zenitech

STEVENS&BOLTON

New Beings

1. AutoGPT

Christopher Lacy-Hulbert
Founder and CTO at Zenitech
Cristian Tintas
Software Developer at Zenitech

2. Copilot

Cristian Tintas
Software Developer at Zenitech

3. SuperAGI

Cristian Tintas
Software Developer at Zenitec

4. Tabnine

Cristian Tintas
Software Developer at Zenitec

A series of articles on AI vol.2

New Beings: AI in Software Development

Zenitech has partnered with law firm, Stevens & Bolton, to create **New Beings** - a series of articles exploring how AI can, and will, be used in software development.

Join us for the journey, as we examine a range of issues, including the legal issues surrounding artificial intelligence, the practicalities of using AI to aid software development, how AI tools can assist with coding, and the issues around security and testing.

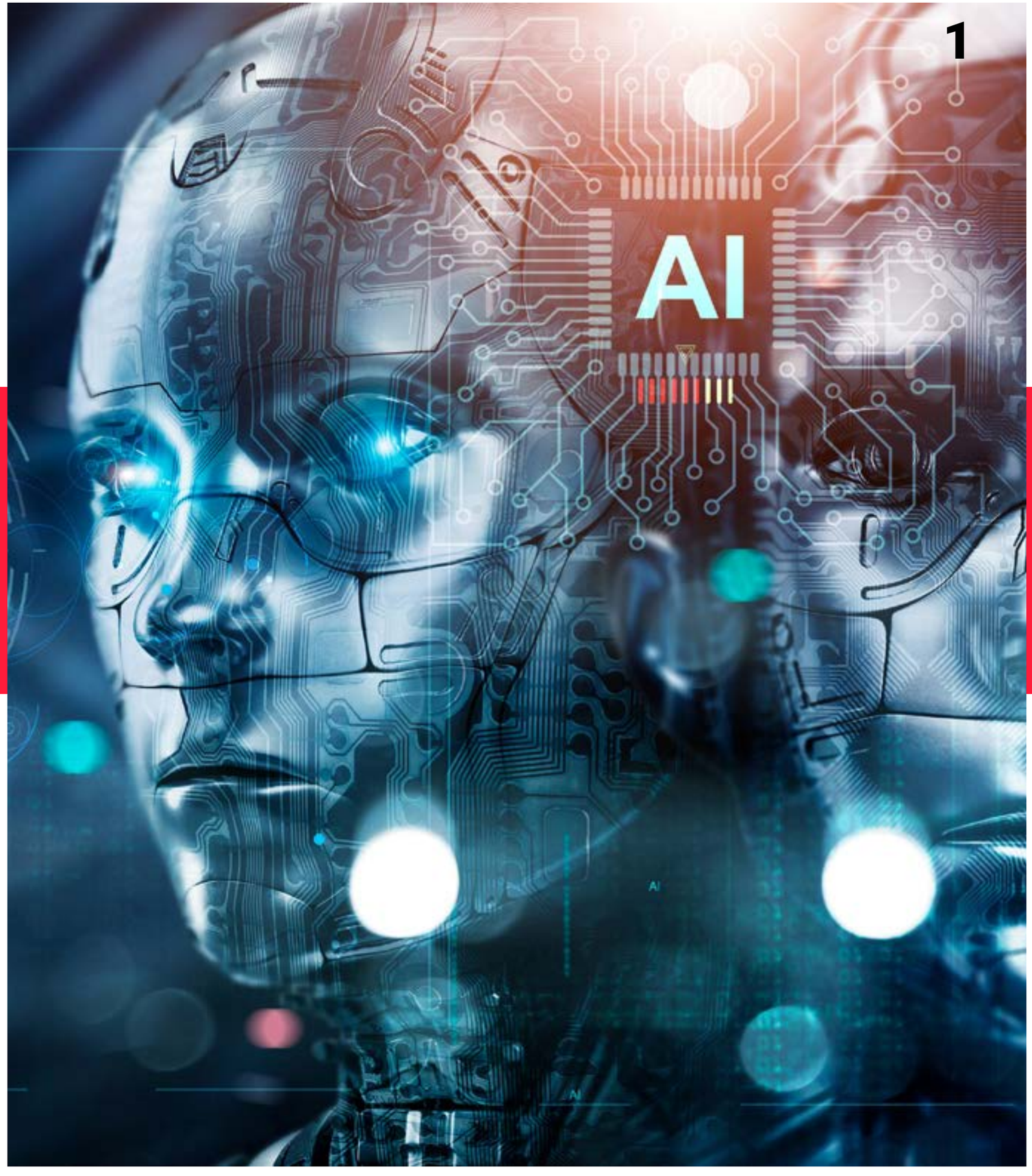
CONTENTS

AutoGPT

Christopher Lacy-Hulbert
Founder and CTO at Zenitech
Cristian Tintas
Software Developer at Zenitech

What is AutoGPT?

AutoGPT is an open-source AI tool that leverages the GPT-4 or GPT-3.5 APIs from OpenAI to accomplish user-defined objectives expressed in natural language. It allows users to define tasks by breaking them down into smaller components and autonomously utilises various resources in a cyclic process to achieve those objectives



1

Features and resources

According to the AutoGPT web site, Auto-GPT's unique strength lies in its self-sufficiency. By utilising the capabilities of ChatGPT, it can take control of tasks and projects, removing the need for continuous user inputs. In doing so, Auto-GPT removes workload by eliminating the need to constantly provide follow-up ideas or responses.

AutoGPT has the following key features:

- Internet access for searches and information gathering
- Long-term and short-term memory management
- GPT-4 instances for text generation
- Access to popular websites and platforms
- File storage and summarisation with GPT-3.5
- Extensibility with Plugins [GitHub - Plugins]

To install AutoGPT, you can clone the GitHub repository from the following link: GitHub. Alternatively, you can download the repository as a zip file. The installation guide and further setup instructions can be found in the repository [GitHub - AutoGPT Guide]. There are also Docker images available, which allow quick set-up of AutoGPT based on existing builds with all dependencies installed and running.

There is also an implementation available with LangChain primitives (LLMs,VectorStores, Embeddings, Tools).

Setup

After cloning the repository, you need to update the ".env" file with your own configurations and API keys. The required keys include the **OpenAI API** key and the GITHUB_API_KEY for authorisation. Additionally, you need to specify your GITHUB_USERNAME and configure other APIs as needed. The installation guide provides detailed instructions on setting up AutoGPT [GitHub - AutoGPT Guide].

In principle, AutoGPT can be used for various coding tasks, such as executing code, analysing code, improving code, generating documentation, debugging, searching Google, web scraping, searching files, and GitHub cloning. It offers a wide range of features and functionalities to assist in coding and debugging processes [GitHub - AutoGPT Guide][GitHub - Plugins]. However, in our experiments, it was very limited and struggled to handle even moderately complex tasks. For very simple use-cases, working on a single file with <50 lines of Python, it was able to autonomously provide bug fixes, documentation and reformatting. Anything more involved than that will send the automaton into infinite loops with no meaningful actions performed.

Concerns about AutoGPT

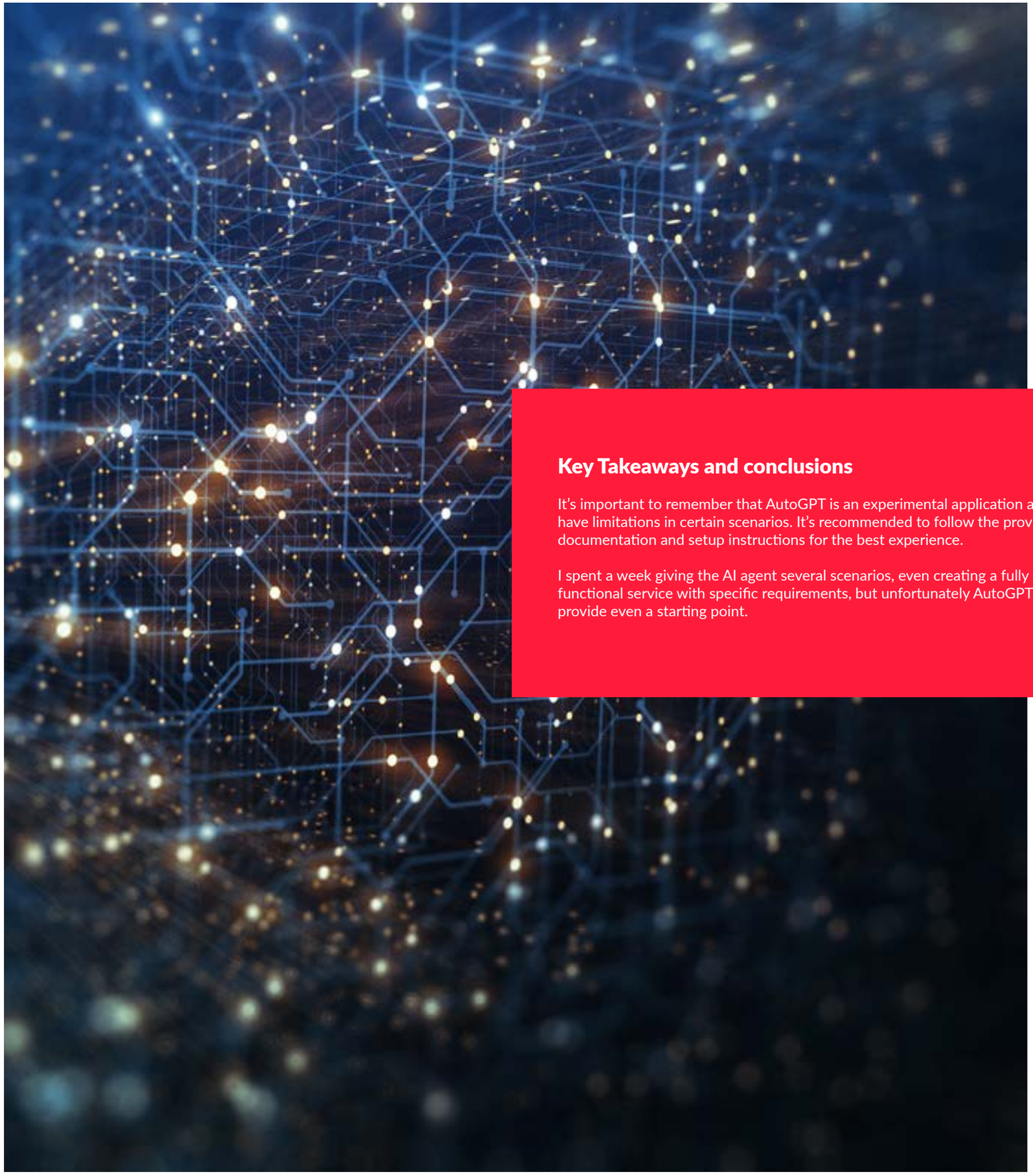
Some potential concerns with AutoGPT include its experimental nature, which means it may not be a polished application or product. It might not perform well in complex, real-world business scenarios, and it can be quite expensive to run, so monitoring API key limits with OpenAI is important [GitHub - Plugins].

We will be looking at the legal position around AutoGPT in New beings: Legal issues and AI.

Alternatives to AutoGPT

There are several alternatives to AutoGPT available in the market, such as other prompt engineering tools or AI-based coding assistants, some similar are **BabyAGI** [GitHub - BabyAGI] and SuperAGI [GitHub - SuperAGI].

Each alternative may have its own unique features, capabilities, and user experiences, so it's important to explore different options and choose the one that best fits your requirements.



Key Takeaways and conclusions

It's important to remember that AutoGPT is an experimental application and may have limitations in certain scenarios. It's recommended to follow the provided documentation and setup instructions for the best experience.

I spent a week giving the AI agent several scenarios, even creating a fully functional service with specific requirements, but unfortunately AutoGPT did not provide even a starting point.

2

Copilot

Cristian Tintas
Software Developer at Zenitec

What is Copilot?

Copilot is an AI pair programmer developed by GitHub that helps developers write code faster and with less effort. It uses OpenAI's Codex, a generative pre-trained language model, to provide context-aware code suggestions and completions based on comments and existing code. Copilot is available as an extension for various integrated development environments (IDEs), including Visual Studio Code, Visual Studio, Neovim, and the JetBrains suite of IDEs [GitHub - Copilot].

The whole point of GitHub Copilot is having an AI assistant suggest to you what to write. There is the upcoming Copilot X that will have integrated ChatGPT functionalities and much more, but the current version is "just" a helper.

The name Copilot is exactly about this. In "pair programming", there are two people working on the same computer: the pilot, the developer actually having the responsibility of writing the code; and the copilot, there to suggest, advise, and check that everything is being developed correctly.
While you are typing, Copilot keeps sending your code to GitHub servers, and when it has a suggestion, it provides it to you by writing it in light grey. If you like what it is proposing, you can press TAB to approve it, or you can keep writing to ignore it, waiting for the next suggestion.

Features and resources

Copilot has the following key features:

- **Code Suggestions:** Copilot suggests individual lines and whole functions in real-time, drawing context from comments and code.
- **Context-Awareness:** The AI model analyses the code you are writing and provides relevant suggestions based on the current context.
- **Multiple Language Support:** Copilot supports various programming languages, including C#, C++, Python, and more [Microsoft - GitHub Copilot extension for Visual Studio].

It also has a GitHub Extension.

Copilot is available as an extension for Visual Studio Code, Visual Studio, Neovim, and the JetBrains suite of IDEs. Developers can install the extension from the respective marketplace or plugin repository for their preferred IDE.

Setup

To use Copilot, developers need to install the Copilot extension for their chosen IDE. The installation process may vary depending on the IDE being used. Once installed, Copilot integrates with the IDE and provides code suggestions and completions as developers write code [Microsoft - GitHub Copilot extension for Visual Studio].

Real-Life Action

In real-life scenarios, developers can leverage Copilot to speed up their coding process and reduce the time spent on repetitive tasks. Copilot analyses the code being written and suggests relevant completions, snippets, or even entire functions. It provides assistance throughout the development process, helping developers write code more efficiently and accurately.

Sometimes, it happens, and when Copilot suggests like the next 10 lines of code and they are all correct, it feels like magic.

Concerns about Copilot

While Copilot can be a valuable tool for developers, there are a few concerns to consider.

Copilot's suggestions are not perfect and may require careful review and testing. It doesn't guarantee flawless code and may generate code that doesn't work or make sense in certain situations.

Copilot also has limited context and may not utilise helpful functions defined elsewhere in a project or even within the same file. It may also suggest old or deprecated usage of libraries and languages.

We will be looking at the legal position around Copilot in XX

Alternatives to Copilot

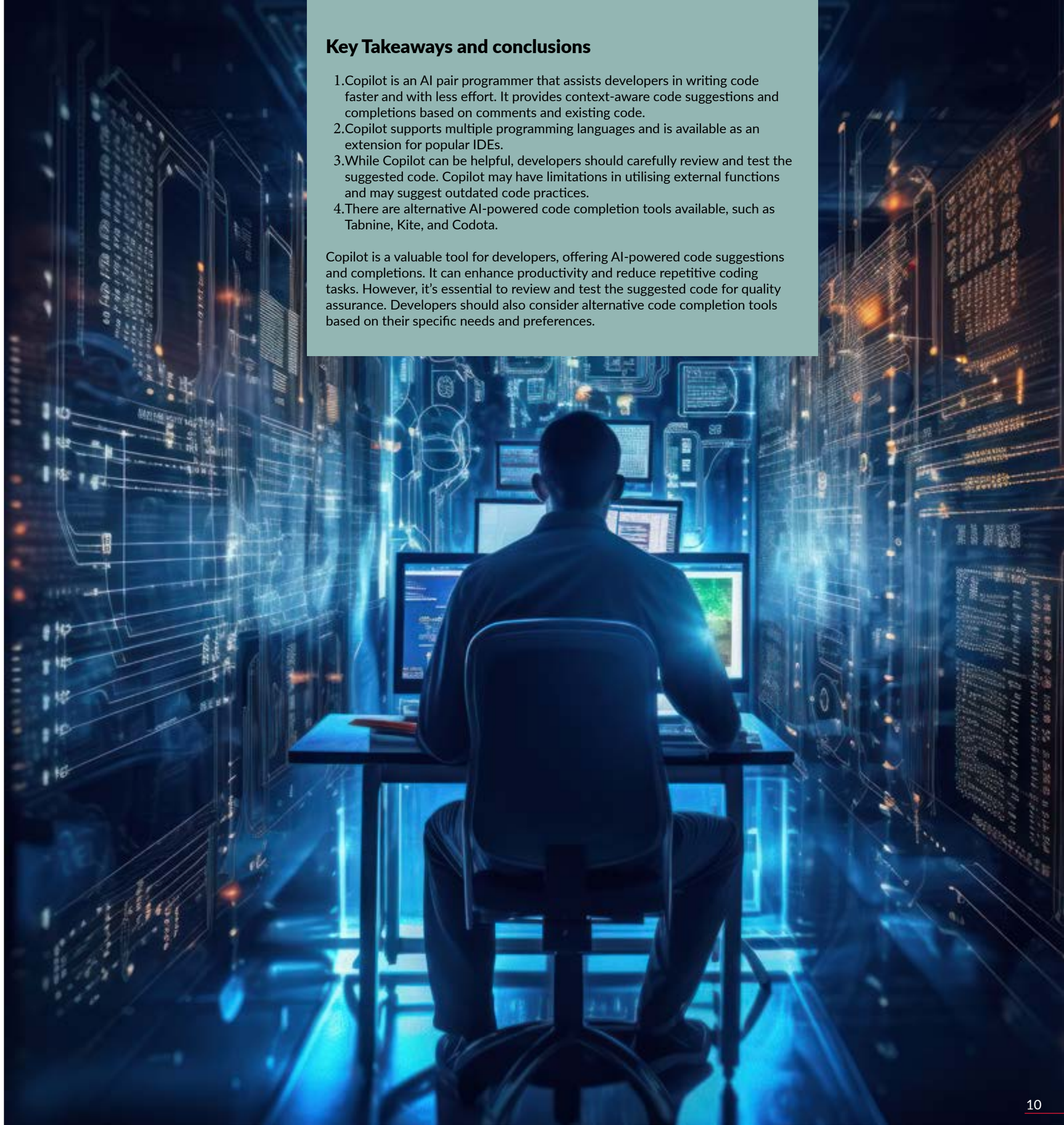
While Copilot is a popular AI pair programmer, there are alternatives available in the market. Some notable alternatives include:

- **Tabnine:** An AI-powered code completion tool that provides intelligent suggestions for multiple programming languages.
- **Kite:** A code completion tool that uses machine learning models to offer contextually relevant code suggestions.
- **Codota:** A code completion tool that leverages machine learning to provide code recommendations based on millions of open-source code examples.

Key Takeaways and conclusions

1. Copilot is an AI pair programmer that assists developers in writing code faster and with less effort. It provides context-aware code suggestions and completions based on comments and existing code.
2. Copilot supports multiple programming languages and is available as an extension for popular IDEs.
3. While Copilot can be helpful, developers should carefully review and test the suggested code. Copilot may have limitations in utilising external functions and may suggest outdated code practices.
4. There are alternative AI-powered code completion tools available, such as Tabnine, Kite, and Codota.

Copilot is a valuable tool for developers, offering AI-powered code suggestions and completions. It can enhance productivity and reduce repetitive coding tasks. However, it's essential to review and test the suggested code for quality assurance. Developers should also consider alternative code completion tools based on their specific needs and preferences.



SuperAGI

Cristian Tintas
Software Developer at Zenitec

What is SuperAGI?

SuperAGI is an open-source platform that provides infrastructure for building autonomous AI agents. It allows developers to spawn, deploy, and manage AI agents with ease.

With SuperAGI, developers can run multiple agents concurrently, extend agent capabilities using various tools, and benefit from a user-friendly graphical interface for agent management.

Features and resources

SuperAGI has the following key features:

- Provision, spawn, and deploy autonomous AI agents.
- Extend agent capabilities with a library of tools.
- Run concurrent agents seamlessly.
- Graphical User Interface (GUI) for easy agent management.
- Action Console for interacting with agents.
- Multi-Modal Agents for customisation using different models.
- Performance telemetry and memory storage for agents.
- Looping detection heuristics and resource manager [GitHub - SuperAGI].

To download the SuperAGI repository, you can clone it using the command git clone https://github.com/TransformerOptimus/SuperAGI.git in your terminal or download it directly from the GitHub page in zip format.

The setup instructions and configuration details can be found in the repository's README file. You will need to create a config.yaml file and provide your unique OpenAI API Key, Google key, and Custom search engine ID to access the required APIs [GitHub - SuperAGI].

Setup

To set up SuperAGI, follow these steps:

- Clone the SuperAGI repository or download it as a zip file.
- Navigate to the SuperAGI directory using the command cd SuperAGI.
- Create a copy of config_template.yaml and name it config.yaml.
- Enter your unique OpenAI API Key, Google key, and Custom search engine ID in the config.yaml file without any quotes or spaces.
- Ensure Docker is installed on your system.
- Run the command docker-compose up --build in the SuperAGI directory.
- Open your browser and go to http://localhost:3000 to access SuperAGI.
- Note: You can change the port it's running on by modifying the docker-compose.yml file [GitHub - SuperAGI].

Real-Life Action

SuperAGI enables developers to build, manage, and run autonomous AI agents. It facilitates tasks such as provisioning agents, extending their capabilities using various tools, running agents concurrently, and providing a graphical interface for agent management. Developers can use SuperAGI to create AI agents tailored to specific tasks and interact with them through the Action Consol.

However, I did not manage to use it to its claimed potential. It ran for more than half a day and was still in its original state.

Alternatives to SuperAGI

There are various alternatives to SuperAGI available for building autonomous AI agents, such as OpenAI's GPT-3 platform, Microsoft's Azure Cognitive Services, and Google Cloud AI. Each alternative offers its own set of features, tools, and capabilities, so it's important to evaluate them based on specific requirements and use cases.

Key Takeaways and conclusions

1. SuperAGI appears to be a powerful open-source platform for building and managing autonomous AI agents. Its features, tools, and concurrent agent capabilities make it a promising solution for developers working with AI agent systems.
2. SuperAGI provides developers with a comprehensive framework for building and managing autonomous AI agents.
3. With its features like concurrent agent support, extendable capabilities, and a user-friendly GUI, SuperAGI simplifies the development and deployment process.

However, it's important to keep in mind that SuperAGI is under active development, and as with any software project, it may have some issues or limitations.

Developers should review the documentation, contribute to the community, and consider their specific requirements when deciding whether to use SuperAGI for their AI agent projects.

Tabnine

Cristian Tintas
Software Developer at Zenitec

What is Tabnine?

Tabnine is an AI assistant for software developers that provides AI-powered code completions and suggestions to enhance productivity and accelerate coding workflows. It uses advanced machine learning models trained on open-source code with permissive licenses to offer intelligent code suggestions across various programming languages and major integrated development environments (IDEs) [Tabnine Get Started].

Features and resources

Tabnine has the following key features:

- AI Completions: Tabnine offers AI-driven code completions that assist developers in writing code faster and more accurately.
- Language and IDE Support: Tabnine supports multiple programming languages, including JavaScript, Java, Python, TypeScript, PHP, C++, Go, Rust, and more. It is compatible with popular IDEs like Visual Studio Code and WebStorm [Tabnine Get Started].
- Privacy and Security: Tabnine ensures the privacy and security of developers' code. It never stores, or shares any user code, and actions that involve sharing code with Tabnine servers require explicit opt-in. Tabnine's generative AI only uses open-source code with permissive licenses for training models [Tabnine].

It provides resources, such as a trust centre, which gives users information about the privacy and security practices employed by Tabnine, ensuring developers that their code remains private and protected [Tabnine].

Tabnine also offers comprehensive documentation and resources to help developers get started with using Tabnine effectively in their coding workflows.

Setup

Developers can install the Tabnine extension for their preferred integrated development environment (IDE). The installation process may vary depending on the IDE being used.

Once installed, Tabnine integrates with the IDE and provides AI-powered code completions as developers write code [Tabnine Get Started].

Real-Life Action

In real-life scenarios, Tabnine acts as an AI assistant for developers, providing intelligent code completions and suggestions. It analyses the code being written and offers relevant suggestions, helping developers write code faster and reduce manual effort.

When used in combination with Copilot it can really enhance developer productivity and streamline the coding process.

Concerns about Tabnine

Although Tabnine strives to provide accurate code suggestions, the quality and relevance of suggestions may vary depending on the context and codebase. Developers should review and test the suggested code for their specific use cases.

We will be looking at the legal position around Tabnine in New beings: Legal issues and AI.

Alternatives to Tabnine

While Tabnine is a popular AI code completion tool, there are alternative options available in the market. Some notable alternatives include:

- Kite: An AI-powered code completion tool that offers contextually relevant code suggestions based on machine learning models.
- Codota: A code completion tool that uses machine learning to provide intelligent code recommendations based on millions of open-source code examples.
- DeepCode: A code review tool that uses AI to analyse code and provides suggestions for code improvements and bug fixes.

Key Takeaways and conclusions

1. Tabnine is an AI assistant that provides developers with AI-powered code completions and suggestions. It supports multiple programming languages and integrates with popular IDEs.
2. Tabnine prioritises privacy and security, ensuring that user code remains private and is not used to train models other than private code models.
3. Developers should consider the accuracy and relevance of suggestions when using Tabnine and review and test the suggested code accordingly.

Tabnine is a valuable AI assistant for developers, offering AI-powered code completions and suggestions to enhance productivity. It supports various programming languages and major IDEs, allowing developers to benefit from its intelligent code suggestions. However, developers should be cautious of taking Tabnine's suggestions, and test their accuracy and relevance