangChain is model-agnostic, meaning it can integrate with different LLMs, such as OpenAI's GPT, Hugging Face Models, DeepSekk R1, and more.
    - Allows developers to choose best model for their use case while benefitting from LangChain's architecture
- Memory Management
    - Supports Memory management, allowing LLM to "remember" context from previous interactions
    - Useful for creatin conversational agents that need context across multiple inputs
    - Memory allows model to handle sequential conversations, keeping track of prior exchanges to ensure system responds appropriately
-