Software Proposal Document for MetaMorph

Ahmed Ehab, Ali Ismail , Mohammed Mamdouh, Mahmoud Khaled
Supervised by: Dr. Nesreen El-Saber, Eng. Toqa El-Rashedy

November 15, 2023

Proposal Version	Date	Reason for Change
1.0	14-November-2023	Proposal First version's specifications are defined

Table 1: Document version history

GitHub: https://github.com/Ahmedehab280202/MetaMorph

Abstract

MetaMorph addresses challenges in the software development life cycle, particularly in the progression from design to implementation. This phase often introduces complexities and communication gaps, leading to deviations from the original vision. To meet these challenges, MetaMorph is designed to ensure a seamless and accurate progression from design concepts to functional software. In the current software industry, efficient development is important, and delays can be costly. MetaMorph emerges as an suitable solution, promising a streamlined transition between design and development, reducing risks, improving productivity, and aligning with industry demands for precision and speed. The system aims to mirror design elements accurately, provide real-time feedback, and ensure a well-organized project structure.

1 Introduction

1.1 Background

In the broad landscape of the software development life cycle, the progression from design to practical implementation has long been a crucial turning point in the software's life cycle. This intersection is crucial because it serves as a benchmark, determining whether the implementation align seamlessly with the vision of the designers and planners. At this point, the software is created to reflect the intended design, guaranteeing a precise and seamless conversion of concepts into practical, useful software. However, it also presents its own challenges. The shift from design to development introduces complexities, such as misinterpretations, inconsistencies, and the potential for communication gaps. These hurdles can result in deviations from the original vision, leading to a disconnect between the design intent and the final product. And It becomes vital to address these challenges to ensure a smooth and efficient progression of the software development process.

1.2 Motivation

1.2.1 Academic

Teams in software development organizations are made up of developers, analysts, product owners, and UI/UX designers, among others. To fulfill the needs of the client, these team members must work together [4]. As stated in [6], integrating design and implementation can be difficult, especially when there is a divide between designers and developers. This is because poor communication and unreasonable design expectations can affect how a project is carried out as a whole. Both teams gain from automating the conversion of UML to back-end code as well as UI design to front-end code. An instrument that makes this process easier increases productivity and lowers mistakes [4]. There are tools in a software category developed to improve communication in software development groups that help close the gap between developers and designers. These technologies serve as channels, enhancing work coordination and communication while enabling automatic interaction between these vital processes from design to development. Stronger security measures to protect important design assets, better prototyping capabilities, and better user input integration are all possible enhancements for these technologies. By offering training resources and customization choices, these technologies can continue to improve and adapt, which makes it easier for UI/UX designers and developers to work together [5].

1.2.2 Business

A continuous pursuit for maximum efficiency characterizes the software development environment. In the rapidly evolving world of software businesses, time and financial resources are interchangeable. Software production delays can be exceptionally costly, particularly in agile environments where rapid development cycles are important.

Furthermore, there are numerous risks associated with the likelihood of misinterpretations or vacancies between the planning stage and the execution of the project. In conditions like these, organizations risk not just monetary losses but also damage to their brand as a consequence of differences between client expectations and the actual outcome.

In today's competitive software market, businesses are in a race against time and securing a competitive edge is crucial. Companies compete fiercely, and being able to produce exceptional software rapidly can set you apart. In an industry where efficiency and innovation are valued greatly, maintaining an advantageous position in the race against time means staying ahead of the competition.

With MetaMorph, an innovative approach ready to revolutionize the software development life cycle, these necessary business needs are met. Through the smooth transition between design and development, MetaMorph minimizes risks, boosts productivity, and eliminates the likelihood of delays, aligning perfectly with the industry's demand for precision, speed, and competitive distinction.

1.3 Problem Statement

During the Software Development Life Cycle, where each step is interconnected and symbolizing the progression of forming a concept for deployment, a seamless SDLC is crucial. Any disruptions in its flow can have far-reaching consequences. And as we transition from one phase to the next, the shift from the design phase to the implementation phase represents a significant turning point since it has a major impact on project outcomes.

But difficulties arise at this crucial point, such as inconsistent manual interpretation, poor communication, and human error resulting in ongoing and frequent modifications. These difficulties may lead to more serious issues, such as excessive time consumption and an inconsistency between the intended and final products."

2 Project Description

MetaMorph, as envisioned by its users, is expected to deliver a seamless transition from design to code by meeting a set of essential requirements. First and foremost, the generated code must faithfully mirror the design elements, ensuring accuracy and fidelity throughout the conversion process. Users expect that the code generated complies firmly with industry best practices and standards, giving them confidence in the stability of the output that is produced. Real-time feedback is essential throughout the design-to-code procedure because it allows for early identification of possible problems. Users also prefer a well-organized file and folder structure that makes things clear and simple to navigate.Efficiency is critical, particularly for large and complex design projects where eager code generation is required.Importantly, users expect minimal delays, ensuring a responsive and timely design-to-code conversion experience. In order to satisfy the diverse needs of its user base, MetaMorph must, be scalable. This makes it easily adaptable to varying project sizes and levels of complexity.

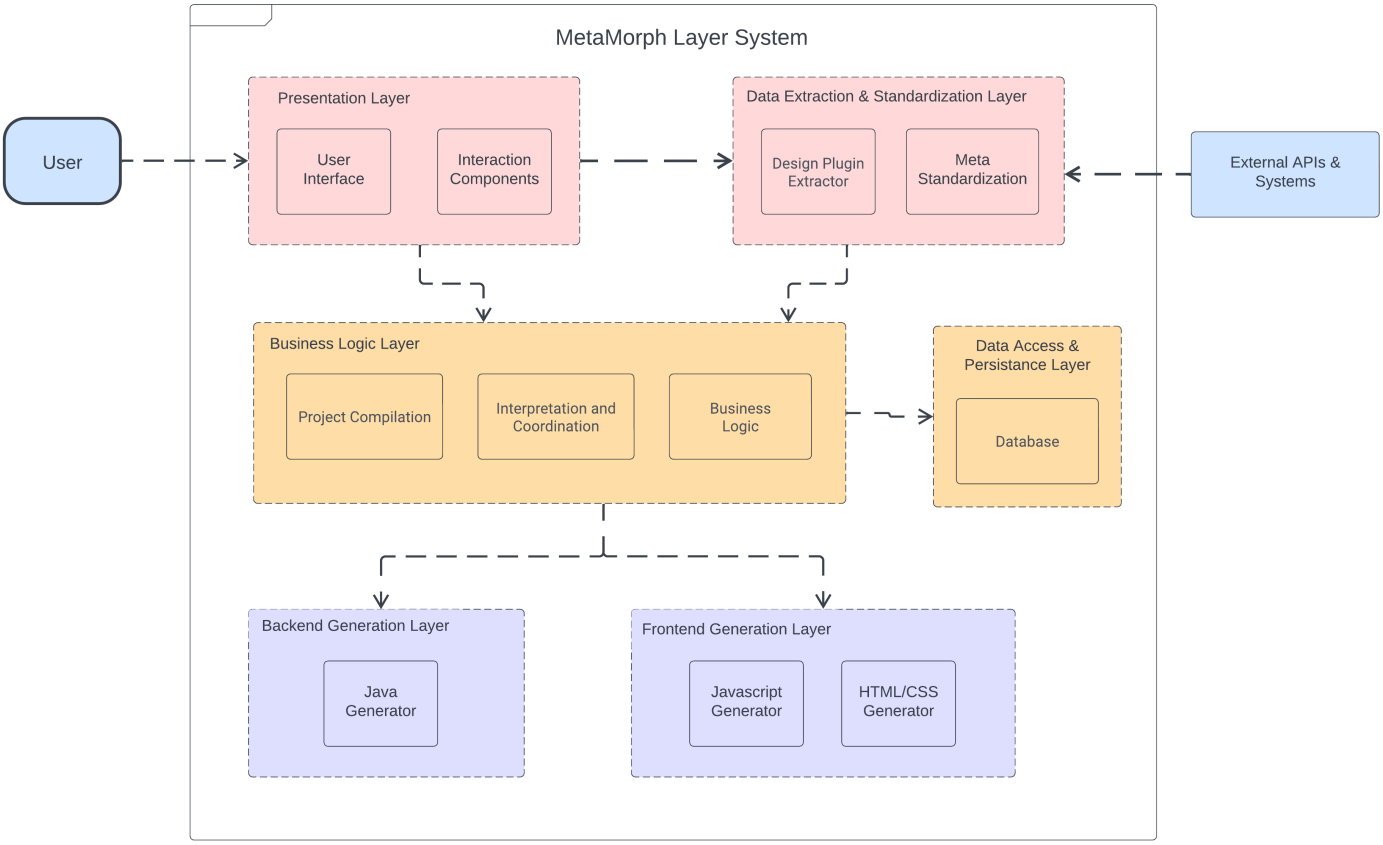

Figure 1: System Overview

1. **Presentation Layer:** This layer engages users through an intuitive interface
   - **User Interface:** Presentation of information to the user
   - **Interaction Components:** Handling user interactions related to design submission, preferences, etc.
2. **Data Extraction & Standardization Layer:** Responsible for interfacing with external APIs and systems for extraction

- **Design Plugin Extractor:** Communicating with external sources, such as design tools' APIs, for extracting JSON data representing design elements
- **Meta Standardization:** Filtering and transforming data extracted from various design tools into a standardized structure

3. **Business Logic Layer:** At the center of MetaMorph, this layer manages fullstack generation, coordination processes, and maintains business logic, ensuring coherence and integrity throughout.
    - **Business Logic:** Ensures consistency, integrity, and coherence in the application's operations
    - **Interpretation and Coordination:** Orchestrating the interpretation of incoming design data and coordinating the execution of necessary generators and compilers
    - **Project Compilation:** Assembling generated code into a coherent project structure based on user preferences
4. **Data Access & Persistence Layer:** Enables efficient storage and retrieval of information
    - **Database:** Interactions with the database, facilitating data storage and retrieval
5. **Backend Generation Layer:** Handling the translation of structural diagrams into server-side logic
    - **Java Generator:** Algorithms responsible for transforming incoming design informative json data into the respective Java code
6. **Frontend Generation Layer:** Handling the transformation of UI designs into frontend web technologies
    - **Javascript Generator:** Algorithms responsible for transforming incoming design informative json data into the respective JavaScript code
    - **HTML/CSS Generator:** Algorithms responsible for transforming incoming design informative json data into the respective HTML/CSS code

2.1 Objectives

- We will reduce the time required for MetaMorph users to generate front-end code from design layers in Figma by 30%, enhancing the efficiency of the design-to-code process.
- We will ensure that MetaMorph generates project directories with an organized structure, reducing the time required for developers to set up projects. Achieve a user satisfaction rating of 80% or higher for the project structure in user feedback surveys within five months.
- Ensure scalability of MetaMorph for projects of different sizes and complexities. Test MetaMorph with a range of project scenarios, ensuring that the system remains efficient and effective across different scales. Achieve a scalability rating of 90% or higher in user testing within six months.

2.2 Scope

MetaMorph helps developers save time, effort, and cost, ensuring seamless integration, efficient project setup, and enhanced user experience. Therefore, to achieve this, the system will contain the following:

- The system will have the ability to generate back-end code through the interpretation of UML class diagram.
- The system seamlessly converts the UI frames into either HTML, CSS, and React.js code.
- It will offer front-end and back-end code integration, enabling the construction of real-time applications by smoothly integrating dynamic data sources into the front-end code.
- Producing a predefined structure of the project which improves project setup and guarantees an efficient, structured development environment.
- It contains integrated code inspection tool that helps in enhancing the quality of the code, also provides export option that make it easy to integrate into other development environments.

2.3 Project Overview

2.4 Stakeholder

2.4.1 Internal

- Ahmed Ehab (Team Leader) : Responsible for Project Overview, Presentation, Data Extraction and Standardization
- Ali Ismail (Team Member) : Responsible for Business Logic and Data Access Persistence
- Mohammed Mamdouh (Team Member) : Responsible for Frontend Generation
- Mahmoud Khaled (Team Member) : Responsible for Backend Generation and Project Compilation

2.4.2 External

MetaMorph aims to target the following front-end, back-end, full-stack developers and UI/UX designers.

3 Similar System

3.1 Academic

3.1.1 closing the gap between designers and developers in a low code ecosystem.

Digital transformation is now necessary for organizations to be competitive in today's market. The increasing need for qualified software engineers is one of the difficulties this transition faces. The reality of established industry practices for UX and UI designers collaborating with front-end developers still leaves a lot to improve in terms of effectiveness and efficiency. To mitigate this pressure point, low-code platforms have risen, allowing people with non-programming backgrounds to craft digital systems capable of solving business relevant problems. This paper proposes an innovative approach that uses model transformation and meta-modelling techniques to transform UX/UI design artifacts into low-code web technology. The approach has been applied to a recognized and established enterprise-grade low-code platform and evaluated in practice by a team of professional designers and front-end developers. Preliminary practical results show savings between 20 and 75 % according to the project complexity in the effort invested by development teams in the above mentioned process. The paper also presents a prototype tool that receives a template

application containing all base elements as input and generates customized Out Systems applications, which represents a significant reduction in the duration of the conversion and customization process carried out by professional teams. topic[6].

3.1.2 Generating Reusable web components from mockups.

The process of generating reusable web components from mockups is a crucial step in web development. It involves analyzing the mockup and identifying repeated UI patterns, which can be extracted and converted into reusable components. This process is often time-consuming and requires manual effort, which can be a bottleneck in the development process. To address this issue, an approach has been proposed that automates the process of generating reusable web components from mockups. This approach uses visual analysis and unsupervised learning to identify and extract repeated UI patterns from the mockup. The extracted patterns are then merged into a template and converted into a component in one of the common front-end frameworks, such as React or Angular. The approach is implemented in a tool called VizMod, which takes a mockup as input and generates a set of web components as output. The evaluation of VizMod on real-world web mockups shows high precision and recall, as well as code reusability. This approach can significantly improve the efficiency and quality of web development, as it reduces the manual effort required to create web components and ensures consistency and maintainability of the UI design. By automating the process of generating reusable web components from mockups, web developers can focus on other aspects of the development process, such as adding business logic, handling events, and connecting to databases or other sources[3].

3.1.3 Auto-Icon: An Automated Code Generation Tool for Icon Designs Assisting in UI Development.

designing icons for UI can be a time-consuming and laborious task for developers. In order to overcome this problem, a group of researchers has developed a novel strategy that uses deep learning and computer vision methods to automate the icon design process. The main objective of this approach is to provide developers with intelligent support that can significantly reduce the development time of icon design in UI. By automatically converting icon images to fonts with descriptive labels, this approach aims to improve UI rendering speed and assist developers with better code accessibility. Additionally, the approach detects the primary color of the icon to provide developers with more knowledge on the image. The authors have incorporated this method into existing automated code generation platforms to extend them beyond effective and descriptive coding. The effectiveness of Auto-Icon has been evaluated through a survey experiment, which showed that it significantly reduced the time and effort required for icon design in UI development[8].

3.1.4 ThingML: A Language and Code Generation Framework for Heterogeneous Targets

ThingML is approach, language and code generation framework developed to bring the benefits of Model-Driven Engineering (MDE) to practical use by a wide range of developers. It talks about the practical applications of MDE and outlines the lessons discovered during the course of the ThingML tool's three major iterations. The resulting strategy offers a family of code generators targeted at heterogeneous platforms and highlights the code generators' capacity for customization to increase productivity. The paper also highlights the use of the ThingML approach in the development of a commercial ambient assisted living system, demonstrating its practical applicability. Additionally, it describes the code generation framework built around the ThingML language, which allows developers to easily customize the code generators for the needs of their specific target platforms and projects[2].

3.1.5 Towards the integration of user interface prototyping and model-based development

Web Development has become one of the main areas of Software Development, covering the development of websites, web services, and web applications. The exponential increase in the use of internet-based services and applications has led to a corresponding rise in demand for web designers and developers. However, the proliferation of languages, frameworks, and libraries has illustrated the current state of immaturity of web development technologies, posing significant challenges in the development and maintenance of web applications. To overcome these obstacles, a novel strategy was developed that combines the conventional user-centered design methodology with model-based user interface development. With this integration, we hope to address the current web development. The approach allows designers to utilize prototyping tools, such as Adobe XD, to design graphical interfaces and then automatically convert them to (Vue.js + Bootstrap) code. This automation is achieved through the interpretation of the SVG file exported from Adobe XD, enabling the creation of an initial implementation for further development. The paper emphasizes the importance of raising the level of abstraction in the development process to address the technological immaturity of web development and streamline the development and maintenance of web applications. By automating the code generation from user interface prototypes, the approach aims to add flexibility, responsiveness, and efficiency in web development[9].

3.1.6 GAMBIT: Addressing Multi-platform Collaborative Sketching with HTML5

GAMBIT is a tool for multi-platform collaborative sketching using HTML5, and explores the importance of prototypes in design activities. prototypes are essential in design activities as they allow designers to test and refine their ideas before committing to a final design.In order to quickly get ideas onto an external medium where they can be discussed, refined, and saved for later reference, designers can find great value in sketching as a tool for creating low-fidelity prototypes. Users can sketch using any device they choose with the GAMBIT system, which supports electronic sketching as the primary method of interaction. The system is built with HTML5 and Javascript, making it accessible on any device with browsing capabilities. The collaborative nature of GAMBIT allows designers and end-users to work together in group sessions, introducing a richer conversation with the working design at hand [1].

3.1.7 Automated Code Generation using UML Design Documentation

The use of UML design documentation to automate code generation is a topic of interest in software development. This approach involves generating code automatically from UML diagrams, which can potentially increase efficiency and accuracy in software development. However, there are also potential drawbacks and challenges associated with this approach, such as managing changing requirements and ensuring ethical and legal compliance. Many code generators modify the Unified Modelling Language (UML) to what is known in the industry as Object Constraint Language (OCL) to get around some ambiguities that UML has. Overall, the use of UML design documentation for automated code generation is a complex and multifaceted topic that requires careful consideration and analysis[7].

3.1.8 From Design to Code: A Study on Generating Production Code From User Interface Design Software

The paper explores the creation of a user-friendly tool to streamline the generation of components for web development projects. It offers a solution that automates the process from UI design program to browser-runnable code, addressing the challenge of redesigning and rebuilding components for various projects. The primary goal is to reduce the setup time for component creation so that components can be generated that are compatible with all static pages and JavaScript frameworks. The tool aims to automate

the entire process, from fetching data from design software to interpreting the response and building usable code. Additionally, the paper discusses the importance of creating a tool that is open source and emphasizes the need for thorough documentation. The research also delves into usability testing and A/B testing to measure the effectiveness of the prototype. Overall, the paper presents a comprehensive approach to addressing the challenges of component generation in web development projects[10].

3.1.9 The UX Design- Frontend Development Boundary: Bridging the gap with WYSIWYG tools

Exploring the possibility of using WYSIWYG (what you see is what you get).minimizing the gap between the UX designer and the frontend developer. discussing the capability of such tools and evaluating their quality and efficiency. The paper proves the time that can be saved by such tools, its limitations in generating code that is hard to read or maintain by humans, and how it might not follow best practices.

[5]

3.2 Business Applications

Front-End Code Generation Tool:

Teleport-HQ: Is an web application design and prototyping tool that with the help of its Figma plugin designers can easily transfer their Figma designs to Teleport-HQ [12]. By doing this the process of design to development can be effortless made. Teleport-HQ goal is to improve the communication between designers and developers

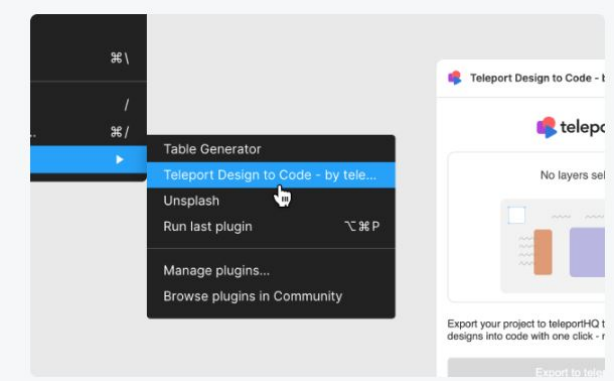

Figure 2: Export prototype wire frame from Figma to Teleport-HQ

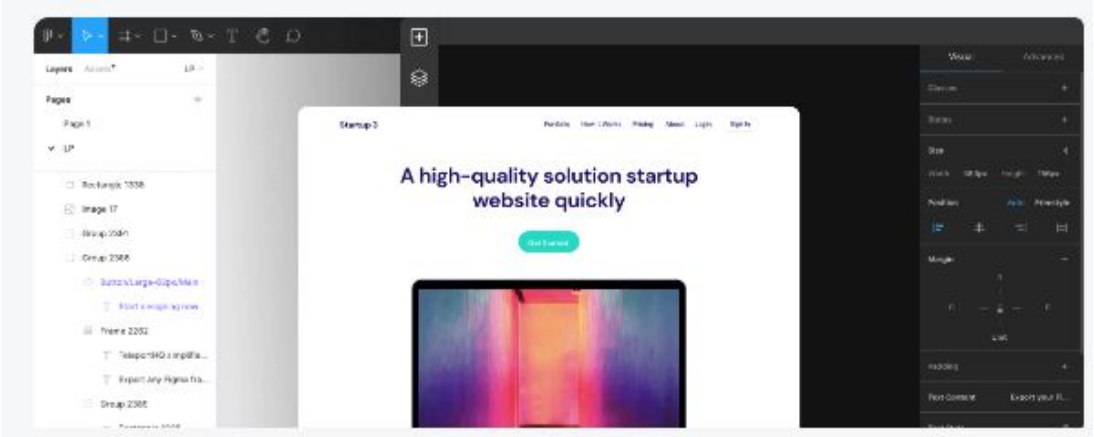

Figure 3: Edit the prototype in Teleport-HQ platform

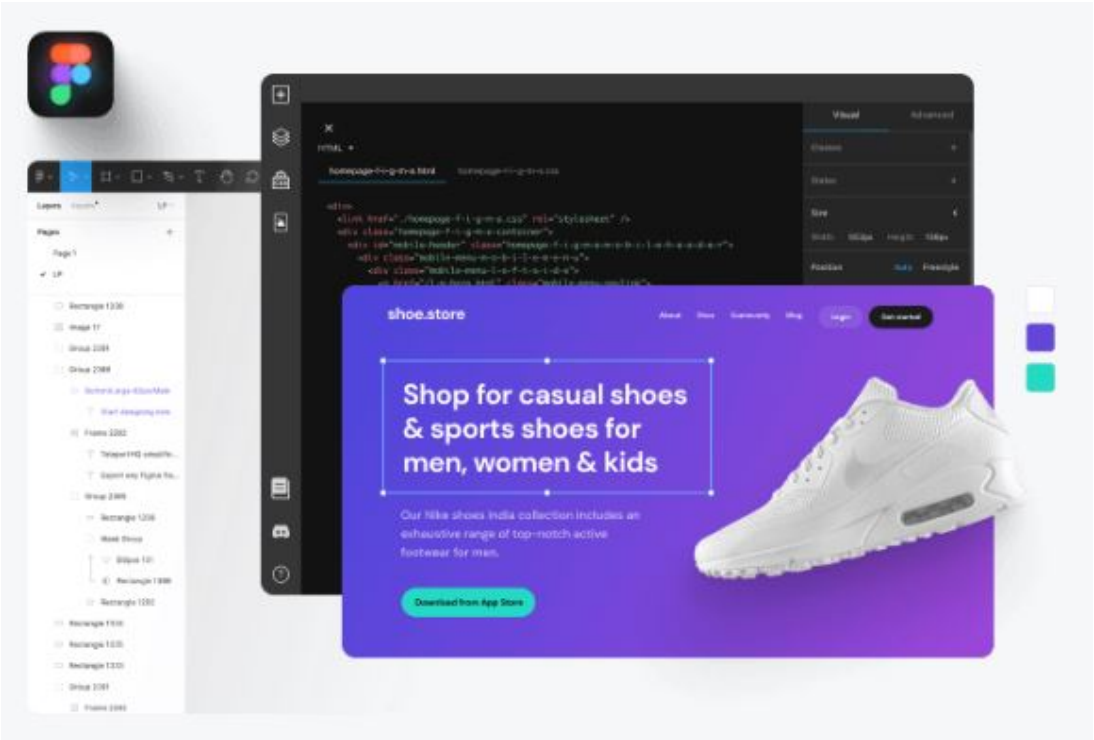

Figure 4: Preview and publish your fully coded responsive website

Back-end Code Generation Tool Using UML:

GenMyModel: Is a web-based modeling tool providing a UML editor with powerful features for creating UML class diagram, directly in the web browser.It supports for now class and use case diagrams and works with GitHub to host the generated code [11].

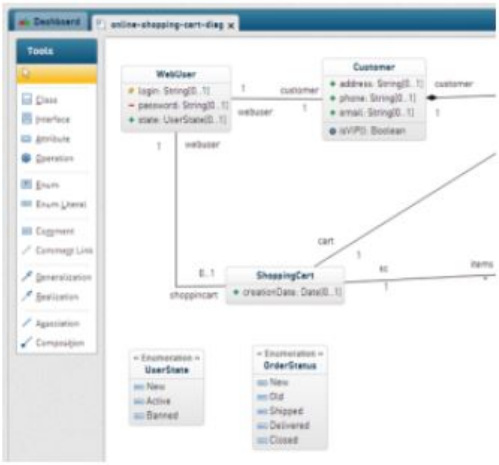

Figure 5: Creating class diagram

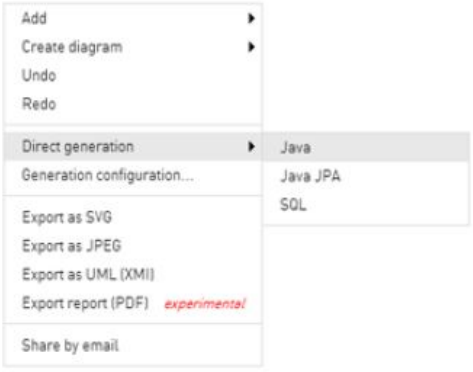

Figure 6: Generate Java code

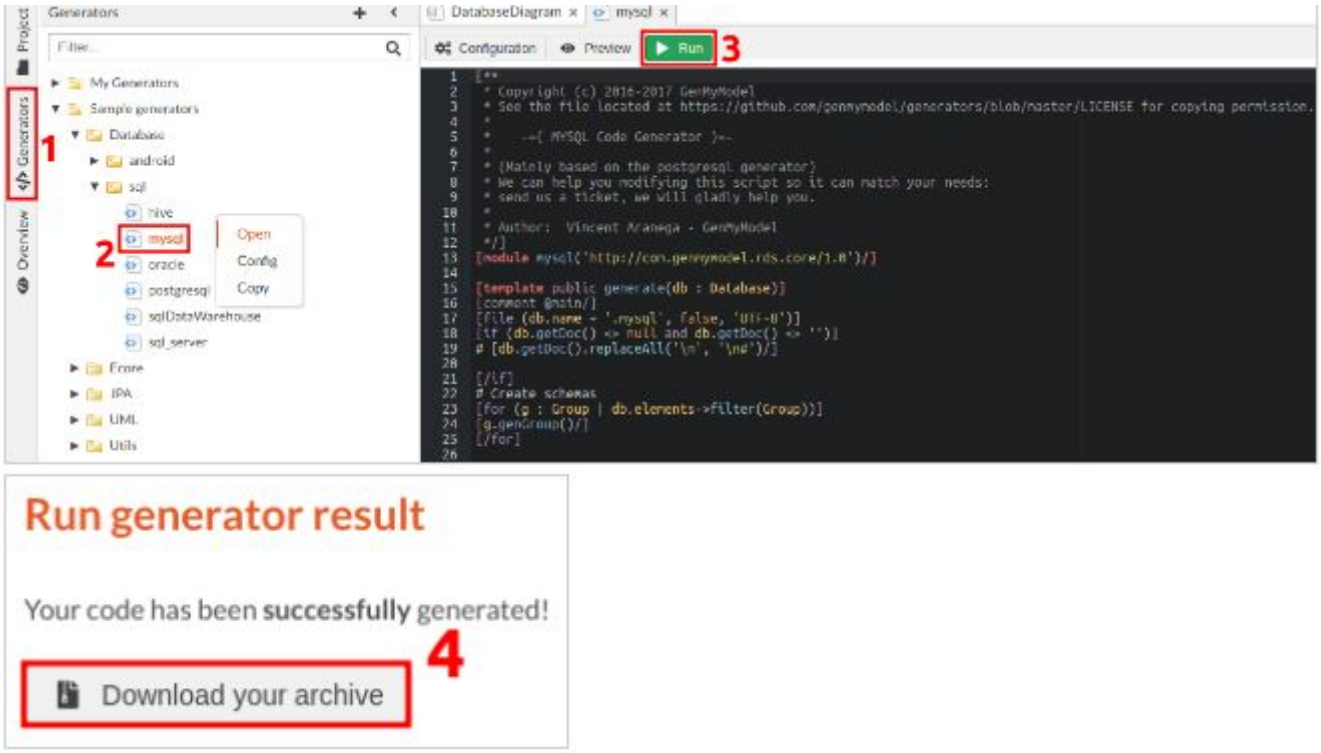

Figure 7: Online Code Generator

4 What is new in the Proposed Project?

- The system includes a back-end code generation feature that produces a dynamic server-side code. This provides the ability to create dedicated controllers to manage the flow of incoming requests and services files through the interpretation of UML class diagram.

- The system seamlessly incorporates the Figma sketching tool to automatically generate dynamic front-end code based on UI frames. This involves the conversion of UI frames into either HTML&CSS or React.js code. Through this integrated process, the system guarantees precision in translating visual designs, facilitating an efficient transition from the design phase to the final implementation of the web application.

- The system generates a complete application project directories with predefined structures. This capability expedites project setup, ensuring a quick and organized development environment. By automatically establishing a well-defined project structure, developers can dive into coding without the need for manual setup, enhancing overall efficiency in the early stages of application development.

- The system provides a tool to enhance user experience, including simultaneous multi-element conversion created for complex projects. It provides a built-in code inspection tool, elevating code quality, and offers versatile export options. This collection of mini features ensures a holistic and user-friendly development experience, promoting efficiency and code quality throughout the project life cycle.

5 Proof of Concept

The following is a showcase for the MetaMorph plugin demo where our objective is to convert figma design layers into HTML and CSS code.

Figure 8: Plugin Demo

Figure 8 shows the design layers in Figma awaiting transformation into front-end code by MetaMorph. In this case, we have a parent layer named "Navbar" and two sub-layers named "Logo" and "Username"

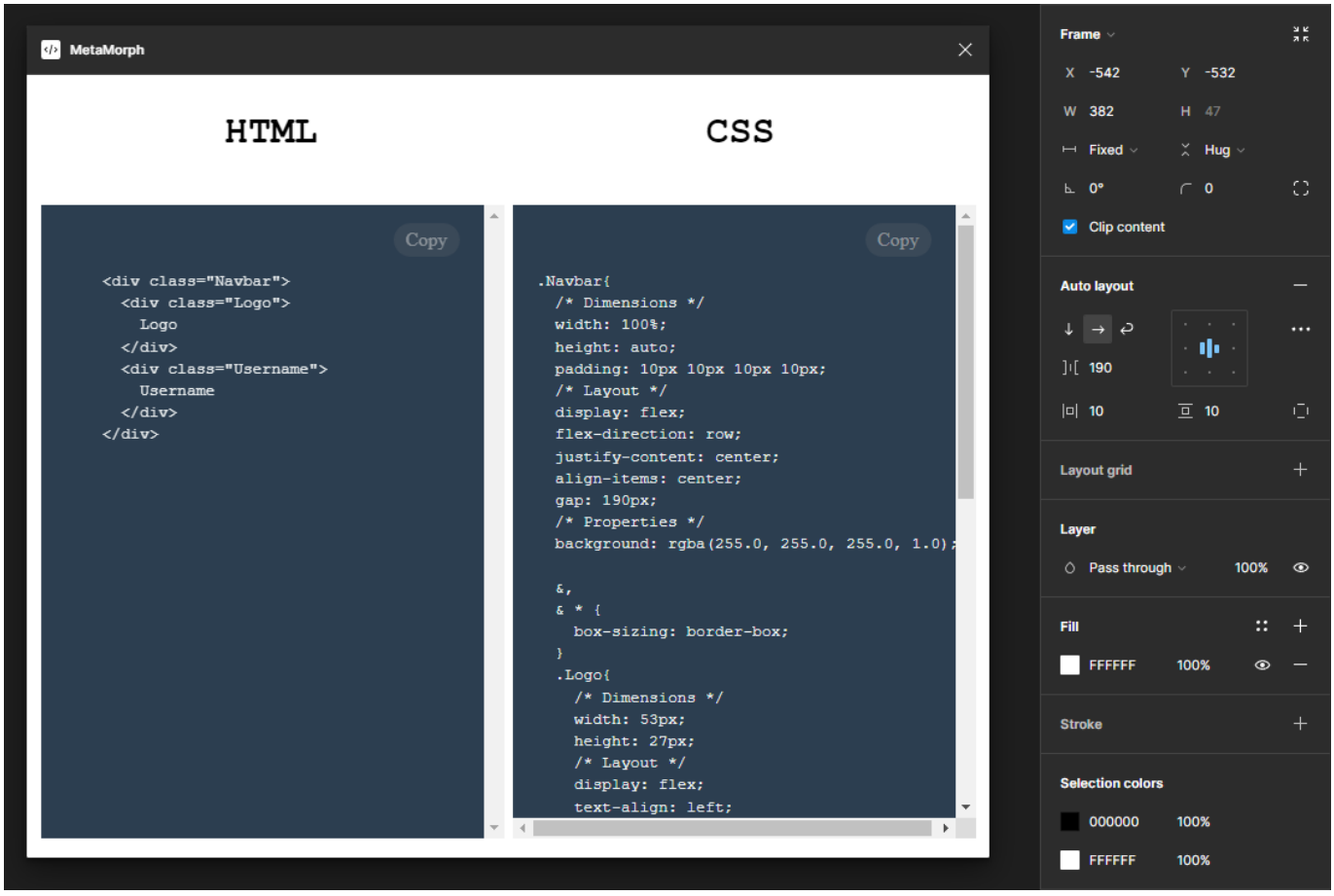

Figure 9: Plugin Demo

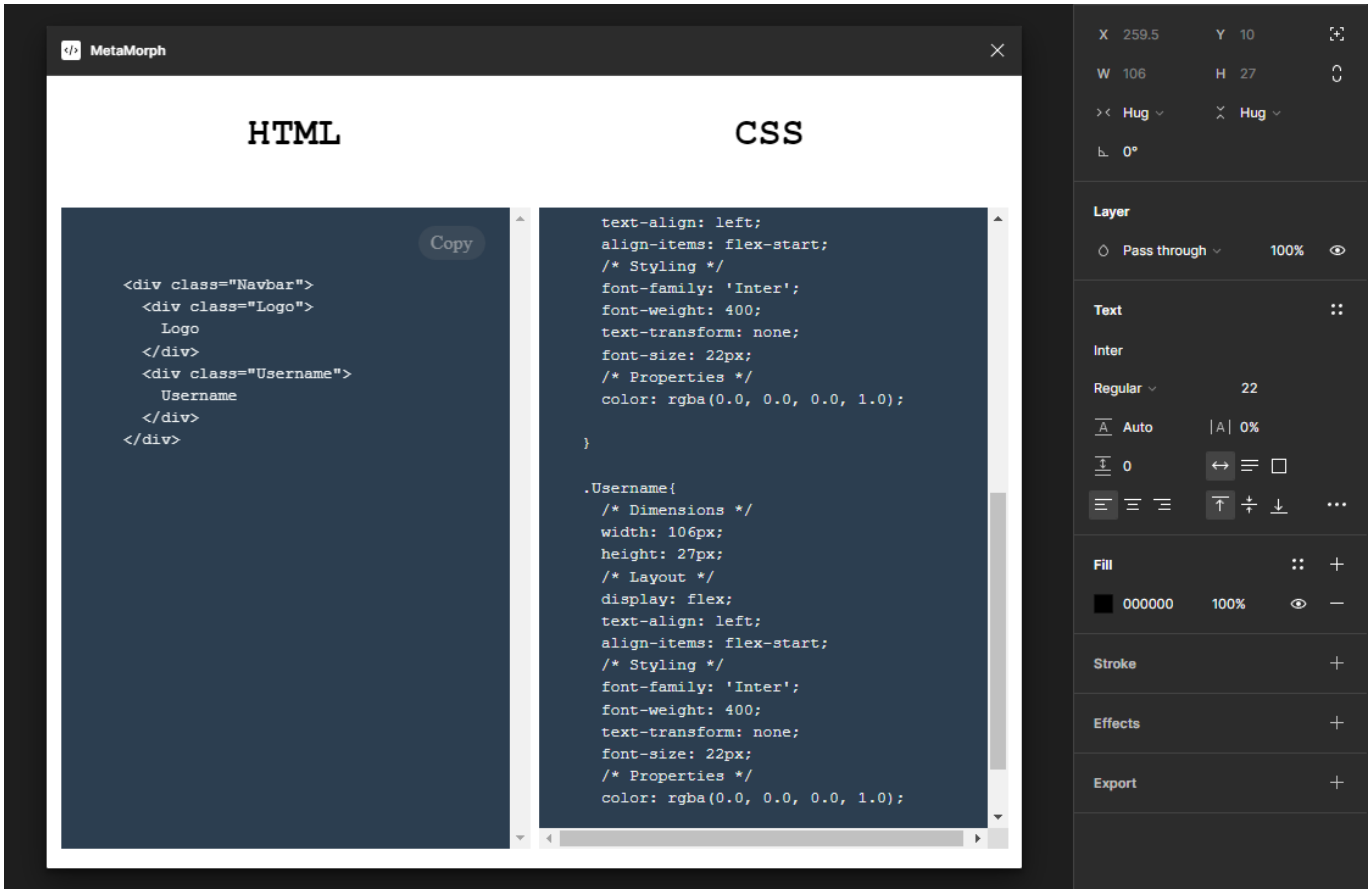

Figure 10: Plugin Demo

In Figure 9 and 10, We can see the MetaMorph plugin in action, the Plugin UI shows the code inspection tool where the user can view the generated code in HTML and CSS.

5.1 Potential Risk

The risk is associated with UML conversion can be the absence of appropriate API. If a suitable API cannot be found, we may have to look into another possibilities such as converting UML diagram using image processing technique.

6 Project Management and Deliverable

6.1 Deliverable

- SRS (Software Requirement Specification): Document describes system Non/functional requirements
- SDD (Software Design Document): Detailed plan of how the system will be developed.
- System Prototype
- Thesis

Table 2: Project Deliverable Timeline

Deliverable	Start Date	End Date	Duration
Proposal	17th October	20th November	34 Days
SRS	20th November	7th January	48 Days
SDD	15th January	7th March	51 days
Prototype	15th March	15th May	61 days
Thesis	7th June	7 July	30 days

6.2 Tasks and Time Plan

Task Name	Start Date	EndDate	Assign
Project Proposal	Oct 17-2023	Nov 20-2023	ALL
Project Architecture Diagrams	Oct 17 ,2023	Dec 4-2023	ALL
SRS	Nov 20-23	Jan7-2024	ALL
User Interface	Nov 20-2023	Dec 4-2023	Ahmed ,Mohamed
Extract Figma Jason Data	Nov 22-2023	Nov27-2023	Ahmed Ehab
Create Generation Module Class Template	Nov 28-2023	Dec 4-2023	Mahmoud ,Mohamed
System module Coordination	Nov 28-2023	December 4,2023	Ali ,Ahmed
Json Standardization for modules	Dec 5-2023	Dec 10-2023	Ahmed Ehab
html/css generation module	Dec 11-2023	Dec 24-2023	Mohamed Mamdouh
Linking export with github	Dec 11-2023	Dec 20-2023	Ali Ismail
System Database	Dec 11-2023	Dec 20-2023	Ali Ismail
Fullstack Integration	Dec 12-2023	Dec 30-2023	Ali ,Mahmoud
react module	Dec 16-2023	Feb 10-2024	Mohamed Mamdouh
Uml Diagram Interpretation	Dec 16-2023	Jan 10-2024	Ali ,Ahmed
Backend Dataservice Generation	Dec 30-2023	Mar 24-2024	Mahmoud ,Ali
Backend Controller Templates	Jan 1-2024	Mar 24-2024	Mahmoud ,Ali
Backend Java Code Generation	Jan 1-2024	Mar 24-2024	Mahmoud ,Ali
Link UI inputs with backend(Project Compilation)	Jan 13-2024	Mar 24-2024	ALL
Finalizing SDD	Jan 15-2024	March 7,2024	ALL
Prototype	Mar 15 -2024	May 15-2024	ALL
Thesis	June 7 -2024	July 7-2024	ALL

6.3 Budget and Resource Costs

In order to improve functionality, the system will make use of APIs, nevertheless, it is crucial to remember that this may result in additional expenses, such as use or subscription fees, depending on the APIs used.

7 Supportive Documents

As shown in figures 11,12,13,14 are the results of the survey made by Developers.

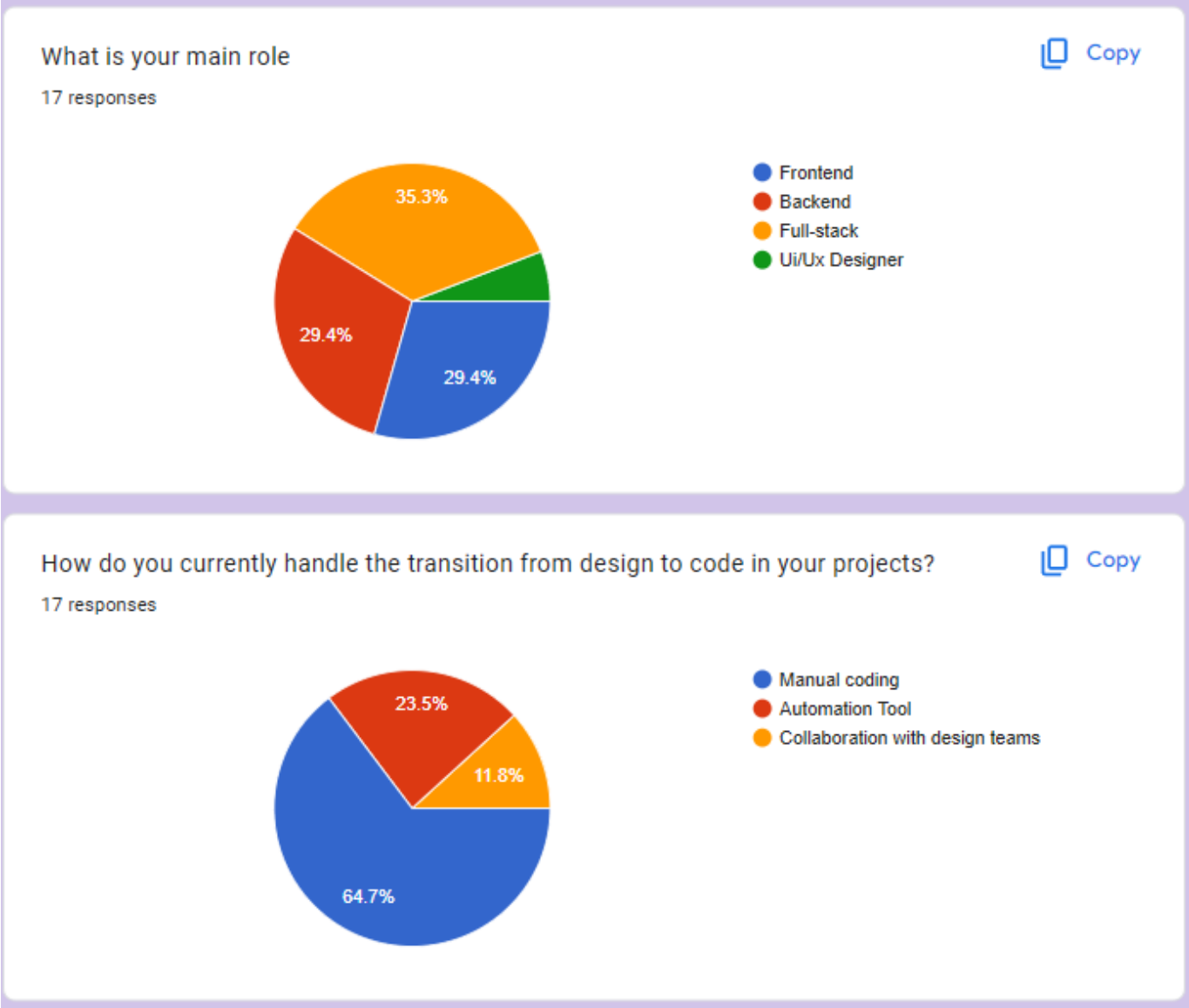

Figure 11: The Percentage of developers roles and the current way they use to change from design to code.

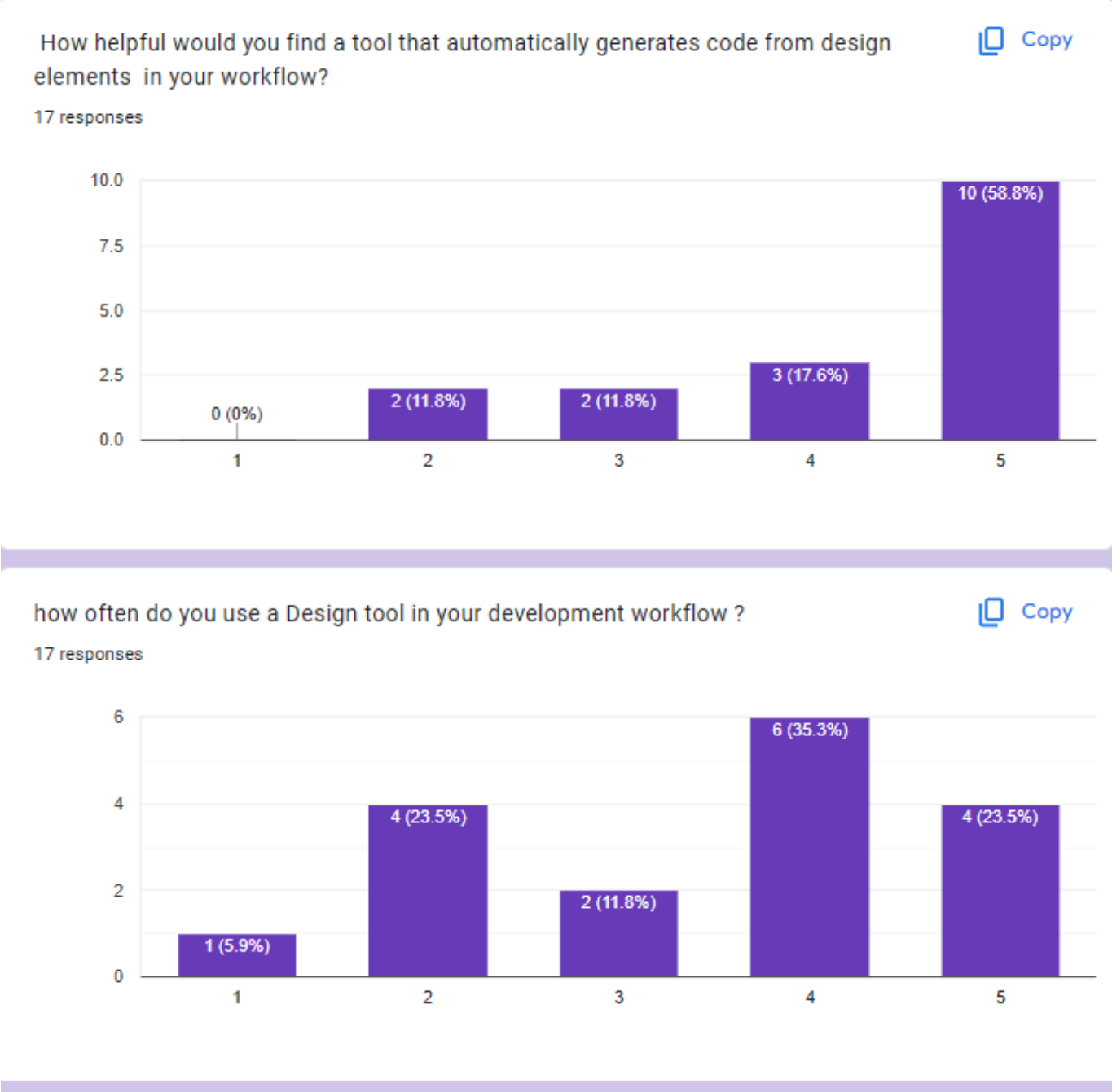

Figure 12: The number of Developers that would find the tool helpful and if they currently use a Design tool

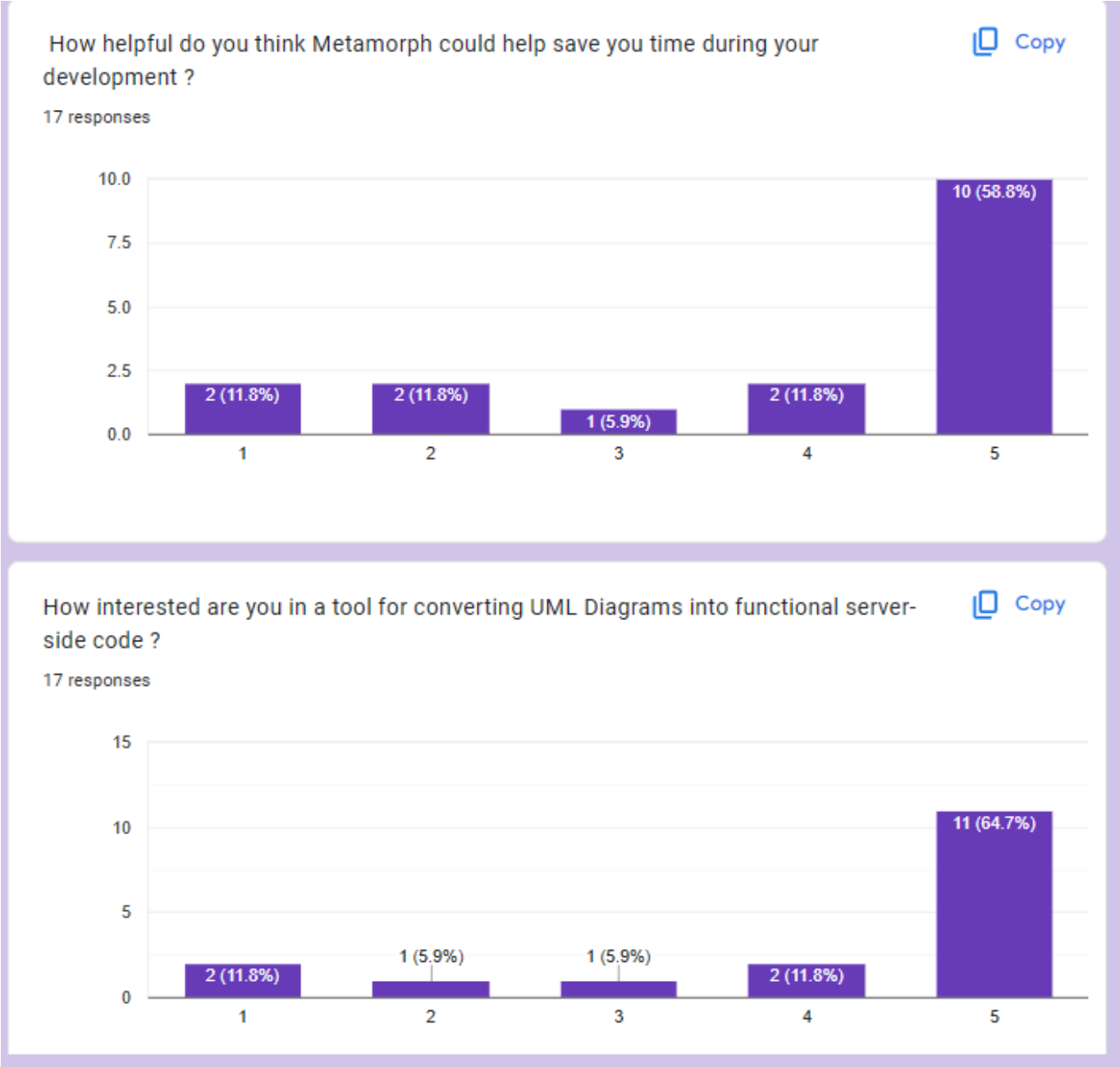

Figure 13: The number of Developers that would use MetaMorph

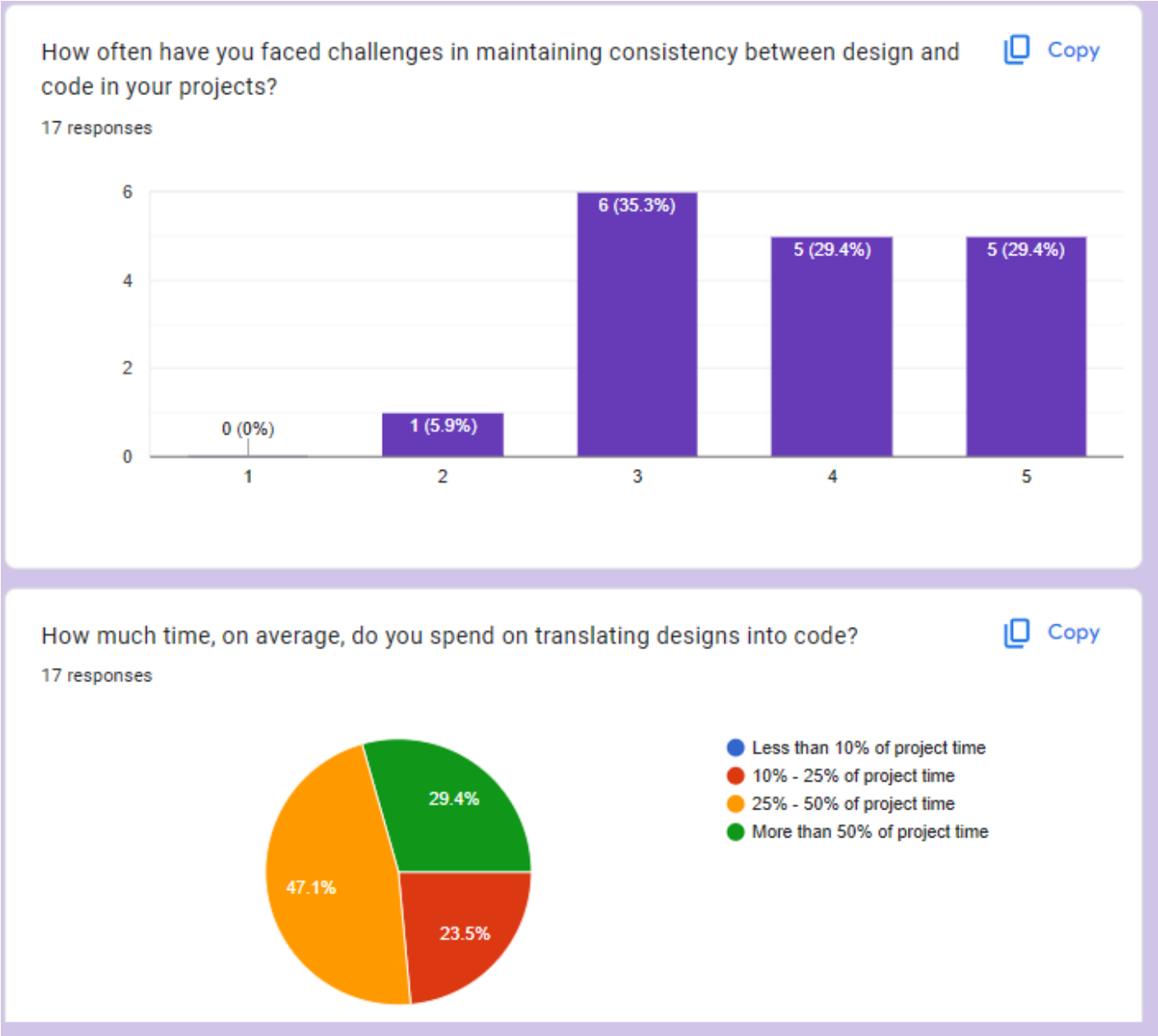

Figure 14: Frequency of Challenges in Maintaining Consistency Between Design and Code

8 References

References

[1] Ugo Sangiorgi and Jean Vanderdonckt. "GAMBIT: Addressing multi-platform collaborative sketching with html5". In: *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*. 2012, pp. 257–262.

[2] Nicolas Harrand et al. "ThingML: a language and code generation framework for heterogeneous targets". In: *Proceedings of the ACM/IEEE 19th international conference on model driven engineering languages and systems*. 2016, pp. 125–135.

[3] Mohammad Bajammal, Davood Mazinanian, and Ali Mesbah. "Generating reusable web components from mockups". In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 2018, pp. 601–611.

[4] Westley Knight. *UX for Developers: How to Integrate User-Centered Design Principles Into Your Day-to-Day Development Work. Apress*. www.apress.com, 2018.

[5] I. Enryd. "The UX-Frontend Development Boundary: Bridging the Gap with WYSIWYG Tools". PhD thesis. Dissertation, 2019.

[6] Mariana Bexiga, Stoyan Garbatov, and João Costa Seco. "Closing the gap between designers and developers in a low code ecosystem". In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 2020, pp. 1–10.

[7] Daniel J Buxton. "Automated Code Generation using UML Design Documentation". In: (2021).

[8] Sidong Feng et al. "Auto-icon: An automated code generation tool for icon designs assisting in ui development". In: *26th International Conference on Intelligent User Interfaces*. 2021, pp. 59–69.

[9] Catarina Machado and José Creissac Campos. "Towards the integration of user interface prototyping and model-based development". In: *2021 International Conference on Graphics and Interaction (ICGI)*. IEEE. 2021, pp. 1–8.

[10] Albin Frick. *From Design to Code: A Study on Generating Production Code From User Interface Design Software*. 2022.

[11] *GenMyModel*. `https://app.genmymodel.com/`. Accessed: 15/11/2023.

[12] *TeleportHQ*. `https://teleporthq.io/`. Accessed: 15/11/2023.