

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technology



Bachelor Thesis

**Evaluation of AI Technologies (Code Generation) for
Application Development**

Alibek Toleukhanov

© 2024 CZU Prague

BACHELOR THESIS ASSIGNMENT

Alibek Toleukhanov

Informatics

Thesis title

Evaluation of AI technologies (code generation) for application development

Objectives of thesis

Main Objective: To evaluate the effectiveness of AI technologies for code generation in application development.

Partial Objectives:

- To analyze the current state of AI technologies for code generation in application development.
- To compare the performance of AI-generated code with manually written code in terms of quality and efficiency.
- To assess the impact of using AI technologies for code generation on the overall productivity and cost-effectiveness of application development projects.

Methodology

The methodology for evaluating the effectiveness of AI technologies for code generation in application development involves conducting a comprehensive literature review to understand the current state of AI technologies used for code generation.

The practical part involves evaluation of three identical simple applications in Flutter app dev., one using AI-generated code, one using author-developed code, and the last one using code from a professional project/developer.

In the end, comparison will be done on their quality, efficiency, and impact on application development. The research will follow scientific methods such as comparison, analysis, and deduction. Findings from literature, development, and testing will be described, evaluated, and will contribute to app. development with the use of AI.

The proposed extent of the thesis

40 –50pages(from Introduction to Conclusion)

Keywords

AI,application development,Flutter,code generation

Recommendedinformaonsources

Hamilton, A.C., 2023. Introduction to Chat GPT: Supercharge Your Productivity With AI – Create Apps, Websites, Google Chrome Extensions & Much More (Artificial Intelligence Uses & Applications). ISBN 9798889551669

raywenderlich tutorial team; M.Katz, K.Moore, V.Ngo, V.Guzzi, 2021. Flutter Apprentice (Second Edition): Learn to Build Cross-Platform Apps. ISBN 9781950325481.

S.Russell, P.Norvig, 2016. Artificial Intelligence: A Modern Approach 3rd Edition. ISBN 9781292153964.

Thomas Bailey, Alessandro Biessek, 2021. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart, 2nd Edition ISBN 9781800565999.

Expected date of thesis defence

2023/24 SS – PEF

The Bachelor Thesis

Supervisor Ing. Jan

Masner, Ph.D.

Supervising

department

Department of Information Technologies

Electronic approval: 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 10. 03. 2024

Declaration

I declare that I have worked on my bachelor thesis titled "Evaluation of AI Technologies (Code Generation) for Application Development" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 15.03.2024

Acknowledgement

I would like to thank Ing. Jan Masner, Ph.D for his advice and support during my work on this thesis.

Title of Bachelor Thesis in English

Abstract

This thesis embarks on a comprehensive exploration of the integration of artificial intelligence (AI) in code generation for application development, with a particular emphasis on the Flutter framework. It begins by laying the foundation with an introduction and a detailed outline of objectives and methodology. The study then dives into an extensive literature review, starting from the basics of AI, including its history, machine learning principles, various types, and practical use cases. Then I examine application development, categorization, development processes, and the rise of cross-platform frameworks, with a special focus on Flutter, including its history and features.

The core of the thesis is an in-depth look at how AI influences code generation in application development. It evaluates the beneficial impacts of AI, addresses the potential negative aspects, and explains the mechanisms behind AI-driven code generation. A list of current tools in this domain is also presented. The practical part of the study involves hands-on application development using Flutter, where AI-generated code is compared against traditional coding methods. This practical experimentation aims to provide a clear understanding of AI's role and efficacy in modern app development. This research aims to bridge the gap between theoretical knowledge and practical application, providing a comprehensive overview of AI's transformative role in application development.

Keywords: AI, Application Development, Flutter, Code Generation.

Hodnocení technologií AI (generování kódu) pro vývoj aplikací

Abstrakt

Tato práce se pouští do komplexního zkoumání integrace umělé inteligence (AI) při generování kódu pro vývoj aplikací, se zvláštním důrazem na framework Flutter. Začíná položením základů úvodem a podrobným nastíněním cílů a metodologie. Studie se poté ponoří do rozsáhlého přehledu literatury, počínaje základy umělé inteligence, včetně její historie, principů strojového učení, různých typů a praktických případů použití. Poté zkoumám vývoj aplikací, kategorizaci, vývojové procesy a vzestup multiplatformních frameworků se zvláštním zaměřením na Flutter, včetně jeho historie a funkcí.

Jádrem práce je hloubkový pohled na to, jak AI ovlivňuje generování kódu při vývoji aplikací. Hodnotí příznivé dopady umělé inteligence, zabývá se potenciálními negativními aspekty a vysvětluje mechanismy generování kódu řízeného umělou inteligencí. Je také uveden seznam aktuálních nástrojů v této oblasti. Praktická část studie zahrnuje praktický vývoj aplikací pomocí Flutter, kde je kód generovaný AI porovnáván s tradičními metodami kódování. Toto praktické experimentování má za cíl poskytnout jasné pochopení role a účinnosti umělé inteligence při vývoji moderních aplikací. Tento výzkum si klade za cíl překlenout propast mezi teoretickými znalostmi a praktickou aplikací a poskytnout komplexní přehled o transformační roli umělé inteligence při vývoji aplikací.

Klíčová slova: AI, vývoj aplikací, flutter, generování kódu.

Table of content

1	Introduction	11
2	Objectives and Methodology	12
2.1	Objectives	12
2.2	Methodology	12
3	Literature Review	13
3.1	Application	13
3.1.1	Categorization	13
3.1.2	Development Process	14
3.1.3	Cross-Platform Development	14
3.1.4	Flutter	16
3.1.4.1	History of Flutter	16
3.1.4.2	Features	17
3.2	Artificial Intelligence	17
3.2.1	History	17
3.2.2	Machine Learning	18
3.2.3	Types	18
3.2.4	Use Cases of AI	19
3.2.5	Neural Network Language Model	20
3.2.5.1	Large Language models	20
3.3	Code generation and AI	21
3.3.1	AI and Application Development	21
3.3.2	Advantages and Disadvantages of code generation with AI	21
3.3.3	How code generation works with AI	22
3.3.4	Code generation tool list	22
3.3.5	The importance of AI generated code	23
4	Practical Part	25
4.1	Flutter SDK setup	25
4.1.1	Libraries	25
4.2	Professionally developed project	26
4.2.1	Home page	27
4.2.2	Create Reminder page	27
4.2.3	Reminder Details page	28
4.3	Author developed application	29
4.4	Chat-GPT generated application	31
4.5	Evaluation	32

4.5.1	Code Quality.....	32
4.5.2	App Performance.....	33
4.5.3	Development Efficiency	34
5	Results and Discussion	36
5.1	Code quality results	36
5.2	Application performance results	37
5.3	Development efficiency results	38
5.4	Final points results	39
6	Conclusion	40
7	References.....	41
8	List of pictures, tables, graphs and abbreviations	43
8.1	List of pictures	43
8.2	List of tables.....	43
8.3	List of abbreviations	43
9	Appendix.....	44

1 Introduction

In today's digital age, application development plays a crucial role in the world of IT, empowering companies, and individual users to create new and innovative solutions for various purposes. The rapid growth of AI technologies in any sphere is undeniable, including areas such as application development, where the concept of creating personalized source code effortlessly seemed akin to science fiction until recent years. The future of application development is probably going in the perfect direction people would say. However, even despite the huge potential of AI in application development, is it possible that there are flaws or is the code generation technology ideal? Hence, it is of utmost importance to unveil the true nature of both technologies through accurate definitions and how they interact with each other.

The content of this thesis begins with a comprehensive theoretical overview touching upon the topics of artificial intelligence and application development, as well as related topics such as code generation and development tools. This theory review is intended to provide an overview of the current state of the art, including comparison methods, which contributes to a deeper understanding of the topic at hand.

2 Objectives and Methodology

2.1 Objectives

Main Objective: To evaluate the effectiveness of AI technologies for code generation in application development.

Partial Objectives:

- To analyse the current state of AI technologies for code generation in application development.
- To compare the performance of AI-generated code with manually written code in terms of quality and efficiency.
- To assess the impact of using AI technologies for code generation on the overall productivity and cost-effectiveness of application development projects.

2.2 Methodology

The methodology for evaluating the effectiveness of AI technologies for code generation in application development involves conducting a comprehensive literature review to understand the current state of AI technologies used for code generation.

The practical part involves evaluation of three identical simple applications in Flutter app dev., one using AI-generated code, one using author-developed code, and the last one using code from a professional project/developer. In the end, comparison will be done on their quality, efficiency, and impact on application development.

The research will follow scientific methods such as comparison, analysis, and deduction. Findings from literature, development, and testing will be described, evaluated, and will contribute to app. development with the use of AI.

3 Literature Review

Before proceeding with the development of applications using the previously mentioned code generation tools, it is crucial to understand the theoretical foundations that are essential for gaining a better understanding of AI, Application tools, etc.

3.1 Application

An application, often called an "app," is software designed for tasks on computers and devices. They cover needs like productivity, entertainment, and communication. It performs specific functions for users or other apps, either standalone or as a group of programs.

The application industry has grown exponentially in the last decade, evolving into a multi-billion-dollar market. According to Statista, in 2023, the current number of smartphone users in the world today is 6.92 billion, meaning 85.95% of the world's population owns a smartphone[16]. This growth is driven by a combination of factors, including increasing smartphone penetration, improved internet connectivity, and the constant demand for new and innovative solutions. App stores, such as the Apple App Store and Google Play Store, serve as platforms for developers to showcase their creations to a massive global audience. The industry has proven to be fertile ground for startups and established companies alike, offering opportunities for revenue generation through in-app purchases, subscriptions, advertisements, and premium version.

3.1.1 Categorization

Applications are diverse due to the interplay between evolving user needs and continuous technological advancements. Sometimes individuals might have many applications, and to categorize this diversity the application categories were done. They are needed to provide a way to quickly identify characteristics of an application[22]:

- **Educational Apps:**

These apps facilitate learning with courses, quizzes, and interactive lessons, catering to both formal and informal educational needs.

- **Lifestyle Apps:**

Lifestyle apps enhance personal interests and routines, covering fitness, meditation, fashion, and more, helping users manage and enrich their daily lives.

- **Entertainment Applications:**

Entertainment apps provide leisure content like streaming services, games, and virtual experiences, offering users enjoyable ways to relax and have fun.

- **Productivity Applications:**

Productivity apps optimize efficiency with tools for tasks, communication, and organization, assisting users in managing work and responsibilities effectively.

- **Social Media Applications:**

Social media apps enable online interactions, connecting users for sharing content, updates, and discussions, fostering connections and information exchange.

- **Game Applications:**

Game apps offer interactive experiences across genres, from puzzles to action, providing users with immersive entertainment and skill-building opportunities.

3.1.2 Development Process

Mobile applications come in three main types: native, web, and hybrid. Native apps are optimized for specific platforms, offering excellent performance and device feature access but require separate development for each platform. Native apps are developed for operating systems (OS), such as iOS or Android, using the platform's programming languages (Swift/Objective-C for iOS, Java/Kotlin for Android). Native apps use APIs (Application Programming Interfaces), which act as a connection between applications and external systems, enabling developers to utilize pre-existing tools and services without the need to create everything anew.

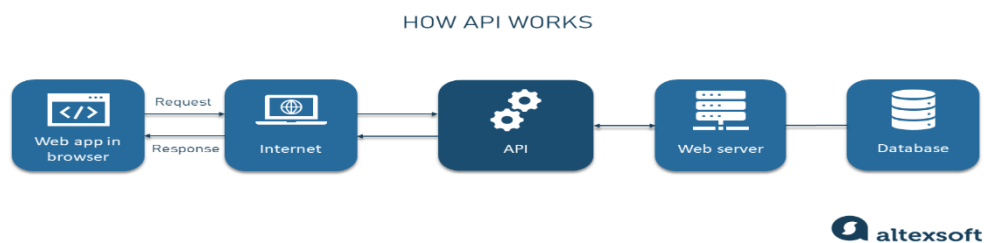


Figure 1: How API's Work **Source:** [1]

Web apps run in mobile browsers, are easily accessible without installation, but might have limited device feature access. Web apps are built using web technologies such as HTML, CSS, and JavaScript. Web mobile apps utilize only a few basic native functions, and developers won't have a ready-made toolkit (SDK-software development kit) to use. Web apps usually have more user traffic which can play a big role in companies, individuals, businesses revenue. Web apps are the reason and the big part of why Amazon is now the third largest public company in the world by revenue, generating \$470 billion in 2021 [18].

Hybrid apps combine native and web app elements, using a native app container to wrap a web app. They leverage native features and device hardware, offering a native-like experience across platforms. These apps are built with front-end technologies like JavaScript, HTML5, Ionic, Cordova, and CSS. They grant access to device APIs, making distribution easy, and updates straightforward. This approach is favored by many companies due to its cost-effectiveness and cross-platform capabilities. Hybrid apps merge web technology with native containers, finding a balance between compatibility and utilizing device features.

3.1.3 Cross-Platform Development

Cross-platform development involves creating software applications that can run on multiple operating systems or platforms with minimal adjustments. When utilizing a cross-platform application, a portion or the entirety of the source code can be distributed across various platforms, like Android and iOS. This approach eliminates the necessity for developers to create mobile assets multiple times. These assets function smoothly on all platforms, eradicating the need for re-coding tailored to each platform.



Figure 2: Cross-Platform App Development **Source:** [5]

In 2019, Shopify employed React Native (framework used in cross-platform development) across all their buyer and merchant apps and at first they estimated 80% code sharing between iOS and Android, and were surprised by the extremely high-levels in practice - 95% (Arrive) and 99% (Compass)[9]. This instance demonstrates how cross-platform development can improve efficiency, promote code reusability, and enhance overall application performance across various platforms.

Mobile applications help their users a lot in terms of efficiency, ease of use, and opportunities. In 2023, there are many available cross platform development tools available for developers to work with. Here is the list of the most popular tools:

- **React Native**

As we already discussed React Native previously, it is one of the most popular development tools for cross platform development. It is an open-source framework which based on React. React Native has specific features That include:

- **Native module:**

React Native allows you to access native platform features like camera, location, or push notifications by writing modules in a native language like Java, Swift, or Objective-C2.

- **Expo:**

Expo is a collection of tools and services that simplify the development and deployment of React Native applications. Expo provides CLI, web-based IDE and client that allows you to run apps on your device without needing to install SDK or emulator2.

- **Hooks:**

Hooks are a new feature in React that allows you to use state and other React features without writing class components. Square brackets make your code more readable and reusable by allowing you to extract logic into custom functions.

- **Ionic**

Ionic is a commonly used platform for creating mobile apps. It came out for the first time in 2013. Apps made with this framework can work on both iOS and Android using the same code.

Ionic allows you to create fast, high-quality apps that work well on different operating systems. Ionic uses Capacitor, a native runtime environment that provides access to native APIs and features.

- **Beautiful design:**

Ionic provides a set of user interface components designed to look great on any platform. Ionic also supports light and dark modes, adaptive UI, and native-like gestures¹³.

- **Standalone framework:**

Ionic works with any JavaScript framework or library, like Angular, React, Vue, or vanilla JS. Ionic also provides integrations and starter templates for these frameworks to make development easier.

- **Xamarin:**

Xamarin is a framework that allows you to build native apps for Android, iOS, macOS, and Windows using C# and .NET³. Xamarin specific features include:

- **Xamarin. Forms:**

Xamarin. Forms is a user interface toolkit that allows you to share your UI code across multiple platforms. Xamarin. Forms provides a set of common controls that map to native controls at runtime³.

- **Xamarin. Essentials:**

Xamarin. Essentials is a library that provides a cross-platform API to access device features such as battery, connectivity, geolocation, or preferences³.

- **Visual studio:**

Visual Studio is an integrated development environment (IDE) that powers Xamarin. Visual Studio provides features like code editing, debugging, testing, publishing, and more.

3.1.4 Flutter

Flutter is that software development framework that will be used in this paper. It is needed to focus on what Flutter is, why it's considered as a cross-platform development tools, its history, and features. Flutter is a versatile UI toolkit for building apps on iOS, Android, web, and desktop platforms like macOS, Windows, and Linux using a unified codebase. It stands out among cross-platform development tools due to its rapid rendering engine, ensuring Flutter apps perform as native ones. The framework's independence from native features is a result of its unique UI elements, called widgets, while still allowing integration with native features when required. With Flutter, you can create apps across various devices and platforms seamlessly using just one codebase. Flutter is supported and developed by a big tech company Google.

3.1.4.1 History of Flutter

In October 2014, Google's dev team heard the call to "Open the sky" and so Sky Engine was born^[15]. Over time, it evolved into what we know today as Flutter – an open-source user interface development kit that took the mobile app development world by storm. The tools were evolving exponentially each year in terms of performance and user experience. In 2018, Flutter 1.0 was

released. In 2021, Google announced the release of Flutter 2.0. And recently in 2022 Flutter 3.0 was announced.

3.1.4.2 Features

The team behind Flutter had a clear vision: to create a platform that not only looks good and works well, but also improves the overall developer experience. But they don't stop there. They have also made it a point to keep the platform open source and flexible, ensuring that developers can customize it to suit their needs. And at the heart of it all is Flutter's programming language, Dart. Dart is a client-optimized language for fast apps on any platform [11]. The programming language compiled as a native code which results in applications that are sharp and responsive. It is considered as one of the flutter's features.

Another feature that Flutter has is "hot reload". Flutter's "hot reload" feature allows developers to instantly see code changes reflected in the UI, speeding up development and enabling error correction, without requiring a complete app restart. It quickly updates the running app's UI and logic, maintaining the app's state, though more complex changes might necessitate a "hot restart" or full restart of the app.

Everything in Flutter is defined as a widget [10]. Flutter employs a widget-based framework, structuring UI elements as reusable and combinable building blocks known as widgets. Widgets are considered as another feature in Flutter. These widgets encompass a wide range of components such as colors, padding, menus, and more, forming the foundation of Flutter's UI definition. This approach enables the creation of intricate and customizable user interfaces, utilizing both built-in collections like Cupertino pack and Material Design, ensuring a seamless user experience.

3.2 Artificial Intelligence

In today's world AI focuses on designing agents-systems that perceive their environment and take actions based on the information they process. The goal is to make machines capable of tasks, like understanding language and recognizing patterns essentially replicating human intelligence. It involves developing algorithms to simulate abilities allowing machines to interact with their surroundings and adapt accordingly. According to Russell & Norvig (2016), AI is the study of how to make computers do things at which, at the moment, people are better [21].

Defining intelligence holds equal importance when discussing AI.

According to Russell & Norvig (2016) Intelligence is the capability to comprehend and adapt to one's surroundings, solve complex problems, reason logically, and learn from experiences [21].

3.2.1 History

History of AI dates to antiquity when myths and stories depicted artificial beings and intelligent automatons. The concept of placing a mind into a body has been present since ancient times, in Greece. This notion can be traced back to mythology dating back to, around 700 B.C., where Talos, a bronze automaton is depicted as the guardian of Crete. The evolution of intelligence (AI) is closely intertwined with the fields of logic and mathematics. However, the formal exploration of AI as a scientific discipline began in the mid-20th century.

The Dartmouth Conference in 1956 is often considered the birth of AI as a field of study. During this conference, John McCarthy and other pioneering researchers articulated the idea of creating "thinking machines" and laid the groundwork for AI research. Also, according to Russell & Norvig (2016), Early AI researchers were optimistic about the prospects of building artificial

general intelligence (AGI), machines that could perform any intellectual task that a human being can do [1].

Artificial Intelligence (AI) has evolved significantly over the past few decades, transforming from a theoretical concept to a practical technology that has found applications in various fields.

3.2.2 Machine Learning

Machine learning, an alluring field, operates within the framework of AI and relies on logic and mathematics for its algorithms. It involves automated extraction of patterns from data, enabling systems to learn and improve independently. Coined by Arthur Samuel in 1952, machine learning empowers computers to autonomously enhance themselves through experience. It's a pivotal subset of AI that enables computers to perform tasks previously restricted to human intelligence, evolving through historical advancements in computing, data, and innovative architectures.

3.2.3 Types

AI can be categorized into various types. There are mainly two types of main categorization which are based on capabilities and based on functionality of AI [16].

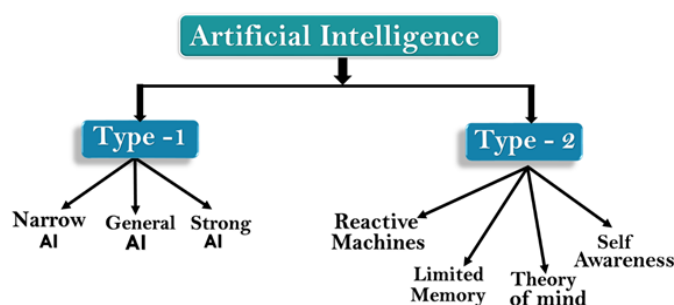


Figure 3: Types of Artificial Intelligence Source: [16]

AI type-1: Based on Capabilities

- **Narrow or Weak AI:**

Narrow AI refers to AI systems that are designed and trained for a specific task or set of tasks. They excel in performing these tasks but lack general human-like intelligence. Examples include virtual assistants like Siri and chatbots.

- **General or Strong AI:**

General AI aims to replicate human-like intelligence, including reasoning, understanding, and learning across a wide range of tasks. However, achieving true general AI remains a complex challenge and is mostly theoretical at present.

AI type-2: Based on Functionality

- **Reactive Machines (Limited Memory AI):**

Reactive machines are AI systems intentionally designed to carry out specific tasks through predefined rules and inputs. However, they lack the ability to retain memory or learn from

prior experiences. While they excel in tasks with clear parameters, they cannot adjust to novel situations. Examples include programs that play chess and certain expert systems.

- **Limited Memory AI:**

AI systems with limited memory encompass a level of data retention to enhance their performance through accumulated experience. These systems leverage historical data and patterns to make decisions, although their specialization remains within designated tasks. Notable instances of this are self-driving vehicles and recommendation systems.

- **Theory of Mind AI:**

In a more conceptual than practical context, Theory of Mind AI revolves around the creation of machines capable of grasping human emotions, beliefs, intentions, and thoughts. Rooted in the concept of comprehending others' mental states, this type of AI is vital for facilitating social interactions.

- **Self-Aware AI (Conscious AI):**

The notion of self-aware AI, also referred to as conscious AI, presents a highly advanced theoretical proposition where machines attain a self-awareness and consciousness comparable to that of human beings. This idea largely resides within the realm of philosophy and speculative conversations

3.2.4 Use Cases of AI

AI has made its way into virtually every industry, functioning smoothly, and revolutionizing various fields, as seen in the notable use cases across different domains:

- **Healthcare:**

AI plays a significant role in medical image analysis, aiding in early disease detection, drug discovery, personalized treatment planning, and patient monitoring. Diagnostic AI systems assist radiologists in identifying medical image abnormalities, contributing to the evolution of personalized medicine. Predictive AI models enhance disease diagnosis and therapeutic response prediction, potentially preventing future illnesses.

- **Finance:**

AI is reshaping finance by enabling fraud detection, risk assessment, algorithmic trading, and portfolio management. Banking sees transformation through AI-driven chatbots enhancing customer service and robot-advisors streamlining investment decisions. In insurance, AI-based chatbots improve customer experience and claim processing, while fraud detection thrives with AI analysing data points. Overall, AI's profound impact has modernized online banking, boosted security, and revolutionized finance management.

- **Gaming:**

AI transforms gaming with strategic gameplay and immersive development. It empowers adaptive NPCs, dynamic content generation, and player engagement analytics, while also advancing anti-cheat systems, matchmaking, and personalized suggestions, shaping the future of interactive entertainment.

- **Information Technology:**

AI has profoundly influenced the field of information technology (IT) with its diverse applications and use cases [17].

AI enriches user interactions by creating personalized experiences through sentiment analysis and emotion recognition, enhancing services and products.

Mobile apps integrated with AI streamline daily business processes, reducing errors through automation, and ensuring high accuracy in data management.

AI-powered mobile apps outperform human speed in tasks like data processing, customer queries, and workflow management, amplifying business efficiency.

From chatbots to personalized experiences, AI's potential in mobile apps mirrors its impact on web applications, encompassing automation, user experience, and security enhancement.

Incorporating AI into IT infrastructure monitoring anticipates issues, optimizes performance, and minimizes downtime. AI-driven cybersecurity systems swiftly detect and counteract threats, adapting to evolving risks. Automated AI testing tools enhance software quality, while AI-based analytics extract insights, enabling data-driven decisions and innovation.

3.2.5 Neural Network Language Model

The neural network language models are the models inspired by the human brain's structure. These models are designed to recognize, interpret, and generate human language in a way that is both meaningful and contextually relevant. They use layers of artificial neurons, they are as well called as the nodes, and they are used to process data, learning to understand and generate language by identifying patterns and nuances in vast amounts of text data. These nodes are just like how in a brain, neurons pass electrical impulses to each other [4]. The neural networks allowed mankind to create many different technologies. Large language models are one of them and it is used in chat bots, where the most known one is Chat-GPT. Apart from that, it is worth mentioning image generators, which generate pictures or images given the text input.

As it was mentioned above, the neural network language models consist of many nodes. These nodes are structured across three main layers. The input layer, hidden layer, and output layer. While these three layers form the basic structure of neural networks, it's common to find networks with additional hidden layers beyond the standard input and output layers.

Each node, regardless of the layer it belongs to, has a specific job: to process the input it receives. This could be from the previous node or, in the case of the first layer, directly from the input layer. Essentially, each node uses a mathematical formula where different variables are given different weights. If the result of this formula applied to the input is above a certain level (or threshold), the node sends the data forward to the next layer in the network. If not, it doesn't send anything onwards.

3.2.5.1 Large Language models

The large language models are the models based on neural networks and they are used as an artificial intelligence tool to recognize and generate the text. A large language models are built on machine learning: specifically, a type of neural network called a transformer model [4]. The "large" in its name comes from the vast amounts of data it learns from, which helps it grasp the intricacies of human language. This extensive reading helps the LLM recognize patterns in language, allowing it to predict and replicate the way humans communicate. Programmers make

sure to feed it high-quality information, so it learns to mimic natural language as accurately as possible.

Apart from just reading, LLMs are also skilled learners thanks to deep learning, a type of machine learning. This technique enables them to notice and remember patterns in text, like which words tend to follow others, without needing someone to guide them through every step. After their initial training, LLMs can be further fine-tuned to specialize in specific tasks, such as translating languages or answering questions.

3.3 Code generation and AI.

In today's fast-paced technological landscape, AI's role in application development is rapidly expanding. This growth is notably evident in the area of code generation, where AI systems are increasingly employed to automate and enhance the coding process.

3.3.1 AI and Application Development

The integration of AI in application development signifies a revolutionary shift. AI algorithms can now understand programming languages and assist in writing code, spotting errors, and even suggesting optimizations. Application development collects vast amounts of data, providing the ideal environment for AI to come in and supercharge digital transformation [23]. By analyzing patterns and common practices in these datasets, AI can generate code snippets, functions, or even entire modules.

3.3.2 Advantages and Disadvantages of code generation with AI

Advantages:

- **Efficiency and Speed:**

AI can rapidly generate code, significantly reducing the time developers spend on routine or repetitive tasks.

- **Error Reduction:**

AI systems are adept at identifying and correcting errors, leading to more robust and reliable code.

- **Learning and Improvement:**

AI systems continuously learn from new data, meaning their code generation capabilities improve over time.

Disadvantages:

- **Dependency Risks:**

Over-reliance on AI for code generation can lead to a decline in fundamental coding skills among developers.

- **Complexity and Interpretability:**

Sometimes, the code generated by AI can be complex and hard to interpret, making debugging and maintenance challenging.

- **Quality Concerns:**

AI-generated code might not always meet the specific quality or standards required for certain projects, especially those requiring highly specialized knowledge. Moreover, AI algorithms can be biased as a result of the data they are trained on, and they might not always be able to correctly predict outcomes [3].

3.3.3 How code generation works with AI.

AI code generation is a process of using machine learning models to generate code automatically. It can help developers write code faster and more efficiently by automating repetitive tasks such as generating boilerplate code, refactoring legacy code, writing test cases, checking for vulnerabilities and much more [19]. There are a lot of AI code generation tools that allow its users to describe what to do in natural non-programming language.

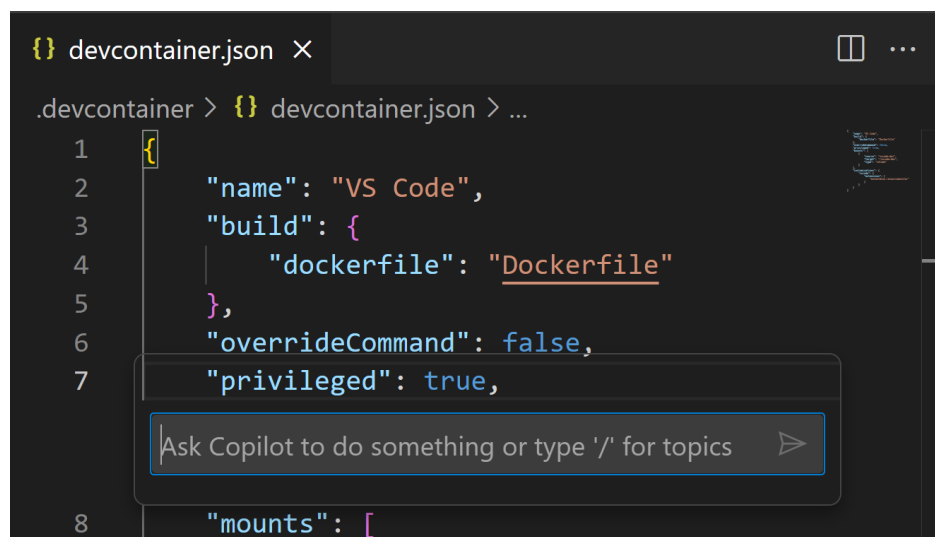
These tools use algorithms that have been trained on a large amount of existing source code, which is usually publicly available and created as part of Open-Source projects. Based on the examples from those projects, the algorithms generate a code that user requested

3.3.4 Code generation tool list

There are plenty of code generation tools existing in the app development market.

- **GitHub Copilot:**

GitHub Copilot is an AI-powered code completion tool developed by GitHub in collaboration with OpenAI. It assists developers by suggesting code snippets, auto-completing lines, and even writing whole functions based on context. It supports multiple programming languages and frameworks.



```
{} devcontainer.json X
.devcontainer > {} devcontainer.json > ...
1 {
2   "name": "VS Code",
3   "build": {
4     "dockerfile": "Dockerfile"
5   },
6   "overrideCommand": false,
7   "privileged": true,
8   "mounts": [
```

Figure 4: GitHub Copilot to enhance your coding with AI. **Source:** [24]

- **TabNine:**

TabNine, now part of GitHub Copilot, was an AI-powered autocompletion tool that used machine learning to predict and suggest code as you type. It could anticipate your next code snippet and provide context-aware suggestions for various programming languages.

- **CodeT5:**

CodeT5 is a variant of the T5 (Text-to-Text Transfer Transformer) model developed by Google. It's designed specifically for code generation tasks. It can generate code from natural language descriptions, answer questions about code, and perform other code-related tasks through text-based input.

- **FlutterFlow:**

FlutterFlow is a low-code development platform that uses AI to streamline app development for Google's Flutter framework. It is a visual builder that lets you build 10x faster by creating beautiful UI, generating clean code, and deploying to the app stores or web in one click [12]. The AI assists in automating various coding tasks, reducing the need for manual coding.

- **ChatGPT:**

ChatGPT, like the one you're interacting with, is a language model developed by OpenAI. While not specifically a code generation tool, it can assist developers by answering coding-related questions, providing explanations, and even generating code snippets based on the context and prompts given by the user.

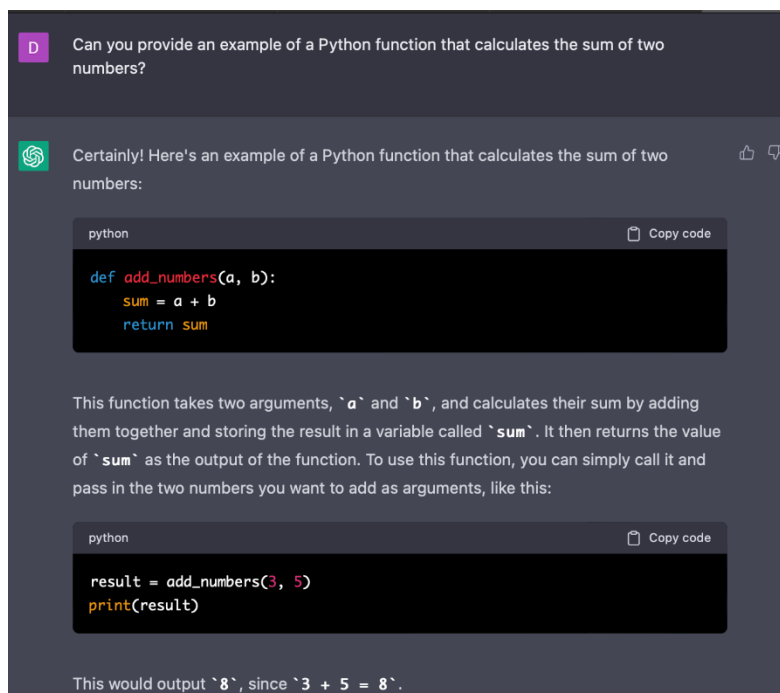


Figure 5: ChatGPT's response for users request. **Source:** [14]

3.3.5 The importance of AI generated code

With AI code technologies, it's an important to look into how the world of software development is changing. AI is changing things up, making it possible to create code faster and

opening up coding to more people, and it is a highly dynamic tool that offers substantial advantages within the right context [20]. This means that whether you're just starting out or you've been coding for years, you can quickly turn ideas into working software.

However, this new approach isn't without its drawbacks. There are concerns about whether the code is as unique or secure as what a human might write, and it might even affect how well developers understand the basics of coding. AI help could lead to more efficient work and better products. But it's not just guesswork—really diving into how AI is mixing things up in software development helps us understand what's actually happening. This is definitely something we should dig into more, especially when we think about where technology is headed and how we're going to build it.

4 Practical Part

This section covers the development process of some medical reminder applications, one utilizing AI-generated code, one author-developed code, and an application from professional project.

The professional app was chosen from GitHub to serve as model project was a thoughtful move. This app is built on a strong, well-organized codebase. This made it a great starting point for learning and comparing it with other two medical reminder apps.

The author developed app had a clear focus on building a strong and organized foundation right from the start. The application development process was followed with the software industry standard to make sure the app would match the features of a professional app, aiming to bring those same high-quality functions to author developed app. Along the way, numerous tutorials were followed, broadening personal skills, and understanding of how to bring the app to life.

On top of that, ChatGPT was chosen for generating code, and it brought an extra touch of innovation to the project. ChatGPT is known for being flexible, easy to access, and has received lots of praise for its capabilities.

The evaluation criteria involved assessing the code quality of all apps in terms of maintainability and complexity, evaluating development efficiency in terms of time speed and cost, as well as performance of each app. Everything was noted in the process of development or analysis.

4.1 Flutter SDK setup

The Flutter SDK was set up on IntelliJ IDEA v. 2023.3 by Jet Brains on Windows 11.

The android virtual device was installed. It is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator [6]. All the applications were run on Pixel 7 Pro phone emulator.

Java is an essential component of the Flutter development environment. The Java version 21 was downloaded as well.

4.1.1 Libraries

There were used Flutter libraries in this project. And below can be seen how each library can be integrated into a Flutter application, contributing to functionalities like state management, data persistence, notifications, and reactive programming:

- **Provider:**

A wrapper around InheritedWidget to make it easier to use and more reusable. It's used for state management in Flutter apps.

- **RxDart:**

Adds additional capabilities to Dart Streams and StreamControllers. It's used for reactive programming.

- **Flutter_local_notifications:**

A plugin for displaying local notifications. It's used for scheduling and displaying notifications within the app.

- **Shared_preferences:**

Provides persistent storage for simple data (key-value pairs). It's commonly used for storing user preferences or settings.

- **Flutter/widgets.dart:**

The Flutter framework's foundational library, providing widgets and other basic functionalities for building UIs.

4.2 Professionally developed project

Mediminder: An Offline Medicine Reminder[13] is a Flutter application built with the Provider and BLoC pattern, offering key features such as a homepage displaying the registered Mediminders, shared preference data storage, and local notifications for daily reminders. Users can add new Mediminders with details like name, dosage, icon selection, reminder intervals, and starting time. The app includes error checking for registration, such as name duplication and empty fields, and allows users to view in-depth details and delete specific Mediminders with corresponding notifications.

Let's open the project to check what it offers to the users. Application meets the user with the main page. User can create a reminder by clicking '+' button. User also can also see his current reminder.

The MedicineReminder app, initiated from the main page, utilizes the Provider package for state management with a GlobalBloc class. Upon launch, it displays the HomePage widget as its main interface, likely showcasing scheduled medicine reminders, functionalities for adding or editing reminders, and additional features for medicine management. The app employs a green-themed MaterialApp with debug banner disabled for a cleaner UI. The GlobalBloc is made available app-wide, indicating the HomePage serves as the central hub for user interaction and management of reminders.

4.2.1 Home page

The HomePage class in this code sets up the main screen for the MedicineReminder app, utilizing Provider for state management with GlobalBloc. It features a Scaffold with a green AppBar and a Container body that includes a top section for displaying app and medicine count info, and a bottom section that dynamically lists medicine reminders using a GridView. The FloatingActionButton navigates to a NewEntry screen for adding reminders. The MedicineCard widget, used in the GridView, displays medicine details and navigates to MedicineDetails on tap, showcasing the use of Hero animations for smooth transitions.

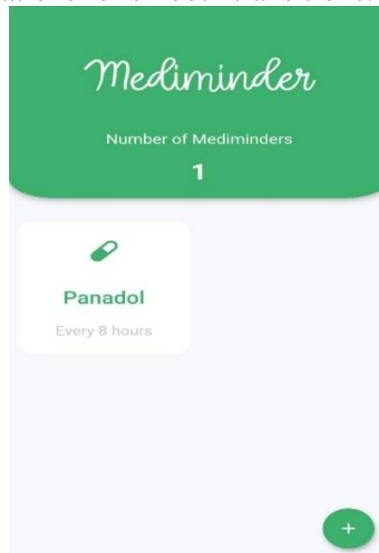


Figure 6: Home page of professional app **Source:** [13]

4.2.2 Create Reminder page

The NewEntry class defines a screen for adding new medication reminders. It includes form fields for entering medication name, dosage, and type, along with a selection for the medication intake interval and starting time. Utilizing the Provider package, it interacts with GlobalBloc and NewEntryBloc for state management and utilizes flutter_local_notifications for scheduling notifications. The UI facilitates user input with validations and feedback, leading to a confirmation action that schedules medication reminders and navigates to a success screen.

The GlobalBloc class in this project manages the application's state related to medicine reminders using Rx Dart for reactive programming and Flutter Local Notifications for scheduling notifications. It maintains a list of Medicine objects as a BehaviorSubject stream, allowing widgets to react to changes in the medicine list. The class provides functions to add, remove, and update medicines in the list, integrating with Shared Preferences for persistent storage of medicine data and using Flutter Local Notifications for managing medicine reminder notifications.

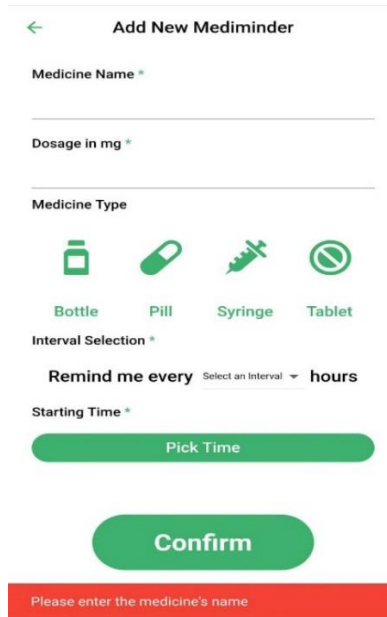


Figure 7: Create reminder page of professional app **Source:** [13]

4.2.3 Reminder Details page

The `MedicineDetails` widget displays detailed information about a selected medicine, including its name, dosage, type, interval, and start time. It provides a comprehensive view within a clean, user-friendly interface, utilizing a white-themed Scaffold with a custom AppBar. The widget incorporates functionalities like deletion of a mediminder with a confirmation dialog. Key components like `MainSection` and `ExtendedSection` are used for structured presentation of medicine data, enhancing the user experience by providing clear, concise information and easy navigation. The medicine name, dosage, type, interval, and time are indicated.

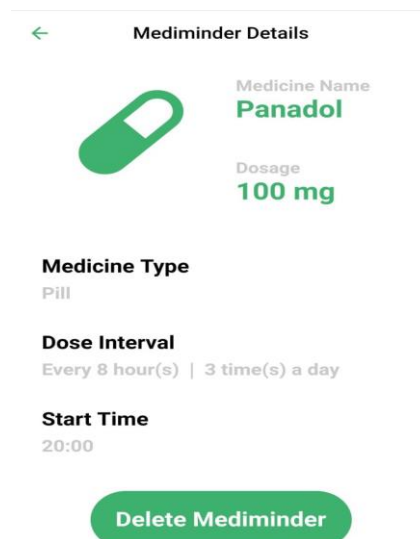


Figure 8: Medicine details page of professional app **Source:** [13]

There were five models used in total: day, duration, errors, medicine, medicine_type.

The `Medicine` model is defined for storing medicine reminders, including notification IDs, medicine name, dosage, type, interval, and start time. Other models like `Day`, `Duration`, `Errors`, and `MedicineType` likely serve similar purposes in modeling different aspects of the app, such as representing days of the week, durations of time, error types, and categorizations of medicine types, respectively, each contributing to the overall functionality of the medication reminder application.

4.3 Author developed application

The author developed app displays similar performance when compared to the professionally developed. The primary distinctions lie in slight structural nuances of the codebase and aesthetic elements of the application's design.

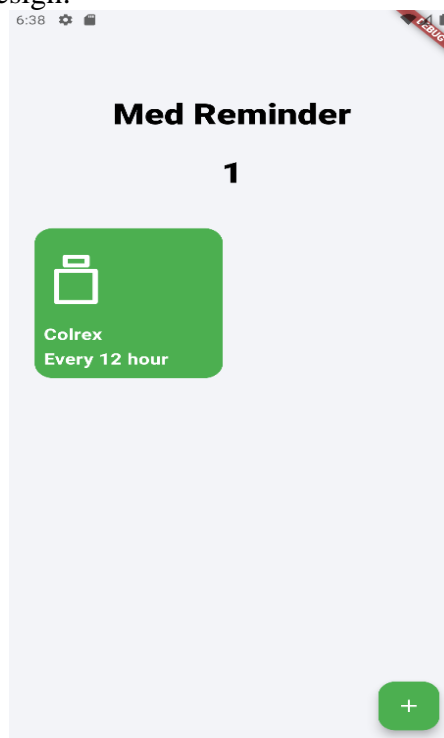


Figure 9: Author developed, home page **Source:** author

The author developed app was slightly changed in terms of of design, colors, and placement.

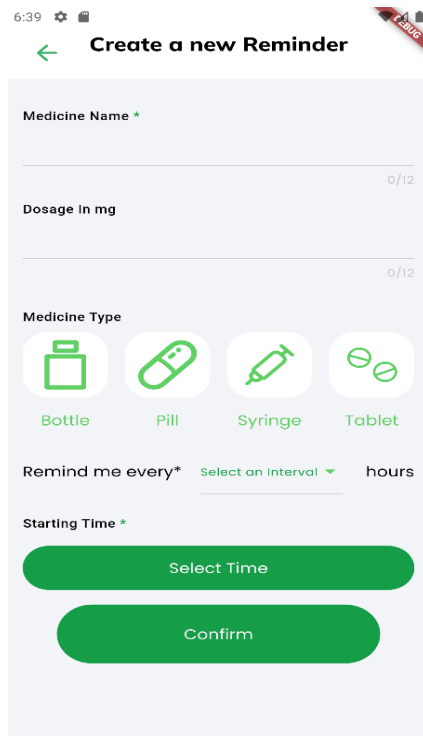


Figure 10: Author developed, create reminder page **Source:** author

Minor but meaningful modifications in the new entry and reminder details pages were also executed, aiming to elevate the overall user interface while maintaining operational consistency.

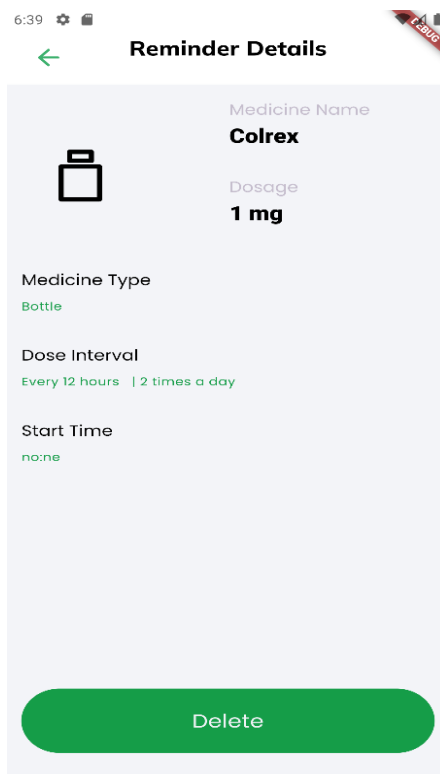


Figure 11: Author developed, medicine details page **Source:** author

For instance, the author developed app also integrates SVG (Scalable Vector Graphics) through a dedicated library for iconography. This implementation ensures high-quality, scalable

icons across different resolutions, contributing to the aesthetic refinement of the app's design. This choice underscores the developer's commitment to a polished and scalable user interface, aligning with modern app development standards.

4.4 Chat-GPT generated application

The Chat-GPT generated app successfully meets its functional objectives, mirroring the capabilities of its counterparts. Notably, the architecture centralizes features within the main page by classifying all page functionalities in one location. The use of SVG for icons, alongside the incorporation of service and provider libraries, marks a small difference from other apps.

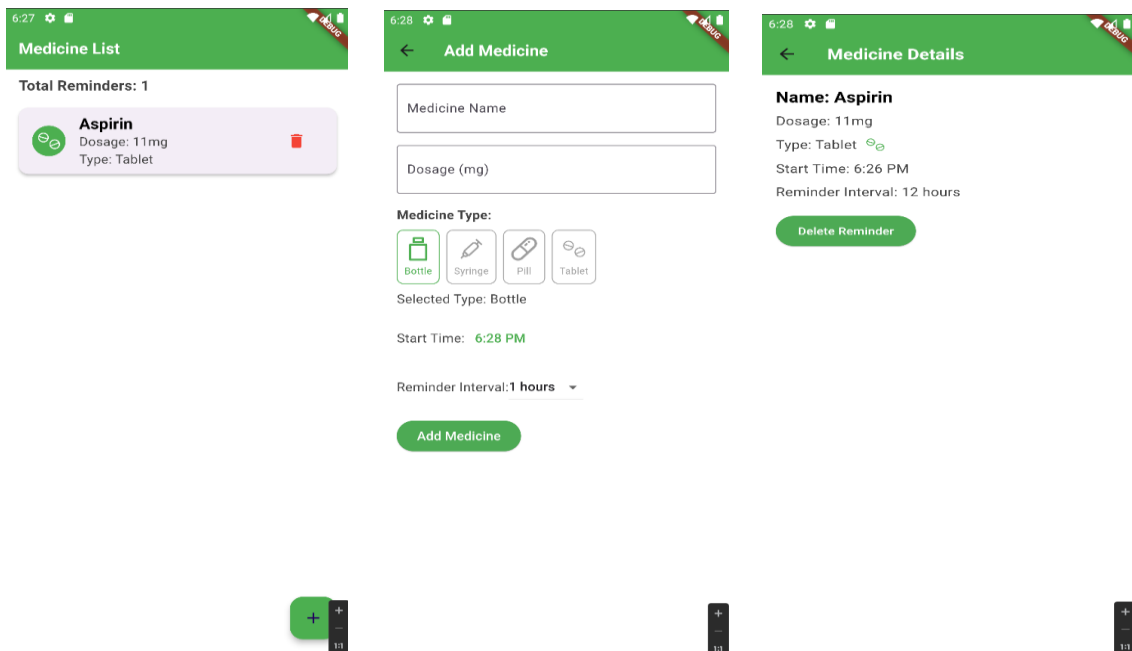


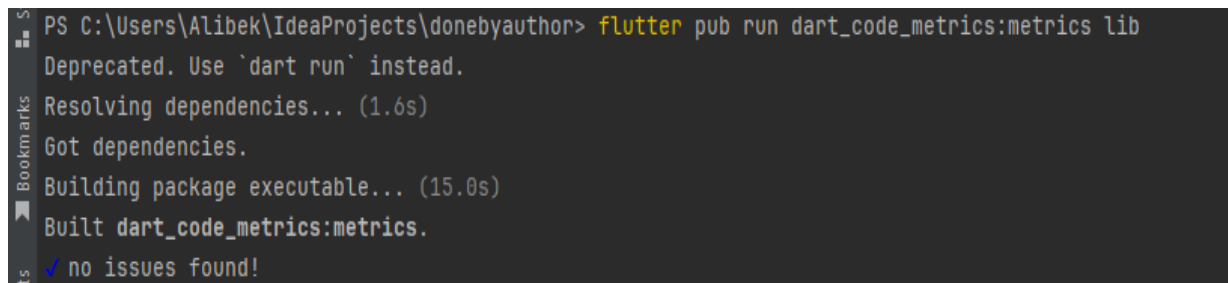
Figure 12: Chat-gpt generated app, all pages Source: author

4.5 Evaluation

The evaluation for the practical part of the thesis was systematically conducted across Code Quality, Application Performance, and Development Efficiency.

4.5.1 Code Quality

The code went through review of the codebase for each app, using static analysis tools like Dart Code Metrics to assess maintainability, adherence to style guides, and overall code complexity.



```
PS C:\Users\Alibek\IdeaProjects\donebyauthor> flutter pub run dart_code_metrics:metrics lib
Deprecated. Use `dart run` instead.
Resolving dependencies... (1.6s)
Got dependencies.
Building package executable... (15.0s)
Built dart_code_metrics:metrics.
✓ no issues found!
```

Figure 13: Terminal result for DCM Source: author

To evaluate all above mentioned factors, the Dart Code Metrics plugin was installed. DCM dependency was added, and its analysis option settings were set. This was needed to configure the analyser, which statically analyses Dart code to check for errors, warnings, lints, etc. After this, the ‘flutter pub run dart_code_metrics:metrics lib’[7] command was run in the terminal for each application separately. As the applications were fully functional, there were no issues found.

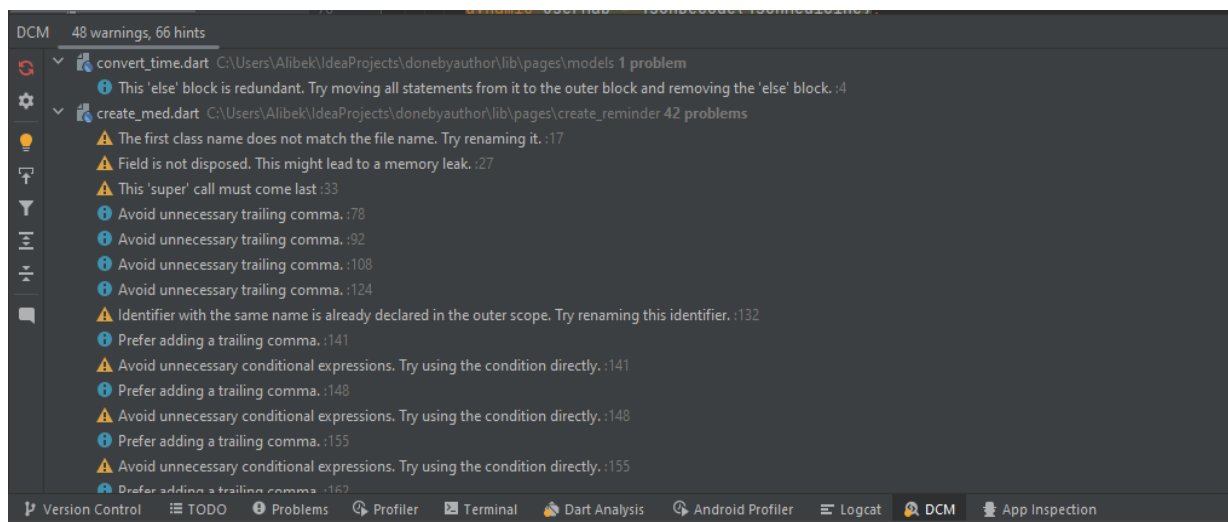


Figure 14: DCM result for the code Source: author

For each app DCM (Dart Code Metrics) analysis was done, the tool identified various warnings and hints, suggesting areas where the code could potentially be improved or amended. Below can be seen DCM test for each application.

DCM results	Professional App	Author developed App	Chat-GPT generated APP
issues	0	0	0
warnings	3	48	5
recommendations	13	66	23
Total memory allocated in Mbs once application is run	16	114	28

Table 1: DCM results for each app. **Source:** author

4.5.2 App Performance

Utilizing Flutter DevTools, the apps were be profiled to measure average load time in milliseconds, and memory consumption.

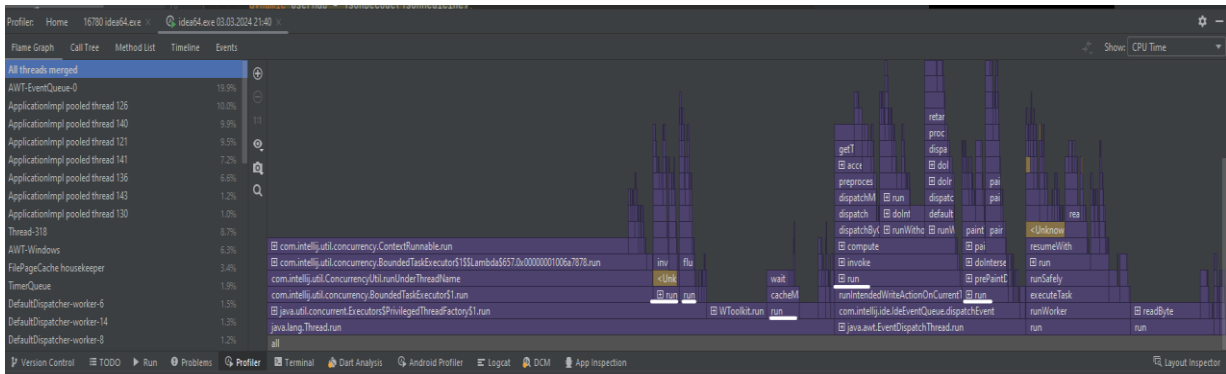


Figure 15: Profiler results for app that was run 5 times **Source:** author

The Profiler recorded the average load time in milliseconds for each app over five attempts, providing an objective measure of application performance.

Average load time in milliseconds	Professional App	Author developed App	Chat-GPT generated APP
1 attempt	433	481	368
2 attempt	456	304	255
3 attempt	716	600	320
4 attempt	908	852	702
5 attempt	981	1187	616
Average load time:	698.8	684.8	452.2

Table 2: Average app load time in milliseconds for each app. **Source:** author

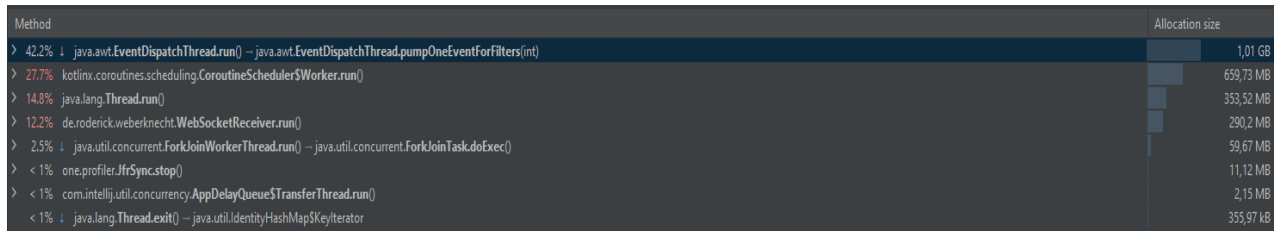


Figure 16: Profiler results for the memory consumption of the app **Source:** author

The total memory allocated for running each program was recorded with the help of profiler. And below it can be seen the table of memory for each program, giving us quite similar numbers.

Memory consumption	Professional App	Author developed App	Chat-GPT generated APP
Total memory allocated in Mbs once application is run	1593,77 Mb	1 377,4 Mb	1 726,56 Mb

Table 3: Memory consumption for each app. **Source:** author

4.5.3 Development Efficiency

The focus here was on documenting the time of development and the overall cost associated with the development process.

Time of development was assessed by comparing the development duration of the author's and the AI-generated apps against the industry standard for a professionally developed app of similar complexity. Detailed records of development time have been meticulously maintained to ensure an accurate assessment of the efficiency in bringing the apps from concept to completion.

time of development	Professional App	Author developed App	Chat-GPT generated APP
days	1-7 days	28 days	3 days
sessions	Up to professional	13 sessions	4 sessions
hours spent/ per session	Up to professional	1,5 hours	1,5 hours

Table 4: Time of development for each app. **Source:** author

The development cost for a professional app can vary widely, often influenced by hourly rates which are dictated by the developers' expertise and the project's complexity. As noted in industry analysis, the cost for simple app development by a professional or team may vary depending on hourly rates, given the fact that in Czech Republic this rate is \$50 – \$149 / hr [8], and time spent on project.

The cost-free designation for the Author developed app highlights the advantage of self-development where labour costs aren't calculated, thus reducing the financial barrier to entry.

The Chat-GPT generated app has a nominal subscription cost of \$25, which could represent access to AI-powered development tools, demonstrating a cost-effective approach to app creation.

cost	Professional App	Author developed App	Chat-GPT generated APP
Development cost in USD	\$50 – \$149 / hr	free	25

Table 5: Development cost for each app. **Source:** author

5 Results and Discussion

The evaluation process across the three domains—Code Quality, Application Performance, and Development Efficiency—was conducted on a scale ranging from 1 to 5. Each domain received a score that reflected the degree to which the applications met the set criteria: the closer to 1, the better the performance, and the closer to 5, the more room for improvement. For each score assigned, detailed explanations were provided, elucidating the rationale behind the ratings to ensure a clear understanding of the applications' performance in each aspect.

5.1 Code quality results

The evaluation of maintainability, complexity, and adherence to style guides is based on the output from Dart Code Metrics.

Maintainability is assessed by looking for issues that could hinder future updates or enhancements, such as non-disposed resources or redundant code blocks. A lower score suggests more such issues were found.

Adherence to Style Guides reflects how well the code follows the Dart language's recommended formatting and structure. Consistent use of trailing commas and proper naming are positive signs, whereas violations lower the score.

Complexity considers the simplicity of logic and structure. Fewer warnings about issues like redundant 'else' blocks or nullable checks suggest lower complexity and a higher score.

Code quality	Professional App	Author developed App	Chat-GPT generated APP
maintainability	1	2	3
complexity	2	3	4
adherence to style guides	1	3	5
total average points	1,3334	2,6667	4

Table 6: Code quality. **Source:** author

- **Professional app:**

Maintainability 1/5: Few to no basic style violations like missing trailing commas or constant constructors, suggesting a strong adherence to Dart's style guide. Easy to maintain due to clear documentation and consistent coding practices.

Adherence to Style Guides 1/5: Optimal use of resources with all streams and controllers being properly disposed of, pointing to good resource management.

Complexity 2/5: Normal complexity with straightforward logic, making it a bit easier for developers to understand and work with the code.

- **Author developed app:**

Maintainability 2/5: The code has several issues that can complicate maintenance, such as resources not being disposed of and redundant blocks.

Adherence to Style Guides 3/5: The consistent use of trailing commas and the enforcement of naming conventions suggest a moderate adherence to Dart's style guide, although there are several violations that need to be addressed.

Complexity 3/5: No direct complexity issues are mentioned, implying the logic may not be overly complex.

- **Chat-GPT generated app:**

Maintainability 3/5: Several issues like non-disposed instances and repeated identifiers suggest potential maintenance challenges. However, these are relatively straightforward to address.

Adherence to Style Guides 5/5: The need for const constructors and naming conventions indicate a deviation from best practices.

Complexity 4/5: The presence of warnings like redundant 'else' blocks and unnecessary nullable checks suggest moderate complexity. The code could benefit from simplification for better readability and maintainability.

5.2 Application performance results

In the comparison of application performance, the average load time and resource consumption were the primary factors considered.

Application performance	Professional App	Author developed App	Chat-GPT generated APP
Average load time in milliseconds	3	3	1
Memory Consumption	3	1	4
total average points	3	2	2,5

Table 7: App performance. **Source:** author

- **Professional app:**

The Professional app showed a balanced performance with middle-ground scores (3/5) in both categories, suggesting it doesn't excel or lag significantly in any particular area but offers a consistent, moderate performance.

- **Author developed app:**

The Author developed app had the best resource consumption score (1/5), which could imply it's the most memory-efficient, even if its load times were slightly less consistent.

- **Chat-GPT generated app:**

The Chat-GPT generated app had the lowest average load time, which is why it received the best score (1/5) in that category. It suggests that the AI-generated app was the quickest to become operational. However, for resource consumption, the Chat-GPT generated app had the highest memory allocation, resulting in the worst score (4/5) in that area. This might indicate that while the AI-generated app is quick to load, it may be less optimized regarding memory usage.

These scores reflect a trade-off between quick accessibility and efficient resource use, which are both valuable metrics in assessing application performance.

5.3 Development efficiency results

Development efficiency	Professional App	Author developed App	Chat-GPT generated APP
time of development	3	5	1
cost	4	1	2
total average points	3.5	3	1.5

Table 8: Development efficiency. **Source:** author

- **Professional app:**

Received a mid-range score for time (3/5) due to a balance of rapid development and professional hours spent, but a lower score for cost (4/5) due to the higher hourly rates typically associated with professional development.

- **Author developed app:**

Author Developed App: Scored the least efficiently for time (5/5) because it took the longest to develop, despite being cost-free (1/5), reflecting no monetary expenditure.

- **Chat-GPT generated app:**

Chat-GPT Generated App: Achieved the best score for development time (1/5), indicating rapid creation capability, while the cost was modest (2/5), thanks to the use of freely accessible AI tools, only incurring a minor subscription fee.

5.4 Final points results

Final points	Professional App	Author developed App	Chat-GPT generated APP
Code quality	1.3334	2.6667	4
Application performance	3	2	2.5
Development efficiency	3.5	3	1.5
Total:	7.8334	7.6667	8

Table 9: Final points. **Source:** author

The final evaluation table consolidates the scores from the three assessment categories for each app. The Professional App, with the lowest total, suggests an overall balance in terms of code quality, application performance, and development efficiency. The Author Developed App scores slightly better, indicating a marginally more efficient development process. The Chat-GPT Generated App has the highest total score due to its lower code quality rating, although it leads in development efficiency.

6 Conclusion

To conclude this thesis, the exploration focused on AI's role in code generation for app development, assessing AI-generated code versus human-written code for quality and efficiency, and examining productivity and development cost implications.

This journey from theoretical underpinnings to empirical evaluation across three applications—AI-generated, author-developed, and professionally developed—revealed that while AI accelerates development and shows efficiency, it varies in quality and performance against human-written code.

The findings suggest that AI code generation, with its rapid development capabilities, still requires refinement for optimal quality. A potential hybrid model, combining AI efficiency with human expertise, might offer a balanced approach. This thesis underscores the evolving role of AI in software development, highlighting areas for future research and development, and suggesting that the integration of AI in coding practices is just beginning.

7 References

- [1] Altexsoft. How API works image [online]. 2019. Available at: <https://www.altexsoft.com/media/2019/06/1.png>. Accessed 12 December 2023
- [2] ASH TURNER. Number of mobile phone & smartphone users[online]. 08 August 2023. Available at: <https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#:~:text=According%20to%20Statista%2C%20in%202023,of%20that%20year's%20global%20population>. Accessed 16 August 2023.
- [3] Binmile. Pros and Cons of AI-Assisted Coding [online]. 01 March 2024. Available at: <https://binmile.com/blog/pros-and-cons-of-ai-assisted-coding>. Accessed 04 March 2024.
- [4] Cloudflare. What is Neural Network? [online]. 2024. Available at: <https://www.cloudflare.com/learning/ai/what-is-neural-network>. Accessed 27 February 2024.
- [5] Cross Platform App development image [online]. 2022 Available at: https://media.licdn.com/dms/image/D4D12AQEgNMKwJkdbHg/article-cover_image-shrink_720_1280/0/1677315403572?e=2147483647&v=beta&t=3tZ4X-_V5RvOtKM8iUjsMSThTcZ1PO2eWqlMxsh0XaA. Accessed 14 December 2023.
- [6] Developer Android. Managing AVDs [online]. 12 March 2024. Available at: <https://developer.android.com/studio/run/managing-avds>. Accessed 06 January 2024.
- [7] Dart Code Metrics. DCM Documentation [online]. 2024. Available at: <https://dcm.dev/docs/>. Accessed 23 February 2024.
- [8] ELENA KONONCHUK. App Development Cost [online]. 24 September 2023. Available at: <https://qubit-labs.com/app-development-cost>. Accessed 23 February 2024.
- [9] FARHAN THAWAR. React Native is the Future of Mobile at Shopify. [online]. 29 January 2023. Available at: <https://shopify.engineering/react-native-future-mobile-shopify> Accessed 16 August 2023.
- [10] FEZA ROHEEL. What are the features of Flutter?[online]. 2024. Available at: <https://www.educative.io/answers/what-are-the-features-of-flutter>. Accessed 18 August 2023.
- [11] Flutter/Dart [online]. Available at: <https://dart.dev/>. Accessed 16 August 2023.
- [12] Flutter Flow [online]. Available at: <https://flutterflow.io/>. Accessed 18 August 2023.
- [13] GitHub. Mediminder [online]. Available at: <https://github.com/HossamElghamry/Mediminder?tab=readme>. Accessed 06 January 2024.
- [14] Google pictures. Chat- GPT interaction [online]. Available at: https://lh3.googleusercontent.com/jImDgwIvP9cDSfYrcE4S40Pd6XKvCBR_wQZXmxDAWzHEH73dOKW9UF2xUIPMAcMZEYVaG-6WUVMsy8PxR0JydEOw0MTvWi6bf1ctG704YxS__LgaWsWnT_dwD-mNx63P1HySAIJvCakGze97wozBJrI. Accessed 06 January 2024.

[15] HARDIK B. A Brief History of Flutter By Farhan Thawar[online]. 22 February 2023. Available at: <https://www.blup.in/blog/a-brief-history-of-flutter>. Accessed 16 August 2023.

[16] Javatpoint. Types of Artificial Intelligence [online]. 2021. Available at: <https://www.javatpoint.com/types-of-artificial-intelligence> . Accessed 16 August 2023.

[17] LEEWAY HERTZ. AI Use Cases and Applications [online]. [Year]. Available at: <https://www.leewayhertz.com/ai-use-cases-and-applications/>. Accessed 16 August 2023.

[18] MIKE KING. Types of Mobile Apps [online]. 13 July 2023. Available at: <https://www.businessofapps.com/app-developers/research/types-of-mobile-apps/>. Accessed 16 August 2023.

[19] RADU MICLAUS. How generative AI code assistants could revolutionize developer experience [online]. 29 July 2023. Available at: [<https://venturebeat.com/ai/how-generative-ai-code-assistants-could-revolutionize-developer-experience/>]. Accessed 18 August 2023.

[20] RAFAEL TIMBÓ. AI-Generated Code [online]. 06 December 2023. Available at: <https://www.revelo.com/blog/ai-generated-code>. 04 March 2024.

[21] RUSSELL, NORVIG, 2016. Artificial Intelligence: A Modern Approach 3rd Edition. ISBN 9781292153964

[22] Sonatype Help. Application Categories [online]. 08 November 2023. Available at: <https://help.sonatype.com/en/application-categories.html>. Accessed 27 February 2024.

[23] Turing. AI Application Development [online]. 2024. Available at: <https://www.turing.com/resources/ai-application-development>. 04 March 2024.

[24] Visual studio. Visual Studio Code Artificial Intelligence [online]. 28 November 2024. Available at: <https://code.visualstudio.com/docs/copilot/overview>. Accessed 14 December 2023.

8 List of pictures, tables, graphs and abbreviations

8.1 List of pictures

Figure 1: How API's Work	14
Figure 2: Cross-Platform App Development	15
Figure 3: Types of Artificial Intelligence	18
Figure 4: GitHub Copilot to enhance your coding with AI.	23
Figure 5: ChatGPT's response for users request.	24
Figure 6: Home page of professional app	27
Figure 7: Create reminder page of professional app	28
Figure 8: Medicine details page of professional app	28
Figure 9: Author developed Home page	29
Figure 10: Author developed Create reminder page	30
Figure 11: Author developed Medicine details page	30
Figure 12: Chat-gpt generated app, all pages.....	31
Figure 13: Terminal result for DCM.....	32
Figure 14: DCM result for the code	32
Figure 15: Profiler results for app that was run 5 times.....	33
Figure 16: Profiler results for the memory consumption of the app.....	34

8.2 List of tables

Table 1: DCM results for each app	33
Table 2: Average app load time in milliseconds for each app.	33
Table 3: Memory consumption for each app.	34
Table 4: Time of development for each app.	34
Table 5: Development cost for each app.....	35
Table 6: Code quality.....	36
Table 7: App performance.	37
Table 8: Development efficiency.....	38
Table 9: Final points.	39

8.3 List of abbreviations

AI – Artificial Intelligence
LLM – Large Language Models
DCM – Dart Code Metrics
SDK – Software development kit
OS- Operating systems
IT- Information technology
HTML – Hyper Text Markup Language

9 Appendix

Additional references (not directly cited)

Forbes. A Very Short History of Artificial Intelligence (AI) [online]. 30 December 2016. Available at:<https://www.forbes.com/sites/gilpress/2016/12/30/a-very-short-history-of-artificial-intelligence-ai/>. Accessed 16 August 2023.

Hamilton, A.C., 2023. Introduction to Chat GPT: Supercharge Your Productivity with AI - Create Apps, Websites, Google Chrome Extensions & Much More (Artificial Intelligence Uses & Applications). ISBN 9798889551669

M.Katz, K.Moore, V.Ngo, V.Guzzi, 2021, Second Edition, Flutter Apprentice: Learn to Build Cross-Platform Apps, ISBN: 9781950325481.

Thomas Bailey, Alessandro Biessek, 2021. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart, 2nd Edition 2nd ed. Edition. ISBN 9781800565999.