Implementing Large Language Model API For Interview Training Based On Job Description

Wisnu Uriawan * , Raden Ibnu Huygenz Widodo , Ray Ramadita , Reza Fahlevi Herdiyanto , Rifqi Syekhi Marsaputra , Silvia Nurrobianti

Posted Date: 1 July 2024

doi: 10.20944/preprints202407.0049.v1

Keywords: large language model; job interview system; agile development

Article

Implementing Large Language Model API For Interview Training Based On Job Description

Wisnu Uriawan *, Raden Ibnu Huygenz Widodo, Ray Ramadita, Reza Fahlevi Herdiyanto, Rifqi Syekhi Marsaputra and Silvia Nurrobianti

Informatics Department, UIN Sunan Gunung Djati Bandung, Jawa Barat, Indonesia; ibnuwdodo@gmail.com (R.I.H.W.); rayramadita12@gmail.com (R.R.); rfahlevih@gmail.com (R.F.H.); rifqisyekhi@gmail.com (R.S.M); silvianurrobianti@gmail.com (S.N.)

* Correspondence: wisnu.uriawan@uinsgd.ac.id

Abstract: Good communication skills are essential for prospective workers, particularly college students, to clearly convey their abilities and understanding during job interviews. Despite the high demand for strong communication skills among employers, many students struggle with inadequate experience and skills, leading to reduced confidence and lower chances of securing desired jobs. Existing tools and resources for improving communication skills are limited, especially free ones, leaving students underprepared and less confident in interviews. To address this issue, a comprehensive solution is needed that provides free, customizable tools to enhance students' interview communication skills. Invisor is a web-based self-interview training application that leverages AI and Machine Learning to analyze users' behavior, responses, and facial expressions during simulated interviews. Accessible via desktop, laptop, or mobile devices, Invisor offers interactive and voice-recorded feedback to help students prepare for interviews effectively. By employing AI technologies, Invisor generates personalized interview questions based on job descriptions, enhancing the realism of the interview experience and providing tailored feedback to improve users' responses. Developed using agile methodologies, Invisor has undergone iterative enhancements, integrating user and stakeholder feedback promptly to improve usability and functionality. The combination of AI-driven simulations and agile development practices enables Invisor to meet current user needs and anticipate future demands, providing a robust tool for interview preparation in a competitive job market. This paper outlines the development and implementation of Invisor, demonstrating its effectiveness in enhancing interview skills and confidence among students and job seekers.

Keywords: large language model; job interview system; agile development

1. Introduction

Good communication skills in interviews are very important to be possessed by prospective workers, in this context, college students. With good communication skills, prospective workers can clearly convey the abilities and understanding they have so far related to the job position they are interested in. According to a recent National Association of Colleges and Employers (NACE) report, 73.3% of employers seek employees with strong written communication skills and 58.8% seek employees with strong verbal communication skills (NACE, 2022). However, in this case not all students have good communication skills and are ready to face the world of work.

Students often face obstacles in facing job interviews due to lack of experience and adequate communication skills. Limited interview experience and lack of knowledge on how to respond to questions or interact with interviewers can reduce their confidence and hinder their chances of getting the job they want. Moreover, finding a job today is more difficult than ever, even for those who are highly qualified and trained in various specialized fields without good communication skills.

 doi:10.20944/preprints202407.0049.v1

Currently, access to many free tools and resources that can help students communicate better is limited. As a result, it is difficult for students to find a fully-featured platform to help them prepare for interviews, so their skills are not really practiced and they feel insecure when facing interviewers.

In response, there is a need for a solution that can help students overcome this problem in improving their communication skills in interviews by providing free access to interview training tools that cover all aspects of interviewing, from preparation to practice answering questions. These tools should also be designed to provide useful guidance and effective practice, and to boost students' confidence.

Communication training should also be personalized according to the needs of each student so as to generate relevant responses. For example, some students may need to address their anxiety during interviews, while others may need to learn to regulate the intonation of their voice.

For a better user experience, these training systems should also provide accurate feedback to students to help them correct their mistakes and improve their ability to better face interviews.

Invisor is a self-paced interview training app allowing students to improve their ability to prepare for interviews with the help of AI. The system uses Machine Learning technology to analyze users' behavior, responses, and facial expressions during simulated interviews. Web-based applications that can be accessed through desktop, laptop or mobile devices as long as the device has a browser. Users can interact with the application by clicking or pressing buttons on the web page. There is also a voice recording that will be requested by the website to provide feedback regarding the user's interview answers. Thus, students can feel more prepared and confident when facing interviews, increasing their chances of successfully landing the desired job.

This paper's remainder is structured as follows: Section 1 introduces the credit scoring and background. Section 2 related work. Section 3 are methods. Section 4 result and discussion. Section 5 concludes this paper.

2. Related Work

Recent research has made AI-based interview systems a major area of interest. For instance, Heimerl, A. et al (2022), aimed to develop and assess systems that use technologies like learning analytics, adaptive learning, and automatic evaluation in order to enhance learning techniques while also lowering bias and guaranteeing fair AI educational systems. Many algorithms, including natural language processing (NLP) algorithms for sentiment analysis, topic extraction, and classification, are used in this study. Deep learning, utilizing models like BERT, is used for sentiment analysis, computerized assessment of student writing, and handwriting identification. The primary findings of this study demonstrate that using visual analytics in video-based learning can enhance student learning outcomes. Handwriting recognition and text analysis are improved thanks to improvements in deep learning models. Adaptive value instruction reduces bias in tasks requiring misinformation classification. A two-stage adaptive testing method satisfactorily balances measurement precision and item exposure.[1].

Lee B. et al. (2021) developed an AI-based interview system to improve the fairness and effectiveness of job interviews. Using deep learning technology, the system analyzes online interview responses and data, including biological signs. In addition, voice-to-text conversion can be checked through natural language processing (NLP). According to their findings, in terms of organizational fit, job fit, and evaluation procedures, five large Korean organizations have a job satisfaction rating of 85% [2].

In the research of Qin C. et al. (2019), a remote recruitment job interview system was created using deep learning technology. This method is used to analyze applicants' body language and their online responses. This method collects information from internal and external applicant sources, such as language, tone of voice, facial expressions, and pulse. The main objective of this study is to create and implement an AI-based interview system that can improve the impartiality and effectiveness of

 doi:10.20944/preprints202407.0049.v1

the recruitment process. The results show that this interview method can improve organizational fit, save time and costs, and increase organizational satisfaction by up to 85% [3].

In addition, Horat S. et al. (2019) conducted a study on virtual job interview training. Using multimodal behavioral analysis, the system evaluates how participants perform interview simulations and provides relevant feedback to help them improve their interview performance. In addition, the algorithm used provides accurate and relevant feedback. The main objective of this study is to create a system that can train job interviewers to provide appropriate and useful behavioral feedback. The findings of this study indicate that this system has the ability to provide appropriate and precise feedback for each person [4].

In the study, Shen D. et al. (2018) used the mean-field variational inference method. The main objective of this study is to develop an intelligent system for assessing job interviews that can use data from various sources, such as resumes, job descriptions, and interview questions, thus allowing for more accurate assessment of candidates for the position. The goal is to improve the recruitment process by using machine learning to simulate complex interactions between components. The results of this study show how well this system models the relationship between interview assessments, resumes, and job descriptions. In addition, the system can distribute important topics related to applicant performance. [5].

Pratap Singh, et al. (2023) examined the role of AI in human resource management (HRM). Using literature review, bibliometric analysis, and data mining methods, the study investigated the use of artificial intelligence in talent development, employee assessment, and candidate selection. In addition, genetic algorithms are used for workforce planning and performance appraisal, and machine learning is used for employee turnover prediction and managerial decision making. According to an analysis of 18 years of data from Scopus, Emerald, and Jstore libraries, publications on AI and HR increased exponentially from 2017 to 2019. The results indicate that the adoption of AI tools and interdisciplinary team collaboration is increasing in HR organizations in India. It also enhances our understanding of how AI can improve the effectiveness and efficiency of HRM practices [6].

To assess verbal expression in audio interviews, Thompson. et all. (2021) used the pairwise comparison method. This method involves raters comparing audio clips based on given definitions and behavioral anchors. Using the results of this comparison, the Elo score of each clip is calculated. This is a technique originally used to assess chess players. The objectives of this study were to measure audio features in video interviews, define verbal expressions, demonstrate the method is valid, provide preliminary validity evidence, and support the application of a deep learning model to automate the measurement of acoustic features in employment screening. The results show that the method is reliable; there is a correlation coefficient (r=.93) between the Elo scores and the Behaviour-Based Rating Scale (BARS). This indicates that there is high agreement between the two methods. In addition, this study shows that significant differences in verbal expression can be identified through the responses given during the interview [7].

A two-cycle action research was used by Dewanti Ratna. et all. (2021) to improve the job interview skills of students of SMK AAG Adisutjipto Aviation. Cycle 1 conducted an initial assessment, introduced artificial intelligence, and observed initial conditions. Cycle 2 concentrated on further assessment and improvement. The results of this study show that AI technology can help students prepare for real-world job interviews by improving fluency, pronunciation, grammar, and other details. With the help of this technology, students can not only develop their skills independently but also receive feedback on what they are doing [8].

Bhere and Dudhat (2023) conducted research to propose the application of artificial intelligence-based chatbot technology that assists users in interview preparation thereby allowing users to interact just like a real interview while increasing their confidence and expertise. The application incorporates methodologies such as voice to text conversion and text to voice synthesis to assist users in improving their interview skills and personality traits. These methodologies allow users to provide verbal responses which are then converted into text by the system, and vice versa, providing verbal

 doi:10.20944/preprints202407.0049.v1

feedback from the chatbot to the user. In addition, the implementation of a virtual Interview Coach in the chatbot aims to improve users' confidence and skills in interview preparation [9].

3. Methodology

Agile methodology is an iterative and incremental approach to software development that focuses on collaboration, flexibility, and rapid response to change. This methodology consists of several main stages which include: Planning, Requirement Analysis, Design, Development, Testing, Deployment, and Review and Retrospective.

The agile methodology is a method being used in software development. Agile methodologies were introduced in the 1990s to drive transformation in software development and are now widely adopted by many software development organizations. It consists of a set of revolutionary practices which allows a team to manage a project by dividing long wait times into several iterations with shorter cycle times. Each iteration is known as a sprint and consists of four phases: requirement phase, implementation phase, testing phase, and customer review to collect feedback that can be worked out in the next iteration.

There are four core principles:

Individuals and interactions over process and tools. Two ways communication among developers and customers is more important than system processes. Working software over comprehensive documentation. Software that is successfully delivered is more important than the process of documentation in software development. Customer collaboration over contract negotiation. Keeping customers in the loop during software development can receive better customer satisfaction than performing contract negotiations with customers. Responding to change over follows a plan. Always ready to adapt and respond to unpredictable changes is better than following a plan. Agile methodology is widely used in different types of projects, including business process management, IoT projects, e-Commerce web applications, and quality assurance processes in software development. Research papers studied show that the agile methodology is being applied in IT projects and software development projects. Compared with applying the waterfall methodology, the success rate of the IT project is 24% higher and 2 times higher in software development projects. Research studies also show that with agile Scrum projects, products are delivered faster at 37% compared to the waterfall model. Scrum, Crystal, XP, and Kanban are some of the agile methods that are constructed based on the four core principles in agile methodology to suit the different project's needs. DevOps culture is combined with agile methodology to improve continuous engagement between the operation team and developer team in a project.

Agile methodologies emerged to reduce the risk of not completing the development of large systems due to budget, technological or resource constraints, among other reasons. The emergence of agile methodologies made it possible to improve the success rate of software development projects and reduce the budget consumption of projects that are finally abandoned.

Agile methodologies divide the overall system into deliverables (modules or subsystems), so that the customer uses or checks those deliverables before the entire system is completed. Agile methodologies are based on 4 values and 12 principles included in the Manifesto for Agile Software Development. These methodologies focus only on the development of a part of the overall system, deploying fully functional products for that part of the system in a short period of time (maximum of 4 to 6 weeks).

In addition, some of the characteristics of agile software development methodologies, in contrast with traditional methodologies, are: teams must be small, the deliverables to be developed must be negotiated with the client and changes can be introduced in the project at any time. However, in some cases, those features could turn into disadvantages. For example, in an enterprise software development project, a small team of developers, as suggested by agile methodologies (in Scrum, 8 developers), is not sufficient when time is pressing. Additionally, whenever an unexpected change needs to be performed to software, there could be budget issues, since the development team could

 doi:10.20944/preprints202407.0049.v1

have to deal with it using the original budget. Moreover, if the decision for the development priority of each deliverable primarily relies on the customer, this may not help the development team to be productive.

3.1. Planning

The planning stage is the beginning of each iteration in the Agile methodology. At this stage, the project team comes together to define the goals of the iteration and determine the features or user stories to be developed. Planning involves stakeholders to ensure that business needs and priorities are aligned with technical goals.

Planning starts with a sprint planning meeting, where the product backlog (a list of all desired features) is evaluated and its priority is determined. The team then decides how much work can be completed during a sprint (a fixed period of time, usually two to four weeks). The work selected for the sprint is put into the sprint backlog, which will be the team's main focus during the iteration.

It is important at this stage to ensure that each team member understands their roles and responsibilities and has a clear picture of the sprint's goals. Project management tools such as JIRA or Trello are often used to keep track of tasks and facilitate collaboration. Effective and transparent communication is key to ensuring that all team members have a common understanding of what should be achieved during the sprint.

3.2. Requirement Analysis

This stage focuses on collecting and understanding user and business requirements. In Agile methods, requirements are usually expressed in the form of user stories (short descriptions of functionality from the user's perspective). Each user story has acceptance criteria that must be met for the story to be considered complete.

Next, the development team works with stakeholders to clarify and prioritize the user stories. Often this process involves discussion sessions to ensure that all requirements are captured properly. Typically, this process uses prototypes or mockups to provide a visual representation of how the feature will look and function.

Once all the requirements have been collected and prioritized, the user stories are put into the product backlog (a document that contains all the features for the final product and is always updated). With a clear backlog, the team can work more efficiently in the next stages.

3.3. Design

The design process in Agile methods focuses on designing solutions that meet the identified needs. In an Agile context, this process is incremental and gradual, allowing for adjustments and changes during development. Usually, the design starts with the creation of a wireframe or basic architecture that decomposes the main components of the system into smaller forms to make it easier for them to interact with each other later. Design helps in indicating equipment and furthermore give a virtual diagram of the framework/programming to be planned.

Design plays an important role in ensuring that the proposed solution is scalable, maintainable, and meets quality standards. Design teams often use UML diagrams, wireframes, and sketches to communicate design ideas with the development team and stakeholders.

During this stage, it is important to maintain close collaboration between the design team and developers to ensure that the design created can be implemented efficiently. Periodic design reviews are conducted to ensure that the proposed solution is still relevant to business needs that may change.

3.4. Development

The development stage is where real code is written to implement the features that have been planned and designed. Development in Agile is done in short sprints, with the goal of producing

 doi:10.20944/preprints202407.0049.v1

increments of a working product by the end of each sprint. The development team works on the sprint backlog and focuses on completing the prioritized user stories.

In Agile development, collaboration and communication are key. Development teams usually work in pairs (pair programming) or small groups in order to maintain code quality and the exchange of information is not too difficult. To ensure that the code written can be integrated into the main system and deployed into the production environment efficiently, the development team will usually implement continuous integration (CI) and continuous deployment.

In addition, it is important to continuously integrate feedback from stakeholders and end-users. This is done to ensure that each iteration of the product is closer to the final expectation. The use of a version control system (VCS) such as Git is also very important in order to track every code change that occurs and allow rollback if an error occurs.

3.5. Testing

The testing process is one part of the Agile method to ensure that each increment of the product functions properly and meets the assessment criteria. This process is carried out continuously, both through manual testing and automated testing in order to ensure that bugs can be fixed as quickly as possible.

Software quality management is a critical segment of development process of a software for the quality engineers. It becomes even more tedious and complex task when the development shifts to usage of Agile tools and techniques. As the methodologies and issues managed in the Agile Software Development (ASD) framework are absolutely poles apart from conventional software development quality assurance architecture and testing phenomenal. In ASD a concentration on people is more focused than processes and methods.

Software testing is one of the most important phases in the software development life cycle. Software testing cannot ensure successful outcomes until and unless done perfectly. For enhanced testing, it is extremely important to identify and prioritize the parameters that influence the testing process.

Every one of the units created, testing process is divided into several stages, ranging from unit testing (checking small parts of the code), to integration testing (making sure the various components work properly), and acceptance testing (making sure the entire system meets user needs). Typically, testing teams use Test-Driven Development (TDD) to conduct tests before the code is actually written in its entirety to ensure all functionality is thoroughly tested.

Conducting early testing is one of the key principles in Agile methods, this is done to enable teams to detect problems early which in turn can reduce repair costs and improve overall product quality. Therefore, the testing process is not a separate phase, but an ongoing part of the development process.

3.6. Deployment

The deployment process in the Agile method focuses on developing the finished product into the production environment.The little units are converted into one useful unit and are tried. This process is done regularly to ensure users can see changes and use new features as quickly as possible. Usually, this is called Continuous Deployment, which is a process where every code change that passes automatic testing will be immediately applied to the production environment.

This process is divided into several stages, from building the build, final testing in the staging environment, and launching the product into production. This process is usually automated using deployment pipelines such as Jenkins, GitLab CI/CD, and Bamboo to ensure each stage is performed consistently and with minimal errors.

After successful deployment, the development team usually conducts monitoring and logging to ensure that the new features implemented can function properly in accordance with the requirements and do not cause problems in the existing system.

 doi:10.20944/preprints202407.0049.v1

3.7. Review and Retrospective

The last process in each iteration of the Agile method is review and retrospective. This is done to assess the results of the completed sprint. The team holds a sprint review meeting where they demonstrate the developed features to the stakeholders. In addition, the review process is an opportunity to get feedback and ensure the results are in line with expectations.

A retrospective is an internal team meeting that focuses on evaluating the team's work process during the sprint. The goal is to identify what went well, what didn't, and how the process can be improved for the next sprint. Teams use various retrospective techniques, such as "Start, Stop, Continue" or "What Went Well, What Didn't Go Well, and Actions for Improvement," to extract insights and implement improvements.

Reviews and retrospectives are key elements of Agile because they encourage a culture of continuous improvement and allow teams to adapt and evolve over time. Through continuous feedback cycles, teams can improve product quality and the efficiency of the overall development process.

4. Result and Discussion

4.1. Result

The implementation of a large language model into a web-based interview application using agile methodology has gone through several stages, each contributing to the refinement and effectiveness of the system. Initially, the integration focused on ensuring that the AI could accurately generate relevant interview questions based on job descriptions provided by users. This stage involved extensive training and fine-tuning of the model to handle diverse inputs and produce meaningful, contextually appropriate questions.

As the development progressed, subsequent stages emphasized user feedback and iterative improvements. By incorporating user insights and conducting regular testing, the application was continuously enhanced to improve its usability, functionality, and overall user experience. The agile methodology facilitated rapid prototyping and the ability to adapt swiftly to changing requirements, ensuring that each iteration brought the system closer to meeting the users' needs effectively. The result is a robust, scalable interview preparation tool that leverages advanced AI capabilities to deliver personalized and accessible training for job seekers.

4.2. Planning

Firstly, the planning phase will begin with a comprehensive sprint planning session. In this session, the development team, including stakeholders and possibly end users (students), will gather to define the objectives of each iteration. This involves breaking down the overall goal of developing an AI-driven interview simulation into manageable tasks and features. Each task will be prioritized based on its importance and dependencies, ensuring that the most critical functionalities, such as job description processing, question generation, and voice-to-text conversion, are addressed early on. This session will also establish the duration of each sprint and the expected deliverables, setting a clear timeline for the development team to follow.

Secondly, communication and collaboration tools using ClickUp and Whatsapp Group for many discussion will be utilized to manage tasks and facilitate team interaction. These tools will help in tracking the progress of each feature implementation and ensure that any changes or updates are documented and communicated effectively across the team. Clear communication channels will be established to address any questions or concerns that arise during the development process, promoting transparency and alignment with the project's objectives. For documenting all of progress and requirement document we use Google Docs because have feature collaboration when doing some requirement document.

 doi:10.20944/preprints202407.0049.v1

Lastly, the planning phase will emphasize the importance of defining roles and responsibilities within the development team. This includes assigning specific tasks related to AI integration, user interface design, backend development, and quality assurance. By clarifying roles early on, team members will have a clear understanding of their contributions to the project and how their work impacts the overall functionality and usability of the Invisor app. Regular review meetings and checkpoints will be scheduled to evaluate progress and adjust priorities if necessary, ensuring that the development process remains agile and responsive to any emerging challenges or opportunities.

4.3. Requirement Analysis

Invisor is an app project that aims to help university students deal with their challenges in finding a job and their lack of experience in job interviews. Based on the observations made, the main need of students in this context is interview training that is independent and accessible for free or for a small fee. Therefore, Invisor was developed to provide support to students for personalized interview training with the help of artificial intelligence (AI) that allows them to develop interview skills independently.

To achieve the goal of Invisor, which is an interview simulation that is expected to help students to be better prepared for interviews, students or users will input job descriptions that they get from job portals or the like into the application, then the application will process them and generate related questions. The user will answer the questions using their voice through their device's microphone which will then be converted to text to allow for correction of wording errors. The app will then score the answers and provide feedback and a score for each answer.

To make Invisor easily maintainable and usable in the long term, the architecture chosen was microservices utilizing technologies such as:

1. Next.js for frontend: Next.js is utilized for frontend improvement since of its progressed highlights to make energetic and intelligently client interfacing.This system gives server-side rendering and inactive location era to progress the execution and SEO of your web applications.The framework's capacity to handle both client-side and server-side rendering guarantees a consistent client involvement, coming about in quicker stacking times and smoother in-app route.Also, Next.js coordinating well with other libraries and instruments, making it a flexible choice for complex web applications like Invisor.
2. Tailwind CSS for UI System: Tailwind CSS was chosen to make strides the plan and responsiveness of the application.This system gives a utility-first approach that permits designers to form custom topics specifically within the markup.It gives a profoundly customizable and proficient workflow with a steady fashion that can be effectively connected over your applications.Tailwind CSS' responsive plan highlights guarantee your app looks and capacities well over a assortment of gadgets, from desktop to versatile, giving clients a outwardly engaging and easy-to-use interface.
3. Flask for API Apparatuses: Flask is utilized as an API tool because of its effortlessness and adaptability in building Tranquil APIs. It may be a lightweight WSGI web application system in Python, outlined to induce begun rapidly and effectively, and with the capacity to scale complex applications. Flask's measured quality permits designers to select the components they require whereas keeping the center basic and extensible. It's perfect for building high-performance, versatile APIs that can handle numerous information demands from the front-end and communicate effectively with back-end frameworks.
4. PostgreSQL and Firebase for client database: PostgreSQL and Firebase are utilized together to oversee client information effectively.PostgreSQL, an progressed social database, gives solid information astuteness, complex inquiry capabilities, and productive capacity of organized information, making it perfect for handling value-based information and the social perspectives of applications.Firebase gives real-time database capabilities and simple integration with other Google administrations, making it appropriate for overseeing client confirmation, real-time overhauls, and versatile cloud capacity.This combination of databases guarantees solid information administration and progresses the by and large execution of your application.

 doi:10.20944/preprints202407.0049.v1

5. Laravel for fundamental backend: Laravel, a PHP system, was chosen as the most backend since of its rich language structure and wealthy highlight set.Laravel rearranges common errands like steering, confirmation, and caching that are basic for building a vigorous backend foundation.Laravel's built-in devices and libraries empower fast advancement and support, and permit applications to handle a wide run of client intuitive and information handling assignments effectively.Bolster for the MVC engineering moves forward code organization and makes it simpler to create, test, and grow applications as required.
6. Python for machine learning backend: Python is utilized for machine learning backend since of its wealthy AI and machine learning libraries and systems such as TensorFlow, PyTorch, and scikit-learn.Python's effortlessness and coherence make it an perfect choice for executing complex machine learning algorithms and models.Within the setting of Invisor, Python is utilized to perform assignments such as common dialect handling, information examination, and composing meet questions.The language's adaptability and broad library environment encourage the fast advancement and integration of machine learning arrangements, guaranteeing that applications give exact and personalized criticism to clients.

The hope is that by using the technology mentioned, Invisor can be accessed easily either through desktop or mobile devices that have a browser.

4.4. Design

4.4.1. Use Case Diagram

The use case diagram of the Invisor application shows the relationship between the user and the main functions available.

 doi:10.20944/preprints202407.0049.v1

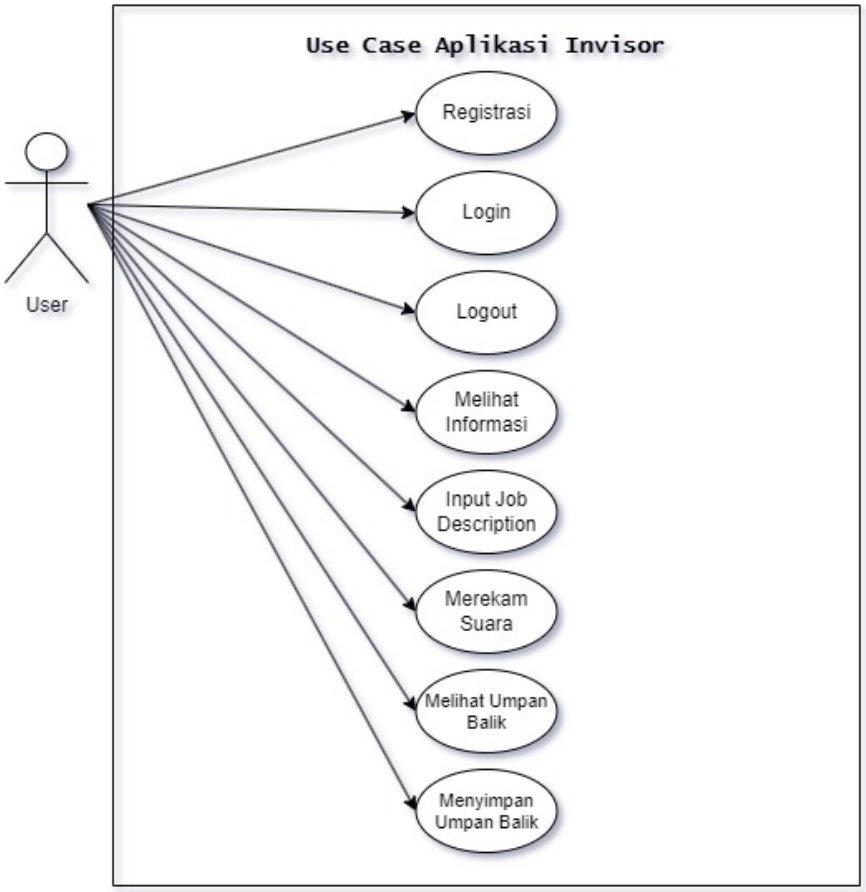

Figure 1. Use Case Diagram

Where users can do:

1. Registration: Create a new account by filling in the required information.
2. Login: Log into the application using the created account.
3. Logout: Exit the application or log out account
4. View Information: Access job details.
5. Input Job Description: Input job description according to the user's interest.
6. Voice Recording: Record the voice for a practice interview session.
7. View Feedback: View reviews and ratings of the interview results.
8. Saving Feedback: Save the feedback received for future reference or analysis.

This use case diagram illustrates how users can manage accounts, take part in interviews, and manage feedback in the Invisor application.

4.4.2. Class Diagram

The class diagram of the Invisor application

 doi:10.20944/preprints202407.0049.v1

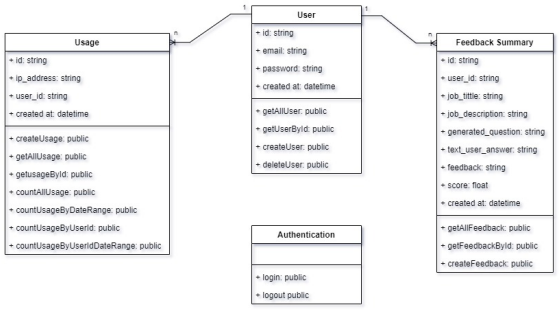

Figure 2. Class Diagram

Describes the data structure and relationships between the various classes in the system. This diagram has four main classes namely: User, Usage, Feedback Summary, and Authentication. For an explanation of each class and its associated attributes and methods as follows:

Class: User

1. **Attributes:**

   (a) id: the id of each user.
   (b) email: The email address of the user.
   (c) password: The user's password.
   (d) created at: The date and time when the user was created.

2. **Methods:**

   (a) +getAllUser(): Retrieves all user data.
   (b) +getUserById(): Retrieves user data by ID.
   (c) +createUser(): Creates a new user.
   (d) +deleteUser(): Deletes the user.

Class: Usage

1. **Attributes:**

   (a) id: Identification for each usage.
   (b) ip address: The IP address of the user.
   (c) user id: The user ID associated with this usage.
   (d) created at: The date and time when the usage was logged.

2. **Methods:**

   (a) +createUsage(): Creates a new usage record.
   (b) +getAllUsage(): Retrieves all usage records.
   (c) +getUsageById(): Retrieves usage records by ID.
   (d) +countAllUsage(): Counts all usage records.
   (e) +countUsageByDateRange(): Counts usage records in a specified date range.
   (f) +countUsageByUserId(): Counts usage records by user ID.
   (g) +countUsageByUserIdDateRange(): Counts usage records by user ID and date range.

Class: Feedback Summary

1. **Attributes:**

   (a) id: Unique identification for each feedback.
   (b) user id: ID of the user who provided the feedback.
   (c) job tittle: Job name
   (d) job description: The description of the job.
   (e) generated question: Interview generated question.
   (f) text user answer: Text answer from the user.
   (g) feedback: Interview result feedback.

 doi:10.20944/preprints202407.0049.v1

(h) score: The score of the interview result.
(i) created at: The date and time when the feedback was created.

2. **Methods:**

(a) +getAllFeedback(): Retrieves all feedback.
(b) +getFeedbackById(): Retrieves feedback by ID.
(c) +createFeedback(): Creates a new feedback.

Class: Authentication

1. **Methods:**

(a) +login(): Method to log in to the system.
(b) +logout(): Method to log out of the system.

Relationships Between Classes

The **User** class has a one-to-many relationship with the Usage and Feedback Summary classes, meaning each user can have multiple usage and feedback records.

4.4.3. Entity Relational Diagram

The Invisor application's Entity Relationship Diagram (ERD) depicts how different entities in the system relate to each other.

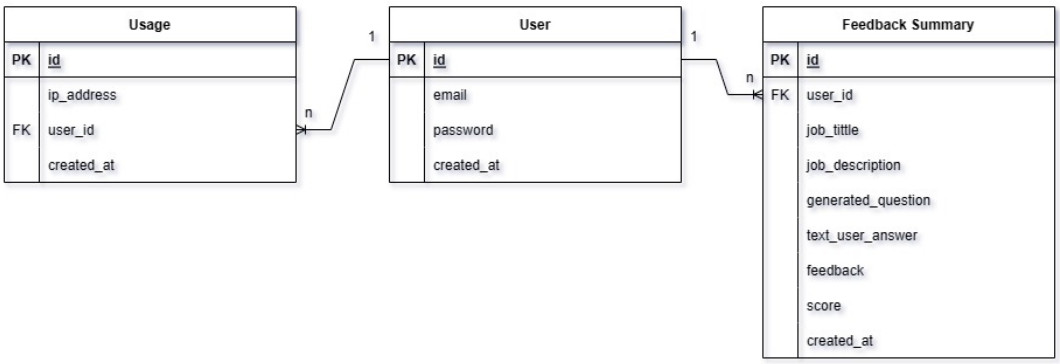

Figure 3. Entity Relational Diagram

This diagram includes three main entities: User, Usage, and Feedback Summary. The following is a detailed explanation for each entity along with existing attributes and relationships:

Entity: User

1. **Attributes:**

(a) id Primary Key (PK): id of each user.
(b) email: The email address of the user.
(c) password: The password of the user.
(d) created at: Date and time when the user was created.

2. **Relation:**

(a) Users have a one-to-many relationship with Usage, meaning one user can have many usage records.
(b) Users also have a one-to-many relationship with Feedback Summary, meaning one user can provide multiple pieces of feedback.

 doi:10.20944/preprints202407.0049.v1

Entity: Usage

1. **Attributes:**

   (a) id: Primary Key (PK) for each usage record.
   (b) ip address: IP address of the user logging usage.
   (c) user id: Foreign Key (FK) that refers to the user ID in the User table.
   (d) created at: The date and time when the usage record was created.

2. **Relation:**

   Usage is linked to User via user id, indicating that each usage record is associated with one specific user.

Entity: Feedback Summary

1. **Attributes:**

   (a) id: Primary Key (PK) id for feedback.
   (b) user id: Foreign Key (FK) that refers to the user ID in the user table.
   (c) job tittle: Name job
   (d) job description: Description job
   (e) generated question: Interview generated questions.
   (f) text user answer: Text answer from user.
   (g) feedback: Feedback provided by the user.
   (h) score: Interview result score.
   (i) created at: Date and time when the feedback was created.

2. **Relation:**

   Feedback Summary is linked to a User via user id, indicating that each piece of feedback is associated with one specific user.

Entity Relationship Diagram (ERD) explains how user data, usage records, and feedback can be interconnected in the Invisor application system. The User entity is the center of this relationship, connecting the Usage and Feedback Summary entities via a foreign key.

4.4.4. Physical Data Model

The Invisor application's Physical Data Model (PDM) describes how data is physically stored in the database, including primary keys and foreign keys. Following is a detailed explanation of the tables and attributes of this diagram:

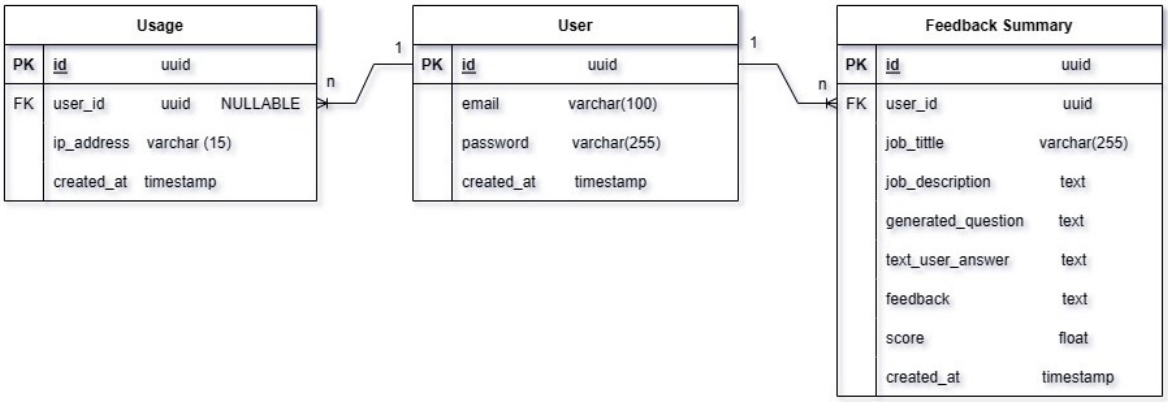

Figure 4. Physical Data Model

The User table has attributes id (uuid, PK), email (varchar(100)), password (varchar(255)), and created at (timestamp), with a one-to-many relationship with the Usage and Feedback Summary tables. The Usage table has attributes id (uuid, PK), user id (uuid, FK, NULLABLE), ip address (varchar(15)),

 doi:10.20944/preprints202407.0049.v1

and created at (timestamp), related to the User table through user id. The Feedback Summary table has attributes id (uuid, PK), user id (uuid, FK), job title (varchar(255)), job description (text), generated question (text), text user answer (text), feedback (text), score (float), and created at (timestamp), also related to the User table through user id. This diagram shows the physical data structure for each entity in the Invisor system, including the relationships between these entities.

4.4.5. Sequence Diagram

1. Case: User can view information about the application

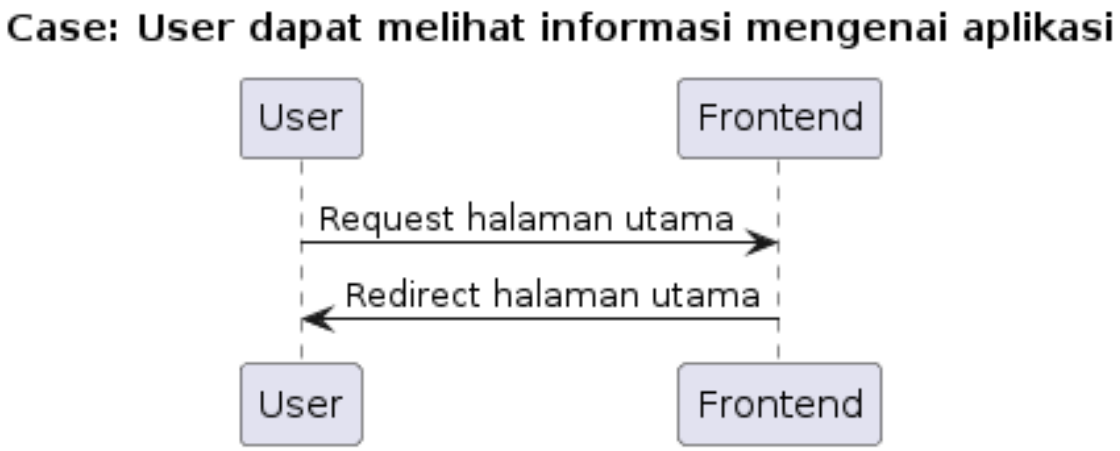

Figure 5. Case 1

This sequence diagram shows the interaction between the user and the frontend of the Invisor app for the case where the user wants to view information about the app. First, the user sends a request to access the main page of the application. The solid arrow from the user to the frontend indicates this request. Upon receiving the request, the frontend processes it and redirects the user to the main page containing the requested information. The solid arrow from the frontend to the user indicates that the frontend returns a response by redirecting the user to the main page.

2. Case: User can input job description then to get a list of questions

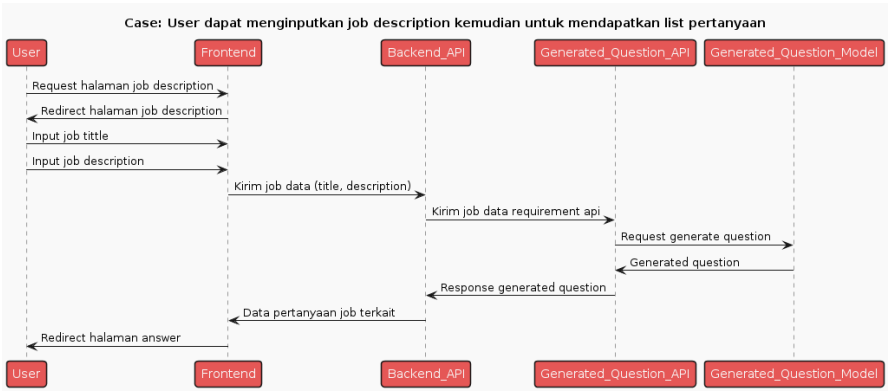

Figure 6. Case 2

In this case, the user interacts with the system to enter a job description and receive a list of related questions. The process starts with the user requesting the job description page, which is then redirected by the 'Frontend' component. Once the user enters the job title and description, this data is sent from the 'Frontend' to the 'Backend API'. The 'Backend API' processes the data and sends it to the 'Generated Question API', which passes it to the 'Generated Question Model'. This

 doi:10.20944/preprints202407.0049.v1

model generates a list of questions based on the given job data, then sends the generated questions back to 'Generated Question API'. The questions are then sent to the 'Backend API', which passes them to the 'Frontend'. Finally, the 'Frontend' displays the list of questions to the user on the answer page, allowing the user to see the questions generated based on the job description they entered.

3. Case: User can record voice answers to questions made by the application, user can see feedback from the application on the answers given, User can save the feedback results in the application.

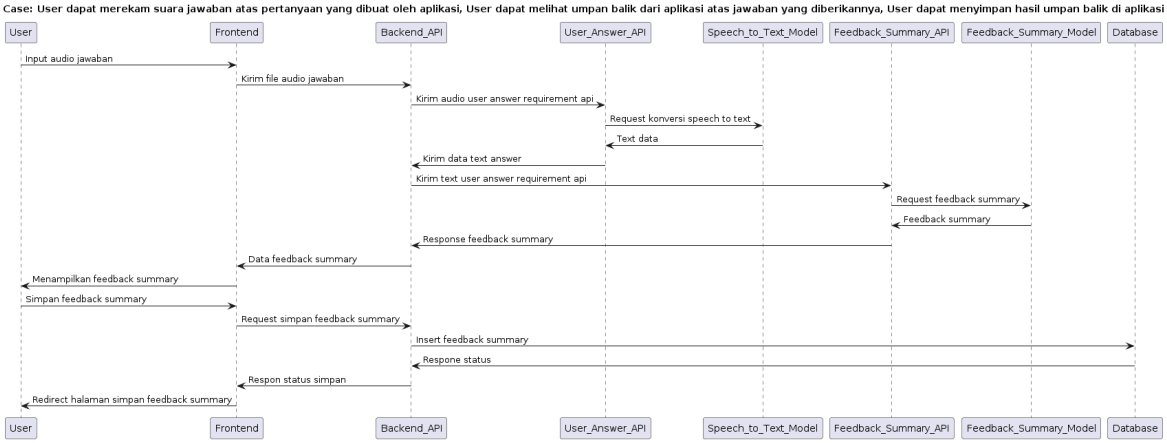

Figure 7. Case 3

This sequence diagram illustrates the process from voice recording of user answers to storing the feedback results in the application. The process starts when the user records and uploads the answer audio file through the 'Frontend', which then sends the file to the 'Backend API' and 'User Answer API'. The 'User Answer API' requests the 'Speech to Text Model' to convert the audio file into text. The text result is sent back through 'User Answer API' to 'Backend API'.

The 'Backend API' requests the feedback summary from the 'Feedback Summary API', which is passed to the 'Feedback Summary Model'. 'Feedback Summary Model' generates a feedback summary based on the answer text and sends it back to 'Feedback Summary API' and 'Backend API'. The 'Backend API' then passes the feedback summary data to the 'Frontend', which displays it to the user.

If the user chooses to save the feedback summary, 'Frontend' sends a save request to 'Backend API', which is then passed to 'Feedback Summary API' and 'Database' for saving. After saving, the 'Frontend' receives the saving status and redirects the user to the confirmation page. This diagram covers the entire flow from audio recording to feedback result saving, involving various system components.

4. Case: User can log in to the application using an account

 doi:10.20944/preprints202407.0049.v1

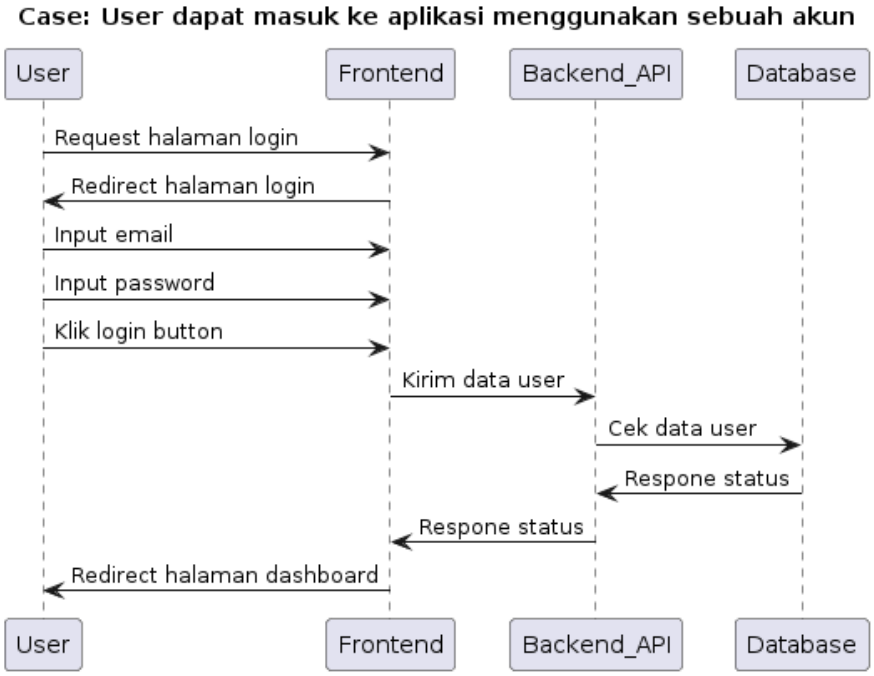

Figure 8. Case 4

This sequence diagram shows how a user logs into the application using their account. The process starts when the user requests the login page and the frontend redirects them to it. The user then enters their email and password and clicks the login button. This login data is sent by the frontend to the Backend API for processing. The Backend API checks the credentials by sending a verification request to the database. The database verifies the credentials and sends a response (success or failure) back to the Backend API. The Backend API then forwards the response status to the frontend. If the login is successful, the user is redirected to the dashboard page; if it fails, the user remains on the login page with an error message. This diagram illustrates the flow of the user authentication process, from login input to verification and redirection to the dashboard page, by showing the interaction between the user, frontend, Backend API, and database.

5. Case: User can log out accounts that are logged in to the app

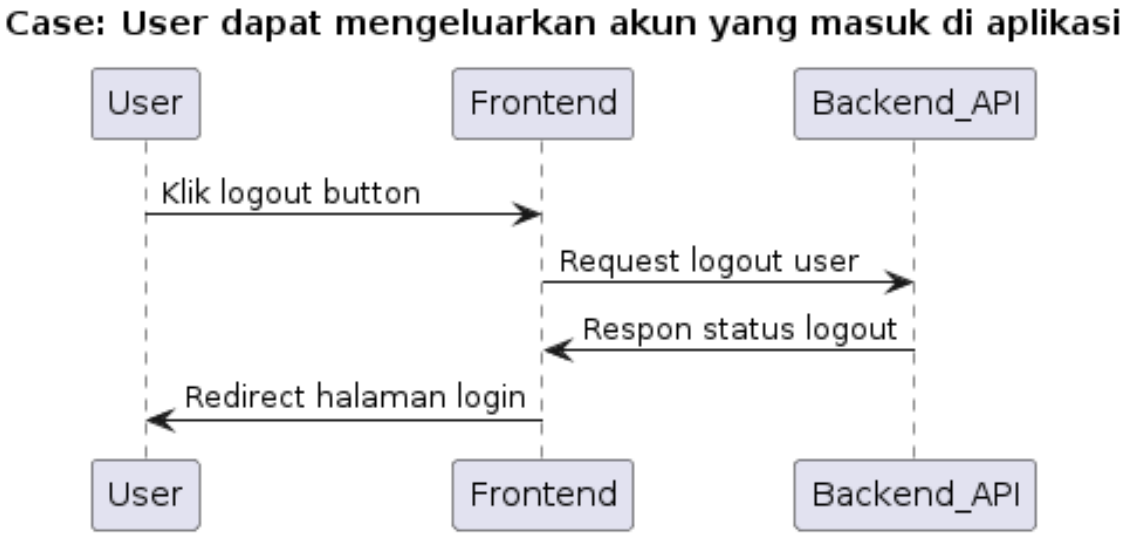

Figure 9. Case 5

 doi:10.20944/preprints202407.0049.v1

This diagram describes the process of logging out of the application. The process starts when the user clicks the logout button in the user interface. The logout request is sent from the frontend to the Backend API. The Backend API processes this request by ending the user session, removing the authentication token, and updating the user status in the database. After that, the Backend API sends a response back to the frontend indicating the status of the logout, whether successful or failed. If the logout is successful, the frontend redirects the user to the login page, signaling that the user must login again to access the application's features. This diagram illustrates the interaction flow between the user, frontend, and Backend API during the logout process.

In this system design stage, we used the Figma application to design the User Interface (UI) and User Experience (UX) of the system. The system design method is applied sequentially, starting with the creation of wireframes as an initial prototype. The creation of this wireframe aims to facilitate the subsequent design process because it allows clear visualization of the structure and layout of the system [10].

After completing the wireframes, the next step is to create the final design. At this stage, we focus on visual refinement and user interaction. Details such as color selection, typography, and interactive elements are carefully designed to produce a UI/UX that is intuitive, attractive, and fits the user's needs. Performing system design early in the application development process has many advantages, including:

1. A structured and clear design helps the development team to understand and translate ideas into code more easily. This saves time and minimizes errors in the development process.
2. Complete and detailed design documentation can help teams communicate more effectively between developers, designers, and other stakeholders. This can minimize miscommunication and ensure that all parties are on the same page.
3. By having a clear design at the beginning, the team can avoid unnecessary changes and revisions later on. This can save time and resources, and improve the efficiency of the overall development process.

To provide a clearer picture of the system design process, here is an example of a wireframe and the final design of one of the application elements:

 doi:10.20944/preprints202407.0049.v1

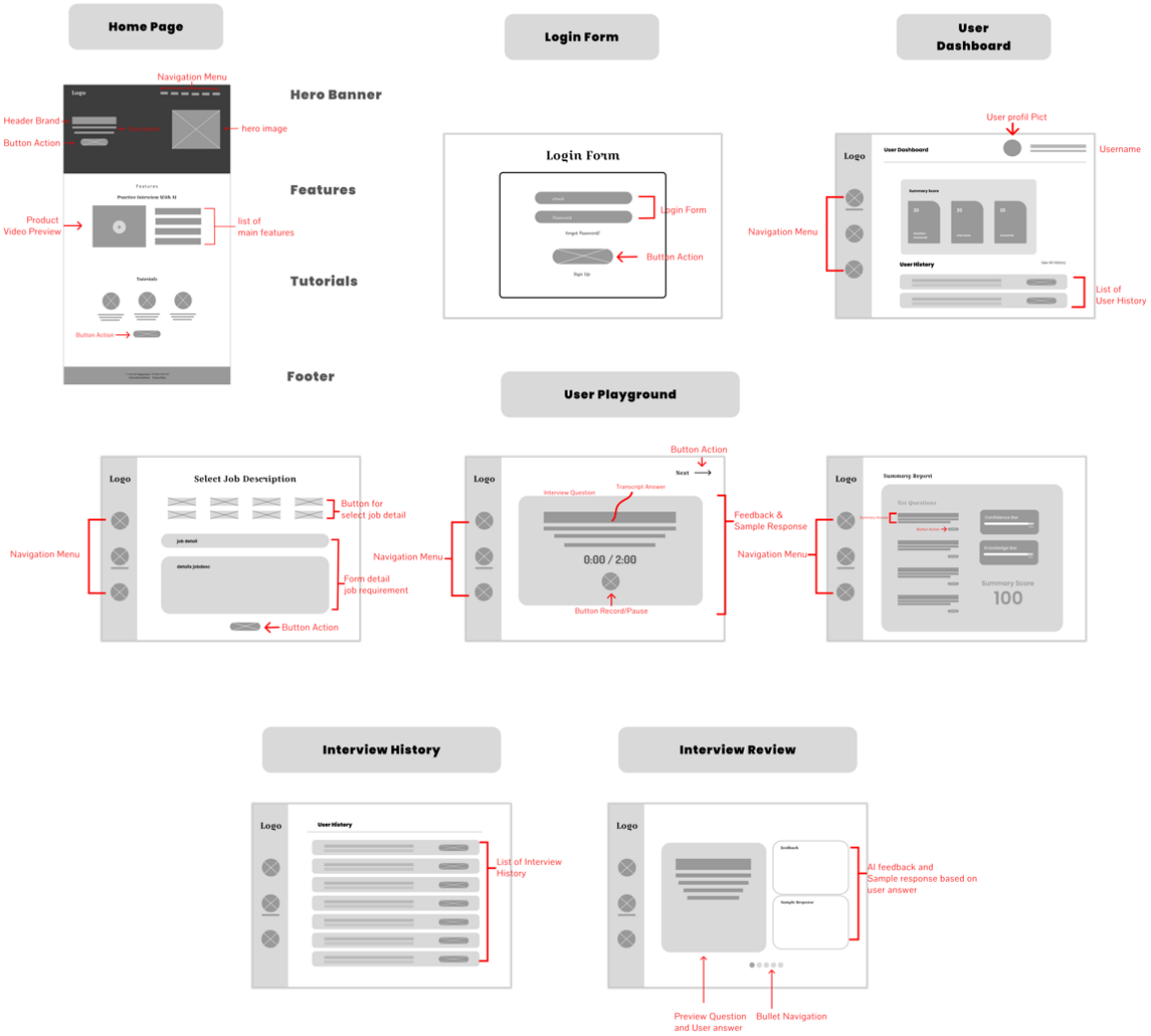

Figure 10. Example of wireframe

 doi:10.20944/preprints202407.0049.v1

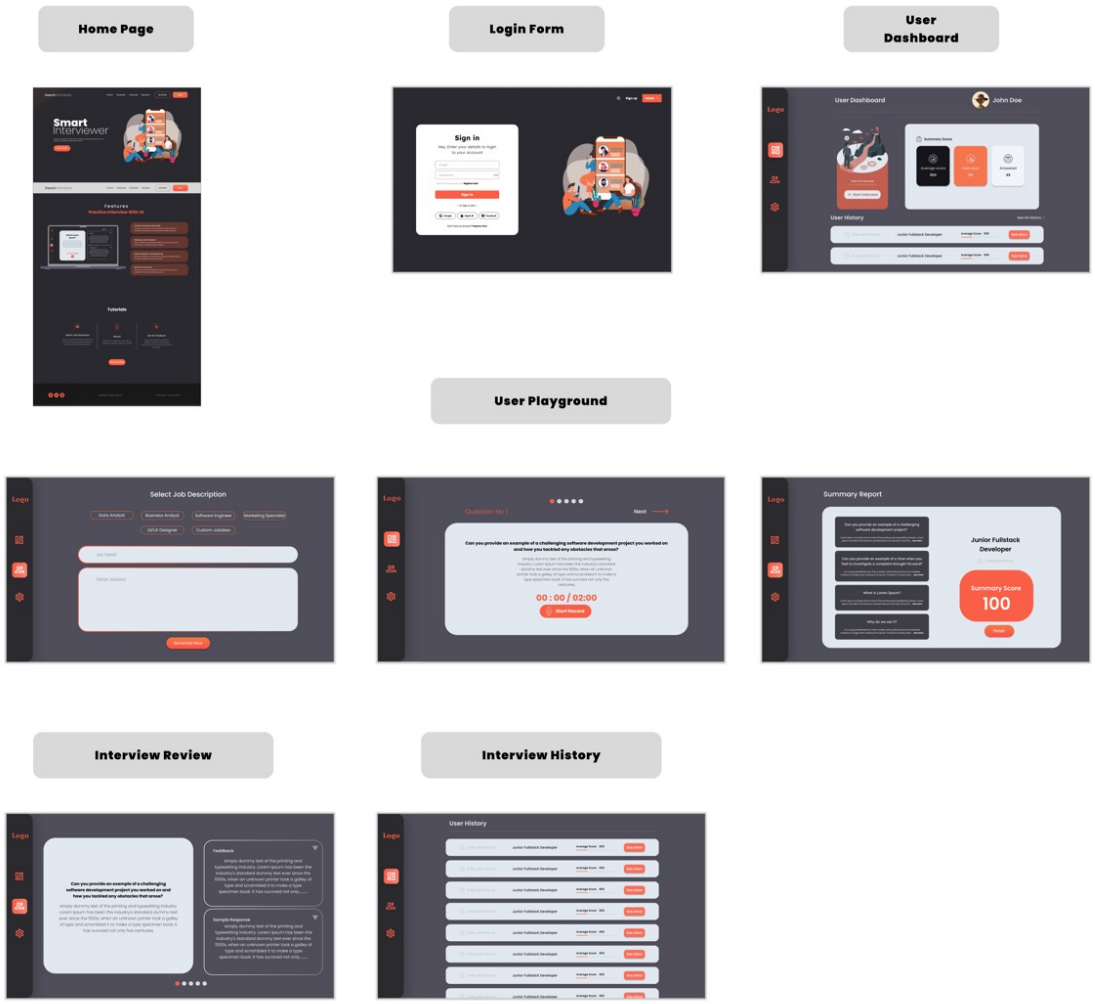

Figure 11. Example of final design

4.5. Development

Invisor's application development phase focuses on leveraging technology to build a robust, scalable system that meets identified requirements.The methodology used pays close attention to system architecture and implementation details, from front-end components to back-end infrastructure.

At the front-end level, the development team focuses on designing a user-friendly interface so that students can interact smoothly. UX/UI designs are transformed into functional components using technologies such as HTML, CSS, and JavaScript frameworks such as React.js and Angular. The focus is on ensuring intuitive navigation, accessibility and responsiveness across all devices to enhance the user experience.

Iterative development cycles enable continuous improvement based on user feedback and usability testing, ensuring that the front end conforms to the planned design and functionality. At the same time, the backend development team focuses on implementing the core logic and data management functions of the Invisor application.

This includes defining and integrating APIs, databases, and server-side technologies such as Node.js, Python, and Java, depending on your project needs. This architecture is intended to support scalability and performance, allowing applications to effectively handle expected user loads and data volumes.

 doi:10.20944/preprints202407.0049.v1

Security measures such as data encryption and user authentication mechanisms are also implemented to protect sensitive information and maintain user trust.During the development phase, rigorous testing procedures are used to verify the functionality and reliability of the system.

Unit testing, integration testing, and end-to-end testing are performed to identify and resolve errors and inconsistencies in application behavior. Continuous integration and deployment (CI/CD) pipelines automate the testing and deployment process, enabling rapid iteration and updates while maintaining overall system stability.

4.5.1. Front-End

The front end of the system is built using a JavaScript framework called Next.js, which allows you to create dynamic and interactive user interfaces. Application data, such as user information, job descriptions, and questions and answers, is stored in PostgreSQL, a powerful relational database management system.

4.5.2. Back-End

Invisor's primary back-end infrastructure is built using Laravel, a popular PHP framework known for its ease and efficacy in managing complex business processes. Critical functions such as user authentication, job vacancy management, and data processing are made possible by Laravel's expressive syntax and many built-in features. Following contemporary web standards and best PHP development practices, this framework ensures secure and scalable application development.

Invisor uses the Queue Worker system integrated in Laravel to optimize performance and simplify data processing. The system effectively manages and prioritizes background tasks, improving overall application efficiency and responsiveness. Additionally, Python libraries and Cohere services support machine learning capabilities in the Back-End. Advanced functions such as text classification and information extraction are supported by these tools, which are critical to leveraging Invisor's capabilities for AI-based interview simulation.

4.5.3. Back-End Architecture

Invisor's back-end architecture uses a RESTful approach to guarantee efficient communication between various parts of its application using JSON over HTTP. This design choice enables standardized and efficient data exchange, which improves interoperability with external systems and services. Additionally, scalability is supported by easy integration with third-party APIs, which allows Invisor to take advantage of the features offered by third-party APIs. To allow for future expansion and upgrades and ensure that the application can evolve as user needs evolve and technology advances, this flexibility is essential.

Additionally, Invisor's RESTful Back-End architecture speeds development and simplifies maintenance. The application offers a structured framework for organizing endpoints and managing data flows, making implementation and maintenance of complex functionality such as user authentication, data processing and interaction with the Front-End easier. By adhering to REST principles, this architecture ensures consistency and reliability in handling requests and responses, thereby providing a robust and responsive user experience.

Invisor's back-end architecture is intended to support its advanced functions without relying on microservices. The RESTful approach lays a strong foundation for future development and optimization while encouraging flexibility in development and scalability of operations. This architectural framework allows Invisor to provide effective interview training to users while remaining flexible to adapt to changing requirements and technological advances in the job market.

4.5.4. Implementation Flow

The system implementation starts from the Front-End to the Back-End, with the following flow:

 doi:10.20944/preprints202407.0049.v1

1. Front-End
    (a) Using Axios, Next.js interacts with the main Back-End and sends and receives data such as job descriptions, questions, and answers, as well as login and registration information. This ensures that the user interface and server interact well, allowing for real-time updates and dynamic rendering of content that is based on user actions and input.
    (b) Using Next.js capabilities, the front-end conveys back-end data to an interactive UI. This creates a responsive and engaging user experience, allowing users to easily navigate features and receive instant feedback on the progress of their interview preparation.
2. Main Back-End
    (a) Data requests from the front-end are sent to the main back-end via Axios. This communications layer handles all incoming requests, ensuring that data is processed correctly and sent to the appropriate services for further processing.
    (b) To ensure the requests are well organized and structured, the back-end uses queue workers to process them. This method helps manage large numbers of requests efficiently, avoid bottlenecks, and ensure timely responses to user interactions. After processing, the data request is sent to the ML Back-End for processing.
3. Machine Learning Back-End
    (a) The machine learning backend receives data requests from the main backend; this section handles the complex calculations and analysis required to generate personalized feedback and interview questions.
    (b) The Back-End calls the Cohere service to perform the machine learning. Using advanced machine learning algorithms and natural language processing (NLP), the system analyzes input data to create relevant, personalized interview scenarios for users.
    (c) The results of the machine learning process are sent back to the main Back-End in JSON format. This standardized data format ensures compatibility and ease of integration, allowing the processed information to be seamlessly utilized by other system components.
4. Main Back-End
    (a) The main Back-End receives the results from the machine learning Back-End. This data is then further processed if necessary and prepared for delivery back to the Front-End.
    (b) The processed results are sent to the Front-End for display to the user. By closing the loop in this data flow, the system ensures that users receive timely and accurate feedback on their interview performance, enhancing their overall training experience.

With this approach, the system is built effectively and efficiently, ensuring all components function synergistically to provide an optimal user experience.

4.6. Testing

Testing is the most commonly used technique to check and verify software quality. Testing is the act of running a program or system with the aim of finding errors. This activity is very important in the Software Development Life Cycle (SDLC). For testing it is divided into frontend and backend. The frontend consists of a view that users access directly through their browser, and the backend is divided into two parts: the main system backend, which contains the main functionality, and the machine learning backend. Handle the request and response process for questions in the creation process and create feedback summaries.

4.6.1. Access and Registration

Testing Invisor's access and registration functionality requires a thorough review of landing page functionality and user interactions. Test your landing page to ensure all elements load correctly, including navigation menus, introductory content, and interactive components. This first interaction

 doi:10.20944/preprints202407.0049.v1

is important because it determines user engagement and makes it easier for users to navigate and understand the purpose of your app from the start. Additionally, we check layout and design consistency across devices and screen sizes to ensure a smooth user experience, no matter what platform you use. The registration process focuses on ease of use and data integrity. Each input field in the registration form is systematically filled with different data sets to check responsiveness and error handling. This testing methodology is intended to identify potential issues such as input validation errors or unexpected behavior, and to ensure that users can register smoothly without encountering any obstacles. Additionally, after completing the registration form and clicking "Register", the speed and accuracy of redirection to the login page is checked to ensure that users are immediately directed to the next step in their interactions within the application. The feedback gathered from these tests not only ensures a good registration process, but also provides insight into potential areas of improvement, including: B. Optimize load times or fix error messages for better user guidance.

4.6.2. Login and User Profile

The login function has been rigorously tested to ensure safe and efficient access to user accounts. In our testing, we started with a login button on a landing page and verified that it triggered a seamless transition to the login interface where users could enter their credentials. Authentication checks are performed to verify correct credentials and confirm the application's ability to seamlessly grant users access to the dashboard. After logging in, the user profile view in the dashboard is checked to ensure it contains the following information for the user: B. reflects his email address correctly. This verification step is important to maintain user trust and provide a personalized login experience. Additionally, user session management and logout functionality has been tested to ensure that users can safely log out of their accounts from any page in the application. To start the logout process, access the Settings menu. The app's response to these actions is monitored to ensure that users are properly directed to landing pages or other designated areas. The goal of this testing is to verify the overall reliability and security of the login process and ensure that users can safely access and manage their accounts. Detailed analysis of login functions and user profiles not only ensures their functionality, but also emphasizes that the app adheres to security best practices and user-centered design principles, thereby increasing overall user satisfaction and trust.

4.7. Interview Simulation

The Interview Simulation feature has been extensively tested to evaluate how well it prepares users for real interviews.Click the "Start Interview" button to start the interview.This will take the user to the job description selection page.It evaluates the application's ability to display relevant job information or allow custom input based on the user's preferences.Central to the testing process is the automatic generation of interview questions and their clear and structured presentation to users.Researchers use the voice input to ensure accurate transcription of responses and real-time scoring.It gives you instant feedback to improve your interview.Interview navigation, such as moving between questions and providing answers, is carefully monitored to ensure smooth navigation and intuitive user control.

The program's ability to display timers and progress indicators accurately mirrors the interview process and increases user engagement and focus.After the interview, testers review the app's evaluation and feedback mechanisms to provide users with meaningful information about performance and opportunities for improvement.Overall, this interview simulation test aims to validate Invisor's effectiveness in teaching practical interviewing skills through a user-friendly and interactive experience.Rigorous testing of this interview simulation feature not only ensures its reliability, but also emphasizes the feature's role in creating a realistic simulation environment where users can practice and improve their interview skills.Iterative feedback from these tests drives improvements aimed at improving user engagement and performance ratings to meet user expectations and industry standards.

 doi:10.20944/preprints202407.0049.v1

4.8. Usage Panel and User History

The panel feature has been tested to provide comprehensive information to users and effectively track interview history.The dashboard interface has been updated so that the dashboard now accurately displays average scores, total number of interviews, and number of questions answered based on user interaction logs.This information is important so that users can track their progress over time and identify areas where they can effectively improve their interviewing skills.The View All History and More buttons on the control panel have been tested to ensure that they correctly direct users to a detailed view of individual interview history and feedback sessions.The login feature has been rigorously tested to ensure secure and efficient access to user accounts.During testing, we started with the login button on the home page and made sure it triggered a smooth transition to the login interface where users can enter their login details.

4.9. Deployment

Software distribution or deployment is the stage between software procurement and use. This stage is performed by DevOps, the agent that takes the software, prepares it to run, and may also run it. Software distribution or deployment can be interpreted as a process consisting of several interrelated activities, such as releasing the software at the end of the development cycle, configuring the software, installing the software into the execution environment, and activating the software.

The Front-End deployment leverages Vercel, a platform specifically designed modern Front-End applications. Vercel provides features such as auto-scaling, serverless functions, and direct intergration with git repositories, which facilitates efficient management and rapid updates of applications. It's reliability in handling static site generation and performance optimization makes it an ideal choice for the Front-End component of this system.

The main Back-End of the system is hosted on a PHP-based shared hosting service. Shared hosting offers a cost-effective and easy-to-manage server solution. The PHP-based applications require standard server configuration. Despite limitation in resources and customization.

For the machine learning Back-End, deployment is carried out using Docker on a Virtual Private Server (VPS). Docker enables isolation of the environment and application dependencies, ensuring consistent performance and ease of configurations and offers more capacity compared to shared hosting. This is good for intensive workloads and specialized for machine learning application

4.10. Review and Retrospective

After deploying the Invisor application and confirming its functionality aligns with the specified requirements, the subsequent phase involves conducting thorough reviews and retrospectives to assess its overall performance and effectiveness in real-world scenarios.

The review process entails evaluating whether the application functions as intended based on the initial requirements analysis. This involves verifying key functionalities such as the ability for students to input job descriptions, receive AI-generated interview questions, and provide accurate answers through voice input. Additionally, the review will focus on validating the scoring and feedback mechanisms, ensuring they provide meaningful insights to help users improve their interview skills. Stakeholders, including developers, testers, and potentially end-users, collaborate to assess the application against predefined acceptance criteria, confirming its ability to meet user needs effectively.

The retrospective phase following the review allows for a critical discussion of the development process itself. The purpose of these internal team meetings is to identify successful practices and areas for improvement. The topics of discussion are the quality of the agile methodology used, team communication, collaboration and problems arising during development and implementation. Retrospectives provide teams with important information and lessons learned that they can use in the future to improve or enhance Invisor applications. This cycle of continuous improvement ensures that

 doi:10.20944/preprints202407.0049.v1

the development process remains relevant and effective while changes based on student conversational needs and feedback.

In general, the overview and retrospective stages will repeat and repeat throughout the life of the Invisor application. A post-deployment review tracks user feedback and helps quickly resolve issues or opportunities for improvement. At the same time, hindsight improves the development culture, optimizes performance and ensures that it meets the changing needs of users. Visor remains a powerful tool to help students face job interviews with confidence.

4.11. Discussion

By integrating artificial intelligence and applying agile methods throughout the life cycle, the development of Invisor allows users to significantly improve their interview preparation.AI technology allows users to simulate interview scenarios individually and effectively.This approach addresses the important need for students and job seekers to improve their interviewing skills in a controlled environment before conducting actual interviews with potential employers.

The use of artificial intelligence in Invisor enables the creation of dynamic and personal interview simulations.Using natural language processing (NLP) and machine learning algorithms, the application generates relevant interview questions based on the job description provided by the user.This not only adds realism to the interview experience, but also ensures that users receive personalized feedback and guidance to improve their answers over time.In addition, the application of agile methodology greatly influenced the iterative development and continuous improvement of Invisor.

Agile practices such as sprint planning, daily meetings and retrospective meetings enable rapid prototyping and further improvement of applications.This iterative approach allows for direct feedback from users and stakeholders to improve usability and functionality across the platform.Combining AI-based interview simulations and agile development practices, Invisor not only meets the immediate needs of its users, but also anticipates the demands of the future labor market.

The platform's ability to adapt and evolve based on user feedback and technology development underscores the platform's commitment to providing effective and efficient tools to prepare for interviews in a competitive professional environment.

5. Conclusion

Currently, the utilization of technology to solve problems in the world has achieved real progress. In this case, it is proven by the integration of artificial intelligence using the Large Language Model (LLM) into the interview preparation system. The application of Advisor with LLM allows users to experience simulated interview scenarios as they should. Because in this case the user can feel a real interaction with the interviewer. Utilization of natural language system capabilities can generate interview questions based on the job description provided by the user. The system is able to generate relevant answers.

In adjusting to the needs of users, accessibility is a very important aspect for us to pay attention to. even more so in today's digital era where distance learning has an important role in developing skills and knowledge. Therefore, Invisor comes as a web-based interview training application. This web-based system can increase accessibility as it can be accessed seamlessly from any internet-enabled device. Invisor also provides various interview training features that can be personalized for skill development and real-world interview readiness.

To conclude, Invisor is an example of how Artificial Intelligence and Web-based Systems can be integrated and revolutionize traditional practices. Invisor offers a structured and efficient solution in preparing interview skills with a mission to empower students to gain skills and confidence by practicing through AI interview simulations through an easily accessible platform. Invisor is not only intended to meet current market demands but also to be a future innovation in career development and education technology.

Acknowledgments: The author's wishes to express sincere gratitude to the Informatics Department of UIN Sunan Gunung Djati Bandung for their partial support and invaluable resources that facilitated this research work. Special thanks are extended to Mr. Wisnu, the course lecturer, whose guidance and expertise greatly contributed to the success of this project. Furthermore, heartfelt appreciation goes to the dedicated team members who worked tirelessly and collaboratively to complete this research, as well as to all colleagues and friends whose encouragement and assistance were instrumental throughout the entire process.

References

1. Heimerl, A.; Mertes, S.; Schneeberger, T.; Baur, T.; Liu, A.; Becker, L.; Rohleder, N.; Gebhard, P.; André, E. Generating personalized behavioral feedback for a virtual job interview training system through adversarial learning. International Conference on Artificial Intelligence in Education. Springer, 2022, pp. 679–684.
2. Lee, B.; Kim, B. Development of an AI-based interview system for remote hiring. *International Journal of Advanced Research in Engineering and Technology (IJARET)* **2021**, *12*, 654–663.
3. Qin, C.; Zhu, H.; Zhu, C.; Xu, T.; Zhuang, F.; Ma, C.; Zhang, J.; Xiong, H. DuerQuiz: A personalized question recommender system for intelligent job interview. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2165–2173.
4. Horat, S.I.; Kara, K.C.; Karakaş, A. Job Pre-Interview System with Artificial Intelligence. 2019 1st International Informatics and Software Engineering Conference (UBMYK). IEEE, 2019, pp. 1–4.
5. Shen, D.; Zhu, H.; Zhu, C.; Xu, T.; Ma, C.; Xiong, H. A joint learning approach to intelligent job interview assessment. IJCAI, 2018, Vol. 18, pp. 3542–3548.
6. Pratap Singh Rathore, S. The Impact of AI on Recruitment and Selection Processes: Analysing the role of AI in automating and enhancing recruitment and selection procedures. *International Journal for Global Academic & Scientific Research* **2023**, *2*, 78–93. doi:10.55938/ijgasr.v2i2.50.
7. Thompson, I.; Cubrich, M.; Qasim, M.; Zhu, Y.E.; Hassenkamp, K.; Koohifar, F.; Koenig, N.; Dudley, N. A Measurement-based Foundation for AI Applied to the Audio of Interviews. TMS Proceedings 2021. American Psychological Association, 2021. doi:10.1037/tms0000005.
8. Pertiwi, D.R.; Kusumaningrum, M.A.D. Developing English job interview skill using artificial intelligence technology. *KACANEGARA Jurnal Pengabdian pada Masyarakat* **2021**, *4*, 221. doi:10.28989/kacanegara.v4i2.915.
9. Bhere, S.; Dudhat, S. AI Interview Trainer. *International Journal of Applied and Advanced Multidisciplinary Research* **2023**, *1*, 223–226.
10. Chen, J.; Chen, C.; Xing, Z.; Xia, X.; Zhu, L.; Grundy, J.; Wang, J. Wireframe-based UI Design Search through Image Autoencoder. *ACM Transactions on Software Engineering and Methodology* **2020**, *29*, 1–31. doi:10.1145/3391613.