

POLITECNICO DI TORINO

Master's Degree in Computer Engineering



Master's Degree Thesis

**Automatic generation of User Interface
(UI) with the help of AI technologies**

Supervisor

Prof. Giovanni MALNATI

Candidate

Arash HONARVAR

October 2024

Abstract

The widespread adoption of digital technologies has resulted in an unparalleled need for user interfaces (UIs) that are both efficient and intuitive on a variety of platforms and devices. Software applications cannot be developed as quickly using traditional approaches of UI design, which frequently take a lot of time and experience. This thesis suggests a novel method for UI generation by integrating artificial intelligence (AI) technology in response to this difficulty.

The main goal of this research is to automate the creation of UI components and layouts using AI algorithms, thus streamlining the UI design process. However, the primary purpose of this thesis is information engineering, aimed at informing designers about the elements that comprise the UI pages based on user-provided project descriptions. This thesis specifically focuses on creating a Figma plugin that works with a Django API server and uses AI capabilities—especially the OpenAI platform and ChatGPT—to understand user-provided project descriptions and produce related user interface designs.

By developing a tool that can quickly prototype user interfaces (UIs) based on high-level project criteria, the proposed system hopes to empower designers and developers and cut down on the time and effort needed for manual design iterations. By utilizing AI, this method not only increases output but also fosters creativity by providing a variety of design ideas that are customized to meet project needs.

The research technique entails designing and implementing the Django API server and Figma plugin, integrating AI models for natural language generation and processing, and evaluating the system through a cost analysis of the plugin and a showcase of the UI designs generated by the Figma plugin. These efforts assess the system's scalability, adaptability, and overall effectiveness in producing high-quality UI designs, with a focus on cost-efficiency and user satisfaction.

The results of this study show the viability and advantages of AI-driven automation in the creative process, which advances UI design approaches. This approach has broad ramifications across multiple areas, such as AI-driven design tools, software development, and human-computer interaction. In the end, the suggested approach has the potential to transform user interface design methodologies, stimulate creativity, and enable designers to fulfill the changing demands of digital consumers in a quickly advancing technological environment.

Acknowledgements

“I would like to express my deepest gratitude to my supervisor, Professor Giovanni Malnati, for his invaluable guidance and support throughout this project. His expertise and encouragement were instrumental in the completion of my thesis. I also wish to extend my heartfelt thanks to my friends and family for their unwavering support and understanding during this journey. Their encouragement and patience provided me with the strength and motivation to persevere.”

Arash Honarvar, Turin, October 2024

Table of Contents

List of Tables	VII
List of Figures	VIII
Acronyms	X
1 Introduction	1
1.1 Overview and Motivation	1
1.2 Contribution	2
1.3 Organization of the thesis	3
2 Related Works	4
2.1 The Significance of Artificial Intelligence	4
2.1.1 The Importance of AI	4
2.1.2 Key Components of AI	4
2.1.3 The Rise of AI in Today’s World	5
2.1.4 Conclusion	6
2.2 Generative AI	6
2.2.1 ChatGPT	7
2.3 Figma: A Comprehensive Overview	9
2.3.1 Introduction to Figma	9
2.3.2 Key Features and Functionalities	9
2.3.3 Figma Plugin Ecosystem	10
2.3.4 Advantages of Using Figma	11
2.3.5 Significance in Contemporary Design Workflows	11
2.3.6 Limitations and Considerations	11
2.4 Uizard	12
2.4.1 Introduction to Uizard	12
2.4.2 Key Features	12
2.4.3 Advantages of Using Uizard	13
2.4.4 Limitations and Considerations	13

2.5	Galileo AI	13
2.5.1	Introduction to Galileo AI	14
2.5.2	Key Features and Functionalities	14
2.5.3	Advantages of Using Galileo AI	15
2.5.4	Limitations and Considerations	15
2.6	Mage (useMage.ai)	15
2.6.1	Key Features and Functionalities	16
2.6.2	Example Use Case	16
2.6.3	Advantages	17
2.6.4	Disadvantages	17
3	Architecture	19
3.1	Introduction	19
3.2	Overall System Architecture	19
3.3	Client-Side Architecture: Figma Plugin	19
3.3.1	Component-Based Design	20
3.3.2	State Management	20
3.3.3	Event-Driven Architecture	20
3.3.4	Integration with Figma API	21
3.4	Server-Side Architecture: Django Python Backend	21
3.4.1	RESTful API Design	21
3.4.2	Request Handling and Routing	21
3.4.3	Integration with External Services	21
3.4.4	Error Handling and Logging	22
3.5	Technologies and Frameworks	22
3.6	Conclusions	22
4	Implementation	23
4.1	Introduction	23
4.2	Setting Up the Development Environment	23
4.2.1	Version Control with Git	23
4.2.2	GitHub Repository	23
4.2.3	Integrated Development Environment (IDE)	24
4.2.4	Frontend Development Setup	24
4.2.5	Backend Development Setup	25
4.3	Frontend Implementation: Figma Plugin	26
4.3.1	Homepage Route	27
4.3.2	Process Route	27
4.3.3	Project Description Route	28
4.3.4	Managing Plugin Logic with 'controller.ts'	29
4.4	Backend Implementation: Django Python Server	33

4.4.1	REST API for Managing the Application	33
4.4.2	Process Endpoint for ChatGPT Communication	33
4.4.3	Interaction with ChatGPT	33
4.4.4	Template-based UI Generation	33
4.4.5	Instruction and Rules for ChatGPT	34
4.4.6	Interaction with OpenAI API for Design Generation	34
4.4.7	Error Handling for OpenAI API	48
4.5	How to Write a Detailed Project Description	48
5	Result and Discussion	51
5.1	Introduction	51
5.2	Example Showcase: UI Generation	51
5.2.1	The outlined project description	51
5.2.2	JSON Output from Django Server	52
5.2.3	Figma UI Designs	52
5.3	Cost Analysis	52
5.4	Intended Purpose and Design Philosophy	53
6	Future Works	56
A	JSON Output from Django Server	57
	Bibliography	70

List of Tables

5.1 Total Cost of Figma Plugin	52
--	----

List of Figures

3.1	Client-server architecture	20
4.1	HomePage screen when the plugin has been launched	28
4.2	Process screen for generating appropriate user interface	28
4.3	Crafting an Effective Project Description Page	50
5.1	Screenshot of Figma Plugin	54
5.2	Screenshot of Figma Plugin - Image 4	55
5.3	Screenshot of Figma Plugin - Image 5	55

Acronyms

AI

Artificial Intelligence

UI

User Interface

UX

User Experience

API

Application programming interface

GAI

Generative modeling artificial intelligence

GAN

Generative Adversarial Network

GPT

Generative Pre-trained Transformer

NLP

natural language processing

ML

machine learning

IDE

Integrated Development Environment

CLI

command-line interface

Chapter 1

Introduction

1.1 Overview and Motivation

The growth of digital technology and the development of design approaches have caused a considerable upheaval in the field of design in recent years. Alongside the rapid development of related tools and software, User Interface (UI) design has emerged as a result of the Internet industry's rapid progress and the widespread use of new media. As a result, there have been notable improvements in the design and use of software interfaces [1].

The field of User Experience and User Interface (UX/UI) is concerned with the design and development of software applications' interactive elements that are visible to end users. It covers both the user's experience—which is given by the application's overall logic and the actions the user must take to complete the task—and the visual components, such as design, elements arrangement, colors, and icons. A well-considered UX/UI design increases conversion rates, purchase probability, and customer loyalty. It also increases recommendation rates and decreases abandonment [2].

To achieve a well-considered UX/UI design, designers rely on powerful tools like Figma[3]. Figma is a cloud-based design tool that has completely transformed the way that designers interact, experiment, and iterate on designs [4]. Figma, which Dylan Field and Evan Wallace founded in 2016, has become well-known in the design community thanks to its cutting-edge functionality and intuitive user interface. In contrast to conventional design software that functions as stand-alone programs, Figma can be accessed fully through a web browser, which makes it platform-neutral and allows team members to collaborate easily from any place [5].

Though Figma comes with a robust feature set right out of the box, there is still room to grow and modify the platform to better meet the particular requirements and tastes of different designers and design teams. This is when creating unique

Figma plugins becomes useful[6]. Using Figma plugins, designers can automate repetitive activities, add new features, and integrate the platform with external data sources to improve its functionality [7].

When designing user interfaces, wireframes, layouts, and visual components are often the result of extensive time and effort invested in the process. It is the responsibility of designers to translate abstract ideas and user requirements into concrete solutions that satisfy both practical and aesthetic standards. But this procedure can be time-consuming and labor-intensive, particularly for intricate projects or design iterations [8].

A further level of difficulty for designers is the increasing desire for dynamic and individualized user experiences. Designers must produce interfaces that are not only aesthetically pleasing but also responsive and flexible since users expect interfaces to adjust to their preferences, habits, and situations [9].

With the increased focus on AI approaches in recent years, they emerged as a logical choice for automation, particularly when taking into account the human aspects associated with UI development and analysis.[10] Thanks to advancements in artificial intelligence (AI), Developers can now include a range of AI functionalities into user-facing systems [11].

By examining user behavior, tastes, and demographics, AI can also be a significant factor in personalizing user experiences by enabling interfaces to be tailored to specific users. Designers can produce dynamic and adaptable interfaces that change in real-time in response to user interactions by utilizing machine learning techniques. For instance, multimedia platforms can modify content suggestions based on user interests and engagement metrics, while e-commerce websites can use AI to recommend products based on a user's browsing history and purchase activity [12].

1.2 Contribution

This thesis project presents a novel method to design assistance by merging OpenAI's[13] ChatGPT model[14] with a Django server API, in addition to developing a bespoke Figma plugin driven by AI. The primary contribution lies in the domain of information engineering, providing a tool that helps designers understand the structure and components of UI pages based on project descriptions, thus facilitating informed design decisions. Through this interface, designers can enter project descriptions into the Figma plugin. The Django server API with ChatGPT then processes these descriptions to produce design recommendations. This partnership between ChatGPT, the Django[15] server API, and the Figma plugin gives designers a cutting-edge approach to AI-powered UI development and design advice. We aim to make it easier for designers to start a project by helping them find the pages

and user interface elements that are specific to their project. This will cut down on the amount of time it takes to start design work.

1.3 Organization of the thesis

The thesis explores the intersection of design and technology through a series of focused chapters.

- **Chapter 2: Related Work:** This chapter comprehensively reviews existing research, literature, and recent advancements relevant to the thesis topic. It critically analyzes the use of Figma, the integration of Artificial Intelligence (AI) in design workflows, and other pertinent technologies to provide a strong theoretical foundation for the project.
- **Chapter 3: Architecture:** This section delves into the technical details of the thesis project's architecture. It outlines the design choices and their rationale, including the implementation of the Django API server, the integration of AI technologies, and the development of a custom Figma plugin.
- **Chapter 4: Implementation:** This chapter details the meticulous implementation of the solution. It provides a comprehensive account of the functionalities of the Figma plugin, the Django API server, and its interaction with ChatGPT. Additionally, it discusses the specific technologies employed, the challenges encountered during development, and the key features incorporated into the project.
- **Chapter 5: Results and Discussion:** This chapter presents the findings from the implemented solution. It includes an analysis of the cost associated with using the Figma plugin and showcases the UI designs generated by the plugin. The chapter discusses how these results demonstrate the effectiveness of the proposed system in automating UI design, contributing to the overall understanding of AI-driven design processes.
- **Chapter 6: Future Work:** The thesis concludes with a forward-looking perspective in Chapter 6: Future Work. Here, the chapter outlines potential avenues for further development and improvement of the project. This might include refining the AI algorithms, expanding the capabilities of the Figma plugin, or exploring new applications or integrations for the technology.

Chapter 2

Related Works

2.1 The Significance of Artificial Intelligence

Artificial Intelligence (AI) has become a revolutionary force in the rapidly changing technology landscape, transforming economies, cultures, and industries globally. Fundamentally, artificial intelligence (AI) is the creation of computer systems that can carry out operations that usually demand for human intelligence, like comprehending natural language, identifying patterns, coming to judgments, and picking up knowledge from past mistakes.

2.1.1 The Importance of AI

The potential of AI to enhance human talents, spur creativity, and resolve challenging issues in a variety of fields makes it significant. Organizations may increase productivity, simplify processes, and seize new chances for expansion and development by utilizing AI.

2.1.2 Key Components of AI

Artificial Intelligence (AI) comprises a wide range of technologies, approaches, and strategies, each with specific uses and functions. Among the essential elements of AI are:

- **Machine Learning:** Without explicit programming, computers can learn from data, spot patterns, and make judgments or predictions thanks to machine learning algorithms. Common methods in machine learning include reinforcement learning, supervised learning, and unsupervised learning.
- **Deep Learning:** Artificial neural networks with numerous layers of abstraction are trained to do complicated tasks like speech recognition, natural

language processing, and image recognition. Deep learning is a subset of machine learning.

- **Natural Language Processing (NLP):** The goal of natural language processing (NLP) is to make it possible for computers to comprehend, interpret, and produce meaningful, contextually relevant human language. NLP is used in a variety of applications, such as chatbots, virtual assistants, sentiment analysis, and language translation.
- **Computer Vision:** Computers can comprehend the content of visual data, such as photos and movies, and extract information from them thanks to computer vision algorithms. Computer vision applications include image classification, object identification, and facial recognition.
- **Robotics:** Robotics combines artificial intelligence (AI) algorithms with physical machinery to allow for intelligent decision-making and autonomous operation. Applications for robotics include driverless vehicles, industrial automation, healthcare, and more.

2.1.3 The Rise of AI in Today's World

There are various reasons why AI is becoming more and more important in today's world:

- **Advancements in Computing Power:** Due to Moore's Law and the development of hardware technologies like GPUs and TPUs, computational power has increased exponentially, making it possible to train large-scale AI models and run sophisticated AI algorithms in real time.
- **Availability of Big Data:** Massive volumes of data have been made available for training AI models and deriving insightful information due to the growth of digital data collected from multiple sources, including as social media, sensors, and Internet of Things devices.
- **Algorithmic Innovations:** Advances in AI algorithms, especially in the fields of reinforcement learning and deep learning, have resulted in notable gains in AI capabilities and performance.
- **Industry Adoption:** AI is being used more and more by a variety of industries, including healthcare, banking, retail, manufacturing, and transportation, to spur innovation, increase productivity, and obtain a competitive edge.
- **Ethical and Societal Considerations:** As AI becomes more pervasive in our daily lives, ethical and societal considerations surrounding issues such as bias,

privacy, transparency, and accountability have come to the forefront, prompting discussions and debates on responsible AI development and deployment.

2.1.4 Conclusion

In conclusion, artificial intelligence offers previously unheard-of possibilities for advancement and change, thereby bringing about a paradigm shift in the way we view and use technology. Through utilizing AI, we can open up new avenues, find solutions to challenging issues, and build a more intelligent, effective, and inclusive future. We will go into more detail on AI in the parts that follow, paying particular attention to generative AI and how it affects creativity and innovation.

2.2 Generative AI

Generative modeling artificial intelligence (GAI) is an unsupervised or partially supervised machine learning framework, which generates manmade relics via the use of statistics, probabilities etc[16]. The two main generative AIs recognized by the body of literature are Generative Adversarial Networks (GAN) and Generative Pre-trained Transformers (GPT) [17][16].

In generative AI, Generative Pre-trained Transformer (GPT) models are a major breakthrough. These models perform exceptionally well in natural language processing (NLP) tasks, exhibiting the capacity to understand and produce text that is similar to that of a human being in a variety of languages [18]. Through the use of massive volumes of publicly accessible digital content data, GPT models are able to produce imaginative compositions that vary in length from brief paragraphs to lengthy research articles [18]. The recent development of Generative Pre-trained Transformer-3 (GPT-3) represents a milestone in the field. With a staggering 175 billion parameters, GPT-3 exhibits enhanced task-agnostic capabilities, striving to rival previous state-of-the-art fine-tuning techniques [19]. Following the popularity of GPT-3, GPT-3.5 was released, expanding on the potential of generative AI by utilizing new developments in model design and training techniques to enhance effectiveness and performance.

GPT-3.5 is now the fundamental NLP engine behind the recently created language model ChatGPT, which has garnered interest across a number of domains [20]. In addition, progress has resulted in the creation of GPT-4, an enhanced version that can handle inputs in the form of text and images [21].

2.2.1 ChatGPT

What is ChatGPT

ChatGPT is an intelligent conversational robot developed by OpenAI[22] that can respond in-depth to a prompt by following instructions [23]. According to the official statement, ChatGPT has demonstrated strong capabilities on a variety of language understanding and generating tasks, including multilingual machine translation, code debugging, sentence composing, acknowledging errors, and even rejecting requests that aren't acceptable. ChatGPT is a better chatbot than its predecessors since it can recall past user comments, facilitating ongoing conversations [24].

Advantages and Strengths of ChatGPT

- The most natural sensation while utilizing ChatGPT is its ability to recognize the user's intent through text prompts (up to 2.5 million tokens) and picture cues (further study is needed), and to produce a variety of text forms during interactive chat [23].
- ChatGPT has surpassed human performance in a number of text creation activities, including question-answering, offering advice, summarizing, and refining documents [25]. When responding to a question, for instance, it will not only provide an accurate response; the generated response text will also disclose the reasoning behind the response and even continually modify and amend the answer in accordance with the user's instructions [23].
- ChatGPT possesses strong reasoning skills, particularly when it comes to responding to inquiries on science, knowledge, and intricate logic[26].

Disadvantages and limitations of ChatGPT

- ChatGPT can combine a variety of resources to produce fluid responses, but these responses frequently contain factual inaccuracies (a phenomenon known as "hallucination problem"). These factual mistakes could have resulted from noise and mistakes in the training set [27].
- The GPT series models employ hundreds of billions of parameters and a vast amount of data, but they just apply the most basic information processing technique known to humankind—predicting a sentence's next potential word [23].

OpenAI API

Large language models can be fine-tuned for particular purposes by users using services offered by numerous well-known generative model providers, such as OpenAI. An API is provided by OpenAI to adjust its fundamental GPT-3, GPT-3.5 and GPT-4 models. This enables users to modify transformer models, such as GPT-3, to enhance their performance in particular applications [28].

With its user-friendly interface, the OpenAI API simplifies the complex process of training and optimizing large-scale models, thereby revolutionizing the accessibility of advanced language models. Using only an OpenAI API key, developers can easily access cutting-edge AI features without needing a great deal of experience with machine learning or natural language processing.

By utilizing the extensive knowledge and contextual understanding built into the underlying language models, the OpenAI API enables developers to improve a multitude of elements of their apps. This makes it possible to develop more intelligent, dynamic, and interactive applications in a variety of fields, from creating sophisticated chatbots and virtual assistants to producing creative content and document summaries.

The OpenAI API's capability to comprehend and produce writing that resembles that of a human being is one of its main features. The models are able to generate responses that are both coherent and contextually relevant since they have been trained on an enormous amount of different data. Because of its ability to comprehend natural language, developers may create programs that can have meaningful conversations, comprehend user inquiries, and produce excellent prose that closely resembles information written by humans.

Additionally, the OpenAI API makes multilingual apps easier to create by supporting a wide range of languages and enabling programmers to create applications that can comprehend and produce content in multiple languages. This makes the API useful for businesses and organizations functioning in a multilingual environment by creating chances for cross-cultural communication, machine translation, and worldwide content creation.

The OpenAI API has several difficulties and things to keep in mind, much like any new technology. It's important to give significant consideration to ethical issues like bias in language models and the appropriate application of AI. Furthermore, it is crucial to guarantee data security and privacy when utilizing the API. In order to safeguard user information and uphold confidentiality, developers need to be aware of potential hazards and take the necessary precautions.

To sum up, the OpenAI API is a significant advancement in AI technology that enables programmers to incorporate cutting-edge language models into their applications. Developers can produce more complex and intelligent software that improves user experiences and opens up new opportunities in a variety of sectors

by utilizing the API's features. To guarantee the ethical and advantageous usage of the OpenAI API in the larger AI ecosystem, it is crucial to address privacy issues, ethical issues, and other difficulties.

2.3 Figma: A Comprehensive Overview

Figma has emerged as a leading collaborative interface design tool, revolutionizing the way designers and teams create digital products. This section provides a comprehensive overview of Figma, encompassing its features, functionalities, advantages, and significance in contemporary design workflows.

2.3.1 Introduction to Figma

Figma is a cloud-based design platform that enables seamless collaboration and real-time editing among designers and stakeholders. Introduced in 2016, Figma has quickly become well-known due to its versatility, accessibility, and ability to streamline the design process [5].

2.3.2 Key Features and Functionalities

Collaborative Editing

Multiple users can collaborate in real-time on the same design file with Figma thanks to its collaborative editing capability. In contrast to conventional design approaches, this promotes collaboration, improves communication, and gets rid of version control problems.

Cloud-Based Storage

With Figma, design files are kept on the cloud and can be accessed from any internet-connected device. This makes sharing and collaborating easier, making it possible for designers to work remotely and across time zones with ease.

Responsive Design

Because Figma adheres to the principles of responsive design, designers can create layouts that adjust to different screen sizes and devices. In order to guarantee consistency and usability across platforms, designers may quickly preview and test their designs across various breakpoints.

Prototyping and Interaction

Designers can produce interactive prototypes with animations, transitions, and user flows with Figma's powerful prototyping features. This facilitates feedback and iteration by allowing stakeholders to experience the design in a real-world setting.

Design Systems and Component Libraries

Reusable component libraries and design systems can be created and managed with Figma. By establishing standardized design patterns, styles, and components, designers may enhance productivity and preserve design coherence over several projects.

2.3.3 Figma Plugin Ecosystem

Figma's plugin ecosystem extends its core functionality by allowing users to enhance their design workflows with custom tools and integrations. Plugins are third-party add-ons created by groups or the community that provide a variety of features and functionality suited to certain design requirements.

Integration with External Tools

Plugins facilitate the smooth integration of external tools and services, giving designers the ability to incorporate content, data, and assets straight into Figma projects. Processes are streamlined, and copy-pasting and manual file transfers are no longer necessary.

Automation and Productivity

Plugins automate repetitive tasks and streamline common design processes, boosting productivity and efficiency. From batch resizing images to generating placeholder text, plugins offer a plethora of time-saving functionalities that expedite the design workflow.

Customization and Extensibility

Users can tailor their design environment to meet their unique needs by using Figma plugins. Plugins enable designers to customize Figma to their own processes, whether it's by adding new features, making bespoke tools, or interacting with proprietary systems.

2.3.4 Advantages of Using Figma

Accessibility and Cross-Platform Compatibility

Because Figma is cloud-based, it is accessible from a variety of devices and operating systems, including Windows, macOS, and Linux. This gets rid of compatibility problems and makes it possible for teams to work together on any platform of their choice.

Real-Time Collaboration

Real-time collaboration eliminates the need for long email exchanges and planned meetings by promoting effective communication and teamwork. The design process can be accelerated by designers' ability to collaborate on decisions, iterate quickly, and offer immediate feedback.

Version History and Revision Control

Design files are automatically versioned by Figma, enabling users to track changes, examine prior iterations, and go back to older versions as needed. In addition to guaranteeing responsibility, this offers protection against unintentional modifications or data loss.

2.3.5 Significance in Contemporary Design Workflows

For design teams in a range of sectors, from startups to major corporations, Figma has emerged as an essential tool. Its emphasis on design systems, adaptable design capabilities, and collaborative features have revolutionized the conception, development, and maintenance of digital goods.

2.3.6 Limitations and Considerations

Although Figma has many benefits, there are some restrictions and things to keep in mind. These could include worries about privacy and data security, reliance on internet access, and the learning curve involved in becoming proficient with the technology. Additionally, while utilizing Figma for private or sensitive projects, companies need to make sure that they are in accordance with all applicable laws and guidelines.

2.4 Uizard

Uizard is a cutting-edge platform that uses artificial intelligence (AI) to make the user interface (UI) design process more efficient. Tony Beltramelli and Andreas Høye founded Uizard in 2017 with the goal of democratizing design by enabling individuals and teams without a lot of design or coding skills to use it [29].

2.4.1 Introduction to Uizard

Using Uizard, a cloud-based design platform, users may create interactive prototypes and fully functional digital interfaces from hand-drawn sketches or wireframes. Uizard uses state-of-the-art AI technology to automate the time-consuming and laborious parts of the design process, freeing up designers to concentrate on originality and creativity.

2.4.2 Key Features

Instant UI Generation

Users only need to draw UI designs on paper or digitally with Uizard to swiftly generate them. The drawings are analyzed by the platform's AI algorithms, which then automatically transform them into high-fidelity user interface designs with interactive features. Furthermore, With the help of Uizard's robust AI features, you can create UI designs from text prompts and change them using a simple drag-and-drop editor.

Real-Time Collaboration

Team members can collaborate easily with Uizard and work together in real-time on UI designs. No matter where they are, designers can collaborate to iterate on designs, share their work with others, and get feedback.

Customization and Iteration

The user-friendly interface of Uizard allows users to modify and refine their creations. The platform provides a number of tools and capabilities that may be used to change layouts, experiment with new designs and themes, and edit design aspects.

Integration with Design Tools

Uizard's seamless integration with well-known design programs like Adobe XD, Figma, and Sketch makes it simple for users to import and export their designs.

Due to its interoperability, Uizard may be integrated into users' current design workflows, hence increasing the platform's flexibility and scalability.

2.4.3 Advantages of Using Uizard

Accelerated Design Process

Compared to traditional approaches, Uizard allows designers to produce high-fidelity prototypes and digital interfaces in a fraction of the time by automating repetitive operations and streamlining the design process. As a result, the design iteration cycle is sped up and the time to market for digital products is accelerated.

Accessibility and Ease of Use

Because of its simple features and intuitive interface, Uizard is usable by designers with varying degrees of technical competence. Designers may easily realize their ideas with the platform's drag-and-drop interface, pre-built templates, and AI-driven support tools.

Seamless Integration

Prototypes can be easily exported to other applications or existing designs can be effortlessly imported into Uizard because to its seamless integration with common design tools and platforms. Designers can use Uizard's capabilities in their preferred design workflow, be it Sketch, Adobe XD, or Figma.

2.4.4 Limitations and Considerations

Accuracy of Code Generation

Although Uizard's sketch-to-code capability is amazing, the resulting code's accuracy could vary based on the design's intricacy and the user's level of detail.

Dependency on AI Algorithms

Because Uizard depends on AI and machine learning algorithms, the platform's functionality and performance could be impacted by the caliber of its training data and the continuous improvement of its algorithms.

2.5 Galileo AI

Galileo AI is an all-inclusive AI-powered platform designed with designers in mind. It provides a variety of tools and features with the goal of improving the

design process, encouraging teamwork, and stimulating creativity. An extensive introduction to Galileo AI is given in this section, along with an emphasis on its salient characteristics, benefits, and relevance to the creative and design industries.

2.5.1 Introduction to Galileo AI

Galileo AI is an AI-powered design platform that helps designers with different parts of the design workflow by utilizing machine learning algorithms and natural language processing (NLP) approaches. Galileo AI is a set of tools and capabilities that was created to help designers overcome common issues. These tools and capabilities are intended to spark creativity, expedite the design process, and make teamwork easier.

2.5.2 Key Features and Functionalities

AI-Powered Design Assistance

Galileo AI helps users with intelligent design by employing cutting-edge AI techniques. This includes tools that make it easier for designers to explore creative options and make well-informed decisions, like automatic layout suggestions, color palette selections, and typography guidance.

Contextual Design Insights

The platform provides pertinent design insights and recommendations by analyzing contextual data and user input. Galileo AI can provide customized recommendations and optimizations that are suited to the unique requirements of each user by comprehending the project context, target audience, and design objectives.

Collaboration and Feedback

Collaboration capabilities in Galileo AI facilitate easy communication and feedback sharing amongst team members. Within the platform, designers may collaborate to iterate on designs, share their work, and get input from stakeholders, promoting a more transparent and iterative design process.

Integration with Design Tools

Popular design tools and software are easily integrated with Galileo AI, enabling users to take use of its AI-powered features from within their current workflows. Using only one program to access Galileo AI's functionality, designers can utilize its features in Sketch, Figma, or the Adobe Creative Suite.

2.5.3 Advantages of Using Galileo AI

Enhanced Design Efficiency

Galileo AI makes design work easier and more productive for designers by automating tedious chores, offering insightful design recommendations, and promoting teamwork. Instead of spending time on tedious manual labor and administrative duties, designers may concentrate their time and energy on innovative ideation and problem-solving.

Data-Driven Decision Making

Galileo AI gives designers the ability to make well-informed decisions at every stage of the design process by utilizing data analytics and AI-driven insights. Through the analysis of user behavior, market trends, and design performance data, the platform assists designers in identifying areas that may be optimized and improved.

Improved Collaboration and Communication

Galileo AI encourages cooperation and communication among design teams with its tools for collaboration and feedback exchange. Regardless of distance or time zone differences, designers may collaborate on projects in real-time, share their work, and get comments from peers.

2.5.4 Limitations and Considerations

Learning Curve

Users who are used to other design tools or traditional design workflows may need some time to become used to Galileo AI's functionality and interface.

Algorithmic Bias

Galileo AI might have biases or limits as a result of the algorithms and training data it uses, just like any other AI-driven platform. It is imperative for designers to conduct a critical assessment of the comments and suggestions supplied by the platform, while considering variables such as cultural background, diversity, and inclusivity.

2.6 Mage (useMage.ai)

The Wasp team created Mage, a cutting-edge web application generator, to make full-stack web application development easier. With the help of the Wasp framework

and AI technology, Mage lets users give a concise description of the web application they want. A complete React-based web application codebase, fueled by Node.js, Prisma, and Wasp, is built and made accessible for download in a matter of minutes. An overview of Mage is given in this section, with emphasis on some of its salient characteristics.

2.6.1 Key Features and Functionalities

Automated Code Generation

With Mage, users may express in simple terms what kind of web application they want to construct. Users can create a fully functional codebase that meets their criteria by providing data about the app's functionality, branding choices, and authentication method. All required parts, written in React, Node.js, Prisma, and Wasp, are included in the generated code, including front-end interfaces, back-end functionality, and database interactions.

Customization Options

Users can alter the brand colors, degree of originality, and authentication mechanism, among other elements of their web application. Users are able to customize their application to match their own needs and vision by adding details like a name and description.

Integration with AI and ChatGPT

Mage uses artificial intelligence (AI) technologies, such as ChatGPT, to help users precisely describe the needs for their online applications. ChatGPT enables smooth communication between the user and the program by interpreting user input and making suggestions that are contextually relevant.

2.6.2 Example Use Case

For instance, a user who wants to build a social media site can use natural language to tell Mage what they need. They might list features like real-time communications, posting capabilities, and user profiles. After that, Mage creates the entire codebase for the required web application, including database schemas for storing user data, frontend interfaces for user profiles and publishing, and backend logic for managing user interactions.

2.6.3 Advantages

Speed and Efficiency

Mage drastically cuts down on the time and effort needed to construct full-stack web apps by automating the code generation process. In only a few minutes, users may move from a concept to code, speeding up the development process and facilitating quick prototyping and iteration.

Accessibility and User-Friendliness

Users with varying degrees of expertise can utilize Mage due to its user-friendly interface and guided workflow, even those with no prior coding experience. Because of the platform's natural language processing capabilities, customers may define their application needs in simple terms without using technical jargon, which streamlines the input process.

Scalability and Versatility

Because of Mage's scalable and adaptable output, users can develop web applications for a variety of industries and use cases. Whether creating content management systems, e-commerce platforms, or data visualization tools, Mage's adaptable architecture guarantees that the applications it generates may change to meet changing business demands and specifications.

2.6.4 Disadvantages

Limited Customization Options

While Mage does offer some customization, customers might discover that there aren't as many alternatives as if they were hand-coding a web application. Mage's automated technique might not offer enough flexibility for projects demanding highly particular design components or sophisticated functionalities.

Dependency on AI Accuracy

Mage's ability to comprehend user input and produce code is mostly dependent on AI algorithms. Therefore, the effectiveness of these techniques determines the produced codebase's correctness and dependability. Errors or less than ideal results in the generated code may result from inaccurate interpretations or misinterpretations of the user descriptions.

Learning Curve

Despite its goal of streamlining the web development process, Mage may still require some learning on the side of users, especially those who are not familiar with AI-driven development tools or the Wasp framework. To fully utilize Mage's capabilities and optimize the generated code for particular project requirements, training and familiarization may be necessary.

Limited Support for Complex Projects

Web applications that are extremely sophisticated or specialized may be too complicated for Mage's automated technique to manage. More manual involvement or customization may be needed for projects with complex business logic, sophisticated data processing needs, or unique architectural limitations than what Mage can offer.

Chapter 3

Architecture

3.1 Introduction

This chapter explores our project's architectural foundation, which acts as the solution's structural blueprint. With ChatGPT's help, we hope to create a dependable and expandable architecture that unifies the Django Python backend and the Figma plugin frontend, enabling the creation of user interfaces from project specifications.

3.2 Overall System Architecture

As illustrated in Figure 3.1 our system follows a client-server architecture, where Django Python powers the server and the TypeScript and React-developed Figma plugin acts as the client. The architecture facilitates a distinct division of responsibilities, allowing the frontend and backend components to function separately and communicate fluidly to achieve our project's goals.

3.3 Client-Side Architecture: Figma Plugin

The Figma plugin functions as the frontend element of our architecture, seamlessly integrating with the Figma environment to offer designers a seamless user experience. Based on the concepts of modularity, extensibility, and reusability, the plugin's architecture makes use of TypeScript and React to produce an engaging and dynamic user interface.

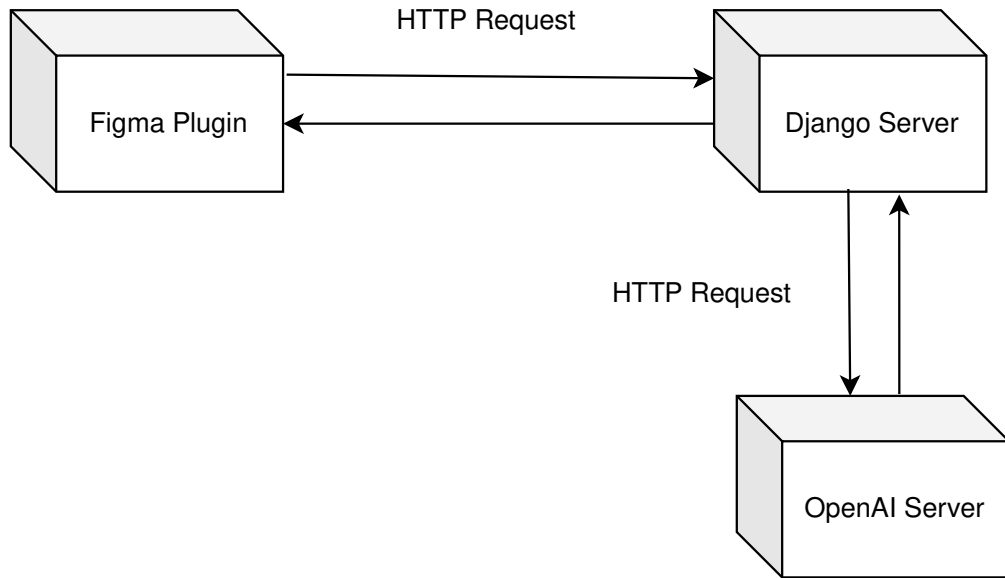


Figure 3.1: Client-server architecture

3.3.1 Component-Based Design

The user interface of the plugin is divided into reusable parts that each contain particular features and actions. With its easy expansion and maintenance, this modular design facilitates the gradual addition of new features and improvements.

3.3.2 State Management

Maintaining the Figma plugin's state is essential to making sure the user experience is responsive and seamless. User input, UI interactions, and asynchronous activities are managed by the application state using state management frameworks like Redux or React Context API. This facilitates fluid data flow and interaction by allowing the plugin to keep a consistent state across many components and panels.

3.3.3 Event-Driven Architecture

The plugin has an event-driven design, meaning that events are handled by event listeners in response to user inputs. This makes it possible for interactive and responsive behavior, guaranteeing a seamless user experience in the Figma environment.

3.3.4 Integration with Figma API

To access project data, modify UI elements, and engage with the Figma environment, the Figma plugin tightly interacts with the Figma API. When interacting with Figma documents, the Figma API offers a wide range of activities, such as obtaining project descriptions, generating and editing layers, and initiating UI modifications. The plugin uses libraries like `axios` or `fetch` to make API calls in order to interact with the Figma API through HTTP queries.

3.4 Server-Side Architecture: Django Python Backend

The server-side part of our design is the Django Python backend, which handles requests from the Figma plugin and uses ChatGPT to coordinate the UI development process. Leveraging the powerful features of the Django framework, the backend architecture is developed with scalability, stability, and maintainability in mind.

3.4.1 RESTful API Design

A RESTful API defined by the backend can be used to handle different tasks like collecting project descriptions, starting UI generation, and sending generated UI designs back to the Figma plugin. To maintain uniformity, readability, and simplicity of use, the API endpoints adhere to RESTful standards for resource name, HTTP method usage, and request/response formats.

3.4.2 Request Handling and Routing

Using URL patterns and view functions, incoming requests from the Figma plugin are forwarded to the relevant handler methods inside the Django application. These handler routines receive the requests, process the data entered, and initiate the appropriate actions (e.g., generating text with ChatGPT).

3.4.3 Integration with External Services

To improve its functionality and offer new features, the Django backend connects with outside services like ChatGPT. Through API calls supplied by the other services, this integration is made possible, enabling the backend to utilize ChatGPT's sophisticated AI capabilities for text synthesis based on project descriptions obtained from the Figma plugin.

3.4.4 Error Handling and Logging

To properly manage exceptions, errors, and unforeseen circumstances, the backend has strong error handling and logging features. To aid in troubleshooting and debugging during development and deployment, comprehensive logs are generated and helpful error messages are returned to the Figma plugin in error responses.

3.5 Technologies and Frameworks

- **TypeScript and React:** used to create the Figma front-end plugin, which offers a cutting-edge and effective framework for creating interactive user interfaces in the Figma environment.
- **Django Python:** used in the construction of the backend server, providing a stable and expandable framework for processing HTTP requests, organizing data, and carrying out business logic.

3.6 Conclusions

We have given a thorough rundown of our project's architecture, including client-side and server-side elements, in this chapter. We have established the framework for a scalable, modular, and extendable solution to the problem statement by utilizing a client-server architecture and the powers of TypeScript, React, and Django Python. We will explore the implementation specifics in the upcoming chapter, offering more information on how our architecture is put into effect.

Chapter 4

Implementation

4.1 Introduction

In this chapter, we transition from theoretical framework to a functional software solution. Our focus is on developing the backend Django Python server and the frontend Figma plugin, aiming to seamlessly integrate AI technologies for UI generation within the Figma environment. This chapter provides a comprehensive overview of the development process, detailing environment setup, frontend and backend implementation, testing, deployment, and other related tasks.

4.2 Setting Up the Development Environment

We will cover in this section the essential steps in creating the development environment both for frontend and backend development. Once established, it guarantees a seamless blending of resources and tools that are indispensable for efficient software development, during collaboration.

4.2.1 Version Control with Git

Version control is essential in modern software development for tracking changes, managing code efficiently, and maintaining project integrity. For this thesis project, I utilize Git as the version control system. Git provides a robust and flexible framework for preserving project histories, enabling a seamless development workflow.

4.2.2 GitHub Repository

To leverage additional features and centralize my project codebase, I hosted my Git repository on GitHub. GitHub serves as a central hub for organizing project

milestones, tracking issues, and storing project files. Utilizing GitHub's features such as branches, commits, and pull requests enhances project management and organization, fostering accountability and transparency in the development process.

4.2.3 Integrated Development Environment (IDE)

I use Visual Studio Code (VS Code) as my primary Integrated Development Environment (IDE) for coding and development tasks. VS Code offers a wide range of features including code editing, debugging, version control integration, and extensibility through plugins and extensions. Its extensive ecosystem and user-friendly interface provide the necessary tools to write, test, and debug code efficiently, contributing to the overall success of this thesis project.

4.2.4 Frontend Development Setup

Setting up the frontend development environment for creating the Figma plugin requires several steps to ensure seamless integration with the Figma desktop application. Here is a detailed explanation:

- **Login and Access Plugin Development Section:** After installing the Figma desktop application, I signed into my Figma account and accessed the "Development" submenu within the "Plugins" section, typically found in the left sidebar menu.
- **Create a New Plugin:** I initiated the creation of a new plugin by clicking the "New Plugin" button within the "Development" submenu. I was prompted to name the plugin and select a project type. Selecting the "Figma" project type aligned with my goal to develop a Figma plugin.
- **Choose Figma Design:** Upon selecting the "Figma" project type, I chose a design template for the plugin. Figma offers various design templates tailored for different functionalities. I selected the "Figma Design" template, ideal for UI-focused plugins.
- **Generate Plugin Folder:** Upon completion of the plugin creation process, Figma automatically generated a folder on my local machine. This folder contained boilerplate code and template files essential for plugin development. I opened this folder in Visual Studio Code, my preferred code editor, to begin development.
- **Add TypeScript and React:** To enhance type safety and development efficiency, I utilized TypeScript as the foundation for the Figma plugin template. Additionally, I incorporated React, a popular JavaScript library for building

user interfaces. Integrating React into the Figma plugin enabled dynamic and interactive UI components, benefiting from its extensive range of components and state management features.

This setup, tailored to my work style and project requirements, establishes a solid foundation for effective frontend development, allowing me to create innovative and user-friendly plugins for the Figma platform.

4.2.5 Backend Development Setup

Setting up the backend development environment for my Django Python server involved several steps to ensure seamless integration and efficient implementation. Below is a detailed explanation of the setup process:

- **Python Installation:** The first step was to install Python on my local machine, as Django development is done in Python. I downloaded and installed the latest version of Python from the official Python website to ensure compatibility with Django and other dependencies.
- **Virtual Environment Setup:** To keep my Python project isolated and manage dependencies effectively, I set up a virtual environment using the 'venv' module. This allowed me to install packages specific to my project without affecting the global Python environment.
- **Django Installation:** With the virtual environment activated, I installed Django using pip, the Python package manager. This command installed the latest version of Django and its dependencies within the virtual environment, ensuring a clean and isolated development setup.
- **Project Initialization:** Using Django's command-line interface (CLI), I initialized a new Django project by running the `django-admin startproject` command followed by the project name. This command generated the necessary project files and directory structure to set up the initial scaffolding for the Django application.
- **App Creation:** Within the Django project, I created one or more Django apps to encapsulate specific functionalities or features. Each app is a modular component of the larger project, containing models, views, templates, and other resources relevant to its domain.
- **Configuration Settings:** I configured various settings in the Django project, including middleware, URL routing, static file handling, and database settings. These configurations were specified in the `settings.py` file within the project directory, allowing customization and flexibility as needed.

- **URL Routing:** Using Django's URL routing mechanism, I defined URL patterns and linked them to views or viewsets responsible for handling incoming HTTP requests. This facilitated the mapping of URLs to specific endpoints within the Django application, enabling the creation of RESTful API endpoints.
- **Integration with External Services:** As part of the backend setup, I integrated external services such as ChatGPT with the Django application to enable seamless communication and collaboration between the frontend and backend components. This involved setting up data serialization and deserialization processes, authentication methods, and API endpoints to facilitate data exchange between the Django server and the Figma plugin.

Following these steps and adhering to Django best practices, I established a solid foundation for building and implementing the Django Python server. This setup enabled effective backend development for my thesis project, facilitating smooth integration with external services and reliable communication with frontend components.

4.3 Frontend Implementation: Figma Plugin

In this section, I provide a detailed account of how I implemented the frontend for my Figma plugin using React. The frontend consists of two main routes: the homepage route ("/") and the process route ("/process"). The following code snippet demonstrates how I configured these routes using React Router:

```
import React from 'react';
import {
  MemoryRouter,
  Routes,
  Route,
} from "react-router-dom";
import './styles/ui.css';
import Homepage from './Homepage';
import Process from './Process';
import Layout from './Layout';

function App() {
  return (
    <MemoryRouter>
      <Routes>
        <Route path="/" Component={Layout} >
```



```
        <Route path="/" Component={Homepage} />
        <Route path="/process" Component={Process} />
        <Route path="/project-description" Component={ProjectDescription} />
    </Route>
  </Routes>
</MemoryRouter>
);
}

export default App;
```

The above code sets up three routes using React Router: one for the homepage ("/"), one for the process page ("/process") and another for the project description page ("/project-description"). Each route is associated with a different component: `HomePage`, `Process` and `ProjectDescription`, respectively.

4.3.1 Homepage Route

The homepage route ("/") serves as the landing page for the Figma plugin. It provides users with an overview of the plugin's features and capabilities. As shown in Figure 4.1, the homepage acts as an introductory page, guiding users through the initial steps required to use the plugin effectively. Key features include:

- **OpenAI API Key:** A critical step for using ChatGPT is obtaining an OpenAI API key. Users are prompted to enter this key to enable the plugin's functionality.
- **Billing Setup:** Users must ensure that billing is enabled in their OpenAI account to cover the costs associated with API calls.
- **Navigation Buttons:** The "Let's Start" button redirects users to the Process route, while the "Give Feedback" button links to a Google Form for collecting user feedback.

4.3.2 Process Route

Users are guided through the process of creating user interfaces with AI technology by the process route ("/process"). Users can input their OpenAI API key and project description using an organized, step-by-step interface, as demonstrated in the image 4.2. Furthermore, the route is built to handle common issues that could occur during communication between the Django server and the OpenAI API, giving users helpful feedback to aid in troubleshooting.

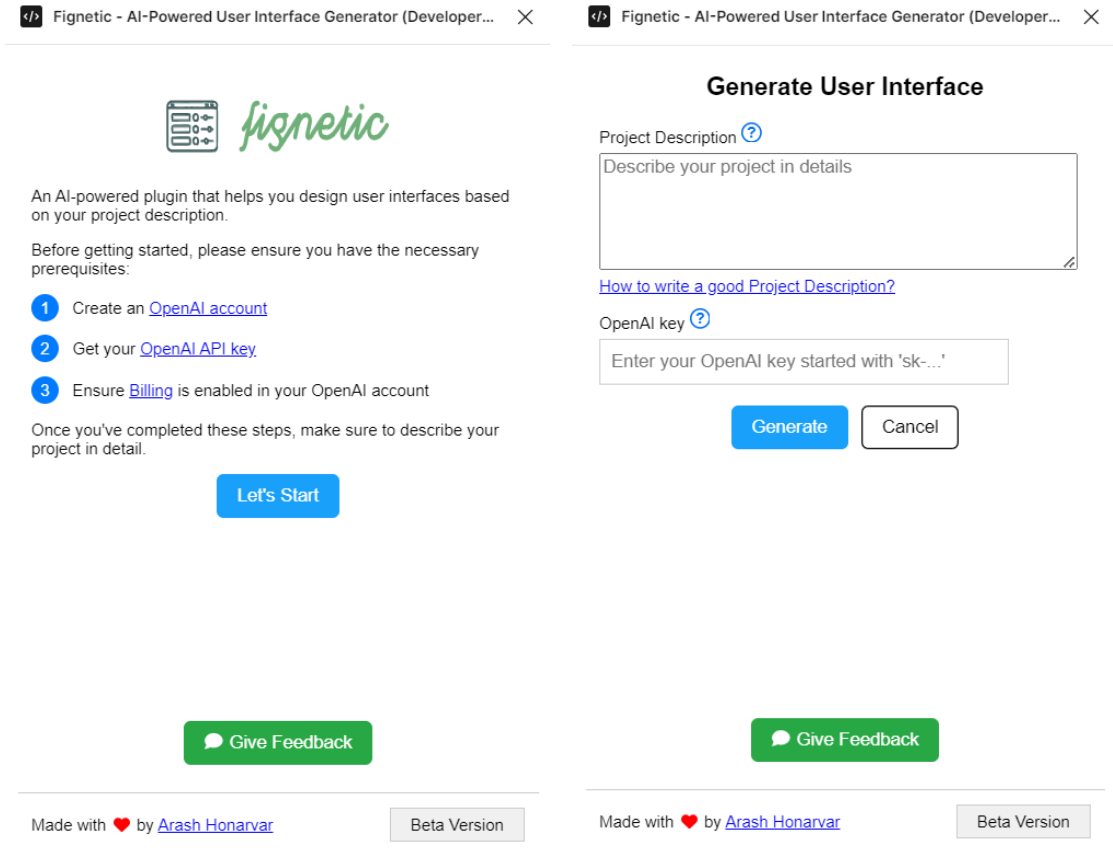


Figure 4.1: Home Page screen when the plugin has been launched

Figure 4.2: Process screen for generating appropriate user interface

4.3.3 Project Description Route

The application includes a dedicated page to assist users in writing a clear and comprehensive project description as demonstrated in the image 4.3. This page, accessible via a link labeled "How to write a good Project Description?" below the project description text box, in the process page as it shown in the image 4.2. It provides essential guidelines to ensure users can maximize the benefits of the plugin. The page titled "Crafting an Effective Project Description" outlines three key areas which is described in the chapter 4.5. The page concludes with a call-to-action button, encouraging users to start generating their UI components based on the crafted description.

4.3.4 Managing Plugin Logic with 'controller.ts'

The 'controller.ts' file functions as the Figma plugin's main logic handler, coordinating user input, server connectivity, and plugin functionality. It functions mostly through event handling, which is an essential component of Figma plugin development.

Event-Driven Architecture

The use of event-driven architecture in the 'controller.ts' file is essential to its functionality. The 'figma.ui.onmessage' event handler, which listens for incoming messages from the plugin UI or other sources, is essential to this design. These messages typically contain instructions or data payloads that trigger specific actions within the plugin.

```
figma.ui.onmessage = async (msg) => {
  // Handle incoming messages based on their type
  switch (msg.type) {
    case 'start-process':
      // Perform actions based on the message type
      // ...
      break;
    default:
      // Handle other message types
      // ...
      break;
  }
};
```

Message Handling

The message is processed by the 'controller.ts' file based on its kind upon receipt. This may entail a variety of duties, including starting a process, resolving issues, or changing the user interface. For example, the controller starts the process by sending the OpenAI API key and project description to the server for analysis in response to a "start-process" message.

```
case 'start-process':
  let projectDescription = msg.description;
  let openAIKey = msg.openAIKey;

  // Start the process by sending the project description
  // and OpenAI API key to the server
```

```
await startProcess(projectDescription, openAIKey);
// ...
break;

async function startProcess(projectDescription, openAIKey) {
  try {
    // Send a POST request to the server's process API endpoint
    const response = await fetch(api_url_prefix + 'api/process/', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        projectDescription: projectDescription,
        openAIKey: openAIKey
      }),
    });

    // Check if the response status is OK (HTTP status code 200)
    if (response.ok) {
      return response.json();
    } else {
      // If the response status is not OK
      await response.json().then((data) => {
        figma.ui.postMessage({
          type: 'process-failed',
          message: data.error,
        });
      });
    }
  } catch (error) {
    // If an error occurs during the request
    figma.ui.postMessage({
      type: 'process-failed',
      message: 'Process failed. Server error.',
    });
  }
}
```

Manipulating Figma Documents for UI Generation

The ability of the 'controller.ts' file to manipulate Figma documents programmatically by utilizing the Figma API is one of its key features. With this feature, the Figma plugin may create user interfaces (UIs) inside the Figma environment that are dynamically generated and customized to meet the unique needs of every project.

The plugin may create UI elements dynamically within the Figma document, including text layers, shapes, photos, and more, by having access to the Figma API. For example, the plugin might generate buttons for navigation, input fields for user input, text layers to show project details, and other UI elements necessary for the project's interface.

Moreover, UI elements can be extensively customized and adjusted using the Figma API in accordance with project requirements and design preferences. The position, size, color, font, and styling of UI elements are among the many properties that developers can manage to guarantee uniformity and adherence to design guidelines.

The 'controller.ts' file allows designers and developers to iterate on UI designs immediately within the familiar Figma environment by integrating smoothly with the Figma document editing workflow. By doing away with the need for additional design tools or manual modifications, this integration improves productivity, speeds up collaboration, and streamlines the design process.

The supplied createText and createRectangle functions serve as an example of how the plugin may be used to programmatically create text layers and rectangles inside of a Figma document. The foundation of the plugin's ability to create dynamic, adaptable user interfaces (UIs) that meet project needs is formed by these and related functions.

```
// Function to create a text node
function createText(parent, x, y, content,
fontSize, type = 'normal') {
  const text = figma.createText();
  text.x = x;
  text.y = y;

  // Check if text length is more than 60 characters
  if (content.length > 60) {
    text.characters = content;
    text.fontSize = fontSize;
    // Allow the text to resize vertically
    text.textAutoResize = "HEIGHT";
  }
}
```

```
    // Set a maximum width of 300 pixels
    text.resize(300, text.height);
  } else {
    text.characters = content;
    text.fontSize = fontSize;
  }

  // Make text bold
  if (type === 'bold') {
    text.fontName = { family: "Inter", style: "Bold" };
  }

  parent.appendChild(text);

  return text;
}

// Function to create a rectangle (representing a navigation
//bar, content area, and footer)
function createRectangle(parent, x, y, width,
height, color, type) {
  const rect = figma.createRectangle();
  rect.x = x;
  rect.y = y;
  rect.resize(width, height);
  rect.fills = [{
    type: 'SOLID',
    color: { r: color[0], g: color[1], b: color[2] }
  }];

  // Set name based on type for identification
  if (type) {
    rect.name = type;
  }

  parent.appendChild(rect);
  return rect;
}
```

4.4 Backend Implementation: Django Python Server

The Backend Implementation of the Figma plugin relies on a Django Python Server to manage the application's logic and handle communication with external services such as ChatGPT. This server-side component serves as the backbone of the plugin, coordinating various tasks and facilitating seamless interaction between different components.

4.4.1 REST API for Managing the Application

The main interface for controlling the functionality of the program is the REST API exposed by the Django Python Server. With the help of this API, clients—like the Figma plugin—can interact with the server and carry out particular tasks, like starting the UI creation process.

4.4.2 Process Endpoint for ChatGPT Communication

The 'process' endpoint, which is intended to handle requests related to UI generation using ChatGPT, is one of the main features of the REST API. The project description and OpenAI key are extracted by the server from the request payload when it receives a POST request to the 'process' endpoint.

4.4.3 Interaction with ChatGPT

After the required data has been retrieved, the Django server contacts ChatGPT to provide user interface designs based on the project description that has been supplied. In order to retrieve the necessary UI elements, the communication procedure usually entails sending queries to ChatGPT and processing the results.

4.4.4 Template-based UI Generation

The server uses a template-based technique to speed up the UI generation process. A pre-made JSON template is created, detailing the UI design's hierarchy down to the page names and related components. The template has different user interface (UI) components in each part, including buttons, lists, forms, navigation bars, and more.

4.4.5 Instruction and Rules for ChatGPT

Furthermore, the server incorporates particular guidelines and directives into the conversation with ChatGPT. Following predetermined criteria and design concepts, ChatGPT is guided by these instructions while generating UI designs using the template that has been provided.

Sample Workflow:

1. The Figma plugin sends a POST request to the Django server's `process` endpoint, providing the project description and OpenAI key.
2. The Django server initiates communication with ChatGPT, requesting UI design generation based on the provided project description.
3. ChatGPT responds with UI design suggestions, which are processed by the Django server.
4. The Django server constructs a JSON template based on the received suggestions, specifying the structure of the UI design.
5. The Django server sends additional instructions and rules to ChatGPT, guiding the UI generation process.
6. ChatGPT generates a JSON-based UI design according to the provided template and instructions, which is then returned to the Django server.
7. The Django server processes the final JSON-based UI design and sends it to the Figma plugin.
8. The Figma plugin uses the Figma API to render the designs within the Figma environment based on the received JSON.

4.4.6 Interaction with OpenAI API for Design Generation

In this section, I describe the interaction with the OpenAI API to generate the necessary design elements for a specific project in Figma. This interaction involved crafting specific prompts to describe the required pages and user interface elements, and then generating a JSON structure containing all the information needed for creating these designs in Figma.

Prompt Crafting and API Interaction

To effectively communicate with the OpenAI API and obtain the necessary design elements, I used two main prompts. The first prompt was to describe all the pages required for the project along with their user interface elements. The second

prompt was to generate a JSON containing all the user interface elements following specific rules.

First Prompt: Describing Pages and UI Elements

The first prompt was designed to list all the pages required for the project along with their user interface elements. Here is the Python code snippet used to send this prompt to the OpenAI API:

```
response_first_api = client.chat.completions.create(
    model="gpt-3.5-turbo",
    temperature=0,
    messages=[
        {
            "role": "user",
            "content": f"""With the help of the project description below,
describe all the pages required for this project with the user
interface elements that the page needs.
Output template: list all of the pages with a numerical list
and for each page list all of
the user interface elements and the items that they contain.
Don't put any additional text.
Project description: {project_description} """,
        },
        {
            "role": "system",
            "content": "I want you to act as a web designer"
        }
    ],
)
```

Description:

- **Purpose:** The primary goal of this prompt is to obtain a detailed list of pages and their respective user interface elements required for the project. This helps in creating a structured blueprint of the website's design.
- **Structure:** The prompt is structured to include a concise and clear project description, followed by a specific output template. This ensures that the API response is formatted in a way that is easy to parse and use in subsequent steps.
- **Components:**

- **Role of User:** The user provides a detailed project description and requests a structured output.
- **Role of System:** The system is instructed to act as a web designer, ensuring the response aligns with design principles.
- **Output:** The expected output is a numerical list of pages, each containing a list of user interface elements. This output serves as the groundwork for generating the JSON structure used in Figma.

Example Response:

1. Home Page
 - Header: Navigation Bar (Home, About, Services, Contact)
 - Main Section: Welcome Text, Image Carousel
 - Footer: Contact Information, Social Media Links
2. About Page
 - Header: Navigation Bar (Home, About, Services, Contact)
 - Main Section: Company History, Team Members
 - Footer: Contact Information, Social Media Links
3. Services Page
 - Header: Navigation Bar (Home, About, Services, Contact)
 - Main Section: Service List, Service Descriptions
 - Footer: Contact Information, Social Media Links
4. Contact Page
 - Header: Navigation Bar (Home, About, Services, Contact)
 - Main Section: Contact Form (Name, Email, Message, Submit Button)
 - Footer: Contact Information, Social Media Links

Second Prompt: Generating JSON for Figma

The second prompt was designed to generate an RFC8259 compliant JSON structure containing all the user interface elements for each page based on the result gathered from the first prompt. This JSON structure follows a specific template and rules. Here is the Python code snippet used to send this prompt to the OpenAI API:

```
response_second_api = client.chat.completions.create(  
    model="gpt-3.5-turbo-1106",  
    response_format={"type": 'json_object'},
```

```
        temperature=0,  
        messages=[  
            {  
                "role": "user",  
                "content": ""Please generate a RFC8259 compliant  
                JSON containing all of the user interface  
                elements for each page based on the provided text,  
                following the template below:
```

Template:

```
{  
  "pages": [  
    {  
      "page_name": "Page 1",  
      "sections": [  
        {  
          "label": "Header",  
          "elements": [  
            {  
              "type": "navigation_bar",  
              "label": "Navigation Bar",  
              "items": []  
            }  
          ]  
        },  
        {  
          "label": "Column 1",  
          "elements": [  
            {  
              "type": "list",  
              "label": "List 1",  
              "headers": [  
                "Item 1",  
                "Item 2"  
              ]  
            }  
          ]  
        },  
        {  
          "label": "Column 2",  
          "elements": [  
            {
```

```
"type": "form",
"label": "Add Book Form",
"inputs": [
  {
    "type": "text",
    "label": "Title"
  },
  {
    "type": "text",
    "label": "Purchase Date"
  },
  {
    "type": "text",
    "label": "Description"
  },
  {
    "type": "button",
    "label": "Submit"
  }
]
},
{
  "type": "form",
  "label": "Update Book Form",
  "inputs": [
    {
      "type": "text",
      "label": "Title"
    },
    {
      "type": "date",
      "label": "Purchase Date"
    },
    {
      "type": "text",
      "label": "Description"
    },
    {
      "type": "button",
      "label": "Update"
    }
  ]
}
```

```
        ]
      }
    ]
  },
  {
    "label": "Footer",
    "elements": []
  }
]
},
{
  "page_name": "Page 2",
  "sections": [
    {
      "label": "Header",
      "elements": [
        {
          "type": "navigation_bar",
          "label": "Navigation Bar",
          "items": [
            "Home",
            "About"
          ]
        }
      ]
    }
  ],
  {
    "label": "Column 1",
    "elements": [
      {
        "type": "image",
        "label": "Image Label",
      },
      {
        "type": "button",
        "label": "Button 1"
      },
      {
        "type": "button",
        "label": "Button 2"
      }
    ]
  }
}
```

```
    ]
  },
  {
    "label": "Column 2",
    "elements": [
      {
        "type": "list",
        "label": "List 1",
        "headers": [
          "Item 1",
          "Item 2"
        ]
      },
      {
        "type": "text",
        "label": "Title",
        "content": "This is a test text."
      }
    ]
  },
  {
    "label": "Footer",
    "elements": []
  }
]
},
'connections':[
  {
    'from_page' : 'page_name',
    'from_type' : 'button',
    'from_label' : 'Button 1',
    'to_page' : 'page_name',
  },{
    'from_page' : 'page_name',
    'from_type' : 'navigation_bar',
    'from_label' : 'Navigation Bar',
    'to_page' : 'page_name',
  }
]
}
```

Instructions:

- Generate a JSON that adheres to the RFC8259 standard.
- Generate only the JSON without any additional information and check that output is a valid JSON.
- The JSON should contain the user interface elements for each page mentioned in the provided text.
- Include all the pages mentioned in the text.
- Each page should have a name and a list of sections.
- Each section should have a label and a list of elements. label is mandatory for all sections.
- Detect appropriate label for the sections.
- The "Header" section should contain a navigation bar (menu) as a "navigation_bar" element.
- If a section can be shown as a table, include a "list" element inside the section and detect the appropriate headers for the table.
- Forms should be represented as "form" elements with their respective inputs included inside the "inputs" label. Do not add elements of the form inside another section.
- Buttons related to a form should be added as elements within the form.
- The "Footer" and "Header" section should be available on each page.
- Do not modify the template; strictly follow the template rules.
- Detect a label for each element inside sections.
- If the element is not a list or a form, it should be a type "text" with the appropriate content inside "content" and an appropriate "label" or it can be an image with type "image" and an appropriate "label".
- If the element is a filter, it should be added as type "form" with appropriate inputs.
- If the connection item type is 'navigation_bar', in the 'from_label' put the label of link item instead of label of Navigation Menu.
- If the navigation_bar elements are long and more than 5 elements, try to add only 4 of them as navigation_bar and add others as buttons inside the pages.
- Try to detect all elements inside the output.

```
        "", },
    {
      "role": "system",
      "content": "i want you to act as a web designer"
    },
    {
```

```
        "role": "assistant",
        "content": response_first_api.choices[0].message.content
    }
    ],
)
```

- **Purpose:** The main objective of this prompt is to generate a structured JSON that adheres to the RFC8259 standard. This JSON contains detailed information about each page and its user interface elements, facilitating the design process in Figma.
- **Structure:** The prompt is structured to provide a comprehensive template and set of rules that the API response must follow. This ensures that the output JSON is consistent, well-organized, and suitable for direct use in Figma.
- **Components:**
 - **Role of User:** The user provides specific instructions and a detailed template that the output JSON must adhere to.
 - **Role of System:** The system is instructed to act as a web designer, ensuring the response aligns with design principles and standards.
 - **Content of First API Response:** The response from the first API call is included in the prompt to provide the necessary context and details about the pages and their elements.
- **Output:** The expected output is an RFC8259 compliant JSON structure that contains all the user interface elements for each page, following the provided template and instructions.

Example Response:

```
{
  "pages": [
    {
      "page_name": "Home Page",
      "sections": [
        {
          "label": "Header",
          "elements": [
            {
              "type": "navigation_bar",
              "label": "Navigation Bar",

```



```
        "items":[
            "Home",
            "About",
            "Services",
            "Contact"
        ]
    }
]
},
{
    "label":"Main Section",
    "elements":[
        {
            "type":"text",
            "label":"Welcome Text",
            "content":"Welcome to our website!"
        },
        {
            "type":"image",
            "label":"Image Carousel"
        }
    ]
},
{
    "label":"Footer",
    "elements":[
        {
            "type":"text",
            "label":"Contact Information",
            "content":"123 Main St, Anytown, USA"
        },
        {
            "type":"text",
            "label":"Social Media Links",
            "content":"Follow us on social media!"
        }
    ]
}
]
},
{
```

```
"page_name":"About Page",
"sections":[
  {
    "label":"Header",
    "elements":[
      {
        "type":"navigation_bar",
        "label":"Navigation Bar",
        "items":[
          "Home",
          "About",
          "Services",
          "Contact"
        ]
      }
    ]
  },
  {
    "label":"Main Section",
    "elements":[
      {
        "type":"text",
        "label":"Company History",
        "content":"Our company was founded in..."
      },
      {
        "type":"text",
        "label":"Team Members",
        "content":"Meet our team..."
      }
    ]
  },
  {
    "label":"Footer",
    "elements":[
      {
        "type":"text",
        "label":"Contact Information",
        "content":"123 Main St, Anytown, USA"
      },
      {
```

```
        "type": "text",
        "label": "Social Media Links",
        "content": "Follow us on social media!"
    }
    ]
}
]
},
{
    "page_name": "Services Page",
    "sections": [
        {
            "label": "Header",
            "elements": [
                {
                    "type": "navigation_bar",
                    "label": "Navigation Bar",
                    "items": [
                        "Home",
                        "About",
                        "Services",
                        "Contact"
                    ]
                }
            ]
        }
    ],
    {
        "label": "Main Section",
        "elements": [
            {
                "type": "list",
                "label": "Service List",
                "headers": [
                    "Service Name",
                    "Description"
                ]
            }
        ],
        {
            "type": "text",
            "label": "Service Descriptions",
            "content": "We offer the following services..."
        }
    ]
}
```

```
        }
      ]
    },
    {
      "label": "Footer",
      "elements": [
        {
          "type": "text",
          "label": "Contact Information",
          "content": "123 Main St, Anytown, USA"
        },
        {
          "type": "text",
          "label": "Social Media Links",
          "content": "Follow us on social media!"
        }
      ]
    }
  ]
},
{
  "page_name": "Contact Page",
  "sections": [
    {
      "label": "Header",
      "elements": [
        {
          "type": "navigation_bar",
          "label": "Navigation Bar",
          "items": [
            "Home",
            "About",
            "Services",
            "Contact"
          ]
        }
      ]
    }
  ],
  {
    "label": "Main Section",
    "elements": [
```

```
{
  "type": "form",
  "label": "Contact Form",
  "inputs": [
    {
      "type": "text",
      "label": "Name"
    },
    {
      "type": "text",
      "label": "Email"
    },
    {
      "type": "text",
      "label": "Message"
    },
    {
      "type": "button",
      "label": "Submit"
    }
  ]
},
{
  "label": "Footer",
  "elements": [
    {
      "type": "text",
      "label": "Contact Information",
      "content": "123 Main St, Anytown, USA"
    },
    {
      "type": "text",
      "label": "Social Media Links",
      "content": "Follow us on social media!"
    }
  ]
}
]
```

```
],
"connections":[
  {
    "from_page":"Home Page",
    "from_type":"button",
    "from_label":"Button 1",
    "to_page":"About Page"
  },
  {
    "from_page":"Home Page",
    "from_type":"navigation_bar",
    "from_label":"About",
    "to_page":"About Page"
  }
]
}
```

4.4.7 Error Handling for OpenAI API

Moreover, error handling techniques for known OpenAI API failures are built into the Django server. This includes looking for typical problems like misplaced API keys or not enough funds in the user's OpenAI account. The server makes sure that the UI generation process runs more smoothly and robustly by proactively addressing these problems.

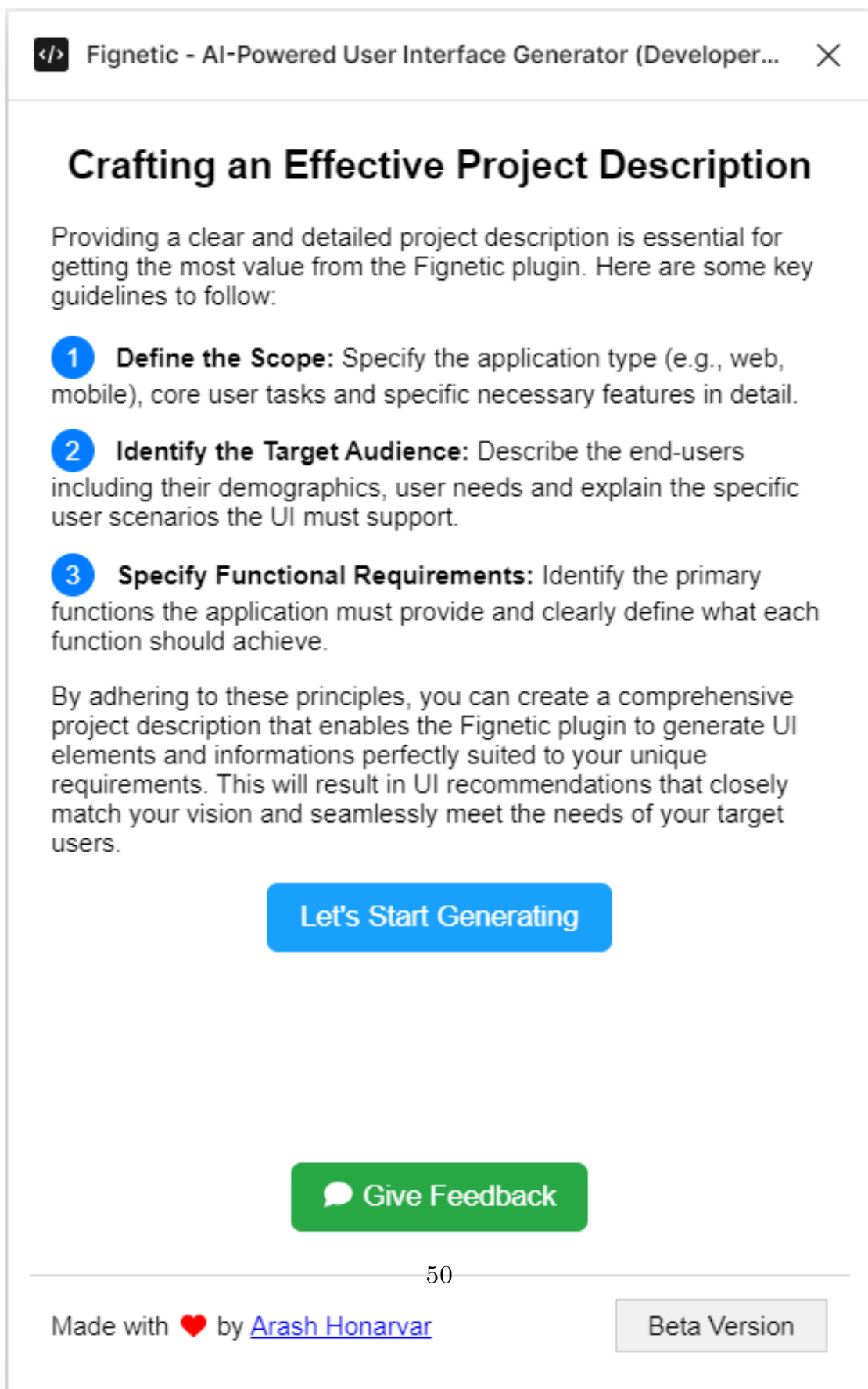
4.5 How to Write a Detailed Project Description

Writing a detailed and clear project description is crucial for generating useful UI recommendations with the Figma plugin. Here are some key points to consider when crafting your project description to ensure the best possible results:

1. **Define the Scope:** Outline the scope of your project. Specify the application type (e.g., web, mobile), primary user tasks and specific necessary features in detail.
2. **Identify the Target Audience:** Explain the end-users including their demographics, user needs and specifics of the users scenarios that the UI must support.
3. **Specify Functional Requirements:** Identify the primary functions that should exist in the application. Outline precisely what every function should

achieve.

It is necessary to follow these guidelines so as to ensure that the project description is comprehensive and informative, therefore, allowing the Figma plugin to create UI elements that are perfectly suitable for your project needs and expectations.



</> Fignetic - AI-Powered User Interface Generator (Developer... ✕

Crafting an Effective Project Description

Providing a clear and detailed project description is essential for getting the most value from the Fignetic plugin. Here are some key guidelines to follow:

- 1 Define the Scope:** Specify the application type (e.g., web, mobile), core user tasks and specific necessary features in detail.
- 2 Identify the Target Audience:** Describe the end-users including their demographics, user needs and explain the specific user scenarios the UI must support.
- 3 Specify Functional Requirements:** Identify the primary functions the application must provide and clearly define what each function should achieve.

By adhering to these principles, you can create a comprehensive project description that enables the Fignetic plugin to generate UI elements and informations perfectly suited to your unique requirements. This will result in UI recommendations that closely match your vision and seamlessly meet the needs of your target users.

[Let's Start Generating](#)

[Give Feedback](#)

Made with ❤️ by [Arash Honarvar](#) Beta Version

Figure 4.3: Crafting an Effective Project Description Page

Chapter 5

Result and Discussion

5.1 Introduction

The results of the implemented solution are shown in the "Result and Discussion" chapter, which also provides a useful example of UI generation utilizing the Figma plugin and Django Python server. This chapter offers a critical analysis of the generated UI designs in addition to insights into the efficiency and performance of the developed system.

5.2 Example Showcase: UI Generation

We demonstrate the features of the Django Python server and the Figma plugin with a particular UI generating scenario. This example shows the entire procedure, from sending a request to the Django server to using the Figma environment to render the final UI designs.

5.2.1 The outlined project description

A system that makes managing assistance tickets possible for consumers of electronic items. There should be a variety of users using the system. Consumers who have the option to register their purchases and open support tickets with the vendor. Professionals are charge of handling open tickets, helping clients, and maybe fixing or replacing damaged goods. Managers are in charge of assigning tickets to experts and must have a dynamic view of how the system is being used.

5.2.2 JSON Output from Django Server

The JSON output returned by the Django server is presented in the appendix A. This JSON structure outlines the hierarchical organization of the generated UI designs, including page names, sections, and UI elements. A comprehensive analysis of the JSON output provides insights into the structure and composition of the UI designs.

5.2.3 Figma UI Designs

Accompanying the JSON output, screenshots of the Figma UI designs are provided in the figures 5.1, 5.2 and 5.3 to visually depict the generated user interfaces. These screenshots showcase the rendered UI elements within the Figma environment, highlighting the layout, styling, and overall presentation of the designs. In addition, it also contains a table which illustrates the connections and links between elements across different pages, showing how navigation and interactions are structured within the application.

5.3 Cost Analysis

Undertaking a cost analysis is an essential step in assessing the proposed solution’s viability and efficacy. The cost effects of using the Figma plugin for UI generation are discussed in this analysis. Through the process of calculating the expenses linked to different use cases, interested parties can make well-informed choices about the acceptance and application of the plugin. For detailed pricing information, readers are directed to refer to the OpenAI pricing page at <https://openai.com/pricing>.

The cost analysis’s findings are displayed in the Table 5.1, which shows the average cost of using the Figma plugin for ten distinct requests. This table acts as a helpful resource for resource allocation and budgeting, as well as transparency regarding the cash outlay made by users.

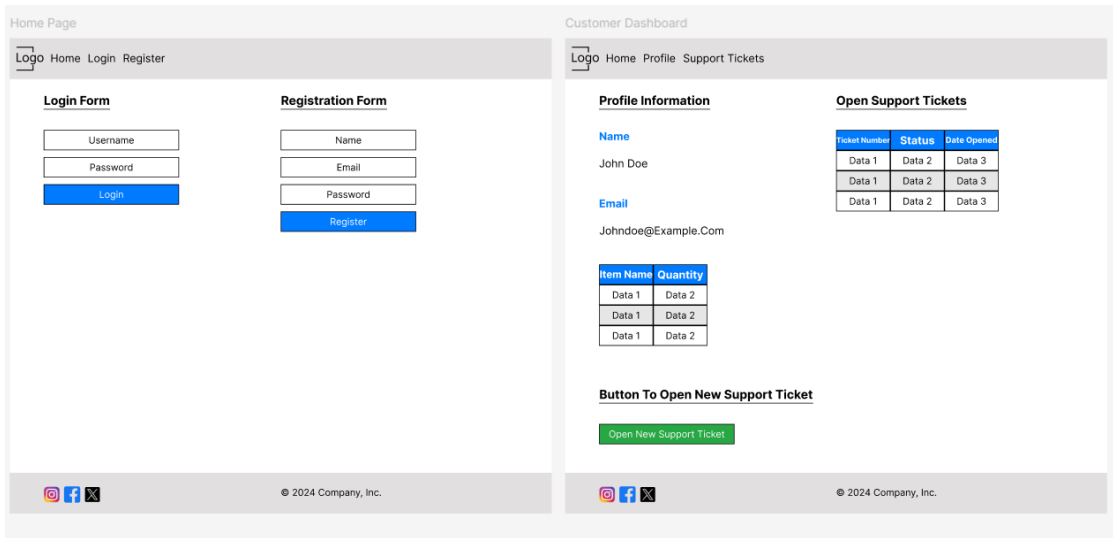
Table 5.1: Total Cost of Figma Plugin

Number of Requests	Number of Context Tokens	Number of Generated Tokens	Total Cost (\$)
10	1395	3483	0.07
2	342	684	0.01
3	531	906	0.015

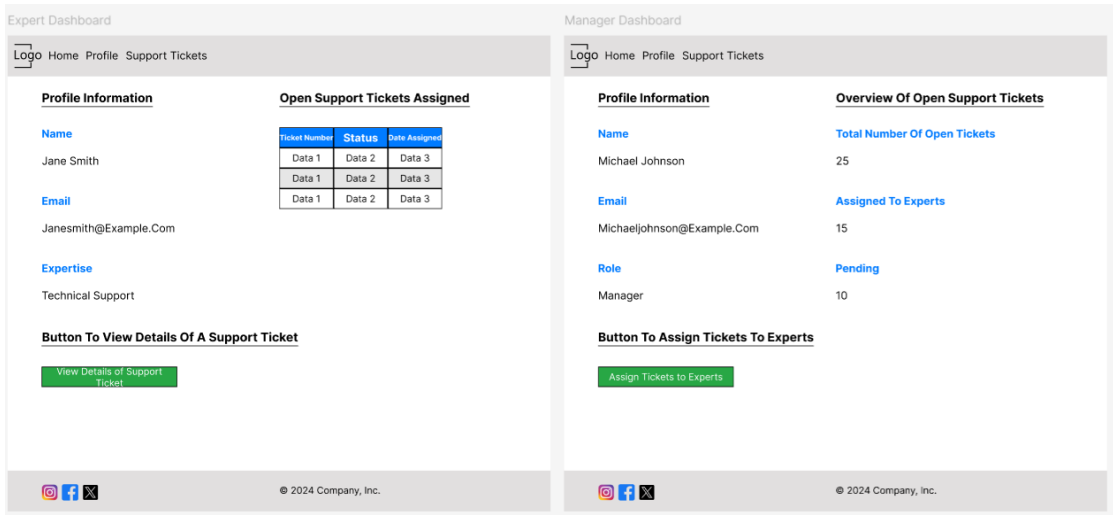
5.4 Intended Purpose and Design Philosophy

Clarifying the intended aim of the generated solution is crucial while discussing the outcomes of this thesis project. This thesis project takes a different approach than standard UI-to-code plugins, which are mostly used by developers to speed up the coding process. It is intended to help designers expedite their workflow by offering recommendations on the structural architecture of user interfaces. This project is created as an information engineering tool rather than merely an AI-driven design generator. The primary purpose is to inform designers about the elements that comprise UI pages based on user-provided project descriptions, thereby enhancing their understanding and decision-making processes. The approach helps designers better conceptualize and organize their concepts, rather than producing code directly. The tool provides significant insights on the arrangement of items on individual pages and the overall interface structure, making it an invaluable tool for designers looking to improve productivity and optimize their design process.

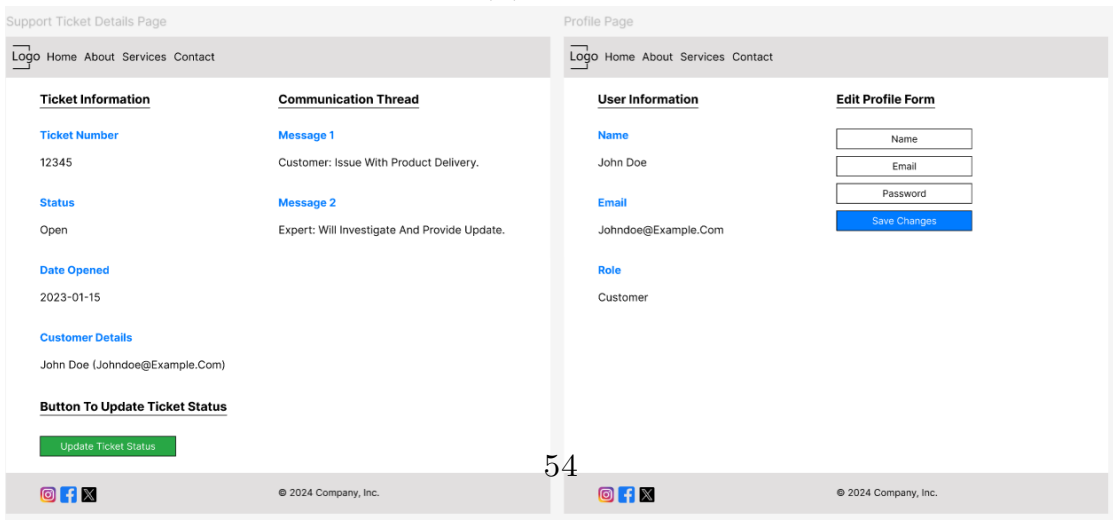
Result and Discussion



(a) Image 1



(b) Image 2



54

(c) Image 3

Figure 5.1: Screenshot of Figma Plugin

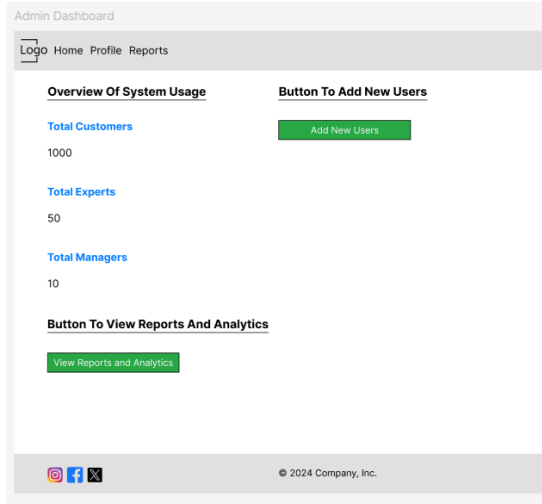


Figure 5.2: Screenshot of Figma Plugin - Image 4

Connections

This table illustrates the connections and links between elements across different pages, showing how navigation and interactions are structured within the application.

Source Page	Interaction Type	Interaction Label	Destination Page
Home Page	Button	Login	Customer Dashboard
Home Page	Button	Register	Customer Dashboard
Customer Dashboard	Button	Open New Support Ticket	Support Ticket Details Page
Expert Dashboard	Button	View Details of Support Ticket	Support Ticket Details Page
Manager Dashboard	Button	Assign Tickets to Experts	Expert Dashboard
Support Ticket Details Page	Button	Update Ticket Status	Manager Dashboard
Profile Page	Button	Save Changes	Home Page
Admin Dashboard	Button	Add New Users	Home Page
Admin Dashboard	Button	View Reports and Analytics	Home Page
Home Page	Navigation Bar	Profile	Profile Page
Home Page	Navigation Bar	Support Tickets	Customer Dashboard
Home Page	Navigation Bar	Support Tickets	Expert Dashboard
Home Page	Navigation Bar	Support Tickets	Manager Dashboard
Admin Dashboard	Navigation Bar	Profile	Profile Page
Admin Dashboard	Navigation Bar	Reports	Home Page

Figure 5.3: Screenshot of Figma Plugin - Image 5

Chapter 6

Future Works

When considering how this project will develop in the future, a number of opportunities for improvement and growth become apparent. Enhancing the user interface (UI) of the created user interface elements is one of the main priorities. Although this version offers insightful information about the structural design of interfaces, both the usability and visual appeal might be strengthened. Future revisions will prioritize implementing design upgrades to guarantee a more visually appealing and intuitive experience for users.

Moreover, a wider range of design elements can be included in the area of approved user interface elements. By providing advanced UI features and adopting a contemporary design approach for interface elements, the tool can significantly enhance the creative freedom and adaptability available to designers. With additional flexibility to accommodate a wider range of project objectives and design preferences, designers will be able to produce more comprehensive and dynamic user interfaces.

As the project enters its next phase of development, it is well-positioned to develop naturally, customizing its features and functionalities to meet the unique requirements and preferences of its user base, thanks to user feedback acting as a compass.

In conclusion, this project's future is quite promising for additional innovation and progress in the field of user interface design. The project is well-positioned to significantly advance the facilitation of more effective, intuitive, and visually appealing user interface design workflows because of its focus to fulfilling the changing needs of designers and its commitment to ongoing improvement.

Appendix A

JSON Output from Django Server

```
1 {
2   "pages":[
3     {
4       "page_name":"Home Page",
5       "sections":[
6         {
7           "label":"Header",
8           "elements":[
9             {
10              "type":"navigation_bar",
11              "label":"Navigation Menu",
12              "items":[
13                "Home",
14                "Login",
15                "Register"
16              ]
17            }
18          ]
19        },
20        {
21          "label":"Login Form",
22          "elements":[
23            {
24              "type":"form",
25              "label":"Login Form",
26              "inputs":[
27                {
28                  "type":"text",
```

```
29         "label ":" Username "
30     },
31     {
32         "type ":" text ",
33         "label ":" Password "
34     },
35     {
36         "type ":" button ",
37         "label ":" Login "
38     }
39 ]
40 }
41 ]
42 },
43 {
44     "label ":" Registration Form ",
45     "elements ":[
46         {
47             "type ":" form ",
48             "label ":" Registration Form ",
49             "inputs ":[
50                 {
51                     "type ":" text ",
52                     "label ":" Name "
53                 },
54                 {
55                     "type ":" text ",
56                     "label ":" Email "
57                 },
58                 {
59                     "type ":" text ",
60                     "label ":" Password "
61                 },
62                 {
63                     "type ":" button ",
64                     "label ":" Register "
65                 }
66             ]
67         }
68     ]
69 },
70 {
71     "label ":" Footer ",
72     "elements ":[
73     ]
74 }
75 ]
76 },
77 }
```



```
78 | {
79 |   "page_name": "Customer Dashboard",
80 |   "sections": [
81 |     {
82 |       "label": "Header",
83 |       "elements": [
84 |         {
85 |           "type": "navigation_bar",
86 |           "label": "Navigation Menu",
87 |           "items": [
88 |             "Home",
89 |             "Profile",
90 |             "Support Tickets"
91 |           ]
92 |         }
93 |       ]
94 |     },
95 |     {
96 |       "label": "Profile Information",
97 |       "elements": [
98 |         {
99 |           "type": "text",
100 |          "label": "Name",
101 |          "content": "John Doe"
102 |         },
103 |         {
104 |           "type": "text",
105 |           "label": "Email",
106 |           "content": "johndoe@example.com"
107 |         },
108 |         {
109 |           "type": "list",
110 |           "label": "Purchased Goods",
111 |           "headers": [
112 |             "Item Name",
113 |             "Quantity"
114 |           ]
115 |         }
116 |       ]
117 |     },
118 |     {
119 |       "label": "Open Support Tickets",
120 |       "elements": [
121 |         {
122 |           "type": "list",
123 |           "label": "Support Tickets",
124 |           "headers": [
125 |             "Ticket Number",
126 |             "Status",
```

```
127         "Date Opened"
128     ]
129     }
130 ]
131 },
132 {
133     "label": "Button to Open New Support Ticket",
134     "elements": [
135         {
136             "type": "button",
137             "label": "Open New Support Ticket"
138         }
139     ]
140 },
141 {
142     "label": "Footer",
143     "elements": [
144     ]
145     }
146 ]
147 },
148 {
149     "page_name": "Expert Dashboard",
150     "sections": [
151     {
152         "label": "Header",
153         "elements": [
154             {
155                 "type": "navigation_bar",
156                 "label": "Navigation Menu",
157                 "items": [
158                     "Home",
159                     "Profile",
160                     "Support Tickets"
161                 ]
162             }
163         ]
164     },
165     {
166         "label": "Profile Information",
167         "elements": [
168             {
169                 "type": "text",
170                 "label": "Name",
171                 "content": "Jane Smith"
172             },
173             {
174                 "type": "text",
175
```

```

176         "label ":" Email ",
177         "content ":" janesmith@example.com "
178     },
179     {
180         "type ":" text ",
181         "label ":" Expertise ",
182         "content ":" Technical Support "
183     }
184 ]
185 },
186 {
187     "label ":" Open Support Tickets Assigned ",
188     "elements ":[
189         {
190             "type ":" list ",
191             "label ":" Assigned Support Tickets ",
192             "headers ":[
193                 "Ticket Number ",
194                 "Status ",
195                 "Date Assigned "
196             ]
197         }
198     ]
199 },
200 {
201     "label ":" Button to View Details of a Support Ticket ",
202     "elements ":[
203         {
204             "type ":" button ",
205             "label ":" View Details of Support Ticket "
206         }
207     ]
208 },
209 {
210     "label ":" Footer ",
211     "elements ":[
212     ]
213 ]
214 }
215 ]
216 },
217 {
218     "page_name ":" Manager Dashboard ",
219     "sections ":[
220         {
221             "label ":" Header ",
222             "elements ":[
223                 {
224                     "type ":" navigation_bar ",

```

```
225         "label ":" Navigation Menu",
226         "items ":[
227             "Home",
228             "Profile ",
229             "Support Tickets"
230         ]
231     }
232 ]
233 },
234 {
235     "label ":" Profile Information",
236     "elements ":[
237         {
238             "type ":" text ",
239             "label ":" Name",
240             "content ":" Michael Johnson"
241         },
242         {
243             "type ":" text ",
244             "label ":" Email",
245             "content ":" michaeljohnson@example.com"
246         },
247         {
248             "type ":" text ",
249             "label ":" Role",
250             "content ":" Manager"
251         }
252     ]
253 },
254 {
255     "label ":" Overview of Open Support Tickets",
256     "elements ":[
257         {
258             "type ":" text ",
259             "label ":" Total Number of Open Tickets",
260             "content ":" 25"
261         },
262         {
263             "type ":" text ",
264             "label ":" Assigned to Experts",
265             "content ":" 15"
266         },
267         {
268             "type ":" text ",
269             "label ":" Pending",
270             "content ":" 10"
271         }
272     ]
273 },
```

```
274     {
275       "label": "Button to Assign Tickets to Experts",
276       "elements": [
277         {
278           "type": "button",
279           "label": "Assign Tickets to Experts"
280         }
281       ]
282     },
283     {
284       "label": "Footer",
285       "elements": [
286       ]
287     }
288   ]
289 },
290 {
291   "page_name": "Support Ticket Details Page",
292   "sections": [
293     {
294       "label": "Ticket Information",
295       "elements": [
296         {
297           "type": "text",
298           "label": "Ticket Number",
299           "content": "12345"
300         },
301         {
302           "type": "text",
303           "label": "Status",
304           "content": "Open"
305         },
306         {
307           "type": "text",
308           "label": "Date Opened",
309           "content": "2023-01-15"
310         },
311         {
312           "type": "text",
313           "label": "Customer Details",
314           "content": "John Doe (johndoe@example.com)"
315         }
316       ]
317     },
318     {
319       "label": "Communication Thread",
320       "elements": [
321         {
322
```

```

323         "type ":" text ",
324         "label ":" Message 1",
325         "content ":" Customer: Issue with product delivery
. "
326     },
327     {
328         "type ":" text ",
329         "label ":" Message 2",
330         "content ":" Expert: Will investigate and provide
update. "
331     }
332 ]
333 },
334 {
335     "label ":" Button to Update Ticket Status",
336     "elements ":[
337         {
338             "type ":" button ",
339             "label ":" Update Ticket Status"
340         }
341     ]
342 },
343 {
344     "label ":" Footer ",
345     "elements ":[
346
347     ]
348 }
349 ]
350 },
351 {
352     "page_name ":" Profile Page ",
353     "sections ":[
354         {
355             "label ":" User Information ",
356             "elements ":[
357                 {
358                     "type ":" text ",
359                     "label ":" Name",
360                     "content ":" John Doe"
361                 },
362                 {
363                     "type ":" text ",
364                     "label ":" Email ",
365                     "content ":" johndoe@example.com "
366                 },
367                 {
368                     "type ":" text ",
369                     "label ":" Role ",

```

```

370         "content ":" Customer "
371     }
372 ]
373 },
374 {
375     "label ":" Edit Profile Form ",
376     "elements ":[
377         {
378             "type ":" form ",
379             "label ":" Edit Profile Form ",
380             "inputs ":[
381                 {
382                     "type ":" text ",
383                     "label ":" Name "
384                 },
385                 {
386                     "type ":" text ",
387                     "label ":" Email "
388                 },
389                 {
390                     "type ":" text ",
391                     "label ":" Password "
392                 },
393                 {
394                     "type ":" button ",
395                     "label ":" Save Changes "
396                 }
397             ]
398         }
399     ]
400 },
401 {
402     "label ":" Footer ",
403     "elements ":[
404     ]
405     }
406 ]
407 },
408 {
409     "page_name ":" Admin Dashboard ",
410     "sections ":[
411         {
412             "label ":" Header ",
413             "elements ":[
414                 {
415                     "type ":" navigation_bar ",
416                     "label ":" Navigation Menu ",
417                     "items ":[
418

```

```
419         "Home",
420         "Profile",
421         "Reports"
422     ]
423 }
424 ]
425 },
426 {
427     "label": "Overview of System Usage",
428     "elements": [
429         {
430             "type": "text",
431             "label": "Total Customers",
432             "content": "1000"
433         },
434         {
435             "type": "text",
436             "label": "Total Experts",
437             "content": "50"
438         },
439         {
440             "type": "text",
441             "label": "Total Managers",
442             "content": "10"
443         }
444     ]
445 },
446 {
447     "label": "Button to Add New Users",
448     "elements": [
449         {
450             "type": "button",
451             "label": "Add New Users"
452         }
453     ]
454 },
455 {
456     "label": "Button to View Reports and Analytics",
457     "elements": [
458         {
459             "type": "button",
460             "label": "View Reports and Analytics"
461         }
462     ]
463 },
464 {
465     "label": "Footer",
466     "elements": [
467
```



```
468         ]
469     }
470 ]
471 }
472 ],
473 "connections":[
474     {
475         "from_page":"Home Page",
476         "from_type":"button",
477         "from_label":"Login",
478         "to_page":"Customer Dashboard"
479     },
480     {
481         "from_page":"Home Page",
482         "from_type":"button",
483         "from_label":"Register",
484         "to_page":"Customer Dashboard"
485     },
486     {
487         "from_page":"Customer Dashboard",
488         "from_type":"button",
489         "from_label":"Open New Support Ticket",
490         "to_page":"Support Ticket Details Page"
491     },
492     {
493         "from_page":"Expert Dashboard",
494         "from_type":"button",
495         "from_label":"View Details of Support Ticket",
496         "to_page":"Support Ticket Details Page"
497     },
498     {
499         "from_page":"Manager Dashboard",
500         "from_type":"button",
501         "from_label":"Assign Tickets to Experts",
502         "to_page":"Expert Dashboard"
503     },
504     {
505         "from_page":"Support Ticket Details Page",
506         "from_type":"button",
507         "from_label":"Update Ticket Status",
508         "to_page":"Manager Dashboard"
509     },
510     {
511         "from_page":"Profile Page",
512         "from_type":"button",
513         "from_label":"Save Changes",
514         "to_page":"Home Page"
515     },
516     {
```

```
517     "from_page": "Admin Dashboard",
518     "from_type": "button",
519     "from_label": "Add New Users",
520     "to_page": "Home Page"
521   },
522   {
523     "from_page": "Admin Dashboard",
524     "from_type": "button",
525     "from_label": "View Reports and Analytics",
526     "to_page": "Home Page"
527   },
528   {
529     "from_page": "Home Page",
530     "from_type": "navigation_bar",
531     "from_label": "Profile",
532     "to_page": "Profile Page"
533   },
534   {
535     "from_page": "Home Page",
536     "from_type": "navigation_bar",
537     "from_label": "Support Tickets",
538     "to_page": "Customer Dashboard"
539   },
540   {
541     "from_page": "Home Page",
542     "from_type": "navigation_bar",
543     "from_label": "Support Tickets",
544     "to_page": "Expert Dashboard"
545   },
546   {
547     "from_page": "Home Page",
548     "from_type": "navigation_bar",
549     "from_label": "Support Tickets",
550     "to_page": "Manager Dashboard"
551   },
552   {
553     "from_page": "Admin Dashboard",
554     "from_type": "navigation_bar",
555     "from_label": "Profile",
556     "to_page": "Profile Page"
557   },
558   {
559     "from_page": "Admin Dashboard",
560     "from_type": "navigation_bar",
561     "from_label": "Reports",
562     "to_page": "Home Page"
563   }
564 ]
565 }
```


Bibliography

- [1] Junfeng Wang, Zhiyu Xu, Xi Wang, and Jingjing Lu. «A Comparative Research on Usability and User Experience of User Interface Design Software». In: *International Journal of Advanced Computer Science and Applications* 13.8 (2022). DOI: 10.14569/IJACSA.2022.0130804 (cit. on p. 1).
- [2] Denis Pimenov, Alexander Solovyov, Nursultan Askarbekuly, and Manuel Mazzara. «Data-Driven Approaches to User Interface Design: A Case Study». In: *Journal of Physics: Conference Series* 2134 (2021). Published under licence by IOP Publishing Ltd, p. 012020. DOI: 10.1088/1742-6596/2134/1/012020 (cit. on p. 1).
- [3] *Figma*. <https://www.figma.com/>. Accessed on February 25, 2024 (cit. on p. 1).
- [4] ZE Ferdi Fauzan Putra, Hamidillah Ajie, and Ika Anwar Safitri. «Designing A User Interface and User Experience from Piring Makanku Application by Using Figma Application for Teens». In: *IJISTECH (International Journal of Information System and Technology)* 5.3 (2021), pp. 308–315 (cit. on p. 1).
- [5] Fabio Staiano. *Designing and Prototyping Interfaces with Figma: Learn essential UX/UI design principles by creating interactive prototypes for mobile, tablet, and desktop*. Packt Publishing Ltd, 2022 (cit. on pp. 1, 9).
- [6] Melissa Zhang. «Speeding Up the Prototyping of Low-Fidelity User Interface Wireframes». PhD thesis. 2022 (cit. on p. 2).
- [7] Lena Hegemann, Niraj Ramesh Dayama, Abhishek Iyer, Erfan Farhadi, Ekaterina Marchenko, and Antti Oulasvirta. «CoColor: Interactive Exploration of Color Designs». In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 2023, pp. 106–127 (cit. on p. 2).
- [8] Brad Myers. «Challenges of HCI design and implementation». In: *interactions* 1.1 (1994), pp. 73–83 (cit. on p. 2).

- [9] Victor Alvarez-Cortes, Benjamin E. Zayas-Perez, Victor Huga Zarate-Silva, and Jorge A. Ramirez Uresti. «Current Trends in Adaptive User Interfaces: Challenges and Applications». In: *Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*. 2007, pp. 312–317. DOI: 10.1109/CERMA.2007.4367705 (cit. on p. 2).
- [10] Julián Grigera, Jordán Pascual Espada, and Gustavo Rossi. «AI in User Interface Design and Evaluation». In: *IT Professional* 25.2 (2023), pp. 20–22. DOI: 10.1109/MITP.2023.3267139 (cit. on p. 2).
- [11] Saleema Amershi et al. «Guidelines for Human-AI Interaction». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300233. URL: <https://doi.org/10.1145/3290605.3300233> (cit. on p. 2).
- [12] Clare-Marie Karat, Jan O Blom, and John Karat. *Designing personalized user experiences in eCommerce*. Vol. 5. Springer Science & Business Media, 2004 (cit. on p. 2).
- [13] OpenAI. *OpenAI*. 2015. URL: <https://openai.com/> (visited on 02/28/2024) (cit. on p. 2).
- [14] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: *OpenAI Blog* (2019). URL: <https://openai.com/research/language-unsupervised/> (visited on 02/28/2024) (cit. on p. 2).
- [15] Django Software Foundation. *Django*. 2022. URL: <https://www.djangoproject.com/> (cit. on p. 2).
- [16] Mladjan Jovanovic and Mark Campbell. «Generative Artificial Intelligence: Trends and Prospects». In: *Computer* 55 (Oct. 2022), pp. 107–112. DOI: 10.1109/MC.2022.3192720 (cit. on p. 6).
- [17] Mohanad Abukmeil, Stefano Ferrari, Angelo Genovese, Vincenzo Piuri, and Fabio Scotti. «A survey of unsupervised generative models for exploratory data analysis and representation learning». In: *Acm computing surveys (csur)* 54.5 (2021), pp. 1–40 (cit. on p. 6).
- [18] Ömer Aydın and Enis Karaarslan. «Is ChatGPT leading generative AI? What is beyond expectations?» In: *What is beyond expectations* (2023) (cit. on p. 6).
- [19] TB Brown et al. «Language Models are Few-Shot Learners Advances in Neural Information Processing Systems 33». In: (2020) (cit. on p. 6).
- [20] Clare Williams. «Hype, or the future of learning and teaching? 3 Limits to AI's ability to write student essays». In: (2023) (cit. on p. 6).

- [21] OpenAI. *GPT-4 technical report*. 2023. URL: <https://cdn.openai.com/papers/gpt-4.pdf> (cit. on p. 6).
- [22] OpenAI. *OpenAI*. 2024. URL: <https://openai.com/> (cit. on p. 7).
- [23] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. «A brief overview of ChatGPT: The history, status quo and potential future development». In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136 (cit. on p. 7).
- [24] Wenxiang Jiao, Wenxuan Wang, JT Huang, Xing Wang, and ZP Tu. «Is ChatGPT a good translator? Yes with GPT-4 as the engine». In: *arXiv preprint arXiv:2301.08745* (2023) (cit. on p. 7).
- [25] Percy Liang et al. «Holistic evaluation of language models». In: *arXiv preprint arXiv:2211.09110* (2022) (cit. on p. 7).
- [26] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. «Is ChatGPT a general-purpose natural language processing task solver?» In: *arXiv preprint arXiv:2302.06476* (2023) (cit. on p. 7).
- [27] Ali Borji. «A categorical archive of chatgpt failures». In: *arXiv preprint arXiv:2302.03494* (2023) (cit. on p. 7).
- [28] Albert Yu Sun, Elliott Zemer, Arushi Saxena, Udith Vaidyanathan, Eric Lin, Christian Lau, and Vaikkunth Mugunthan. «Does fine-tuning GPT-3 with the OpenAI API leak personally-identifiable information?» In: *arXiv preprint arXiv:2307.16382* (2023) (cit. on p. 8).
- [29] Uizard. *About Uizard*. 2022. URL: <https://uizard.io/about/> (cit. on p. 12).