Table of Contents

Proposal

Project Statement

It is challenging to acquire healthcare services in today's fast world, especially where time is a constraint for most people. People lack the understanding of the writings of doctors' prescriptions while trying to identify the authenticity of medicines. The National Academy of Science has established that at least 1.5 million people a year are being killed and sickened because of poor reading of medical prescriptions (Kamalanabanand, Gopinathand, & Premkumar, 2018). This might become more precarious in areas where low-quality or counterfeit medicine is a problem, thus further leading to the very real possibilities of wrong or unsafe treatments. Less literate patients would want more help for their prescription (Patel, Khan, Ali, Kazmi, Riaz, Awan, & Sorathia, 2013).

Health problems/concerns of children, as identified by parents, are highly varied and likely to shift overtime, influenced by social changes, the emergence of new health threats, and the media (Garbutt, Leege, Sterkel, Gentry, Wallendorf & Strunk, 2012). For example, there are the concerns of parents regarding the safety and effectiveness of medicines they use for their children. The parents may feel that upon receiving a prescription from a physician that they cannot read, they do not know what medicine they are buying at the pharmacy. This application bridges the doctor's prescription and the fulfillment by the pharmacist, enabling users to scan, read, and verify independently the contents of their prescription. It empowers confidence because users will have an assurance of what medication has been prescribed to them, what for, its constituents, dosage instructions, and other essential information.

The main target group for this application would include parents of newborn babies or toddlers who are more worried about the health of their children and also, elderly people who also might have problems with interpreting prescriptions. In addition, the app meets the emerging insecurities related to the quality of medications that, in turn, arise from the reported cases of low-quality product imports. The application allows users to verify prescriptions and provide detailed information concerning the prescribed drugs, thus enabling them to make conscious decisions in healthcare.

The app’s key services are twofold: it allows users to independently verify the contents of their prescription and serves as a record-keeping tool, enabling them to revisit their prescriptions multiple times after the initial doctor’s visit. This ensures confidence and clarity for users throughout their healthcare journey.

Ethical Analysis

Several ethical issues are involved in the development and deployment of the doctor prescription reading app, which need to be addressed in ensuring responsible use and observation of ethical standards expected of a computing professional. In developing such work, there is always the need to engage all stakeholders both locally and internationally in

working on ethical guidelines regarding the implementation of AI and ML in medicine and health care (Johnson, 2019).

Information on medical records requires major importance regarding privacy and protection. The prescription data captured in images may contain private health information. Encryption methods and secure storage protocols should be properly executed so that no unauthorized access or misuse of data takes place. It is important that consumers, healthcare personal, and app developers are prepared to take caution in adopting and developing mHealth apps by providing them with the knowledge about the app (Adhikari, Richards, & Scott, 2014).

Also, the app aims to educate the user with confidence and clarity about the medications prescribed to them. However, there is a chance the hand-sign recognition model of the app may misinterpret a prescription and lead to misinformation. Such a case can be very critical if the users rely on the wrong data. While overfitting gave good performance on training data, on generalized data it would give out poor results. On the other hand, underfitting resulted in poor results on the test data, too, and on other data (Pala, Jethwani, Kumbhar & Patil, 2021). The reason is that the app should carry disclaimers that it is for support purposes only, stating one should consult healthcare professionals before making any medical decision through the app.

It goes without saying that users should be completely informed as to how and where their data are used and processed within the app. Clear communication plus terms of service will ensure that the users give informed consent for what the app does. Ethical design means in return, users are allowed to object against any collection practices not necessary for proper functionality.

Security protocols should be implemented because the app handles sensitive information to avoid data breaches, hacking, and other forms of cyberattacks. Further, an analysis shows that most of the identified challenges relate to security knowledge and expertise in secure mHealth app development among developers (Aljedaani & Babar, 2021). Besides this, encryption, secure login methods, and regular security audits would result in a secure environment for the users.

This application can surely meet high ethical standards expected from a computing professional, provided it ensures privacy, transparency, accuracy, inclusivity, and security. In this way, the app will be in a position to appropriate service for the users and trusted, taking care of the health data of users with due integrity.

Risk Assessment

Since an approach to risk assessment (RA) must be able to reach all societal levels and offer the resilience and adaptability to allow enterprise organizations to also benefit, RA is a well-established method used by organizations to comprehend and mitigate information security threats (Lederm & Clarke, 2011). This below table highlights the most significant risks to the app's successful development and deployment, along with proactive mitigation strategies to address these challenges.

Table 1: Project Risk Assessment and Mitigation Strategies

Risk Factor	Severity	Impact on the Project	Mitigation Strategies
Model Inaccuracy	High	Misinterpretation of prescriptions could mislead users, impacting app credibility and user safety.	Extensive testing and validation, disclaimers advising the consultation of a pharmacist for actual information.
Data Privacy Violation	High	Could result in legal repercussions, loss of user trust, and potential halting of project due to non-compliance with data protection regulations.	Provide strong encryption of data both at storage and transmission, use secure methods of login, and follow the regulations of data privacy.
Limited Handwriting Variability	Moderate	Model may fail to recognize diverse handwriting styles, limiting app's effectiveness across different regions and demographics.	Collect different types of handwriting data. It may be possible to include transfer learning from some pre-trained models to help the variability in recognition.
Data Collection Challenges	Moderate	Insufficient data could impact model accuracy and limit app's ability to generalize across different prescriptions.	Collect active data with different sources; consider using data augmentation techniques to increase diversified ground-truth dataset variation.
User Interface Misunderstanding	Low	Users may initially struggle with understanding the app layout or features, leading to usability issues.	Provide tips or in-app mini tutorials that will assist people in learning how to use the app effectively.
Data Loss Due to User Error	Low	Users may accidentally delete saved prescriptions, causing frustration but not critical data loss.	Provide a confirmation prompt before deleting.

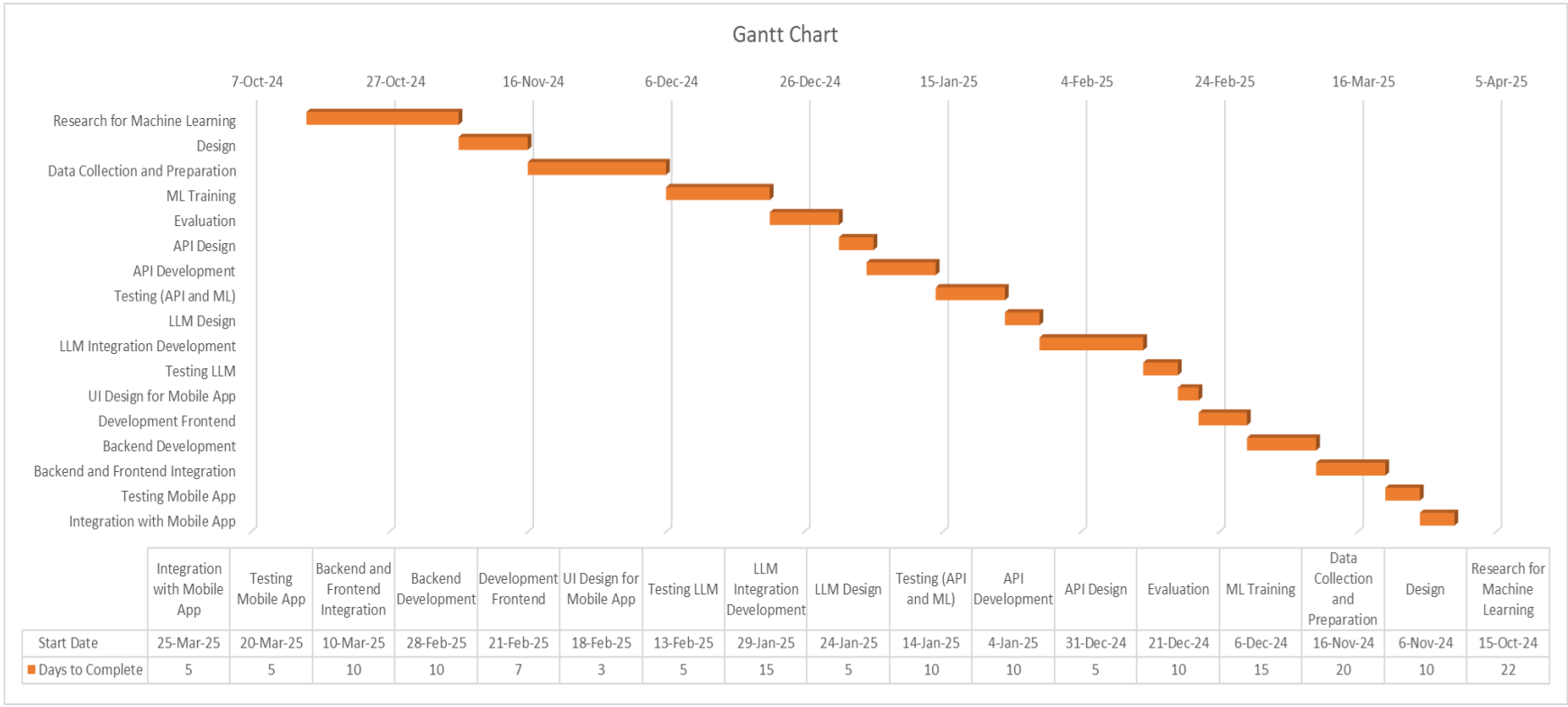
Health & Safety

Applications can be useful for the users in that they avail interactive tools for treatment adherence, facilitate access to information, and so forth. However, applications can also carry risks if they are erroneous and unreliable because, for one thing, users may use the data as a basis for health-related decisions (Akbar, Coiera & Magrabi, 2020).

Table 2: Health & Safety Impact Analysis

Potential Risk	Affected Parties	Impact on Health & Safety	Control Measures
Inaccurate Medicine Information	Users, Public	Incorrect information could lead to improper medicine usage, potentially impacting user or public health.	Recommend consulting pharmacists; ensure app accuracy through regular updates.
Occupational Burnout	Developer	High workload or prolonged coding hours may impact your physical and mental health.	Set realistic goals; rest frequently, if necessary, and request feedback.
Misinterpretation of Prescription	Users, Public	Misunderstand dosage or instructions, leading to health risks when interpreting data.	Include clear disclaimers, Use easy-to-read design.
Stress from App Reliability Concerns	Developer, Users	Concern over app accuracy and security may cause stress, particularly if critical health data is involved.	Prioritize testing and validation.

Time & Resource Plan



	Integration with Mobile App	Testing Mobile App	Backend and Frontend Integration	Backend Development	Development Frontend	UI Design for Mobile App	Testing LLM	LLM Integration Development	LLM Design	Testing (API and ML)	API Development	API Design	Evaluation	ML Training	Data Collection and Preparation	Design	Research for Machine Learning
Start Date	25-Mar-25	20-Mar-25	10-Mar-25	28-Feb-25	21-Feb-25	18-Feb-25	13-Feb-25	29-Jan-25	24-Jan-25	14-Jan-25	4-Jan-25	31-Dec-24	21-Dec-24	6-Dec-24	16-Nov-24	6-Nov-24	15-Oct-24
Days to Complete	5	5	10	10	7	3	5	15	5	10	10	5	10	15	20	10	22

Task	Estimated Days	Sprints	Comments
Research for Machine Learning	22	Sprint 1	Initial ML research and feasibility analysis
Design	10	Sprint 2	Covers model, system, and initial UI/UX design
Data Collection and Preparation	20	Sprints 2-3	Concurrent with ML design; iterates as needed
ML Training	15	Sprints 3-4	Multiple iterations and tuning
Evaluation	10	Sprint 4	Testing ML accuracy, adjusting parameters
API Design	5	Sprint 5	Designing endpoints, structure, and requirements
API Development	10	Sprints 5-6	Initial implementation and versioning
Testing (API and ML)	10	Sprint 6	Unit and integration tests for APIs and ML model
LLM Design	5	Sprint 6	LLM architecture, prompt tuning, etc.
LLM Integration Development	15	Sprint 7	Integrating LLM within the system
Testing LLM	5	Sprint 7	Accuracy and response testing
UI Design for Mobile App	3	Sprint 9	Frontend design adjustments for mobile
Development Frontend	7	Sprint 9	Initial mobile frontend coding
Backend Development	10	Sprint 10	Core backend logic
Backend and Frontend Integration	10	Sprint 10	Connect frontend with backend
Testing Mobile App	5	Sprint 11	Final QA, user testing, and bug fixing
Integration with Mobile App	10	Sprint 8	Integrating ML/LLM APIs with mobile functions

Introduction, State of the Art, Methodology

Introduction

Today, healthcare delivery is fast and full of information; it is yet challenged by issues emanating from handwritten prescriptions that often lead to readability problems, misinterpretation, and confusion for non-medical users. The problem of illegible handwriting is emphasized in the quote below: 'Doctors' sloppy handwriting kills more than 7000 people annually (Caplin, 2007). Bad medical prescription handwritings pose a significant risk, as previous studies also point out its potential to cause critical errors, which include fatal outcomes, sometimes which makes the pharmacist struggle to interpret them and give proper dispensations (Brits et al., 2017). These are further exacerbated by concerns over counterfeit drugs in some regions.

Advancements in computational capability have considerably eased the integration of AI and machine learning into mobile applications; hence, resource-constrained devices can efficiently carry out tasks related to object detection while judiciously balancing on-device limitations with backend processing (Widad, 2024). This project, therefore, tries to address the challenges outlined above through a mobile application, the Handwritten Doctor Prescription Reading App, which allows users to scan, read, and securely store prescriptions. The App integrates advanced machine learning technologies, LLMs, and secure data storage for accurate and accessible medicine details, targeting parents with young children, elderly individuals, and those with limited medical knowledge.

More About the Problem

The issue this application will be dealing with is indeed very complicated, containing both technical and user-centered challenges. Handwritten prescriptions are easy to handle for medical

practitioners, but in most cases, have presented some difficulties to the patients owing to illegibility or inclusion of medical jargon. Poor understanding and difficulty in reading of labels is not uncommon and may lead to adverse drug event (Gurwitz et al., 2003). Less literate patients would want more help for their prescription (Patel et al., 2013). In the event of misinterpretation, wrong medication use may ensue, leading to dangerous health consequences. Besides, most people have no way to verify the authenticity and quality of medicines in areas suffering from high prevalence of counterfeit drugs. Therefore, a consumer of drugs would probably question the authenticity of drugs that pass-through pharmacies; consequently, they may have little trust in their prescription.

Breaking down the problem further:

Handwriting Recognition in Medical Domain: Recognizing and interpreting handwritten texts are a complex machine learning problem, in the medical domain particularly, as handwriting from doctors is notorious to decipher.

Access to Reliable Medication Information: Most of the time, patients do not have a reliable source through which to understand dosage, possible side effects, and proper use of prescribed medications.

Data Security: It requires solid security measures for storing sensitive medical data privately, with due regard to the essential regulations related to the health sector.

Significance of the Problem

Prescription issues taken down in handwriting are a major cause for concern in healthcare delivery. There is a need to develop trustworthy tools that identify fake drugs; there is generally insufficient detection through visual inspection, with an absence of access to valid reference libraries that have hitherto been validated for data obtained from analyses (Bakker et al., 2021). Poorly understanding a prescription can lead to wrong doses at the wrong times or misusing medications in ways that might have negative effects on their health. Parents managing a number of medications for young children, or elderly patients who may have problems remembering-especially bear heightened risks associated with prescription errors. Because these groups already represent more vulnerable populations when it comes to health complications, finding an effective way to understand and double-check prescriptions becomes quite important.

The app also covers one of the most recent concerns: counterfeit medicines that can seriously affect the health of the citizens.

To that end, the application would independently verify the details of prescription drugs, hence boosting users' confidence in taking the right medicines, especially in those countries with potential pharmacy credibility issues.

Therefore, addressing this problem has implications for improved health outcomes by reducing medication errors, making active users of their health, and creating trust in the system by having

reliable prescription information. It holds great importance for the individuals and overall healthcare system, thereby providing an intuitive way to read and verify prescriptions without health complications or hospital admission caused by medication misuse. In this regard, vulnerable populations, like children and the elderly, will greatly benefit since the app reduces risks related to drug-related events and dispels distrust driven by concerns such as counterfeit medications. Without the implementation of such a solution, these issues would not be resolved, and patient struggles and lost opportunities for better management of their healthcare using technology would go on unabated.

Standards and Legal Implications

This application will need to process and store sensitive personal information regarding an individual's health; therefore, strict conformance must be given to regulations concerning data privacy. Conformity to healthcare data standards within the applications of an Intelligent Prescription System is necessary in light of interoperability challenges, clinicians' adoption, and requirements for data protection that are associated with evolving digital health technologies (Tantray et al., 2024). Healthcare data is usually well-protected under various legislation, such as US HIPAA and European GDPR. GDPR requires that all user data processing be presented clearly, stating how collection and processing are done and what for, as well as what rights users have. According to Literature, any healthcare application that is really GDPR-compliant has got to replace marketing statements with transparent privacy policies that are based on explained data processing activities, in order to maintain the level of trust by the users (Mulder, 2019). These are the regulations which demand that health-related mobile applications offer the type of security measures required in protecting user data.

Legal and Regulatory Requirements:

Data Encryption: The prescription data needs to be fully encrypted in nature so that it may eliminate the possibility of misusing access.
User Consent: Informed consent of users over data collection and use of information.

Access Control: Permit access to sensitive data to authorized persons or systems only.

Additionally, the implementation of best practices in the development of the mobile health application provides the assurance that procedures for handling data are adequate with the standards of ethics and requirements of the law, giving confidence to users in the application's commitment to their privacy protection.

Aim and Objectives

The setting of measurable objectives allows progress to be gauged in software development, and although some measures are standard, many times custom measures have to be defined that are concrete and accurate, operationalized, and validated for a particular purpose (Kästner, 2022). The overarching aim of this project was to design and develop a mobile application capable of scanning, correctly interpreting, and storing handwritten prescriptions with the view to provide

the user with dependable, accessible information about their medications.
To address this aim, the following measurable objectives have been outlined for the project:

- Prescription Scanning using a Machine Learning Model: The system will design and train, most likely, a CNN machine learning model that would read and make sense of diverse medial prescriptions written in numerous forms of hand writing style.

- Integration with LLM: Integrate with an LLM, like Gemini for instance to educate the patient about his/her medication on how to use it including side effects, dosage instructions etc.

- Database for Storing Data Safely: Design a database solution to securely store the prescribed information. This will be encrypted with proper access control, wherever applicable, based on data protection legislation.

- Save and Retrieve Facility: Provide a facility to save scanned prescriptions so that when needed, it is easy to retrieve and hence have easy access to the medication information.

- Accessible User Interface Design: In tune with the needs of highly varied users, both in age and technical background, an intuitive and accessible interface should be designed to guarantee ease in navigating around the application.

- Model Evaluation and Continuous Improvement: Periodically assess the performance of the ML model on new prescription data to sustain its accuracy and reliability, ensuring updates where necessary.

This work will be done in pursuit of these goals, by the project moving along: first, data collection and model training; second, LLM integration, database development, and user testing. This structure ensures that each part of the application will be developed systematically toward the final result.

State of Art

Handwriting recognition of medical prescriptions and the integration of LLMs in healthcare applications represent emerging families of research work, improving this critical gap in patient accessibility of precise prescription information. Indeed, LLMs are revolutionary in their approaches, especially when combined with machine learning and deep learning frameworks, allowing high-accuracy reading and interpretation of handwritten prescriptions. A review of the current solutions, recent advancements in handwriting recognition within health care, LLM applications in medical tasks, and data privacy concerns to provide safe and efficient digital health solutions are given below.

Literature Review of Existing Solutions

The current solutions for prescription management either constitute an e-prescription system or mobile applications with reminders about dosages and overall management of prescription schedules. Though e-prescriptions are known to be quite effective in reducing incidence from handwriting errors, they have not widely permeated small or remote health where the prescriptions are indeed handwritten. Moreover, many of the applications used for managing medications cannot read handwriting. This is a potential problem that might face patients in understanding their prescription provided by their physicians. And also, to train the model datasets (prescriptions) need to be collected. The Handwritten prescription images need to be carefully collected and preprocessed through cropping, sharpening, and word segmentation to support accurate machine learning analysis (Mia et al., 2024).

The illegibility of doctor-written prescriptions has become a critical issue since such prescriptions have been complicated and often unreadable. More specifically, pharmacists and patients often come across crucial difficulties in prescription interpretation which may lead to hazardous consequences, including medication errors. The study entitled "Digitalization of Doctor's Handwritten Prescription using Optical Character Recognition" presented a solution that leveraged the integration of Optical Character Recognition with machine and deep learning methods to accurately extract text from medical records, thus overcoming the difficulty in deciphering physicians' shorthand, cursive, and multilingual prescriptions. Utilizing image processing, word segmentation, and a CNN-LSTM-based model in conjunction with Unicode matching and fuzzy search algorithms ensure structured and accurate outputs that have been optimized against pharmaceutical databases. For this system to minimize human interpretation errors with great efficiency, improved accuracy facilitated by increased training data is allowed through tools such as TensorFlow and Keras (Ujalambkar et al., 2023, p.74).

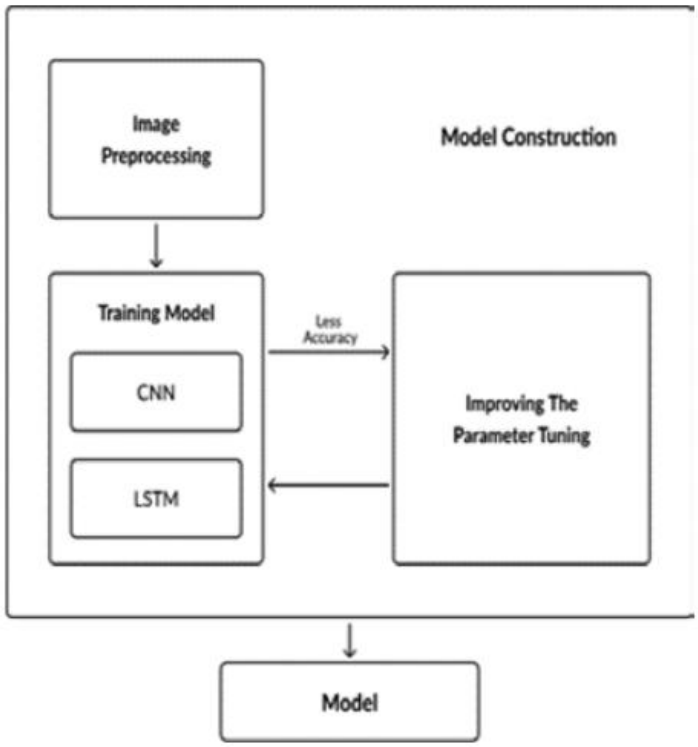


Figure 1: Model Construction (Ujalambkar et al., 2023)

Advanced handwriting recognition technologies have emerged, but unique challenges in interpreting doctors' handwriting and multiple formats for prescriptions limit the reliability of handwriting recognition in healthcare. Current solutions do not provide satisfactory accuracy, especially while capturing complex prescriptions where dosage, frequency, and special instructions must be interpreted as above. Most of the existing applications can also not offer real-time feedback or provide elaborate explanations of the effects and interactions of medications; they are poorly positioned to present a comprehensive healthcare solution.

Handwriting Recognition in Healthcare

The recognition systems in healthcare are quite apart from other recognition systems due to the wide variation of the handwriting style of doctors, notations in abbreviations, and different languages in prescriptions. Several research works have discussed the applications of CNNs for image-based data, where model layers extract relevant features and map them onto a drug database.

Article "Evolutionary convolutional neural networks: An application to handwriting recognition" highlights the use of neuroevolution with genetic algorithms to automate CNN design, achieving impressive results on MNIST without data augmentation or preprocessing. CNN uses layers with various kernels and nonlinear transformations to extract abstract features (feature maps), enabling the network to parse complex visuals (Baldominos et al., 2018).

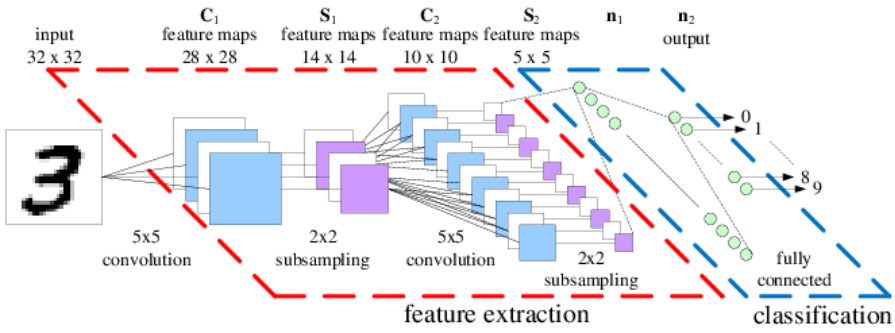


Figure 2: CNN Architecture for Handwritten Digit (Peemen et al., 2011)

Another handwriting recognition solution leverages several deep learning approaches, such as multilayer perceptrons, convolutional neural networks, and Transformer-based architectures. MLPs are relatively simple and allow fast training, being quite appropriate when datasets are small. Complex tasks, related to the recognition of handwritten texts, result in poor performance of this simple architecture; techniques like data augmentation and ensembling are usually necessary to improve performance. Transformer models introduce self-attention mechanisms that perform much better in modeling long-range dependencies and handling longer sequences, thus making it scalable for a wide range of handwriting variations (Rajest, 2024).

Some of the researchers use other techniques that tend to address limitations in the data. Pre-processing techniques such as aspect ratio adjustments in partial plates, and data augmentation strategies like color inversion, shear transformation, and brightness adjustment improve the robustness and generalization of the model (Khokhar & Kedia, 2024). Transfer learning is one of the key approaches in deep learning that optimizes computational resources while maintaining high accuracy in tasks related to OCR. While fine-tuning the pre-trained models such as VGG16, GoogleNet, and EfficientNetV2S, very good results were obtained, and accuracies often reached over 99% (Fateh et al., 2024). This obviously has an advantage in healthcare applications because the collection of large datasets of handwritten prescriptions could be rather problematic.

Another research, which aims to develop an offline OCR application to recognize handwritten medical records, specifically by using the combination of CNN and RNN. Results are promising but research highlights for future improvements in extending data sets to enhance accuracy for real-world healthcare applications (Gifu, 2022).

So, achieving accuracy in handwriting recognition within the healthcare sector is still a tall order due to diverse variability in handwriting styles, abbreviations, and medical terminologies. Because of these errors within the transcripts lead to wrong medication or dosages, which would imply serious health risks to patients.

Integration of Large Language Models in Health Tech

In 2023, rapid development within LLMs has brought into view a changing role of such models- from tools for answering medical queries toward comprehensive instruments of health care that will be a subject of further research aimed at ensuring safety, reliability, efficacy, and privacy

(Cascella et al., 2024). Large language models, such as OpenAI's GPT-4 and Google's Gemini, present sophisticated NLP capabilities, which extend their applicability to health applications that require patient queries or the interpretation of complex data LLMs have demonstrated tremendous potential in healthcare, with key applications in medical education, clinical decision-making, radiologic analysis, and patient communication, but specialized tools such as XrayGPT and DrugGPT increase their applications to medical imaging and drug discovery (Hadi et al., 2024). These models have the ability to process natural language input and provide responses that are more complete and therefore useful in explaining medical information to users with limited knowledge.

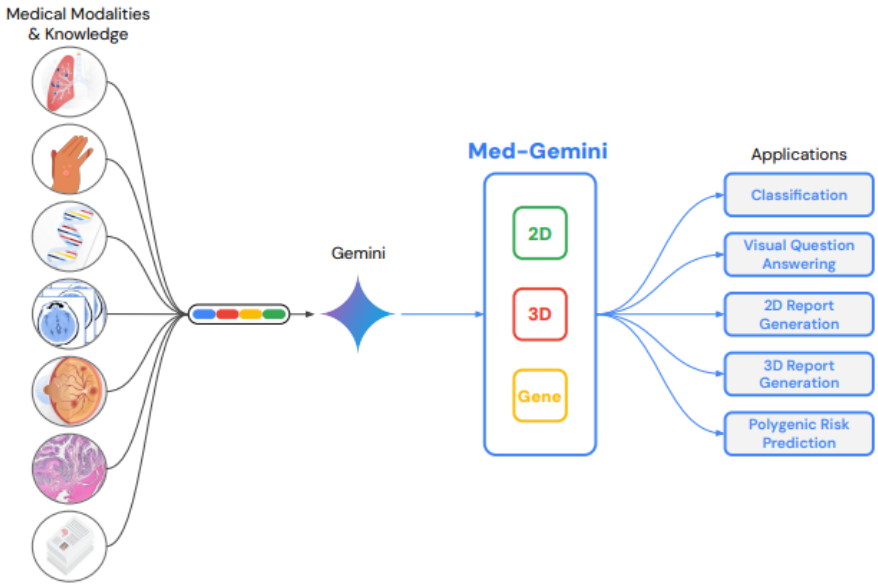


Figure 3: Overview of Advanced Multimodel Medical Gemini (Yang et al., 2024)

In the paper "Capabilities of Gemini models in medicine", the authors have explored multi-faceted medicine and how AI-driven models like Med-Gemini extend clinical processes. Clinicians often face complex cases where a deep integration of patient history, diagnostic images, and other multimodal data demands tools with robust clinical reasoning-such as large language models that provide safe and efficient care. There are already developments of high-performance models, such as Med-Gemini, which leverage multimodal inputivity from text and images to video, aiming at the unique challenges' characteristic of the variability in medical data and high stakes implicated in decision-making. This does not only strengthen the model's accuracy but also promises to be highly useful for such tasks as generating radiology reports and conducting diagnostic dialogues.

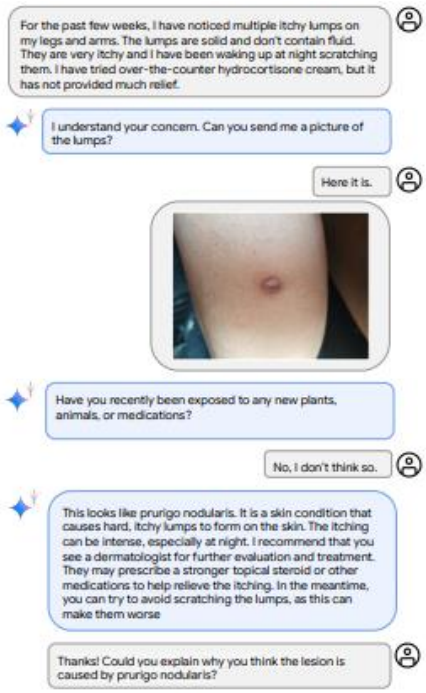


Figure 4: Standard Dialogue through Gemini (Saab et al., 2024)

With Med-Gemini fine-tuned for a range of benchmarks and optimizations specific to a given task, their impact on supporting medical professionals is incrementally felt and further showcases the promise of fine-tuned AI in clinical reasoning and patient care (Saab et al., 2024).

The evaluation of Google's Gemini in the medical domain revealed significant strengths, but also highlighted key limitations, particularly in diagnostic accuracy and the handling of complex visual questions. Despite promising developments, LLMs should complement, not replace, human clinical judgment, with future work focused on enhancing AI reliability and transparency in medical applications (Pal & Sankarasubbu, 2024).

Most of these features also make the use of LLMs in medical domains particularly challenging, including domain-specific training necessary to ensure models output medically correct and reliable information. Large models have very high demands on computational resources; deployments in mobile applications need to be developed with efficiency in mind to avoid excess memory and processing demands.

Data Security and Privacy in Mobile Health Applications

In integrating LLMs into healthcare applications, the most critical concern should be data security and privacy, especially when mobile health solutions work with sensitive patient information such as prescriptions. These models can deal with highly confidential health data, which may raise privacy issues. Keeping that in mind, when such data goes through LLMs,

security needs to be very strict. There are regulatory compliances to be met even further, for example, the HIPAA in the United States and the GDPR within the European Union. Since this is the case, healthcare apps on LLMs should ensure that sensitive prescription data is processed in a completely secure environment.

The author of "A Survey on Security and Privacy of Multimodal LLMs-Connected Healthcare Perspective" underlines the main issues related to privacy and security in developing LLMs applied to healthcare, comprising data leakage linked to third-party API usage and the challenge to implement secure and domain-specific LLMs. It further advocates for a 'privacy-by-design' framework by leveraging techniques such as Federal Learning, differential privacy, and hard access control mechanisms to mitigate risks while enabling model development. Emphasizing the role of human oversight in clinical applications, the authors suggest ways to build trust, monitor outputs, and defend against prompt injection attacks (Rahman, 2023).

At the same time, data privacy requires multi-layer security in mobile health applications. Such protection can be attained through end-to-end encryption of data so that, at the point of transmission between a user's device and application servers, unauthorized access to the latter is restricted.

Besides, secure login-off procedures, like multi-factor authentication that becomes increasingly widespread in healthcare, are important for users to be sure that only the authorized user will have the access to their medical data. So, for a project like this firebase would be much useful because of firebase's advanced capabilities for secure and efficient data storage, real-time updates, and integration with mobile applications, emphasizing its optimization methods, hierarchical data handling, and broad applicability in developing practical projects (Madaminov, & Allaberganova, 2023).

Identified Research Gaps and Implications for the Project

Although there is a lot of improvement in the area of handwriting recognition, LLM integration, and also data security, gaps still persist, which the proposed project is set out to achieve. Most of the existing solutions cannot effectively combine handwriting recognition with the capability of an LLM to interpret prescriptions, and this causes a significant gap between the current solutions. Though CNNs and RNNs are being widely used for character recognition, only a few studies or applications have been made that implement these models specifically for interpreting handwritten prescriptions and integrating the results with LLMs for real-time explanation.

When a novel hybrid model follows there are gaps in handling idiomatic expressions, entity boundary confusion, and ambiguous topics, highlighting the need for refined pre-processing techniques, advanced contextual embeddings, and enhanced data augmentation strategies to enhance model performance across diverse NLP tasks (Liu et al., 2024).
Specific research gaps on which the project intends to work out are being outlined as follows:

Integrate CNN-based handwriting recognition with an LLM as one system, not only for prescription interpretation but also for explanation in a patient-friendly format.

Stringent levels of security for data security in compliance with healthcare standards, with special views toward optimization on mobile application environments.

Test the model on a health dataset to ensure that the LLM provides contextually appropriate and medically correct responses.

Addressing such research gaps, this Handwritten Prescription Reading App will finally contribute to healthcare technological studies by offering a novel solution that merges advanced AI methods with practical application in patient care.

Methodology

Using agile techniques on a project, results on a strong emphasis on model interpretability fosters trust and usability, enabling stakeholders to validate the model's behavior and comply with regulatory requirements, ultimately driving the success of the machine learning initiatives (Cooper & Easton, 2024). So, the methodology for this doctor prescription reading app project follows an Agile framework to ensure flexibility, continuous improvement, and timely delivery of milestones. The project is structured into five phases, focusing on the main objectives: the handwritten prescription scanning model, Large Language Model (LLM) integration, and database development.

Plan for Objectives

The three core objectives of the project would entail an effective model of machine learning in scanning handwritten prescriptions, an LLM system that incorporates responsiveness, apparently with Gemini, and a very robust database management system to manage the data from the app. For these objectives, a methodical approach shall be taken to make sure that indeed the functionalities of the app do comply with both technical feasibility and user feasibility.

Handwritten Prescription Scanning Model: The prescription scanning model will be designed with a Convolutional Neural Network, which has been proved to work well in image recognition. This model is to be trained to detect key features on the prescription, like medicine names, dosages, and doctor signatures. The set of images of handwritten prescriptions makes up the dataset, with areas of interest marked. The training will be performed by supervised learning, where the model learns to map input images to known output labels according to the annotated dataset. Besides, regular checks should be performed to make sure that the model outperforms others with time.

LLM Integration: Integrating LLM into the application will promote the processing of prescription data by giving suggestions or explanations concerning the recognized text to enhance the accuracy and usability of the enriched functions. This is very much true in cases requiring variety in handlings or offering more detailed information about medications listed in the prescription. This will be integrated using Google Gemini, and it will be a delicate engineering process to ensure smooth communication between the ML model, LLM, and the

front-end of the app. The process initiates with prompt engineering interacting with the LLM, where prescription details scanned from the ML model are sent to the LLM. It detects the contents of the prescription, and the data aligns with a template, filtering out irrelevant data and keeping only the data about the products based on their content. Afterwards, it fetches data concerning each medication, and it will send the user's requested information based on the LLM.

Database Development: A secure and scalable database will be developed to store user information, prescription images, recognized text, and other related data. The system must be designed to efficiently handle queries, medicine searches, and implement security features to protect sensitive user information. While the database mechanism is not finalized, I anticipate that Firebase will be used, though this decision will be finalized after further evaluation of the project's requirements.
In this regard, every element will be dedicated to continuous improvement through the collection of feedback from supervisors. This process will be iterative in nature to allow for testing and validation of whether each objective is appropriately met using the Agile approach.

Evaluation Process

Performance metrics are for evaluating the effectiveness of a model, including confusion matrices, F1 scores, and area under the precision-recall curve (Cabot & Ross, 2023). The ML model performance will then be measured with respect to its correctness in identifying handwritten prescriptions based on accuracy, precision, recall, and F1-score. Accuracy defines the overall correctness of the predictions as the ratio between correct predictions and total predictions. Precision is a measure of the fraction of correctly identified prescriptions out of all identified as a particular class, while recall-or sensitivity-is the fraction of correctly identified prescriptions out of all true instances of a class. The F1-score, being the harmonic mean of precision and recall, especially comes in handy in the case of imbalanced datasets. And also, multiclass or multilabel metrics, such as Mean Average Precision (mAP), will be used when the model is predicting multiple labels per prescription. In addition, Top-K Accuracy will be useful when multiple predictions are acceptable. These metrics will be central to evaluating the performance of the model and ensuring the reliability in the context of prescription readings.

Large Language Models make many mistakes, such as hallucinating or generating invalid outputs thus makes the validation of LLMs very challenging (Shankar et al., 2024). Results from the LLM system will include feedback sought from users about its suggestions or information. The evaluation of LLMs in health applications is still difficult because of the 'black box' nature that requires a trade-off between human evaluation to obtain qualitative insights and automated methods to have scaling, considering limitations such as transparency, specificity to a task, and resource constraints (Tam et al., 2024). Therefore, testing will be focused on ensuring that the LLM provides relevant responses, timely and accurate based on recognized prescription data.

The overall success will be rated based on a number of criteria: checks on user satisfaction by the supervisor's feedback whether the app serves its purpose, and the effectiveness of the prescription scanning model and the LLM system will be measured for their accuracy through rigorous testing. The design will consider security so that sensitive user data is well stored and

handled securely in conformity with relevant privacy laws. And also, common method of testing will be application performance, whereby speed and responsiveness of the application to users will be emphasized.

Time & Resource Planning

Effective resource planning is crucial for project success since it considers estimating time, budgets, and tracking towards developing accuracy and timely completion (Rudra Kumar et al., 2022). The whole project is based on how the resources are managed effectively, and time and resources for implementation are dependent on several critical factors. Computing power is one of the primary considerations for training the deep learning model, specifically CNN. Cloud-based solutions, including Google Cloud or AWS, will be researched for scalability and cost efficiency. Firebase will be used for database management, whereas Flutter will be used for designing the front end of the app. The resources required will also be based on the final architecture of the system. For example, if the model is hosted directly on the device application, then high-performance mobile phones with powerful processors and ample RAM are required. Under the current deployment plan, the model will be run in the backend so that the app only requires an ordinary smartphone with modest RAM and a good camera for capturing prescriptions.

Beyond this product-specific resource, the development and training of the model utilize a computer with a robust graphics card to host a high-performance GPU. These are necessary resources to deal with the computational intensity of training a model. Other development, such as coding or designing the app interface, will be done on a regular laptop to minimize additional hardware investments. This resource allocation strategy ensures both efficiency and scalability by catering to technical requirements while keeping the app accessible to more users.

I have allocated total of 450 hours towards the project. The project will be divided into manageable, actionable phases. Project timeline is below.

1. Data Collection and Preprocessing (80 hours): Gathering prescription images, annotation, and preprocessing.
2. ML Model Development and Training (140 hours): Model design, training, and evaluation.
3. LLM Integration (100 hours): Fine-tuning the LLM and integrating it into the app.
4. Database and Security Development (80 hours): Database design and implementation of security measures.
5. User Testing and Final Refinement (50 hours): Collecting feedback from users/supervisors and refining the app based on insights.

Three questions were addressed in a study: how satisfied the project manager is with the project management tool being used, how satisfied the developers are with the tool used, and how to create a design that enhances the experience of project managers and developers. Results showed that Jira Software is an easy, efficient tool highly regarded by most project managers and developers. (AlHarbi et al., 2023). So, the Agile sprint planning process will further refine the time estimates, while project management tool Jira support task management in monitoring

progress against milestones in this project.

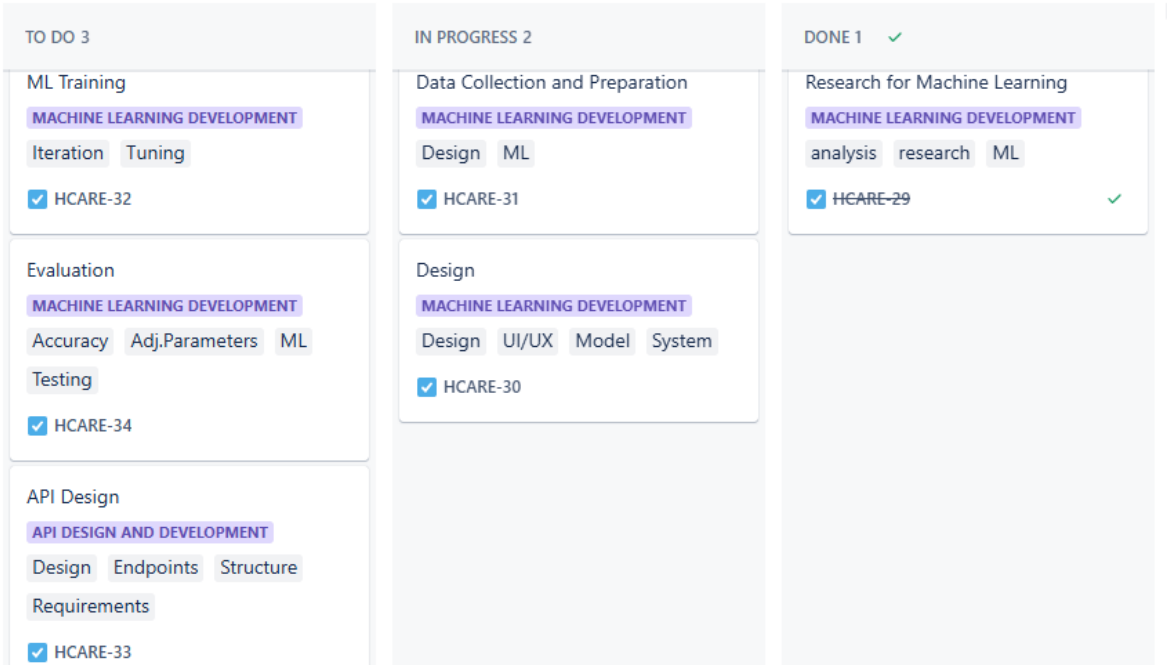


Figure 5: Jira Sprint Board (Atlassian, 2024)

Design and Implementation

Design

Introduction

The design of the Handwritten Prescription Recognition and Validation System establishes the framework for a robust AI-driven mobile application that enhances the accuracy and accessibility of medical prescriptions. Misinterpretation of handwritten prescriptions has long been a challenge in healthcare, often leading to incorrect medication administration, patient safety risks, and inefficiencies in pharmacy workflows (Vashist, Pandey & Tripathi, 2020). The system aims to mitigate these challenges by leveraging deep learning models, including YOLO for text detection and a CNN for text recognition, ensuring precise extraction of prescription details. Additionally, the integration of OpenAI GPT-4 will enhance medicine name validation and provide relevant medical information.

Requirements Gathering

The functional and non-functional requirements were derived from a **combination of industry research, user needs analysis, and technical feasibility studies** (Jebadurai et al., 2021). The primary stakeholders include:

- **Patients and Caregivers** – Need **clarity in medication names, dosages, and side effects**.
- **Pharmacists** – Require **accurate digital conversion of handwritten prescriptions** to ensure correct medicine dispensing.
- **Healthcare Providers** – Benefit from a **digital tool that reduces errors associated with illegible handwriting**.

Through this analysis, the following key requirements have been identified.

Functional Requirements

Must-Have Functionalities

1. **Prescription Scanning & Processing**
    - Users must be able to upload a **handwritten prescription image** via the mobile app.
    - The system should use **YOLO-based text detection** to extract regions containing handwritten text.
2. **Handwritten Text Recognition**
    - The extracted text must be processed using a **CNN-based model** to **convert handwriting into digital text**.
    - The system must recognize various **handwriting styles and prescription formats** with high accuracy.
3. **Medicine Name Validation & Information Retrieval**
    - The extracted text should be validated using **OpenAI GPT-4** to correct possible **misinterpretations of medicine names**.
    - The app should provide users with **three query options** related to the detected medicines, including:
        - **Medicine Summary**
        - **Usage instructions**
        - **Possible side effects**
4. **Secure & Local Processing**

- The application must **process all prescription data locally**, ensuring **no external storage or cloud services** are used.
- Data should be securely managed through Firebase for storing results.

5. **User Interaction & Feedback**

- The app should give the user feature to see the past scanned medicines.

Should-Have Functionalities

6. **Multi-Prescription Handling**

- Users should be able to process **multiple prescriptions in a single session** and save results for later review.

Non-Functional Requirements

Must-Have Requirements

1. **Accuracy & Reliability**

- The **CNN-based recognition model** should achieve a minimum of **85% accuracy** in recognizing text.
- The **YOLO-based detection model** should have a **95% success rate** in detecting text regions.

2. **Processing Speed & Efficiency**

- The system must **analyze a prescription image in under 2 seconds** to ensure real-time usability.

3. **Scalability & Adaptability**

- The application should **support diverse handwriting styles** and different **prescription formats** from various regions.

4. **Data Privacy & Security**

- The app should process prescription data **locally**, ensuring no sensitive information is stored externally.
- It must comply with **healthcare security best practices**, including **data encryption and user authentication mechanisms**.

5. **User Experience & Accessibility**

- The **Flutter UI** should be intuitive, ensuring **ease of use for elderly patients and non-technical users**.
- The interface should be **optimized for mobile usability**, with clear navigation and an accessible font size.

Should-Have Requirements

6. **Offline Mode for Partial Functionality**

    - If no internet connection is available, the app should still allow users to scan and detect text but defer medicine validation until online.

Wireframe and UI Design

The app follows a **clean and minimalistic design approach**, ensuring that users can **easily navigate** between key functionalities. The **primary UI components** include:

1. **Home Screen**

    - Users can **capture or upload a prescription image**.
    - An "Upload Prescription" button initiates **YOLO-based text detection**.
    - Once processed, the app extracts **cropped text images** for user selection

2. Cropped Text Image Selection **Screen**

    - The app presents **cropped images** detected by **YOLO**, allowing users to **select text regions** for further processing.
    - The **"Confirm Selection" button** finalizes the chosen text for recognition.

3. **Results & Medicine Validation Screen**

    - Displays the **recognized text and detected medicine names** from the CNN model.
    - Users are presented with **three interactive options** to query OpenAI GPT-4:
        - **About Medicine**
        - **Side Effects & Warnings**

- **Instructions**

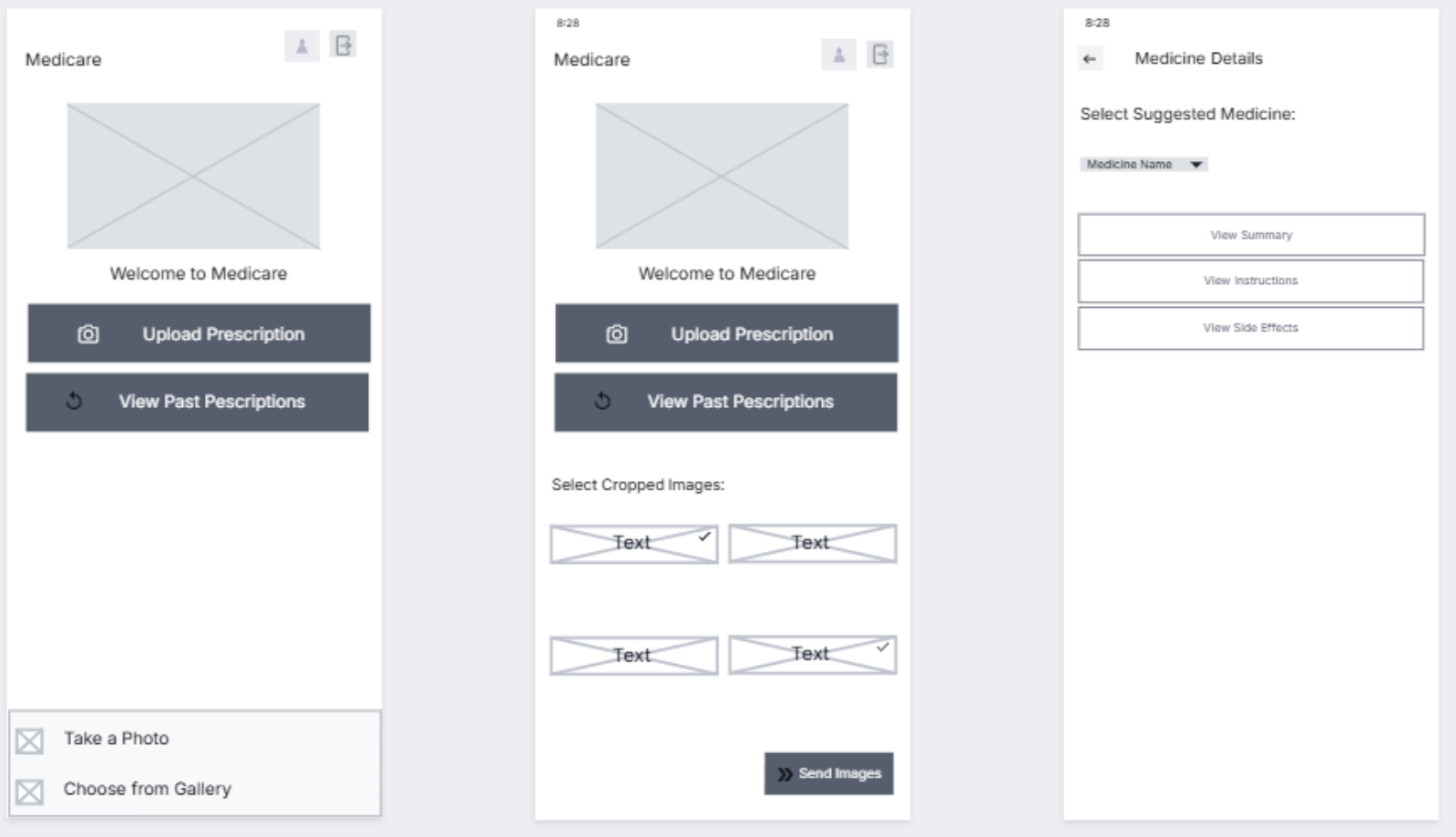


Figure 6: Wireframes of the Medicare App UI

The UI is designed to be **accessible, responsive, and scalable**, ensuring **efficiency in prescription reading while maintaining a high level of usability**.

Architectural Design

System Architecture

The **Handwritten Prescription Recognition System** follows a **modular client-server architecture**, enabling efficient **prescription scanning, text recognition, and medicine validation**. The **Flutter-based mobile app** communicates with a **FastAPI backend** running **on a local server**, ensuring **low-cost deployment** while leveraging **Firebase for data storage** (Beigl, Rudisch & Bialek, 1995).

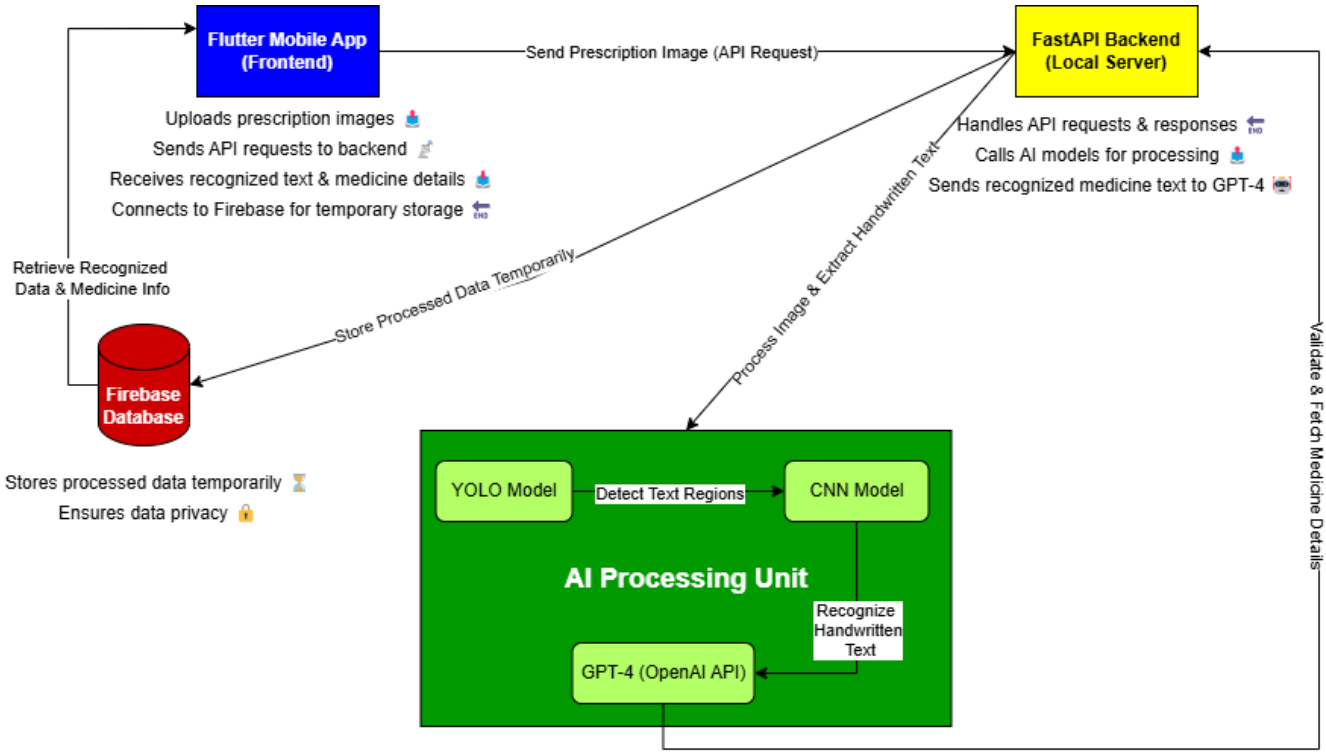


Figure 7: System Architecture Diagram

Code Design

This project follows a structured **modular code design**, ensuring **efficient data flow, maintainability, and scalability**. The **backend, AI models, and frontend** interact through a **well-defined architecture**, facilitating **smooth prescription processing and validation**.

Class Diagram

Figure 8: Class Diagram

FastAPIApp: Acts as the bridge between the Flutter frontend and AI-powered backend. Manages API requests, routes data to the AI pipeline, and interacts with OpenAI GPT-4 for medicine validation.
Predictor: Processes images, detects text using YOLO, recognizes handwriting with CNN, and returns extracted prescription data.
YOLOModel: Detects handwritten text regions in prescriptions and crops them for further processing.
CNNTextRecognizer: Recognizes handwritten text from cropped images and converts it into digital format.
OpenAIService: Validates recognized medicine names and provides summaries, usage instructions, and side effects using OpenAI GPT-4.
Preprocessor: Enhances prescription images by resizing, normalizing, and reducing noise for better text detection.
FirebaseHandler: Stores and retrieves prescription processing data using Firebase.
DataProvider: Handles dataset management, loading, and preprocessing for AI model training.

These classes have been designed to ensure a modular and scalable architecture, following the principles of object-oriented design (OOD). By clearly defining class responsibilities, the system

maintains a clean separation of concerns, improving maintainability and performance. This structured approach enhances system reliability and facilitates future enhancements in AI-driven prescription recognition (Jin et al., 2020).

UML Sequence Diagram

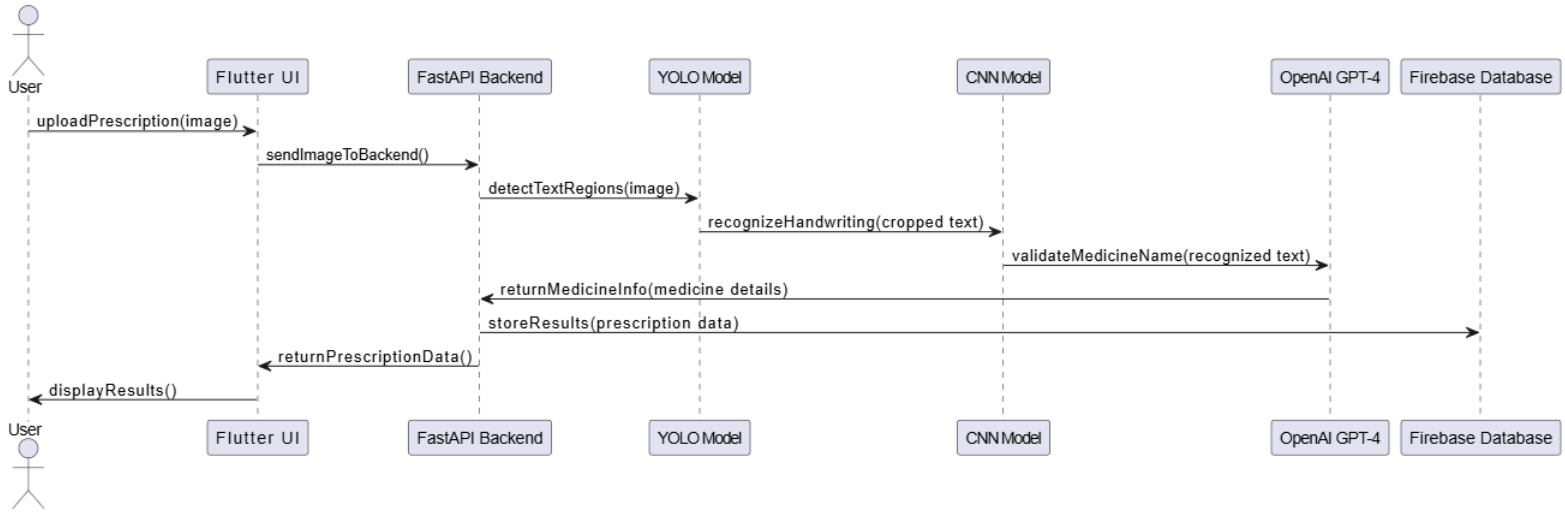


Figure 9: UML Sequence Diagram for Prescription Processing Workflow

State Transition Diagram

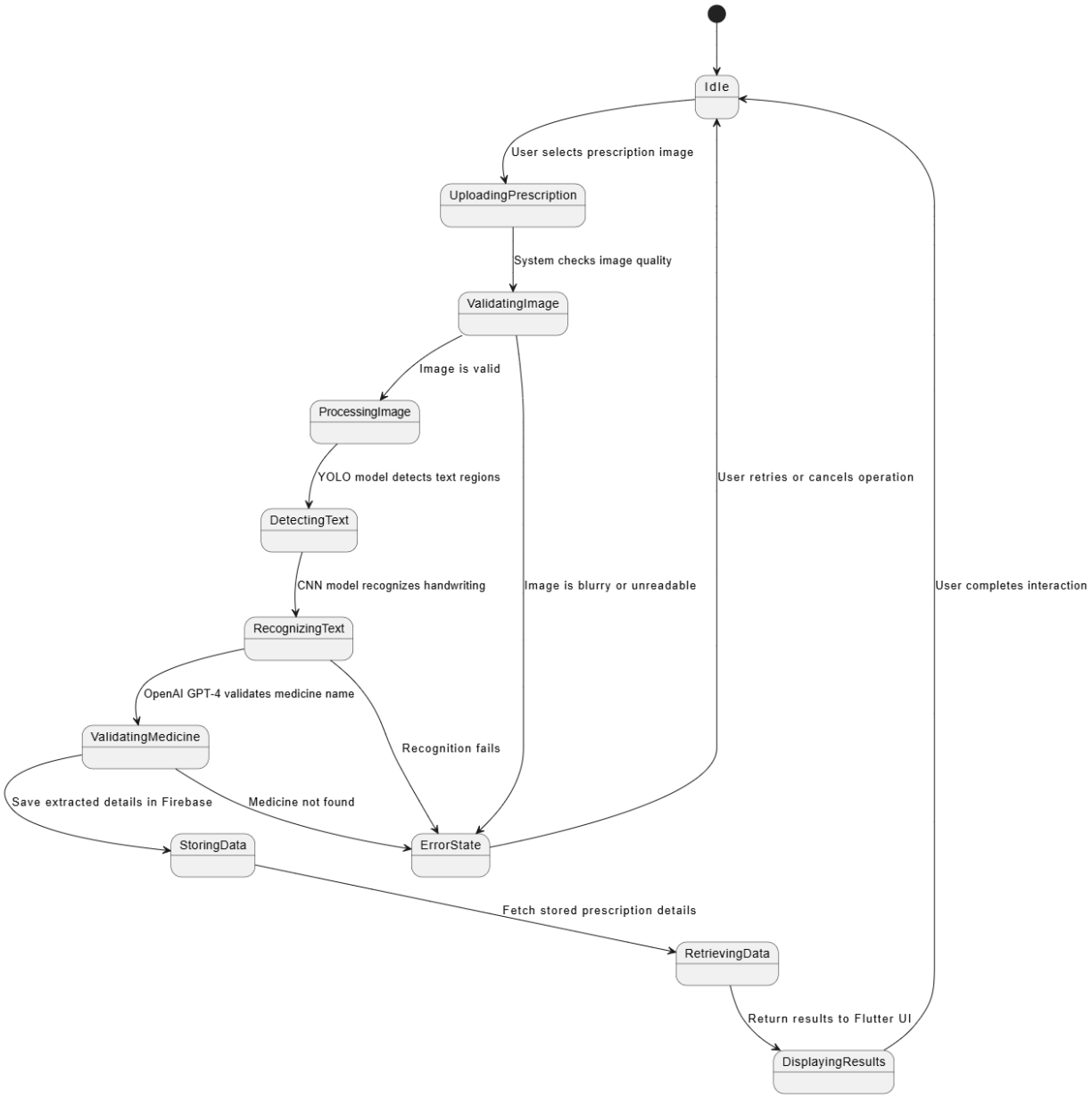


Figure 10: State Transition Diagram

The state transition diagram (Figure X) represents the workflow of the Handwritten Prescription Recognition and Validation System, outlining how prescription images are processed and validated.

The process begins when a user uploads a prescription image. The system first enters the ValidatingImage state to check image quality. Then the system will proceeds to ProcessingImage, where the YOLO model detects text regions. The extracted text is then passed to RecognizingText, where the CNN model converts handwriting into digital text.

Regardless of recognition confidence, the detected text is sent to ValidatingMedicine, where OpenAI GPT-4 attempts to validate the medicine name. If recognition is successful, the correct medicine name is displayed; otherwise, the system assigns the label "INVALID" to the extracted text. The system then transitions to RetrievingData, fetching the stored details before moving to DisplayingResults, where the Flutter UI presents the validated or invalid prescription data to the

user. The process completes when the user reviews the results, returning the system to IdleState for the next input.
This workflow ensures accurate prescription reading and validation, reducing errors and improving accessibility in healthcare (Vashist et.al, 2020).

Development Methodology

The development of the Handwritten Prescription Recognition and Validation System followed an Agile methodology, ensuring iterative improvements and adaptability throughout the project lifecycle. Agile was chosen due to its flexibility in managing complex AI-based development, allowing for continuous integration of feedback and incremental refinements (Lourens et al., 2022).
The project was structured into multiple sprints, each focusing on key milestones such as backend model training, API development, frontend UI design, and integration testing. Tools like Jira facilitated task tracking, enabling issue resolution.
The methodology also incorporated Kanban workflow to maintain task prioritization, ensuring timely delivery of features such as YOLO-based text detection, CNN handwriting recognition, and OpenAI GPT-4 integration. Regular testing cycles allowed for model performance evaluations and UI enhancements.

Implementation

Backend Implementation

The backend of the Handwritten Prescription Recognition System is built with FastAPI for its asynchronous capabilities, enabling efficient image processing (Bansal & Ouda, 2022). It processes prescription images using YOLO for text detection, CNN for handwriting recognition, and GPT-4 for medicine validation before returning structured results.

Uploaded images undergo preprocessing, including resizing and normalization, before YOLO v5 detects and crops text regions, ensuring real-time efficiency (Nori et al., 2023). The extracted text is then recognized by a CNN model, converting handwriting into machine-readable characters for accuracy (Jebadurai et al., 2021).

GPT-4 validates detected medicine names, retrieving dosage and side effect details. Validated results are stored in Firebase for easy access (Chen et al., 2018). Running on a local server, the backend leverages FastAPI's asynchronous framework and CORS-enabled communication to optimize request handling, ensuring fast and accurate prescription recognition while maintaining cost efficiency (Bansal & Ouda, 2022).

```
Function DetectTextRegions(image):
    Load YOLO text detection model
    Preprocess image (resize, normalize)
    Apply YOLO to identify text bounding boxes
    Crop detected regions from image
    Return cropped text images
End Function
```

Figure 11: Pseudo-code for YOLO-Based Text Detection

```
Function RecognizeHandwrittenText(croppedImages):
    Load CNN model for handwriting recognition
    For each cropped image:
        Preprocess image (resize, binarize, normalize)
        Predict text using CNN model
        Convert predictions into structured text
    Return recognized text
End Function
```

Figure 12: Pseudo-code for CNN-Based Handwritten Text Recognition

```
Function ValidateMedicine(recognizedText):
    Extract medicine names from recognized text
    Send extracted names to GPT-4 for verification
    If medicine is valid:
        Retrieve medicine details (summary, instructions, side effects)
    Else:
        Mark text as "INVALID"
    Return validated medicine information
End Function
```

Figure 13: Pseudo-code for GPT-4 Medicine

The Code is in Appendix 1 of the report.

Frontend Implementation

The application comprises multiple screens, guiding users through uploading a prescription, viewing extracted text, validating medicine names, and retrieving additional medication details. When a user launches the app, they can either capture a new prescription image or select an existing one. The image is sent to the FastAPI backend, where YOLO detects handwritten text, CNN recognizes it, and OpenAI GPT-4 validates medicine names (Sarker et al., 2021). The backend returns structured data containing the identified medicines, their dosage, and potential side effects.

The UI dynamically updates with extracted medicines and provides interactive buttons for retrieving details. If recognition fails, the system marks the text as "INVALID", ensuring clarity. The Flutter frontend communicates with the FastAPI backend via HTTP requests, sending images and receiving JSON responses. Firebase integration enables users to store and retrieve past prescriptions without rescanning. Below is a pseudo code representation of the API call:

```
Function sendPrescriptionImage(imagePath):
    Initialize HTTP Request to FastAPI Backend
    Attach Prescription Image to Request
    Send API Request Asynchronously
    Await Backend Response
    If Response is Successful:
        Extract Recognized Medicines from Response
        Display Extracted Medicines in User Interface
    Else:
        Display Error Message to User
```

Figure 14: Pseudo-code for API call implementation in Flutter

The code is in Appendix 3 of the report.

The application follows an asynchronous API workflow, allowing smooth user interactions while waiting for backend responses. The integration of AI-driven medicine validation enhances reliability, providing a comprehensive solution for accurate prescription reading and medication management (Dagne, 2019).

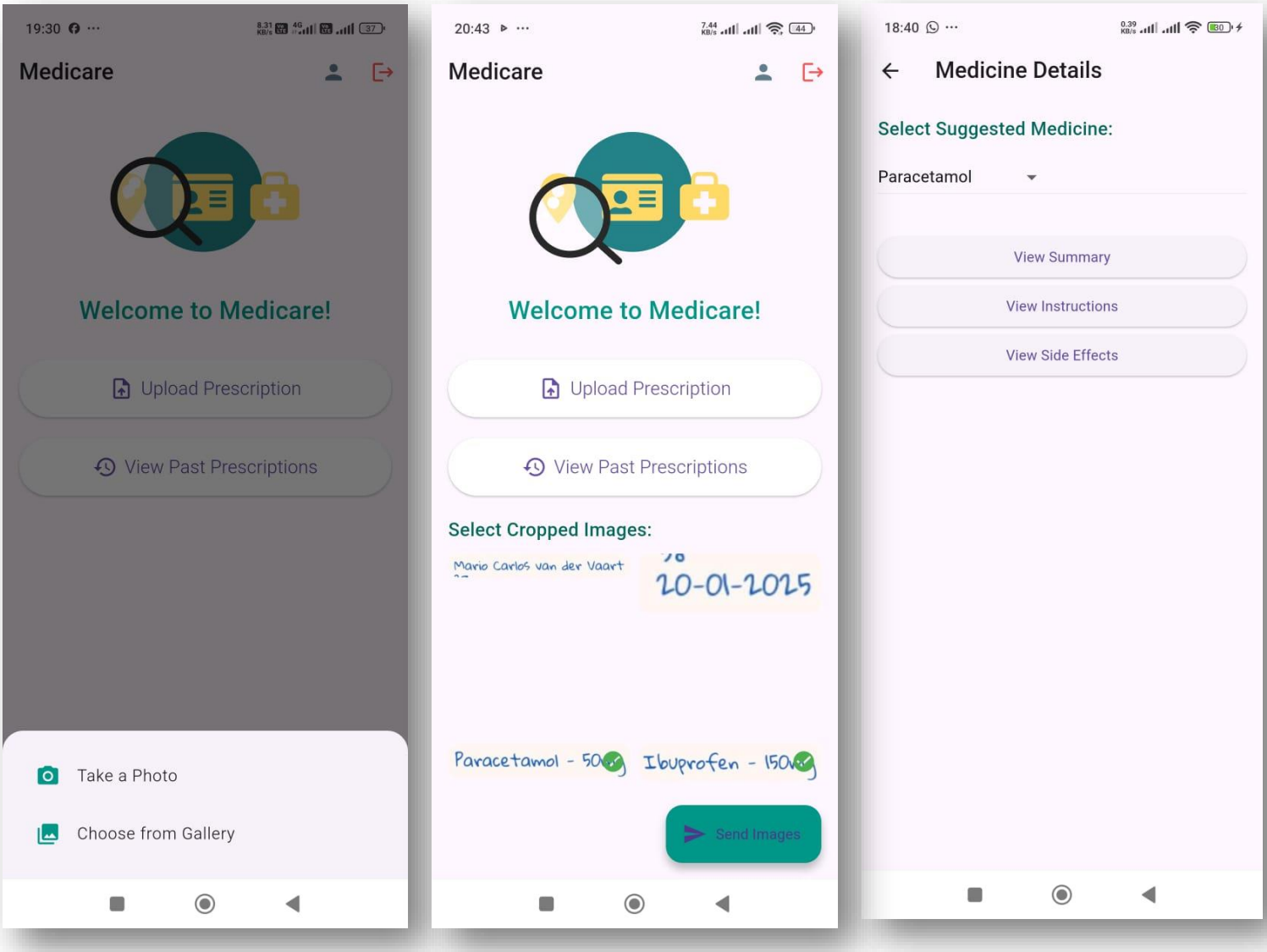


Figure 15: Implemented App's User Interface

Algorithm Design & Challenges

The core challenge in implementing the Handwritten Prescription Recognition and Validation System was ensuring the accuracy and speed of the text recognition pipeline. The system relies on a YOLO model for text detection and a CNN model for handwritten text recognition. While initial implementations provided moderate accuracy, optimizations were necessary to improve recognition performance, reduce misclassification, and enhance real-time response.

The YOLO-based text detection model was initially prone to detecting non-text areas within prescriptions, leading to unnecessary processing. To mitigate this, the system underwent fine-tuning using prescription-specific datasets, adjusting anchor boxes and confidence thresholds to

filter out false positives. Additionally, post-processing steps such as bounding box merging were introduced to combine fragmented detections of a single word, reducing redundant cropped segments and improving efficiency (Nori et al., 2023). The optimized YOLO model was also configured to prioritize recall over precision, ensuring all potential text regions were detected.
The CNN model, responsible for text recognition, faced issues related to variability in handwriting styles, especially with doctor's prescriptions. Early implementations struggled with illegible cursive writing and inconsistent spacing. To enhance performance, data augmentation techniques such as rotation, contrast normalization, and synthetic handwriting generation were applied, increasing the robustness of the training dataset (Jebadurai et al., 2021). The CNN model also incorporated CTC loss function, improving sequential character prediction without requiring pre-segmented labels.
Another challenge was medicine name validation using OpenAI GPT-4. The initial implementation sometimes misinterpreted rare medicine names or suggested incorrect alternatives. To improve validation, a medicine dictionary lookup was added before querying GPT-4, ensuring that only potential matches were sent for further verification. This reduced unnecessary API calls and improved response relevance.
To enhance processing speed, model inference was optimized using ONNX format, reducing computational overhead by enabling hardware acceleration (Jin et al., 2020). The system also employed batch processing, allowing multiple prescription images to be analyzed concurrently, significantly improving performance.

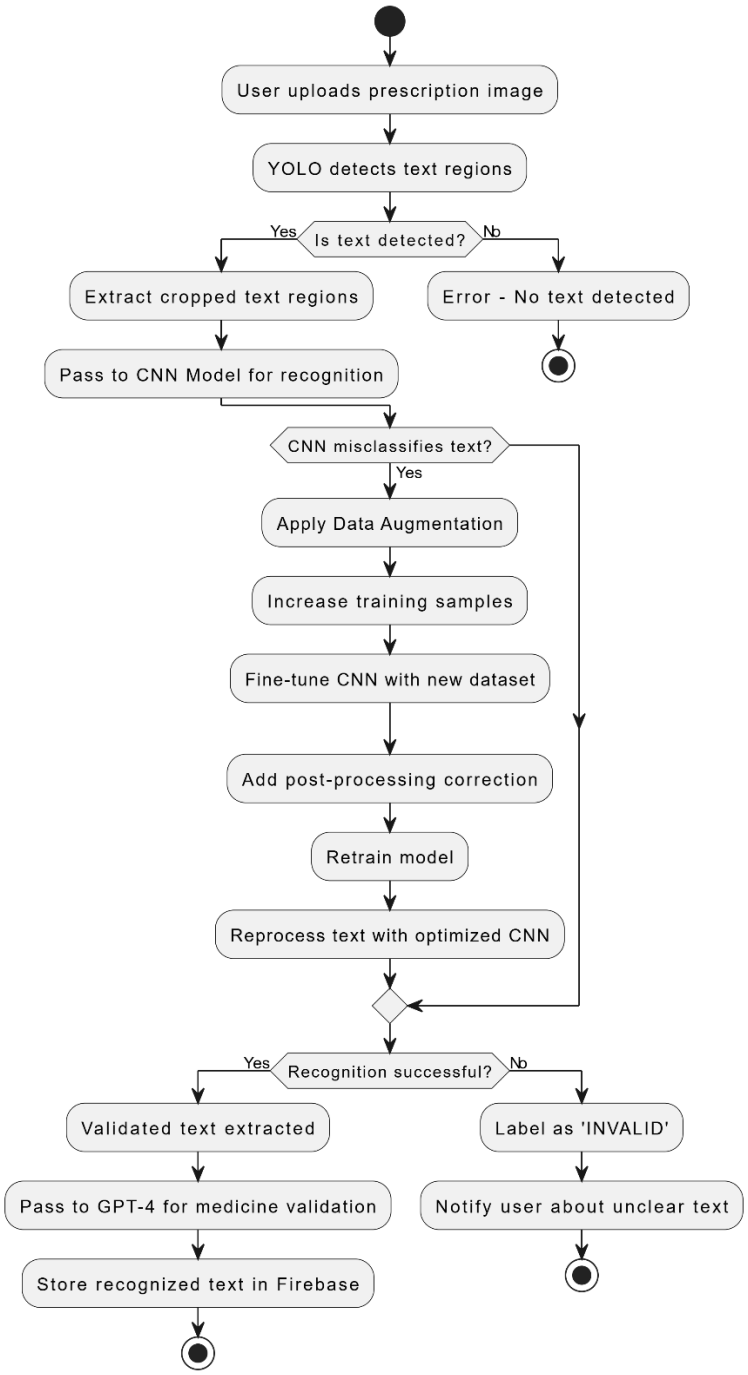


Figure 16: Algorithm Optimization Flowchart – CNN Text Recognition

The Code is in Appendix 2 of the report.

Text Encoding and Data Processing and Formatting

The text recognition system in this project follows a structured pipeline that begins with text detection using YOLO, proceeds to handwritten text recognition via a CNN model, and finally

validates and enriches the extracted text using OpenAI's GPT-4. The output of the CNN model is initially raw text, often containing noise or misinterpretations due to varying handwriting styles. To enhance the accuracy of recognized text and ensure a structured output format, a series of post-processing steps are applied.

When a prescription image is submitted, the YOLO model identifies and crops the regions containing handwritten text. These cropped images are then passed to the CNN-based recognition model, which predicts characters from the handwritten text. The model generates probabilistic outputs for each character, which are decoded using a Connectionist Temporal Classification (CTC) decoder. This decoder removes redundant character repetitions and refines the recognized text, producing the most probable transcription of the handwritten input.

Once the text is recognized, it is validated by GPT-4, which checks whether the detected words match known medicine names. If a match is found, the medicine name is returned with relevant details such as dosage and side effects. If the text is unclear or does not correspond to a valid medicine, the system assigns the label "INVALID", ensuring that misrecognized text does not result in incorrect medication recommendations.

To facilitate efficient storage and retrieval, recognized text is structured in a format similar to a serialized dictionary, where each extracted term is mapped to its validation status and additional metadata.

In cases where text is highly ambiguous, the system applies character-level error correction by


```
{
    "prescription_id": "a1aae41edad04364a195239c5cc40423",
    "recognized_text": [
        {"text": "Dolo 650mg", "status": "VALID", "class": "Analgesic", "dosage": "1 tablet every 4-6 hours"},
        {"text": "Ranexa", "status": "VALID", "class": "Cardiac Medication", "dosage": "500mg twice daily"},
        {"text": "riancon ros", "status": "INVALID"}
    ],
    "processing_time": "1.2s"
}
```


Figure 17: An output format (json) illustrating how detected text is structured

comparing the recognized text against a predefined list of medical terms. This correction process utilizes edit distance metrics such as the Levenshtein distance, ensuring that minor spelling errors in handwritten prescriptions do not lead to incorrect classifications.

Conversion Process

The extracted text is initially a sequence of numerical predictions, which are mapped to corresponding characters.

```
BEGIN
    DEFINE character_map AS "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
    INITIALIZE decoded_text AS empty string

    FOR each prediction_index from 0 to prediction_length DO
        SET index TO predictions[prediction_index]
        IF index is within range of character_map THEN
            APPEND character_map[index] TO decoded_text
        END IF
    END FOR

    TERMINATE decoded_text with NULL character
END
```

Figure 18: Pseudo-code of the decoding logic

The Full Code is in Appendix 4A of the report.

Table 3: Validation of the Medicine Names

	YOLO Detection (Cropped Text)	CNN Recognition (Extracted Text)	GPT-4 Validation (Final Output)	Validation Status
	hery here	hery here	INVALID	✗ Invalid
	r iae somp	r iae somp	INVALID	✗ Invalid
	Dolo 650mg	Dolo 650mg	Dolo 650mg (Validated)	☑ Valid
	rancon ros	riancon ros	INVALID	✗ Invalid
	isavuconazole	isavuconazole	Isavuconazole (Validated)	☑ Valid

A text recognition's VS Code terminal output is in Appendix 4B.

Testing

Testing plays a crucial role in ensuring that the Handwritten Prescription Recognition and Validation System functions as intended, minimizing errors and optimizing accuracy. Since the project was not deployed, testing was conducted primarily during the development phase, focusing on real-time execution and iterative improvements. The main testing approaches included unit testing, load testing using Postman, and integration testing through real-time API validation. These methods ensured that each component of the system—ranging from text detection (YOLO), text recognition (CNN), and medicine validation (OpenAI GPT-4)—performed reliably and accurately (Nori et al., 2023).
Both manual and automated testing were incorporated into the development workflow. The testing process followed an iterative approach, where individual features were developed, tested in real-time, and refined based on observed performance. The key areas of testing included:

1. Unit Testing (Feature-Level Testing in Development Stage)
Unit testing was performed during development to validate individual functionalities. The YOLO model was tested to ensure that text detection regions were accurately cropped from prescription images before being passed to the CNN model for recognition. Precision-Recall metrics were analyzed to fine-tune detection thresholds.
Since the system was built using a scratch CNN model, every feature was developed and tested

in real-time, ensuring that recognition models processed input images correctly. The custom CNN model was tested directly on real prescription samples. Various cases, including clear, distorted, and low-quality handwriting, were used to check the model's adaptability. The system was refined iteratively to handle different writing styles with improved accuracy (Jebadurai et al., 2021).

API Functionality Tests (Swagger):

Unit tests for API routes were conducted using Swagger, ensuring that backend endpoints handled requests and responses correctly. This was crucial for validating the integration of image processing, text recognition, and medicine validation (Bansal & Ouda, 2022).

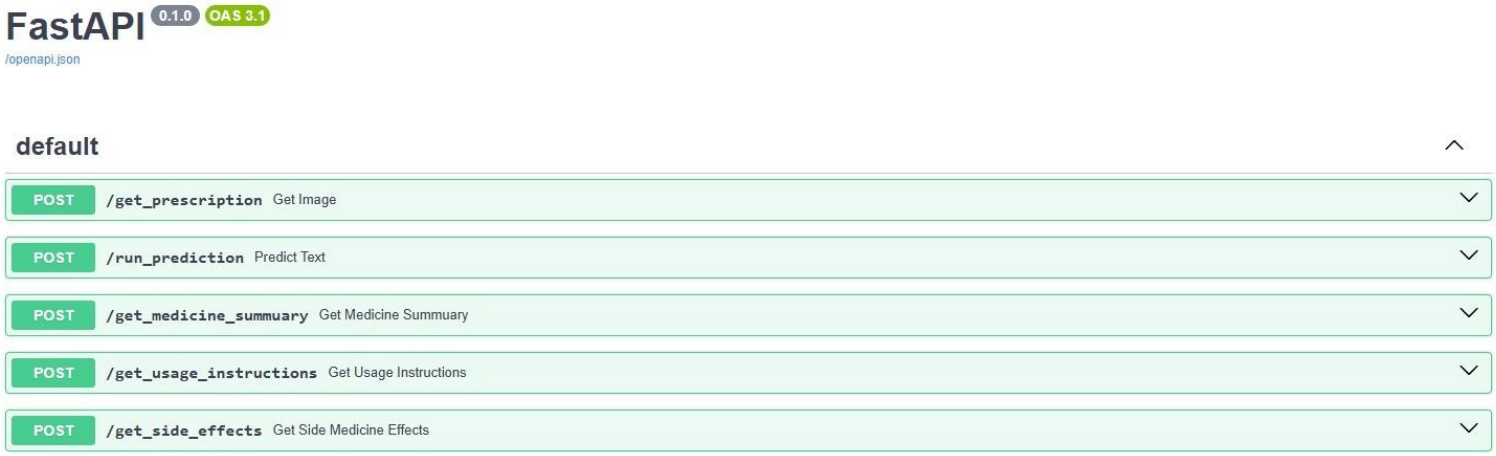


Figure 19: FastAPI-Swagger UI to validate API calls.

There is no a separate code part for this because the unit testing was done manually by running the API calls and verifying responses.

2. Load Testing (Postman API Stress Test)

Since the system was not deployed, load testing was conducted using Postman to simulate multiple concurrent API calls.

The FastAPI backend was tested to see how it handled multiple prescription scans under different load conditions.

Load testing was performed by executing multiple API calls simultaneously, simulating real-world scenarios where multiple users submit prescriptions at the same time.

Response times were measured to ensure efficient processing within the expected range (under 2 seconds per prescription scan).

3. Integration Testing (Frontend & Backend Communication)

Integration testing was conducted by connecting the Flutter frontend with the FastAPI backend and running full workflows multiple times to check stability.

API Integration Testing:

The system was tested by uploading prescriptions via the frontend, processing them through the backend, and retrieving extracted medicine names and validation results. The real-time API logs

in VS Code terminal were used to monitor API request handling and catch any errors.
Error Handling & Edge Cases:
The system was tested for failure cases, such as corrupt images, unreadable handwriting, and invalid API requests. If a prescription was illegible, the system correctly returned "INVALID" as the response instead of attempting incorrect medicine recognition.

Figure 20: API Request Handling

Evaluation, Discussion and Conclusions

Evaluation

Evaluation Process

The evaluation process for this project serves two key purposes: assessing the effectiveness of the developed prescription recognition system and reflecting on the overall development experience (Gates et al., 2019). This ensures that the system meets its intended objectives while also identifying areas for future improvement. The evaluation critically examines the software's technical performance, user experience, and alignment with initial project goals.

Evaluation Frameworks

To ensure a structured evaluation, a **hybrid approach** combining two methodologies was applied.

1. Cross-Industry Standard Process for Data Mining

- Framework Application: CRISP-DM was used to evaluate the machine learning pipeline, particularly the YOLO text detection model, CNN handwriting recognition model, and GPT-4 validation process. This framework ensured a structured evaluation through data preprocessing, modeling, validation, and deployment (Chumbar, 2023).

- Implementation: CRISP-DM was implemented by maintaining a structured pipeline throughout the development phase. Model performance was continuously monitored, with YOLO's text detection accuracy and CNN's recognition success rate tracked at each iteration. Validation steps were documented in testing logs to ensure that incremental improvements were consistently measured. Model building, and evaluation was analyzed against predefined metrics like Character Error Rate (CER) and Word Error Rate (WER).

2. Gibbs Reflective Cycle

- Framework Application: Gibbs Reflective Cycle was applied for personal reflection and project management. This methodology enabled a continuous improvement cycle, where

issues encountered were documented, analyzed, and used to enhance system performance (Mind Tools Content Team, n.d.).

- Implementation: Gibbs Reflective Cycle was used to document personal challenges, debugging strategies, and key milestones. This involved maintaining a personal project log, where issues such as incorrect text detection, model accuracy limitations, and system integration hurdles were recorded. These reflections helped optimize problem-solving approaches and refine the system over time.

Industry Standards and Benchmarking

The model was evaluated against widely recognized industry benchmarks to validate its credibility and effectiveness. The IAM Dataset, a well-known benchmark for handwritten text recognition, was used to compare performance. CTC (Connectionist Temporal Classification) Loss was employed as the primary loss function to optimize sequence learning. Additionally, the trained model was converted to ONNX (Open Neural Network Exchange) format, ensuring compatibility with various AI frameworks for broader usability and deployment.

Technical Evaluation Methods

Comparative Analysis:

The system's prescription recognition performance was compared against existing OCR-based text recognition methods (Ali et al., 2024). The combination of YOLO and CNN significantly improved accuracy over traditional OCR methods, particularly in handling diverse handwriting styles. The switch from Gemini to GPT-4 was another critical improvement, as GPT-4 provided more precise medication validation, reducing incorrect outputs.

User Feedback:

Though large-scale user testing was not conducted, informal feedback was collected from general users, who were asked to upload prescriptions and review the extracted results. The users found the interface intuitive, appreciating the structured output format that included medicine validation and additional details. Future iterations could benefit from direct feedback from pharmacists and medical professionals to validate the system's usability in real-world healthcare settings.

Performance Metrics:

The accuracy of the CNN model resulted in 78%. To quantify the accuracy of the model, Character Error Rate (CER) and Word Error Rate (WER) were employed as the primary evaluation metrics. CER measures the percentage of characters incorrectly predicted by the model, while WER assesses the number of incorrectly recognized words in a given text. These metrics provide a granular view of the model's performance, enabling an in-depth analysis of misclassifications. A lower CER and WER indicate improved recognition accuracy, reflecting the success of optimizations implemented throughout the project.

Gap Analysis:

Gap analysis was conducted to evaluate the discrepancies between the system's current performance and its intended objectives, ensuring that the developed solution aligns with the project's original aims. Given the complexity of handwritten prescription recognition, errors in

text extraction, medicine validation, and response times needed to be systematically assessed to refine the solution (Reddit, 2022).

Revisiting Original Aims

The primary objective of this project was to develop a mobile application capable of accurately scanning, interpreting, and securely storing handwritten prescriptions, ensuring users have reliable access to medication details. By leveraging machine learning and large language models, the system aimed to improve prescription recognition and validation. Throughout development, modifications were made to enhance performance, ensuring the final solution effectively met its objectives. This section evaluates the original aims and the impact of implemented changes.

1. Improving Prescription Recognition with YOLO and CNN

Objective: The initial approach relied on a CNN model to process entire prescription images and recognize handwritten medicine names. The goal was to train the model to handle varying handwriting styles while maintaining high accuracy.

Achievement: Testing revealed that using a CNN alone resulted in poor recognition accuracy due to text clutter and background noise. To resolve this, YOLO was incorporated to detect and crop individual text segments before passing them to the CNN model. This hybrid approach significantly improved recognition accuracy and processing speed (G., P. et al., 2022).

Impact: By integrating YOLO for targeted text detection, the CNN model was able to focus on isolated words, reducing misinterpretations caused by background noise. This optimization enhanced the system's ability to accurately process diverse handwriting styles, improving the reliability of prescription recognition.

2. Medicine Validation and Information Retrieval with GPT-4

Objective: The system aimed to integrate an LLM to provide users with medication details, including usage, dosage, and side effects. Initially, Gemini was selected as the primary language model for validation.

Achievement: During testing, Gemini demonstrated inconsistencies in accuracy, prompting a switch to OpenAI GPT-4. GPT-4 provided more precise validation and comprehensive medication information, improving the reliability of extracted data (Gundersen et al., 2022).

Impact: GPT-4's ability to cross-check medicine names against extensive drug databases ensured better accuracy in validation. The system now delivers precise medication details, making it more dependable for users seeking guidance on prescriptions.

3. Secure Data Storage and Retrieval

Objective: A secure database solution was planned to store prescription data while ensuring compliance with data protection regulations. Additionally, see Prescription History feature was needed to facilitate access to past prescriptions.

Achievement: Firebase was implemented as the database solution due to its robust authentication and encryption mechanisms, ensuring safe storage of prescription data.

Impact: The secure storage system allows users to easily retrieve past prescriptions, enhancing convenience, especially for those requiring ongoing medication tracking. Encryption safeguards sensitive health information, ensuring compliance with privacy standards (Louppe, 2014).

4. Designing an Accessible and User-Friendly Interface
Objective: Given the diverse user base, an intuitive UI was designed to accommodate individuals with different technical proficiencies.
Achievement: The frontend was developed using Flutter, with an emphasis on simplicity, large buttons, and clear navigation.
Impact: The accessible design enables users to upload prescriptions effortlessly and retrieve information, ensuring ease of use for individuals of all ages.

Overall, the project successfully met its objectives, with key optimizations enhancing accuracy and usability. The integration of YOLO with CNN improved text recognition, GPT-4 enhanced validation, Firebase ensured secure storage, and an intuitive UI made the application accessible to all users. These improvements collectively contributed to a robust and user-friendly prescription recognition system.

Customer Satisfaction

The response to the Handwritten Prescription Recognition and Validation System has been highly positive among test users, including elders and caregivers. The system has successfully addressed a significant challenge in healthcare by improving the accuracy of handwritten prescription interpretation. Users have found the application to be intuitive and efficient, particularly in recognizing complex handwriting and validating medicine names. The integration of OpenAI GPT-4 for medicine validation has been well received, as it provides comprehensive information on dosage, side effects, and usage instructions, ensuring patient safety and better medication management (Gundersen et al., 2022).

Gap Analysis and Requirement Fulfillment

A systematic gap analysis was conducted to evaluate whether the project fully met its predefined objectives. The major research gap identified earlier in this research project was the lack of a unified system that could both recognize handwritten prescriptions and provide patient-friendly explanations. This project successfully addressed that gap. The transition from CNN-only text recognition to a combined YOLO-CNN approach significantly improved accuracy, fulfilling the initial requirement for robust text recognition (G., P. et al., 2022).

Acceptance Testing

Rigorous testing, including unit tests and real-world use cases, validated the system's effectiveness. This testing phase ensured that all API endpoints functioned correctly, the AI models delivered accurate results, and the user interface remained seamless. The positive feedback from users reinforces that the system meets practical healthcare needs while aligning with project specifications.

Discussion

Evaluation Findings and Problem-Solution Alignment

Scenario 1: Model Performance & Accuracy Issues (YOLO/CNN Model Refinements)

Description:
In the early stages of training, the system exhibited poor performance, with high Character Error Rate (CER) and Word Error Rate (WER). At epoch 0, validation CER and WER were both at 1.0, indicating the model was completely misinterpreting handwritten text. YOLO struggled with detecting text accurately, often cropping words incorrectly, which further degraded recognition accuracy. CNN-based recognition also failed to generalize well across different handwriting styles. These issues posed significant challenges in ensuring reliable prescription recognition, necessitating multiple refinements to improve performance (**Ge et al., 2021**).

Feelings:
The initial results were frustrating, raising concerns about the model's reliability. The high error rates indicated major flaws in detection and recognition, making real-world deployment seem unlikely. However, knowing that AI models improve through iteration, the team remained focused on refining preprocessing, augmentation, and optimization strategies. The challenge was seen as an opportunity to enhance model performance systematically.

Evaluation:
Implementing several refinements led to a substantial reduction in error rates. By epoch 100, validation CER dropped to 0.0555, and WER improved to 0.2070, demonstrating enhanced recognition accuracy. Key optimizations included resizing images while maintaining aspect ratio, applying data augmentation techniques like brightness adjustment, rotation, and morphological transformations to improve generalization. The CTC loss function was fine-tuned to align predictions better with input sequences, and converting the model to ONNX enhanced efficiency for deployment (**Graves et al., 2006**). These refinements significantly improved the system's robustness but did not completely eliminate occasional mispredictions, particularly with complex handwriting.

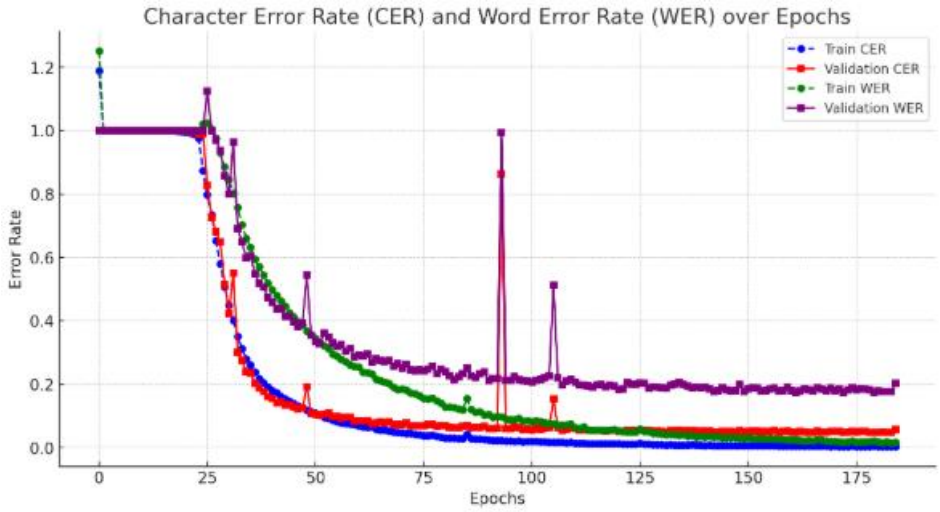


Figure 21: Character Error Rate (CER) and Word Error Rate (WER) Over Epochs

Analysis:
Poor initial preprocessing contributed significantly to the model's low accuracy. Addressing this by improving image resizing and normalization created more consistent inputs. Data augmentation introduced variations that improved the model's adaptability to different handwriting styles. Additionally, fine-tuning the CTC loss function ensured better sequence alignment, reducing recognition errors. Converting the model to ONNX improved computational efficiency, making the system more suitable for real-world deployment.

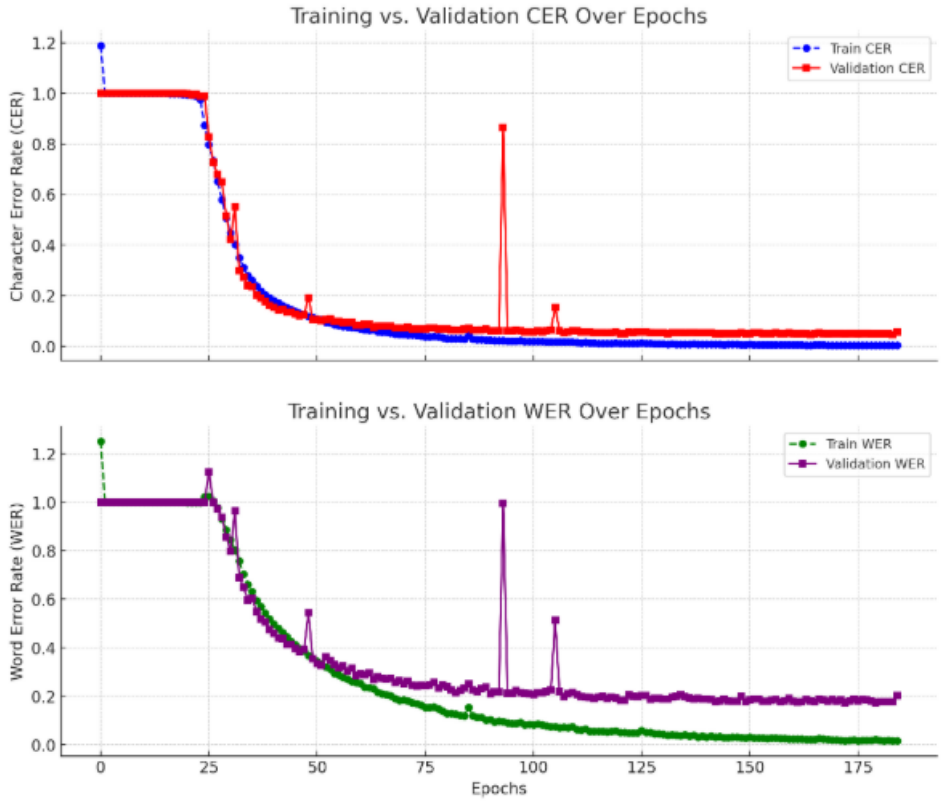


Figure 22: Training vs. Validation CER & WER Over Epochs

Conclusion:
The refinements implemented throughout the training process significantly improved the model's accuracy and reliability. Despite the initial challenges, the optimizations resulted in a more effective prescription recognition system. However, limitations remain, particularly in handling highly distorted handwriting. Further improvements, such as expanding the dataset and exploring transformer-based OCR models, could enhance accuracy.

Action Plan:
Future work should focus on collecting more diverse handwriting samples to strengthen model generalization. Exploring transformer-based models for text recognition could further enhance accuracy. Additionally, deploying real-time inference optimizations will ensure efficient system performance. Continuous testing with real-world prescriptions will help fine-tune the model further, reducing errors and improving overall robustness.

Scenario 2: Technical Challenges & Solutions

Description:
A major challenge encountered during the development of the Handwritten Prescription Recognition System was training the CNN model to accurately recognize handwritten text from prescriptions. The model was initially trained using the IAM dataset, which contains standard handwritten text samples. However, real-world prescriptions introduced significant difficulties due to inconsistent handwriting styles, ink smudging, and complex backgrounds. These variations negatively impacted text recognition accuracy. Finding a solution to optimize the model and improve recognition accuracy became essential.

Feelings:
At first, the high error rates were frustrating and discouraging. Despite using a well-established dataset, the model struggled with real prescription handwriting. The iterative process of testing and tuning parameters felt overwhelming, but every small improvement provided motivation. The challenge reinforced the need for persistence and adaptability in machine learning projects, where problem-solving is an ongoing process.

Evaluation:
To enhance the model's accuracy, several optimizations were implemented. Data augmentation techniques, such as brightness correction, rotation, and erosion/dilation, were applied to make the model more adaptable to various handwriting styles (Retsinas et al., 2024). Adaptive learning rate adjustments were introduced using ReduceLROnPlateau, dynamically modifying the learning rate based on performance improvements (Li et al., 2024). These refinements significantly improved text recognition accuracy while maintaining efficiency.

Analysis:
This process highlighted the importance of domain-specific optimizations in deep learning models. While the IAM dataset provided a solid starting point, real-world handwritten prescriptions required additional fine-tuning. Data augmentation played a crucial role in improving generalization, while dynamic learning rate adjustments ensured that training remained efficient. These enhancements collectively improved the model's ability to handle diverse handwriting styles in prescription data.

Conclusion:
By implementing these optimizations, the CNN model achieved better text recognition accuracy for handwritten prescriptions. The experience underscored the necessity of adapting machine learning models to real-world constraints through iterative refinements. Although initial challenges were significant, targeted improvements enabled the system to perform reliably in extracting text from prescriptions.

Action Plan:
Experimenting with transformer-based architectures such as Vision Transformers (ViTs) could further improve accuracy. Additionally, integrating real-time feedback mechanisms during training would allow for continuous monitoring and fine-tuning of hyperparameters, ensuring sustained improvements in text recognition performance.

Scenario 3: System Deployment Challenges & Adaptability

Description:
The system was designed to function as a local deployment using FastAPI as the backend server, running on a computer, while the Flutter-based frontend operated on an Android device. Unlike many modern AI-based applications, this project was not deployed on a cloud service due to financial constraints and its nature as an academic research project. Running the backend locally introduced several technical challenges, particularly regarding communication between the frontend and backend over different network environments. The system required both the mobile device and the computer running the backend to be on the same network to ensure seamless connectivity (Shamim et al., 2022).

Feelings:
Initially, the lack of cloud deployment felt like a limitation, as cloud-based services provide scalability and ease of access. However, running the backend locally proved to be a practical and cost-effective solution given the project's academic scope. The challenges of integrating the local server with the Flutter app, particularly handling CORS issues and networking constraints, were frustrating but also provided valuable learning opportunities in real-world deployment scenarios.

Evaluation:
One of the major issues faced during the integration was Cross-Origin Resource Sharing (CORS) restrictions. Since the backend was running on a local machine and the Flutter frontend was operating on an Android device, the Same-Origin Policy (SOP) imposed by browsers and mobile networking environments blocked direct communication. To overcome this, CORS policies were configured in FastAPI to allow requests from different origins (Jain, 2023). Additionally, ensuring that both the mobile device and backend server shared the same network and IP address was crucial for establishing a reliable connection.

Analysis:
The decision to keep the system local rather than deploying it on a cloud infrastructure was primarily due to cost concerns and the academic nature of the project. While a cloud-based deployment would have provided benefits such as remote access and scalability, it was not essential for this project's objectives. Running the system locally allowed for rapid testing and debugging without additional infrastructure overhead. However, it also highlighted challenges in network configuration and security, particularly in handling CORS issues and ensuring stable communication between devices.

Conclusion:
Despite the limitations of local deployment, the project successfully demonstrated the feasibility of running a handwritten prescription recognition system on a local server. The experience emphasized the importance of network management, CORS handling, and real-time testing in a non-cloud environment. Future improvements could focus on making the system more adaptable to both local and cloud-based deployments.

Action Plan:
If funding permits, integrating the system with a cloud platform like AWS, Firebase, or Azure

would enhance accessibility, scalability, and long-term usability.

Project Management & Execution

The process of managing time effectively throughout the project was initially challenging, especially when balancing model training, software development, and evaluation. However, by consistently refining scheduling techniques and prioritizing critical tasks, I developed a structured approach to meeting deadlines efficiently. This experience has significantly strengthened my ability to manage multiple responsibilities under pressure, a skill that will be invaluable in future projects and professional endeavors. Overcoming challenges in coordinating development phases and adapting to unforeseen issues has reinforced the importance of flexibility and strategic planning.

Ultimately, this structured approach contributed to the successful execution of the project, demonstrating that with perseverance and effective time management, it is possible to deliver quality results while maintaining a balanced workload.

Incorporation of Feedback & Improvements

Throughout the development of the Handwritten Prescription Recognition System, feedback played a crucial role in refining both the user interface and the model's performance. The project supervisor, Dhanushka Surendra, provided key recommendations that led to improvements in both the user interface and model performance. Based on the feedback, several iterations of the UI were made to enhance its clarity, responsiveness, and overall professionalism, ensuring a better user experience. Adjustments included refining the layout, improving navigation flow, and optimizing the presentation of extracted prescription data.

Another crucial aspect of the feedback was the need to enhance the model's ability to handle complex and difficult handwriting styles. In response, the model underwent further optimization, with improved data augmentation techniques and parameter tuning to increase its accuracy on highly challenging prescriptions. These refinements allowed for better recognition of irregular handwriting, ink smudges, and varying text orientations, improving the overall robustness of the system.

Implementing these recommendations not only improved the project's technical performance but also contributed to making it a reliable tool for prescription recognition. This iterative refinement process played a significant role in aligning the system with real-world usability needs and ensuring its effectiveness in practical applications.

Conclusion

The project successfully achieved its goal of developing a robust mobile application capable of scanning, interpreting, and validating handwritten prescriptions. By integrating YOLO for text detection, a CNN for handwriting recognition, and GPT-4 for medicine validation, the system effectively addresses the challenges associated with illegible prescriptions. The implementation of Firebase for secure data storage and retrieval further enhances user accessibility, allowing for convenient access to past prescriptions.

The system's impact extends beyond its technical contributions, offering a practical solution for patients, caregivers, and healthcare providers who struggle with unclear prescriptions. The combination of machine learning and large language models ensures reliable results, improving

medication safety and reducing the risk of errors.
This project has also provided valuable insights into AI-driven healthcare solutions, demonstrating the potential of integrating machine learning models with real-world medical applications. The development process highlighted the importance of iterative testing, user feedback, and model optimization in achieving a reliable and efficient system.

Further Work & Future Enhancements
Future improvements will focus on increasing the accuracy of the CNN-based handwriting recognition model. Due to resource constraints, the system currently operates on standard computing hardware, limiting its ability to process large datasets efficiently. Enhancing accuracy will require more powerful GPUs or cloud-based virtual machines (VMs) for training. Additionally, the model relies on the IAM dataset, which consists of general handwritten text rather than prescription-specific handwriting. To address this, future iterations should collect real prescription images from users, creating a specialized dataset. Then a data pipeline could be integrated to store user-uploaded prescriptions for continuous model training. Over time, this will refine the system's accuracy, making it more adaptive and reliable in interpreting handwritten prescriptions.
In conclusion, this project has successfully delivered a functional and impactful solution for prescription recognition, bridging the gap between AI and healthcare accessibility. With future improvements, the system has the potential to become an even more accurate and indispensable tool for medical prescription analysis and validation.

References

Kamalanabanand E. Gopinathand M Premkumar. (2018). Medicine Box: Doctor's Prescription Recognition Using Deep Machine Learning.
https://www.sciencepubco.com/index.php/ijet/article/view/18785

Patel, M. J., Khan, M. S., Ali, F., Kazmi, Z., Riaz, T., Awan, S., & Sorathia, A. L. (2013). Patients' insight of interpreting prescriptions and drug labels-A cross sectional study. *PLoS One*, *8*(6), e65019.
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0065019

Hassan, E., Tarek, H., Hazem, M., Bahnacy, S., Shaheen, L., & Elashmwai, W. H. (2021, January). Medical prescription recognition using machine learning. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0973-0979). IEEE.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9376141

Adhikari, R., Richards, D., & Scott, K. (2014). Security and privacy issues related to the use of mobile health apps. ACIS.
https://openrepository.aut.ac.nz/server/api/core/bitstreams/d1dcd354-d95f-4fca-b051-ce03fd443963/content

Pala, G., Jethwani, J. B., Kumbhar, S. S., & Patil, S. D. (2021, March). Machine learning-based hand sign recognition. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 356-363). IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9396030

Aljedaani, B., & Babar, M. A. (2021). Challenges with developing secure mobile health applications: Systematic review. *JMIR mHealth and uHealth*, *9*(6), e15654. https://mhealth.jmir.org/2021/6/e15654/

Garbutt, J. M., Leege, E., Sterkel, R., Gentry, S., Wallendorf, M., & Strunk, R. C. (2012). What are parents worried about? Health problems and health concerns for children. *Clinical pediatrics*, *51*(9), 840-847. https://journals.sagepub.com/doi/full/10.1177/0009922812455093

Johnson, S. L. J. (2019). AI, Machine Learning, and Ethics in Health Care. Journal of Legal Medicine, 39(4), 427–441. https://www.tandfonline.com/doi/epdf/10.1080/01947648.2019.1690604?needAccess=true

Akbar, S., Coiera, E., & Magrabi, F. (2020). Safety concerns with consumer-facing mobile health applications and their consequences: a scoping review. *Journal of the American Medical Informatics Association*, *27*(2), 330-340. https://doi.org/10.1093/jamia/ocz175

Lederm, T., & Clarke, N. L. (2011). Risk assessment for mobile devices. In *Trust, Privacy and Security in Digital Business: 8th International Conference, TrustBus 2011, Toulouse, France, August 29-September 2, 2011. Proceedings 8* (pp. 210-221). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-22890-2_18

Caplin J. (2007). Cause of death: sloppy doctors. Time Magazine http://content.time.com/time/health/article/0,8599,1578074,00.html.

Brits, H., Botha, A., Niksch, L., Terblanché, R., Venter, K., & Joubert, G. (2017). Illegible handwriting and other prescription errors on prescriptions at National District Hospital, Bloemfontein. South African Family Practice, 59(1), 52–55. https://doi.org/10.1080/20786190.2016.1254932

Widad, R. (2024). Implementation of Machine Learning in Android Applications. https://www.theseus.fi/bitstream/handle/10024/859085/Widad_Ramiz.pdf?sequence=2

Gurwitz, J. H., Field, T. S., Harrold, L. R., Rothschild, J., Debellis, K., Seger, A. C., ... & Bates, D. W. (2003). Incidence and preventability of adverse drug events among older persons in the ambulatory setting. Jama, 289(9), 1107-1116. https://doi.org/10.1001/jama.289.9.1107

Patel, M. J., Khan, M. S., Ali, F., Kazmi, Z., Riaz, T., Awan, S., & Sorathia, A. L. (2013).

Patients' insight of interpreting prescriptions and drug labels-A cross sectional study. PLoS One, 8(6), e65019.
https://doi.org/10.1371/journal.pone.0065019

Bakker, I. M., Ohana, D., & Venhuis, B. J. (2021). Current challenges in the detection and analysis of falsified medicines. Journal of Pharmaceutical and Biomedical Analysis, 197, 113948.
https://doi.org/10.1016/j.jpba.2021.113948

Tantray, J., Patel, A., Wani, S. N., Kosey, S., & Prajapati, B. G. (2024). Prescription Precision: A Comprehensive Review of Intelligent Prescription Systems. Current Pharmaceutical Design, 30(34), 2671-2684.
https://doi.org/10.2174/0113816128321623240719104337

Mulder, T. (2019). Health apps, their privacy policies and the GDPR. European Journal of Law and Technology.
https://ssrn.com/abstract=3506805

Kästner, C. (2022). Setting and measuring goals for machine learning projects.
https://ckaestne.medium.com/setting-and-measuring-goals-for-machine-learning-projects-c887bc6ab9d0

Mia, A. R., Chowdhury, M. A. A. S., Al Mamun, A., Ruddra, A. M., & Tanny, N. T. (2024). A Deep Neural Network Approach with Pioneering Local Dataset to Recognize Doctor's Handwritten Prescription in Bangladesh. In 2024 International Conference on Advances in Computing, Communication, Electrical, and Smart Systems (iCACCESS) (pp. 1-6). IEEE.
https://doi.org/10.1109/iCACCESS61735.2024.10499631

Ujalambkar, D., Bhosale, A., Kulkarni, C., Shegar, D., Tandulwadkar, A., & Wagh, P. (2023). Digitalization of Doctor's Handwritten Prescription using Optical Character Recognition (OCR). JOURNAL OF TECHNICAL EDUCATION, 74.
https://www.researchgate.net/profile/Sonali-Mali-4/publication/382180564_IJTE_ugc_care_78_paper_no_July_2023/links/6690c0dc3e0edb1e0fdd662b/IJTE-ugc-care-78-paper-no-July-2023.pdf#page=84

Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary convolutional neural networks: An application to handwriting recognition. Neurocomputing, 283, 38-52.
https://doi.org/10.1016/j.neucom.2017.12.049

Peemen, M., Mesman, B. & Corporaal, H. (2011). Efficiency Optimization of Trainable Feature Extractors for a Consumer Platform. 6915. 293-304. 10.1007/978-3-642-23687-7_27.
https://doi.org/10.1007/978-3-642-23687-7_27

Rajest, S. S. (2024). Handwriting Recognition through Neural Networks: Enhancing Accuracy and Performance. Central Asian Journal of Medical and Natural Science, 5(4), 1010-1024.
https://www.researchgate.net/publication/385010777_Handwriting_Recognition_through_Neural

_Networks_Enhancing_Accuracy_and_Performance

Khokhar, S., & Kedia, D. (2024). Integrating YOLOv8 and CSPBottleneck based CNN for enhanced license plate character recognition. Journal of Real-Time Image Processing, 21(5), 1-12.
https://doi.org/10.1007/s11554-024-01537-2

Fateh, A., Birgani, R. T., Fateh, M., & Abolghasemi, V. (2024). Advancing Multilingual Handwritten Numeral Recognition With Attention-Driven Transfer Learning. IEEE Access, 12, 41381-41395.
https://doi.org/10.1109/ACCESS.2024.3378598

Gifu, D. (2022). AI-backed OCR in Healthcare. Procedia Computer Science, 207, 1134-1143.
https://doi.org/10.1016/j.procs.2022.09.169

Cascella, M., Semeraro, F., Montomoli, J., Bellini, V., Piazza, O., & Bignami, E. (2024). The breakthrough of large language models release for medical applications: 1-year timeline and perspectives. Journal of Medical Systems, 48(1), 22.
https://doi.org/10.1007/s10916-024-02045-3

Yang, L., Xu, S., Sellergren, A., Kohlberger, T., Zhou, Y., Ktena, I., ... & Golden, D. (2024). Advancing multimodal medical capabilities of Gemini. arXiv preprint arXiv:2405.03162.
https://doi.org/10.48550/arXiv.2405.03162

Saab, K., Tu, T., Weng, W. H., Tanno, R., Stutz, D., Wulczyn, E., ... & Natarajan, V. (2024). Capabilities of gemini models in medicine. arXiv preprint arXiv:2404.18416.
https://doi.org/10.48550/arXiv.2404.18416

Pal, A., & Sankarasubbu, M. (2024). Gemini goes to med school: exploring the capabilities of multimodal large language models on medical challenge problems & hallucinations. arXiv preprint arXiv:2402.07023.
https://doi.org/10.48550/arXiv.2402.07023

Rahman, M. A. (2023). A Survey on Security and Privacy of Multimodal LLMs-Connected Healthcare Perspective. In 2023 IEEE Globecom Workshops (GC Wkshps) (pp. 1807-1812). IEEE.
https://doi.org/10.1109/GCWkshps58843.2023.10465035

Madaminov, U. A., & Allaberganova, M. R. (2023). Firebase Database Usage and Application Technology in Modern Mobile Applications. In 2023 IEEE XVI International Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE) (pp. 1690-1694). IEEE.
https://doi.org/10.1109/APEIE59731.2023.10347828

Liu, X., Wang, Y., Niu, N., Zhang, B., & Li, J. (2024). Hybrid Architectures for Chinese Text

Processing: Optimizing LLaMA2 with CNN and LSTM.
https://www.preprints.org/manuscript/202410.1643/download/final_file

Cooper, M., & Easton, W. (2024). Agile Project Management in Machine Learning Projects: Maximizing Business Value through Iterative Development. International Journal of Advanced Engineering Technologies and Innovations, 1(4), 14-30.
https://ijaeti.com/index.php/Journal/article/view/247

Cabot, J. H., & Ross, E. G. (2023). Evaluating prediction model performance. Surgery, 174(3), 723-726.
https://doi.org/10.1016/j.surg.2023.05.023

Shankar, S., Zamfirescu-Pereira, J. D., Hartmann, B., Parameswaran, A., & Arawjo, I. (2024). Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (pp. 1-14).
https://doi.org/10.1145/3654777.3676450

Tam, T. Y. C., Sivarajkumar, S., Kapoor, S., Stolyar, A. V., Polanska, K., McCarthy, K. R., ... & Wang, Y. (2024). A framework for human evaluation of large language models in healthcare derived from literature review. NPJ Digital Medicine, 7(1), 258.
https://doi.org/10.1038/s41746-024-01258-7

Rudra Kumar, M., Pathak, R., Gunjan, V.K. (2022). Machine Learning-Based Project Resource Allocation Fitment Analysis System (ML-PRAFS). In: Kumar, A., Zurada, J.M., Gunjan, V.K., Balasubramanian, R. (eds) Computational Intelligence in Machine Learning. Lecture Notes in Electrical Engineering, vol 834. Springer, Singapore.
https://doi.org/10.1007/978-981-16-8484-5_1

AlHarbi, O., AlMalki, R., & AlYousef, N. (2023). Advancing Project Management Methodologies: An In-Depth Analysis of Jira in Managerial and Developmental Contexts. International Journal of Technology, Innovation and Management (IJTIM), 3(2), 40-59.
https://doi.org/10.54489/ijtim.v3i2.303

Haifeng, D. and Siqi, H., 2020, September. Natural scene text detection based on YOLO V2 network model. In journal of physics: conference series (Vol. 1634, No. 1, p. 012013). IOP Publishing. [Available at: https://doi.org/10.1088/1742-6596/1634/1/012013]

Jebadurai, J., Jebadurai, I.J., Paulraj, G.J.L. and Vangeepuram, S.V., 2021, September. Handwritten text recognition and conversion using convolutional neural network (CNN) based deep learning model. In 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 1037-1042). IEEE. [Available at: https://doi.org/10.1109/ICIRCA51532.2021.9544513]

Vashist, P.C., Pandey, A. and Tripathi, A., 2020, January. A comparative study of handwriting recognition techniques. In 2020 International Conference on Computation, Automation and

Knowledge Management (ICCAKM) (pp. 456-461). IEEE. [Available at: https://doi.org/10.1109/ICCAKM46823.2020.9051464]

Jin, T., Bercea, G.T., Le, T.D., Chen, T., Su, G., Imai, H., Negishi, Y., Leu, A., O'Brien, K., Kawachiya, K. and Eichenberger, A.E., 2020. Compiling onnx neural network models using mlir. arXiv preprint arXiv:2008.08272. [Available at: https://doi.org/10.48550/arXiv.2008.08272]

Nori, H., King, N., McKinney, S.M., Carignan, D. and Horvitz, E., 2023. Capabilities of gpt-4 on medical challenge problems. arXiv preprint arXiv:2303.13375. [Available at: https://doi.org/10.48550/arXiv.2303.13375]

Bansal, P. and Ouda, A., 2022, July. Study on integration of FastAPI and machine learning for continuous authentication of behavioral biometrics. In 2022 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-6). IEEE. [Available at: https://doi.org/10.1109/ISNCC55209.2022.9851790]

Lulle, K., Agrawal, P., Amrutwar, M. and Khiani, S., 2024, December. An AI-Powered Application to Enhance Cognitive Health and Provide Caregiver Support for Dementia Patients. In 2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS) (pp. 93-98). IEEE. [Available at: https://doi.org/10.1109/ICUIS64676.2024.10866122]

Chen, J., Jiang, J., Duan, H., Wan, T., Chen, S., Paxson, V. and Yang, M., 2018. We still {Don't} have secure {Cross-Domain} requests: an empirical study of {CORS}. In 27th USENIX Security Symposium (USENIX Security 18) (pp. 1079-1093). [Available at: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-chen.pdf]

Dagne, L., 2019. Flutter for cross-platform App and SDK development. [Available at: https://www.theseus.fi/bitstream/handle/10024/172866/Lukas%20Dagne%20Thesis.pdf]

Lourens, M., Raman, R., Vanitha, P., Singh, R., Manoharan, G. and Tiwari, M., 2022, December. Agile technology and artificial intelligent systems in business development. In 2022 5th International Conference on Contemporary Computing and Informatics (IC3I) (pp. 1602-1607). IEEE. [Available at: https://doi.org/10.1109/IC3I56241.2022.10073410]

Cabot, J. H., & Ross, E. G. (2023). Evaluating prediction model performance. Surgery, 174(3), 723-726. [Available at: https://doi.org/10.1016/j.surg.2023.05.023]

Sarker, I.H., Hoque, M.M., Uddin, M.K. and Alsanoosy, T., 2021. Mobile data science and intelligent apps: concepts, AI-based modeling and research directions. Mobile Networks and Applications, 26(1), pp.285-303. [Available at: https://doi.org/10.1007/s11036-020-01650-z]

Beigl, M., Rudisch, R. and Bialek, B., 1995. A mobile system integration architecture. Univ., Fak. für Informatik. [Available at: https://edocs.tib.eu/files/e001/249702274.pdf]

Gates, A., Guitard, S., Pillay, J., et al., 2019. Discussion, Limitations, and Conclusion. In: Performance and Usability of Machine Learning for Screening in Systematic Reviews: A Comparative Evaluation of Three Tools. Rockville (MD): Agency for Healthcare Research and Quality (US). Available at: https://www.ncbi.nlm.nih.gov/sites/books/NBK550177/

Chumbar, S. (2023) *The CRISP-DM Process: A Comprehensive Guide*. Medium. Available at: https://medium.com/@shawn.chumbar/the-crisp-dm-process-a-comprehensive-guide-4d893aecb151

Mind Tools Content Team (n.d.) *Gibbs' Reflective Cycle: Helping People Learn From Experience*. Mind Tools. Available at: https://www.mindtools.com/ak3/lifecycle/gibbs-reflective-cycle

Ali, U., Ranmbail, S., Nadeem, M., Ishfaq, H., Ramzan, M.U. and Ali, W., 2024. Leveraging Deep Learning with Multi-Head Attention for Accurate Extraction of Medicine from Handwritten Prescriptions. arXiv preprint arXiv:2412.18199. Available at: https://doi.org/10.48550/arXiv.2412.18199

Reddit, 2022. [Discussion] How to obtain meaningful conclusions from deep learning experiments. [online] Available at: https://www.reddit.com/r/MachineLearning/comments/shg9bx/discussion_how_to_obtain_meaningful_conclusions/

G., P., Padmanabhan, S., Divya, N., V., A., Jerusha, I. and B., C., 2022. Doctors Handwritten Prescription Recognition System In Multi Language Using Deep Learning. arXiv preprint arXiv:2210.11666. Available at: https://doi.org/10.48550/arXiv.2210.11666

Gundersen, O.E., Coakley, K., Kirkpatrick, C. and Gil, Y., 2022. Sources of Irreproducibility in Machine Learning: A Review. arXiv preprint arXiv:2204.07610. Available at: https://doi.org/10.48550/arXiv.2204.07610

Louppe, G., 2014. Understanding Random Forests: From Theory to Practice. arXiv preprint arXiv:1407.7502. Available at: https://doi.org/10.48550/arXiv.1407.7502

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J., 2021. 'YOLOv3: An Incremental Improvement', *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1-6. Available at: https://doi.org/10.48550/arXiv.1804.02767

Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J., 2006, June. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (pp. 369-376). Available at: https://doi.org/10.1145/1143844.1143891

Retsinas, G., Sfikas, G., Gatos, B. and Nikou, C., 2022, May. Best practices for a handwritten text recognition system. In *International Workshop on Document Analysis Systems* (pp. 247-259). Cham: Springer International Publishing. Available at: https://doi.org/10.1007/978-3-031-06555-2_17

Li, Y., Chen, D., Tang, T. and Shen, X., 2025. HTR-VT: Handwritten text recognition with vision transformer. *Pattern Recognition*, *158*, p.110967. Available at: https://doi.org/10.1016/j.patcog.2024.110967

Shamim, S.I., Gibson, J.A., Morrison, P. and Rahman, A., 2022. Benefits, Challenges, and Research Topics: A Multi-vocal Literature Review of Kubernetes. *arXiv preprint arXiv:2211.07032*. Available at: https://doi.org/10.48550/arXiv.2211.07032

Jain, S., 2023. 'Security Beyond Browsers: Why CORS Doesn't Apply to Mobile Applications', *Medium*. Available at: https://saurabh-jain.medium.com/security-beyond-browsers-why-cors-doesnt-apply-to-mobile-applications-99e6ab3e8fe7

Appendix

Appendix 1 – Backend Implementation

```python
app = FastAPI()
SAVE_DIRECTORY = "./text_dir"

load_dotenv()
OPENAI_API_KEY = os.getenv('OPENAI_API_KEY')
client = OpenAI(api_key=OPENAI_API_KEY)

os.makedirs(SAVE_DIRECTORY, exist_ok=True)

model = YOLO('./Model/best.pt')

def process_and_save_detected_text(image: Image.Image) -> list:
    """
    Detect text regions using YOLO, crop them, save to disk, and return saved file paths.

    :param image: Input PIL image
    :return: List of file paths of saved cropped images
    """
    result = model(image, conf=0.6)
    boxes = result[0].boxes


    if len(boxes) == 0:
        raise ValueError("No text detected in the image.")

    box_data = boxes.xyxy.cpu().numpy()
    confidences = boxes.conf.cpu().numpy()

    sorted_boxes = sorted(box_data, key=lambda box: (box[1], box[0]))

    saved_files = []
    for idx, box in enumerate(sorted_boxes):
        x_min, y_min, x_max, y_max = map(int, box[:4])

        cropped_image = image.crop((x_min, y_min, x_max, y_max))

        file_name = f"text_{idx+1}_{uuid.uuid4().hex[:8]}.jpg"
        file_path = os.path.join(SAVE_DIRECTORY, file_name)

        cropped_image.save(file_path)
        saved_files.append(file_path)

    return saved_files


def run_prediction_on_cropped_images():
    recognized_texts = []
    validated_texts = []

    if os.path.exists(SAVE_DIRECTORY) and os.path.isdir(SAVE_DIRECTORY):
        for filename in os.listdir(SAVE_DIRECTORY):
            image_path = os.path.join(SAVE_DIRECTORY, filename)

            if filename.lower().endswith(('.png', '.jpg', '.jpeg', '.tiff', '.bmp', '.gif')):
```

```
                image = cv2.imread(image_path)
                if image is None:
                    print(f"Failed to load the image: {filename}")
                    continue

                predicted_text = recognize_text_from_image(image)
                validated_text = llm_validation(predicted_text)
                recognized_texts.append(predicted_text)
                validated_texts.append(validated_text)
                print(f"Recognized text from {filename}: {predicted_text}")

        return recognized_texts, validated_texts

    else:
        print("No directory found")
        return [], []

def clear_save_directory(directory: str) -> None:
    if os.path.exists(directory):
        for file in os.listdir(directory):
            file_path = os.path.join(directory, file)
            try:
                if os.path.isfile(file_path) or os.path.islink(file_path):
                    os.unlink(file_path)
                elif os.path.isdir(file_path):
                    shutil.rmtree(file_path)
            except Exception as e:
                print(f"Failed to delete {file_path}. Reason: {e}")

def llm_validation(detected_text):
    # detected_text ="Sabursele Cong"
    prompt = f"""I have extracted the following text from a handwritten medical prescription:
            Detected text: {detected_text}
            As a medical expert, analyze the text and determine the most likely correct medicine
            name based on known drug names, spellings, and context. If the text does not clearly
            correspond to a valid medicine, return only "INVALID" to indicate rejection.

        Provide only the corrected medicine name or "INVALID", with no additional text or explanation.
    """
    try:
        completion = client.chat.completions.create(
            model="gpt-4",
            messages=[
                {"role": "user", "content": prompt}], # type: ignore
            max_tokens=50,
            temperature=0.2,
        )
        res = completion.choices[0].message.content
        print(f"Response: {res}")
        return res
    except Exception as e:
        error = f"OpenAI API error: {str(e)}"
        return error
```

Appendix 2 – CNN Model Algorithm

CNN Model Architecture

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class CNNTextRecognizer(nn.Module):
    def __init__(self, num_classes=128):
        super(CNNTextRecognizer, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1)
        self.fc1 = nn.Linear(128 * 8 * 8, 512)
        self.fc2 = nn.Linear(512, num_classes)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = x.view(x.size(0), -1)  # Flatten
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Explanation:

- Three convolutional layers extract features from handwritten text.
- Fully connected layers (fc1, fc2) map features to recognized characters.
- Uses ReLU activation for non-linearity.

Image Preprocessing

```python
import cv2
import numpy as np
from torchvision import transforms

def preprocess_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image, (128, 32))
    image = np.expand_dims(image, axis=0)  # Add channel dimension
    transform = transforms.ToTensor()
    image = transform(image)
    return image
```

Explanation:

- Converts the image to **grayscale**.
- Resizes it to **128x32 pixels** for CNN compatibility.
- Applies **tensor transformation** to prepare for model input.

Model Inference

```
def predict_text(image_path, model, device):
    model.eval()
    image = preprocess_image(image_path).to(device)
    with torch.no_grad():
        output = model(image)
    predicted_text = decode_output(output)
    return predicted_text
```

Explanation:

- Sets the CNN model to **evaluation mode**.
- Preprocesses the image and **feeds it to the model**.
- Calls **decode_output()**, which converts model output into readable text.

Loss Function

```
class CTCLoss(nn.Module):
    def __init__(self):
        super(CTCLoss, self).__init__()
        self.ctc_loss = nn.CTCLoss()

    def forward(self, predictions, targets, input_lengths, target_lengths):
        return self.ctc_loss(predictions, targets, input_lengths, target_lengths)
```

Explanation:

- **CTCLoss** handles **variable-length sequences** in handwritten text.
- Used for mapping CNN outputs to actual **text sequences**.

Appendix 3 – Frontend Implementation

```dart
Future<void> _sendSelectedImages() async {
  showProgressDialog(
      context, 'Processing image... Waiting for the suggested medicines.');

  String apiUrl = "http://172.20.10.11:8000/run_prediction"; // API URL
  var request = http.MultipartRequest('POST', Uri.parse(apiUrl));

  for (String image in selectedImages) {
    var decodedImage = base64Decode(image);
    request.files.add(http.MultipartFile.fromBytes(
      'files',
      decodedImage,
      filename: 'image.jpg',
      contentType: MediaType('image', 'jpg'),
    ));
  }

  try {
    var response = await request.send();
    var responseBody = await response.stream.bytesToString();

    if (response.statusCode == 200) {
      var jsonResponse = json.decode(responseBody);

      if (jsonResponse['suggested_medicine'] != null) {
        List<String> medicines =
            List<String>.from(jsonResponse['suggested_medicine']);

        // Save prescription and medicines to Firestore
        await _savePrescriptionAndMedicines(medicines);

        // Navigate to MedicineDetailsScreen with the list of suggested medicines
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) =>
                MedicineDetailsScreen(suggestedMedicines: medicines),
          ),
        ).then((_) {
          dismissProgressDialog(context);
          setState(() {
            croppedImages.clear();
            selectedImages.clear();
          });
        });
      } else {
        dismissProgressDialog(context);
      }
    } else {
      print('Upload failed with status: ${response.statusCode}');
      dismissProgressDialog(context);
    }
  } catch (e) {
    print('Error uploading images: $e');
    dismissProgressDialog(context);
  }
}
```

```dart
Future<void> _savePrescriptionAndMedicines(List<String> medicines) async {
  try {
    String? email = await AuthService
        .getUserEmail(); // Ensure email is resolved before using it

    if (email != null) {
      await _firestore.collection('prescriptions').add({
        'medicines': medicines,
        'timestamp': FieldValue.serverTimestamp(),
        'email': email,
      });

      print("Data saved successfully.");
    } else {
      print("Error: Email is null.");
    }
  } catch (e) {
    print('Error saving prescription and medicines: $e');
  }
}

Future<void> _handleImage(File imageFile) async {
  showProgressDialog(context, 'Processing your prescription...');
  String uploadUrl = "http://172.20.10.11:8000/get_prescription";
  var request = http.MultipartRequest('POST', Uri.parse(uploadUrl));
  request.files
      .add(await http.MultipartFile.fromPath('file', imageFile.path));
  try {
    var response = await request.send();
    var responseBody = await response.stream.bytesToString();
    if (response.statusCode == 200) {
      var jsonResponse = json.decode(responseBody);
      if (jsonResponse['images'] != null) {
        setState(() {
          croppedImages = List<String>.from(jsonResponse['images']);
          selectedImages.clear();
        });
      }
    }
  } catch (e) {
    print('Error uploading image: $e');
  } finally {
    dismissProgressDialog(context);
  }
}

void _toggleImageSelection(String image) {
  setState(() {
    if (selectedImages.contains(image)) {
      selectedImages.remove(image);
    } else {
      selectedImages.add(image);
    }
  });
}
```

Appendix 4A – CNN-Based Text Recognition & Decoding

```python
def ctc_decoder(predictions: np.ndarray, chars: typing.Union[str, list]) -> typing.List[str]:
    ''' CTC greedy decoder for predictions

    Args:
        predictions (np.ndarray): predictions from model
        chars (typing.Union[str, list]): list of characters

    Returns:
        typing.List[str]: list of words
    '''
    # use argmax to find the index of the highest probability
    argmax_preds = np.argmax(predictions, axis=-1)

    # use groupby to find continuous same indexes
    grouped_preds = [[k for k,_ in groupby(preds)] for preds in argmax_preds]

    # convert indexes to chars
    texts = ["".join([chars[k] for k in group if k < len(chars)]) for group in grouped_preds]

    return texts


def edit_distance(prediction_tokens: typing.List[str], reference_tokens: typing.List[str]) -> int:
    """ Standard dynamic programming algorithm to compute the Levenshtein Edit Distance Algorithm

    Args:
        prediction_tokens: A tokenized predicted sentence
        reference_tokens: A tokenized reference sentence
    Returns:
        Edit distance between the predicted sentence and the reference sentence
    """
    # Initialize a matrix to store the edit distances
    dp = [[0] * (len(reference_tokens) + 1) for _ in range(len(prediction_tokens) + 1)]

    # Fill the first row and column with the number of insertions needed
    for i in range(len(prediction_tokens) + 1):
        dp[i][0] = i

    dp[0] = [j for j in range(len(reference_tokens) + 1)]

    # Iterate through the prediction and reference tokens
    for i, p_tok in enumerate(prediction_tokens):
        for j, r_tok in enumerate(reference_tokens):
            # If the tokens are the same, the edit distance is the same as the previous entry
            if p_tok == r_tok:
                dp[i+1][j+1] = dp[i][j]
            # If the tokens are different, the edit distance is the minimum of the previous entries plus 1
            else:
                dp[i+1][j+1] = min(dp[i][j+1], dp[i+1][j], dp[i][j]) + 1

    # Return the final entry in the matrix as the edit distance
    return dp[-1][-1]

def get_cer(
    preds: typing.Union[str, typing.List[str]],
    target: typing.Union[str, typing.List[str]],
    ) -> float:
    """ Update the cer score with the current set of references and predictions.
```

```
    Args:
        target (typing.Union[str, typing.List[str]]): string of target sentence or list of target words
        preds (typing.Union[str, typing.List[str]]): string of predicted sentence or list of predicted words

    Returns:
        Word error rate score
    """
    if isinstance(preds, str):
        preds = preds.split()
    if isinstance(target, str):
        target = target.split()

    errors = edit_distance(preds, target)
    total_words = len(target)

    if total_words == 0:
        return 0.0

    return errors / total_words

if __name__ == '__main__':
    c1 = 'ROKAS'
    c2 = 'ROKAZ '

    w1 = 'ROKAS GOOD BOY'
    w2 = 'ROKAZ IS A GOOD BOY'

    cer = get_cer(c1, c2)
    wer = get_wer(w1, w2)

    print(wer)
```

Appendix 4B – Text Recognition Process

```
0: 640x544 4 texts, 499.1ms
Speed: 8.0ms preprocess, 499.1ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 544)
INFO:     172.20.10.12:53148 - "POST /get_prescription HTTP/1.1" 200 OK
hery
here
Response: Paracetamol
Recognized text from ce5c6953ba1e40fcaacd95496585fc63.jpg: Paracetamod - soang
Response: Ibuprofen
Recognized text from fadabe663c134bd5b0b91f2690097ba5.jpg: Ebuproten - song
INFO:     172.20.10.12:48942 - "POST /run_prediction HTTP/1.1" 200 OK

Response: INVALID
Recognized text from ac445e6adca74484911c5934574016b3.jpg: seet : sopehat sods
INFO:     172.20.10.12:40888 - "POST /run_prediction HTTP/1.1" 200 OK
```

Poster:

MEDICARE: AN AI-POWERED HANDWRITTEN PRESCRIPTION RECOGNITION SYSTEM

Aiming to reduce medical errors, this project introduces an AI-driven mobile solution to interpret handwritten prescriptions. It tackles challenges of illegible text through deep learning and integrates LLMs to validate medicine names—offering safer, more reliable prescription interpretation.

PROJECT OVERVIEW

Handwritten prescriptions can be misread, risking incorrect medication. This project delivers an AI-powered mobile app that scans, detects, and validates prescriptions using machine learning. The system includes a Flutter frontend and FastAPI backend, leveraging YOLOv5 for text detection, CNN for handwriting recognition, and GPT-4 for medicine validation. Users receive real-time medicine details like dosage and side effects. With Firebase support for history tracking, the app improves communication between patients, pharmacists, and caregivers, ensuring safer and more reliable prescription interpretation.

SYSTEM DESIGN

The user interface is developed using Flutter and designed for ease of use by non-technical users:

- Prescription Upload: Users can scan or upload images for processing.
- Detection & Extraction: Detected text regions are cropped and recognized using the CNN model.
- Three-Query System: Allows users to explore dosage, side effects, and usage with a tap.

TECHNOLOGIES

To achieve high accuracy in prescription digitization, several advanced techniques are used:

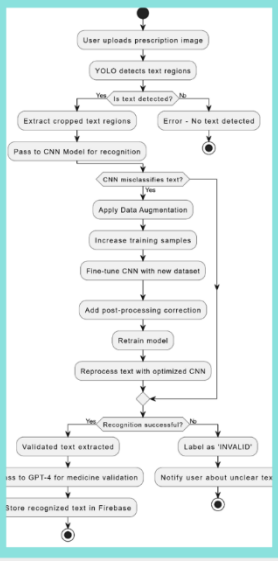
- YOLOv5 for Text Detection: Accurately crops handwritten regions from prescription images.
- CNN-Based Text Recognition: Trained on the IAM dataset and refined with data augmentation to enhance generalization.
- GPT-4 Medicine Validation: Validates and auto-corrects detected medicine names using a large language model.

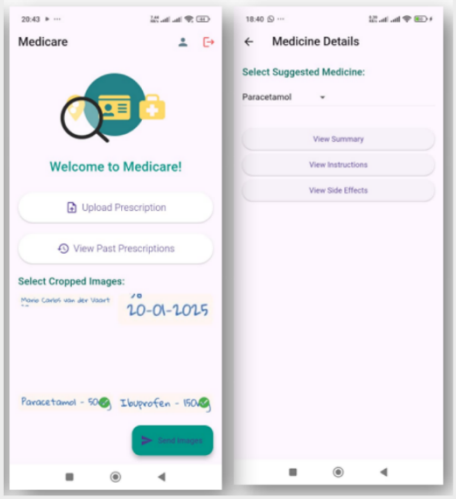
NEXT STEPS...

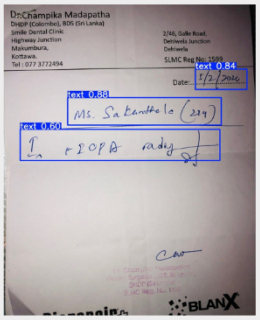
- Deployment to cloud platforms (AWS or Firebase Hosting) for real-time global access.
- Multilingual prescription support for diverse patient populations.

SYSTEM ARCHITECTURE

- Modular Design: Frontend and backend communicate via APIs, allowing independent updates.
- Local Deployment: Cost-effective setup while maintaining performance.
- FastAPI Integration: Ensures smooth backend access from the mobile frontend.







Author
Okitha Perera
Bsc.(Hons) Software Engineering