CS 491 - Senior Design Project I

Raporla.ai

Analysis & Requirements Document

Kerim Eren - 22103759

Ömer Amir Akdağ - 22102918

Ali Cevat Erçal - 22103341

Göktuğ Serdar Yıldırım - 22103111

Efe Tokar - 22103299

Table of Contents

1. Introduction

In the contemporary world, the health system has many challenges. Most of those challenges also apply to Turkey. When statements of the doctors from Turkey are evaluated, it appears that one of the biggest problems is that with the increasing number of patients, the time that doctors can spend per patient is severely limited. In order to fix this problem, automated technologies can be used.

One of the main solutions that many doctors need is the systems that can automatically make preliminary diagnosis. For example, for the fields like neurology or radiology, test results can be evaluated by image processing to make a preliminary diagnosis to help the doctors by reducing their time that is spent by evaluating the already evident findings. However, both the available data and necessary technology is not enough for this idea to become a senior project, even though it may be efficient.

So, our project Raporla.ai focuses on the other main problem that takes up doctors' time unnecessarily long: reports. Doctors must write formal report documents for their examinations and surgeries. These reports may take up unnecessary times since they should be written for every patient detailly. Especially in the fields like neurology, radiology, and pathology, the formal reports may be long and complex, therefore taking up a long time for doctors.

Furthermore, there is no standardized report format in any field. A doctor may write the same examination type’s report very differently from another doctor. This makes it difficult to understand and evaluate a patient's examinations by different doctors at different times.

By recognizing these challenges of the doctors from Turkey, Raporla.ai introduces an automated, speech-to-text report system that is specialized in Turkish and medical literature. By this, we aim to reduce the time spent on procedural things for doctors, and encourage them to focus on detailed examinations and treatments for the patients. Also, by introducing common templates, we aim to standardize the reports among different hospitals and different doctors. Last but not least, the application will also have a preliminary diagnosis suggestion system to create even more time for doctors by helping them to find out evident diagnosis in advance. This system will be purely based on medical datasets to minimize the model’s bias as much as possible.

2. Current System

Existing AI medical reporting tools, such as RadMate AI [1], Amazon Transcribe Medical [2], and Rad AI [3], demonstrate the feasibility of automating healthcare documentation but fail to meet the needs of Turkish professionals. They mainly support English, lack integrated Turkish medical literature, and do not provide standardized templates or preliminary diagnostic suggestions. Raporla.ai addresses these gaps by focusing on Turkish language requirements,

incorporating local medical references, and delivering more comprehensive, context-aware assistance.

3. Proposed System

3.1 Overview

Raporla.ai aims to reduce the unnecessary time that doctors spend on procedural things. As mentioned in the introduction, especially in specific fields like neurology and radiology, writing reports may take a long time since doctors should examine the image results detailly. To this end, Raporla.ai introduces a speech-to-text model that is specialized in Turkish and specifically in Turkish medical literature. Beside these, our application will also have common report templates to bring standardized reporting systems between hospitals and doctors. In addition, there will be a suggestion system that suggests probable preliminary diagnosis to doctors.

By recognizing the importance of time management for doctors, the automated report generation system will dramatically reduce the time spent on reporting. Doctors will only record the examination report in audio. Afterwards, the model will detect the parts of the doctor's statements related to the fields of the report template. Finally, after making the inference, the model will place the necessary parts to the report template and will give the generated report in a formal format.

Furthermore, by introducing common report templates, the communication between the doctors by reports will be easier. There will be common report templates created for each field and each type of examination and surgery. Doctors also will be able to create their own templates and share with other doctors, if approved. The templates will be created, or approved, based on their coverage of the necessary report type. The report templates will be a union of all the necessary fields that must be stated for a particular examination or surgery. If there are empty fields that are not included in the doctor's statements, there will be warnings for them. If the doctor still finds them unnecessary, he/she may ignore the warnings and download the generated report.

In addition, Raporla.ai will have a preliminary diagnosis suggestion feature, too. After generating the report, the system will choose the keywords from the report's related fields. Then, it will search these keywords in a medical dataset, such as PubMed. Its information will be based on previous articles and reports from that medical dataset. By this, we aim to minimize the bias of the model's suggestions. After examining and evaluating the (a limited number of) relevant articles and reports, the system will give probable diagnosis to the doctor along with the sources it used for that particular diagnosis.

All these features aim to create more time for doctors and make them focus on special examinations and surgeries for each patient. Also, by introducing a standardized way of reporting, they aim to make communications between doctors more understandable and clear.

3.2 Functional Requirements

Raporla.ai basically has two sides of functional requirements. For both client and server, the application must be able to:

3.2.1 Client Side Requirements

- Allow users to log in with their credentials.
- Provide a speech-to-text module to doctors to dictate reports.
- Enable doctors to select predefined report templates.
- Support customization of report templates to remove or add fields as needed.
- Contain a validation system that highlights missing fields.
- Allow doctors to approve and finalize reports after validation.
- Display anticipated diagnosis based on keywords taken from the report and verified against medical datasets like PubMed.
- Download the report.
- Delete an existing report.
- Delete their account.
- Search an existing report
- Offer a user-friendly interface in Turkish, optimized for medical terminology.

3.2.2 Server Side Requirements

- Authenticate users during login and manage session security
- Generate standardized reports using predefined templates using the speech-to-text model fine-tuned for Turkish medical terms.
- Map the transcribed text into the selected report template fields.
- Validate the generated report by checking for missing fields.
- Store predefined and user-customized report templates in the database.
- Store generated reports securely
- Query external medical databases (e.g., PubMed) using keywords extracted from the report.
- Handle deletion of user data or reports.
- Support real-time processing for interaction between the client and backend services.
- Support role-based access control, such as differentiating between doctors and administrators.

3.3 Non-Functional Requirements

3.3.1 Usability

Raporla.ai will be used for a considerable number of users. So, the interface should be user-friendly and easy to use. Raporla.ai is designed to be highly usable so that doctors do not need to go through any heavy training and can get hold of it in no time. The idea behind the approach was to fit into the daily routines of doctors seamlessly. Moreover, the application integrated with the Turkish language and medical terminology on accuracy in transcription, report generation, and interaction specified toward the needs of Turkish doctors [4].

3.3.2 Reliability

The Raporla.ai system is designed to maintain performance in all conditions necessary for its operation that is not to experience a crash or failure against demanding conditions. The Speech-to-Text functionality is engineered to maintain accuracy, effectively handling background noises and voice tone or accent variations. It has been designed to consider the most critical needs of healthcare professionals to ensure consistency and dependability of the results [4].

3.3.3 Performance

Raporla.ai is prioritized for high performance to provide an ideal user experience. The system is designed to generate reports within seconds, therefore minimizing the delay and providing a smooth workflow is important. All this emphasis on speed and efficiency means the application raises productivity without sacrificing the accuracy and dependability expected from medical reports and decision-making [4].

3.3.4 Supportability

The application should be easy to update for incorporating new medical datasets and templates [4].

3.3.5 Scalability

Raporla.ai must be able to manage growing user counts and concurrent audio uploads without experiencing appreciable performance drops. New features like integration with hospital management systems should be supported by the application [4].

3.4 Pseudo Requirements

- Python will be used for the backend, the speech-to-text functionality and report generation processes and also libraries such as NumPy and Pandas.
- FastAPI will be used as the backend framework.

- React.js will be used for the frontend interface.
- MySQL will be used for storing user and report data.
- AWS or Google Cloud will be used for hosting.
- Google Colab will be used for training and experimentation of machine learning models with GPU resources.
- Git with GitHub will be used as the version control system.

3.5 System Models

3.5.1 Scenarios

3.5.1.1 Create Report

Use Case Name: Create Report

Participating Actors: User and all actors that extends User

Flow of Events:

1. The actor logs in / register to the application.
2. In the application page, the actor clicks the plus (+) button on the left side of the page, above the previously created reports.
3. The actor fills the necessary information displayed in the opening form (i.e. report's name, patient's name, date)
4. The actor clicks the "Create" button.

Entry Condition: The actor is on the application page.

Exit Condition: The actor clicks the "Create" button or anywhere on the screen except the opening form.

3.5.1.2 Delete Report

Use Case Name: Delete Report

Participating Actors: User and all actors that extends User

Flow of Events:

1. The actor logs in / register to the application.
2. In the application page, the actor selects the report that is wanted to be deleted from the left side of the screen.
    a. The actor can search the report by scrolling and passing next pages.
    b. The actor can search the report by filtering.
    c. The actor can search the report by using the search bar above.

3. The actor clicks the delete button above the opening report.
4. The actor confirms that he/she wants to delete the report from the opening pop-up.

Entry Condition: The actor is on the application page.

Exit Condition: The actor clicks the “Delete” button and then confirms the action.

3.5.1.3 Create Report Template

Use Case Name: Create Report Template

Participating Actors: User and all actors that extends User

Flow of Events:

1. The actor logs in / register to the application.
2. In the application page, the actor clicks the plus (+) button on the top-right section of the page.
3. The actor creates and adds fields that he/she wants to add to the template by clicking “add field” button and typing the field’s name.
4. The actor clicks the “Create” button.

Entry Condition: The actor is on the application page.

Exit Condition: The actor clicks the “Create” button or anywhere on the screen except the opening form.

3.5.1.4 Update Report Template (User)

Use Case Name: Update Report Template

Participating Actors: User and all actors that extends User

Flow of Events:

1. The actor logs in / register to the application.
2. On the application page, the actor navigates to the "Templates" section on the sidebar.
3. The actor selects the specific report template they want to update or delete from the displayed list of templates.
4. The actor selects the specific report template they want to update from the displayed list of templates.
5. The actor clicks on the "Edit" button next to the template name.
6. Once the modifications are complete, the actor clicks the “Update” button.

Entry Condition: The actor is on the application page.

Exit Condition: The actor clicks the “Update” button or anywhere on the screen except the opening form.

3.5.1.5 Send Invitation For the Group (Group Admin)

Use Case Name: Send Invitation For the Group

Participating Actors: Group admin and all actors that extends group admin

Flow of Events:

1. The actor logs in / register to the application.
2. On the application page, the actor opens the "My Groups" section.
3. The actor selects the specific group they want to invite members to.
4. The actor clicks on the “Invite” button.
5. A form or pop-up appears, requesting the email address of the individual(s) to be invited.
6. The actor clicks the “Send Invitation” button

Entry Condition: The actor is logged into the application and has navigated to the group management page.

Exit Condition: The invitation(s) have been successfully sent or the actor clicks anywhere on the screen except the opening form.

3.5.1.6 Get Transcript from Voice Recording

Use Case Name: Get Transcript from Voice Recording

Participating Actors: User and all actors that extends User

Flow of Events:

1. The actor logs in / register to the application
2. The actor opens a report
3. The actor clicks on the microphone icon on the report page.
4. The system displays a red recording icon to show that recording is in progress.
5. The actor speaks into their device’s microphone, providing the content they want to transcribe.
6. When finished, the actor clicks the “Stop Recording” button.
7. Once the transcription is complete, the system displays the generated transcript to the actor.

Exit Condition: The transcript text is successfully generated.

3.5.2 Use-Case Model

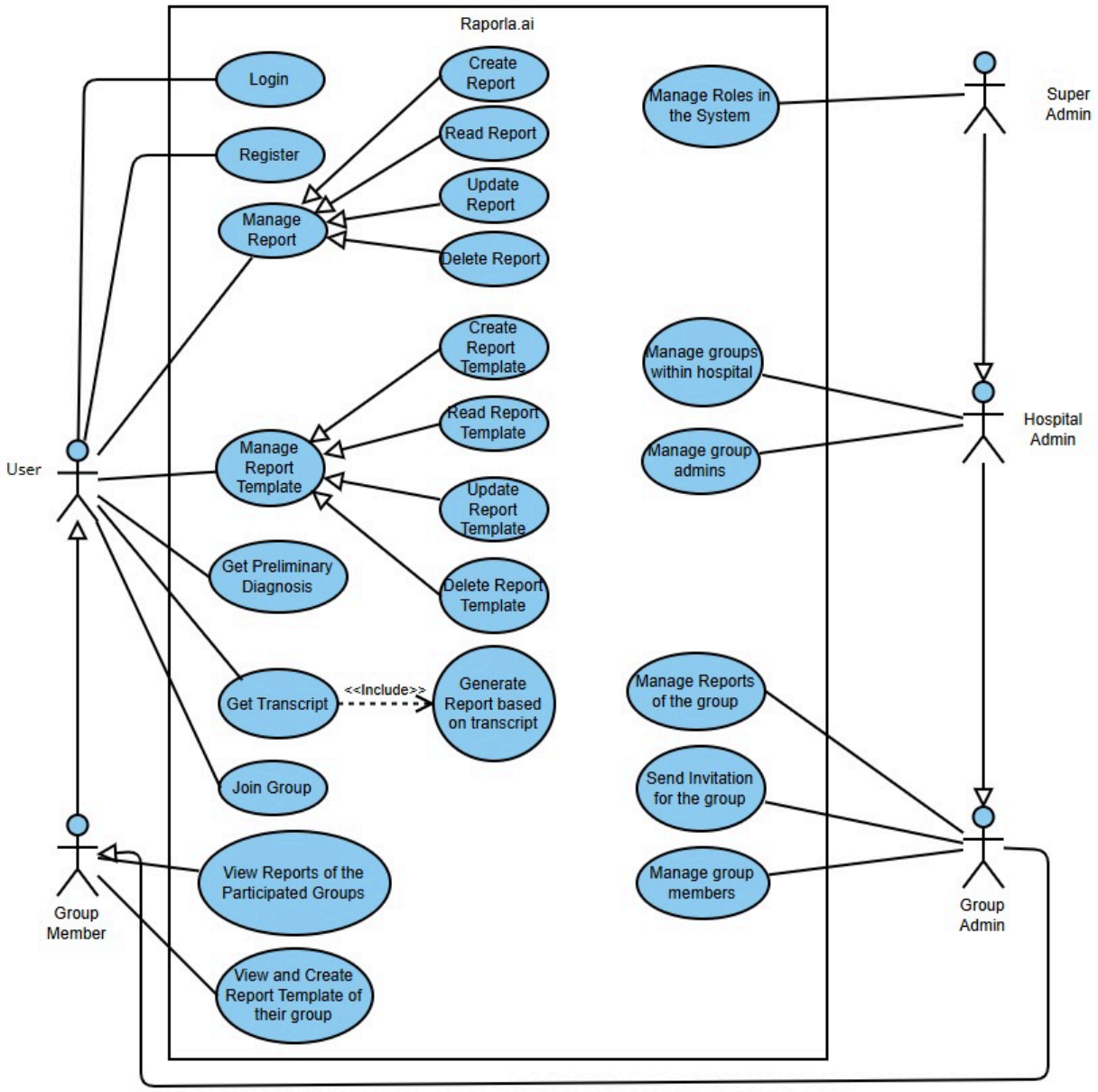

Fig. 1: Use Case Model

3.5.3 Object and Class Model

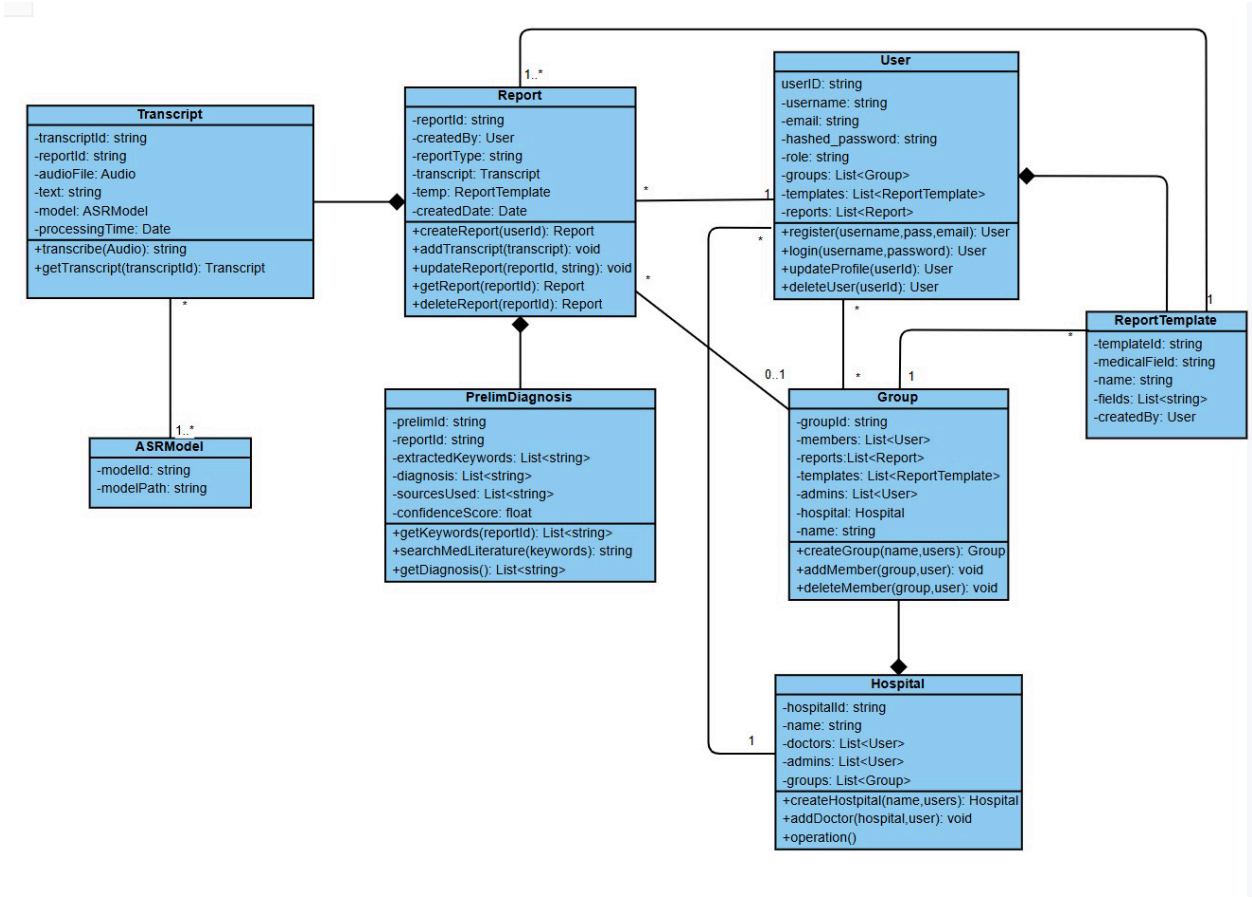

Fig. 2: Object and Class Model

3.5.4 Dynamic Models

3.5.4.1 Sequence Diagrams

3.5.4.1.1 User Authentication

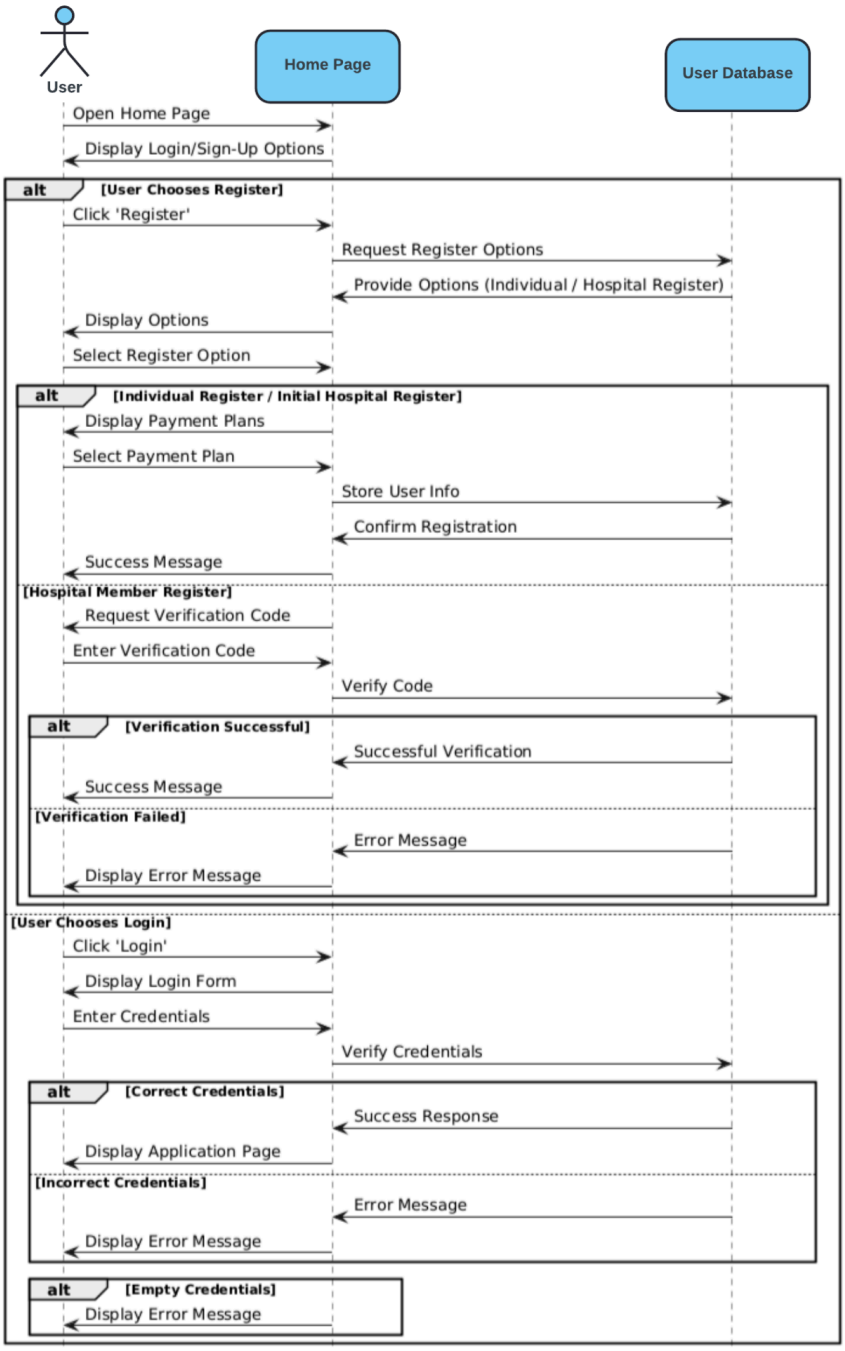

Fig. 3: User Authentication Sequence Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. The user opens the home page.
   2.1. The system displays the login and sign-up options.
3. The user chooses to register.
   3.1. The user clicks on the "Register" option.
   3.2. The system requests registration options from the user database.
   3.3. The user database provides the available options:
      3.3.1 Individual Register/Initial Hospital Register.
      3.3.2 Hospital Member Register.
   3.4. The system displays these options to the user.
4. If the user selects Individual Register/Initial Hospital Register:
   4.1. The system displays the payment plan options.
   4.2. The user selects a payment plan.
   4.3. The system stores the user's registration details in the user database.
   4.4. The user database confirms the registration.
   4.5. The system displays a success message to the user.
5. If the user selects Hospital Member Register:
   5.1. The system requests the user to enter a verification code.
   5.2. The user enters the verification code.
   5.3. The system sends the verification code to the user database.
   5.4. The user database verifies the code.
      5.4.1. If the code is valid:
         5.4.1.1 The user database confirms successful verification.
         5.4.1.2 The system displays a success message to the user.
      5.4.2. If the code is invalid:
         5.4.2.1 The user database sends an error message.
         5.4.2.2 The system displays an error message to the user.
6. The user chooses to log in.
   6.1. The user clicks on the "Login" option.
   6.2. The system displays the login form.
   6.3. The user enters their credentials (username/email and password).
   6.4. The system sends the credentials to the user database for verification.
      6.4.1. If the credentials are correct:
         6.4.1.1 The user database confirms success.
         6.4.1.2 The system displays the application page.
      6.4.2. If the credentials are incorrect:
         6.4.2.1 The user database returns an error.
         6.4.2.2 The system displays an error message to the user.

6.4.3. If the credentials are empty:

6.4.3.1 The system displays an error message to the user.

3.5.4.1.2 Speech-To-Text

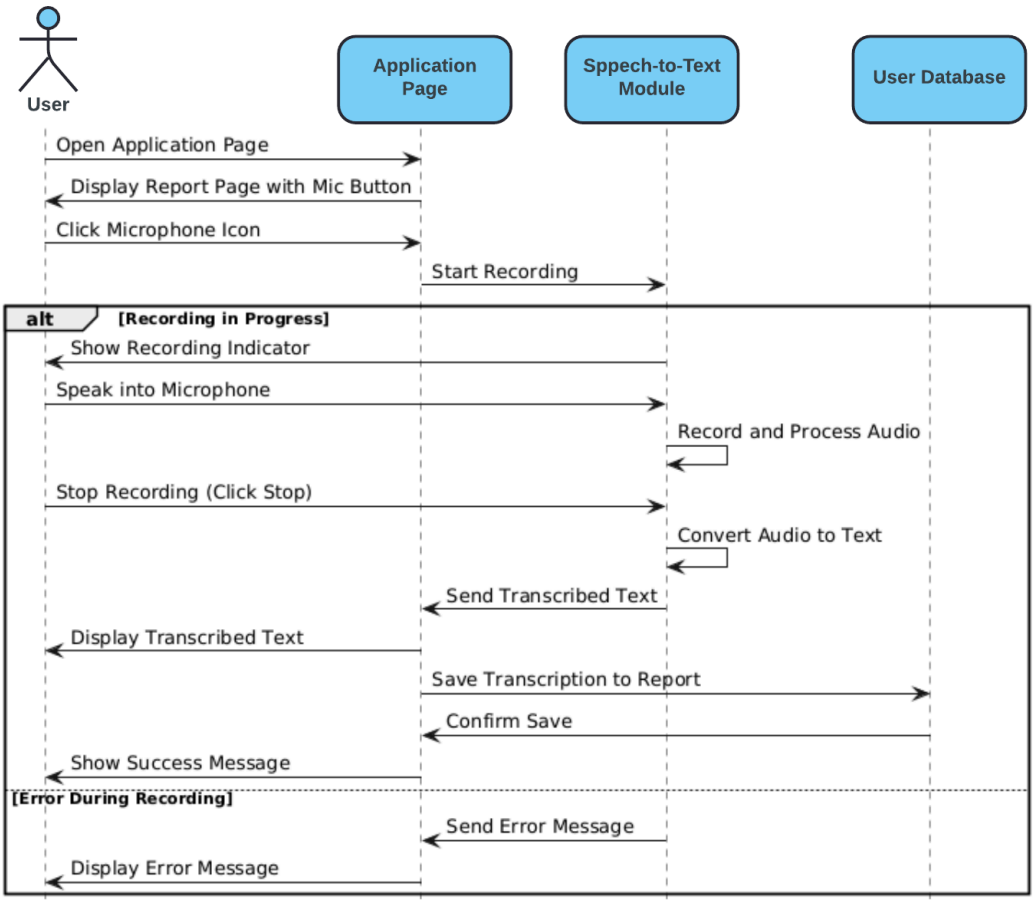

Fig. 4: Speech-To-Text Sequence Diagram

Flow of Events:

1. The user opens the application page.
   1.1. The system displays the report section within the application page with a microphone button.
2. The user clicks the microphone icon.
   2.1. The system starts the recording process using the Speech-to-Text module.
3. Recording is in progress:
   3.1. The system shows a recording indicator to inform the user.
   3.2. The user speaks into the microphone, providing the audio input.
   3.3. The system records and processes the audio input.

4. The user stops the recording:
   4.1. The user clicks the "Stop" button to finish the recording process.
   4.2. The system converts the recorded audio to text using the Speech-to-Text module.
5. The system processes the transcription:
   5.1. The system sends the transcribed text to the application page.
   5.2. The system displays the transcribed text to the user.
   5.3. The system saves the transcribed text into the user's report in the database.
   5.4. The database confirms that the transcription is saved successfully.
   5.5. The system shows a success message to the user.
6. Error during recording (alternative flow):
   6.1. If an error occurs during recording, the Speech-to-Text module sends an error message.
   6.2. The system displays the error message to the user.

3.5.4.1.3 Report Generation

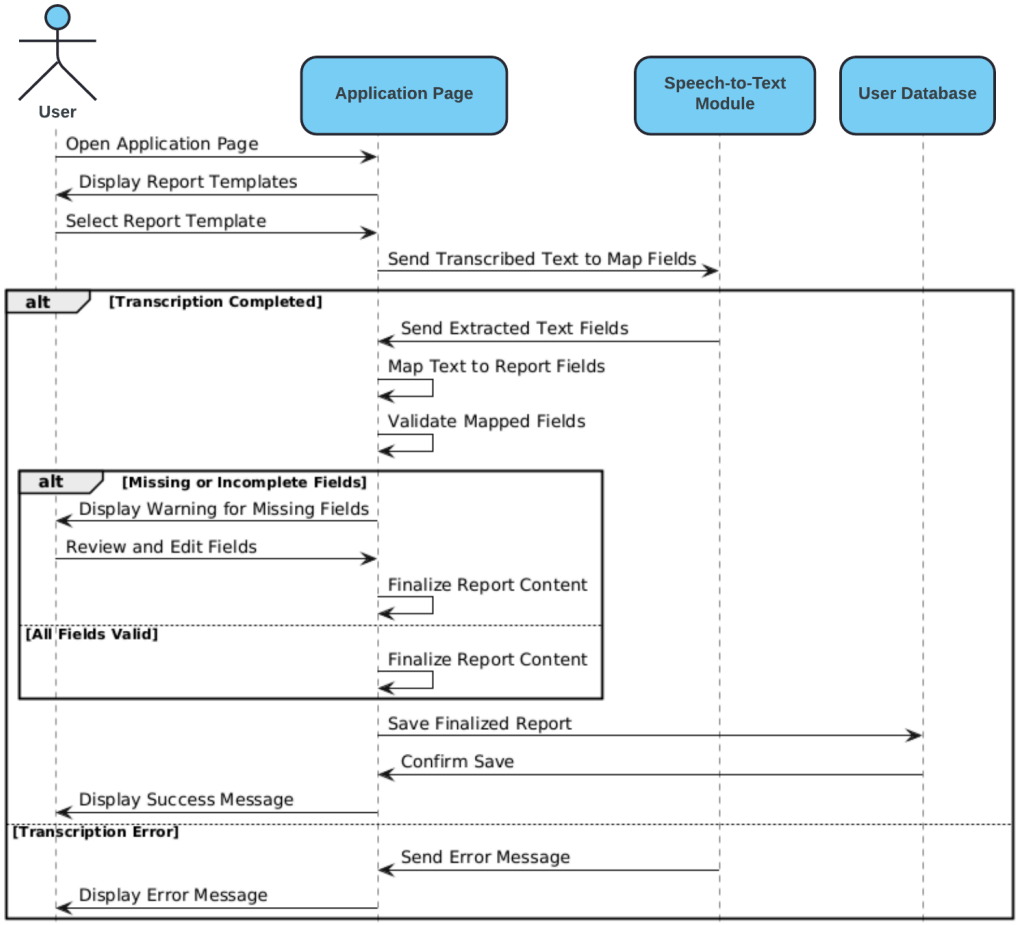

Fig. 5: Report Generation Sequence Diagram

Flow of Events:

1. The user opens the report page within the application page.
   1.1 The system displays the available report templates.
2. The user selects a report template.
   2.1 The system sends the transcribed text to map fields in the selected template.
3. The transcription is completed.
   3.1 The system sends the extracted text fields to the application page.
   3.2 The system maps the transcribed text to the corresponding fields in the selected template.
   3.3 The system validates the mapped fields to check for completeness.
4. If there are missing or incomplete fields:
   4.1 The system displays a warning for missing fields.
   4.2 The user reviews and edits the incomplete fields.
   4.3 The system finalizes the report content.
5. If all fields are valid:
   5.1 The system finalizes the report content without requiring edits.
6. The finalized report is saved.
   6.1 The system saves the report to the user database.
   6.2 The user database confirms the save operation.
   6.3 The system displays a success message to the user.
7. If a transcription error occurs:
   7.1 The system sends an error message to the application page.
   7.2 The system displays the error message to the user.

3.5.4.1.4 Preliminary Diagnosis Suggestion

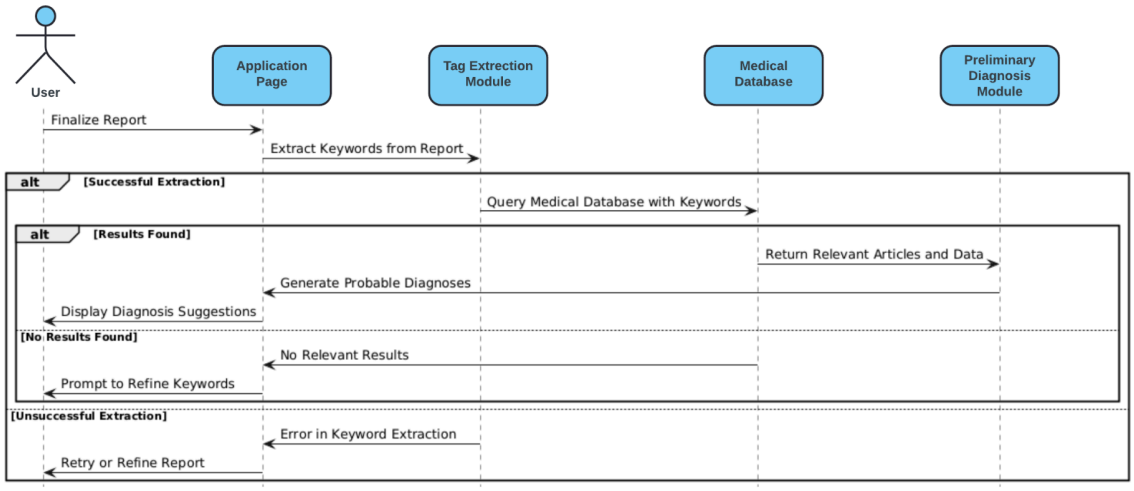

Fig. 6: Preliminary Diagnosis Suggestion Sequence Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. The user finalizes the report to the point where it wants to see preliminary diagnosis suggestions.
   2.1 The system transitions to the keyword extraction state.
3. The system extracts keywords from the finalized report.
   3.1 If the extraction is unsuccessful:
      3.1.1 The system sends an error message to the user.
      3.1.2 The user can retry the extraction or refine the report content.
4. The system queries a medical database (preferably PubMed) using the extracted keywords.
   4.1 If no results are found:
      4.1.1 The system prompts the user to refine the keywords.
      4.1.2 The system retries the query with alternative terms.
   4.2 If results are found:
      4.2.1 The system retrieves relevant articles and medical information.
5. The system generates probable diagnoses based on the retrieved results.
   5.1 The system processes the retrieved data.
   5.2 The system identifies possible diagnoses.
6. The system displays the diagnosis suggestions to the user.
   6.1 The process is completed successfully.

3.5.4.2 State Diagrams

3.5.4.2.1 User Authentication

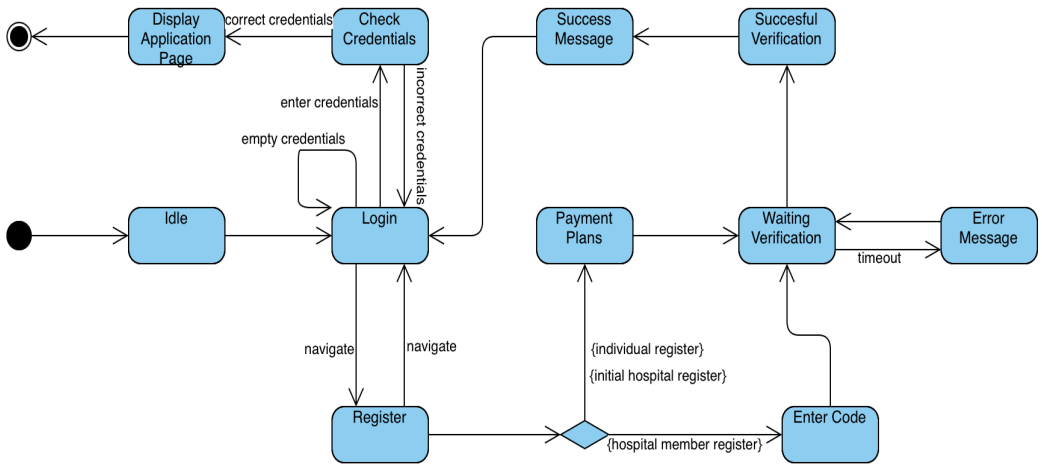

Fig. 7: User Authentication State Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. The user tries to log in / register to the application.
3. If they don't have an account continue the register process
   3.1. The user selects whether to continue as an individual register/initial hospital register or hospital member register.
   3.2. If chooses individual register/initial hospital register, select a payment plan and register.
   3.3. If chooses hospital member register, enter the code given and register.
4. If they have an account, continue the login process.
   4.1. If they leave the fields empty, return to the initial state.
   4.2. If they enter incorrect credentials, return to the initial state.
   4.3. If they enter correct credentials, continue.
5. Then display the application page.

3.5.4.2.2 Speech-To-Text

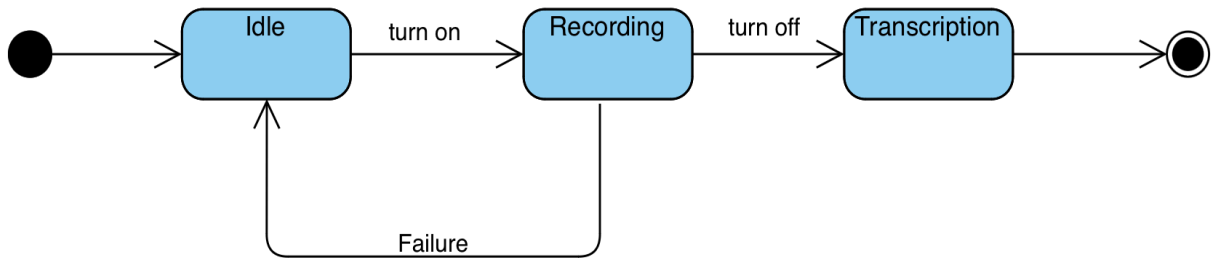

Fig. 8: Speech-To-Text State Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. The user initiates the recording process by pressing the "Kayıt al" button.
3. The system records the audio input provided by the user.
   3.1. If an error occurs, the system goes back to idle state with an error message
4. The user stops the recording by pressing the "stop" button or when the recording reaches a predefined time limit.
5. The system processes the recorded audio and converts it into text using the speech-to-text module.

3.5.4.2.3 Report Generation

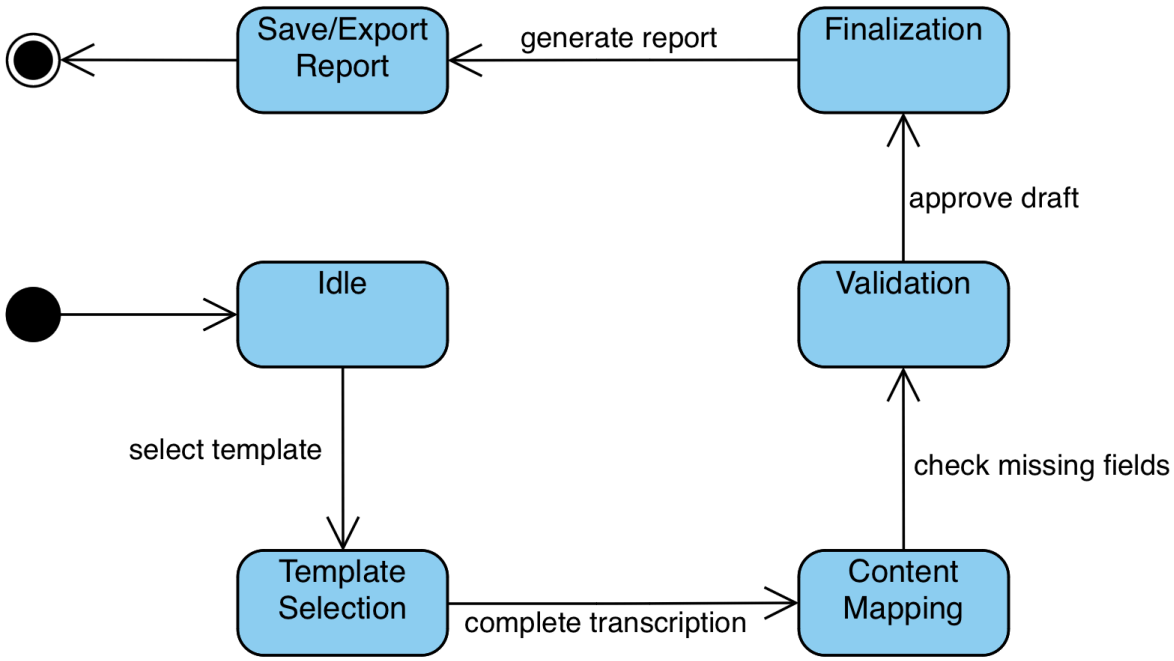

Fig. 9: Report Generation State Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. The user selects a predefined report template.
3. After transcription is completed, the system maps the extracted text to the corresponding fields in the selected template.
4. The system validates the content by checking for missing or incomplete fields in the mapped report.
5. The user reviews the draft report and approves it for finalization.
6. The finalized report is generated and saved. The user can export the file.

3.5.4.2.4 Preliminary Diagnosis Suggestion

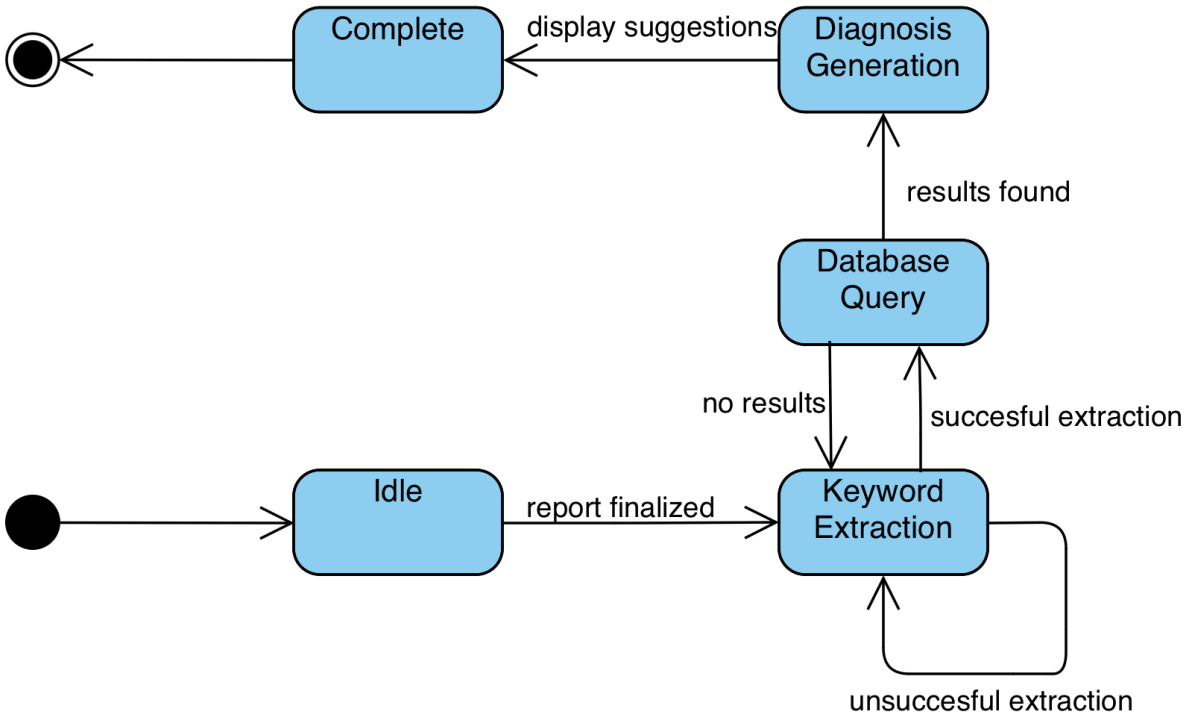

Fig. 10: Preliminary Diagnosis Suggestion State Diagram

Flow of Events:

1. The system is in an idle state, waiting for an action.
2. Once a report is finalized, the system transitions to the "Keyword Extraction" state.
3. The system extracts keywords from the finalized report to identify relevant terms for diagnosis suggestions.
   3.1. If it is an unsuccessful extraction, retry the process.
4. The system queries a medical database (e.g., PubMed) using the extracted keywords to find relevant information for diagnosis.
   4.1. If no results are found, the system prompts the user to refine keywords or retries the query with alternative terms.
5. The system uses the database results to generate probable diagnoses and suggestions.
6. Then display the suggestions and complete the process.

3.5.4.3 Activity Diagrams

3.5.4.3.1 Login

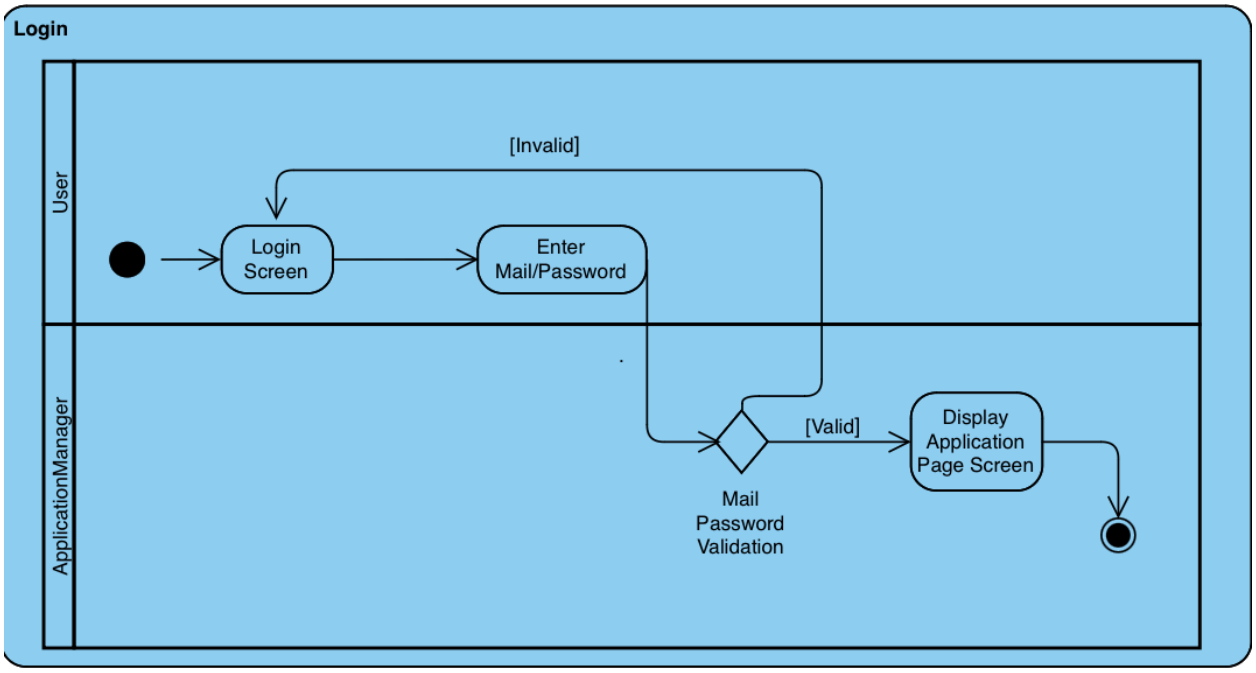

Fig. 11: Login Activity Diagram

3.5.4.3.2 Signup

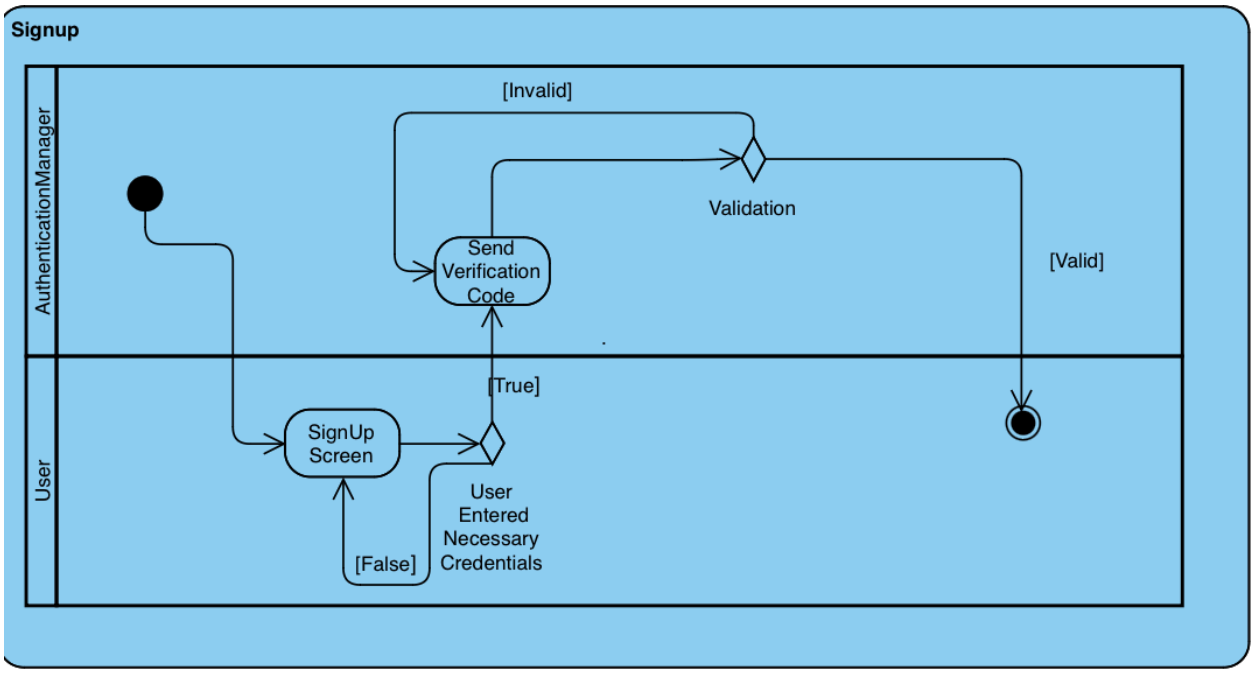

Fig. 12: Signup Activity Diagram

3.5.4.3.3 Report Management

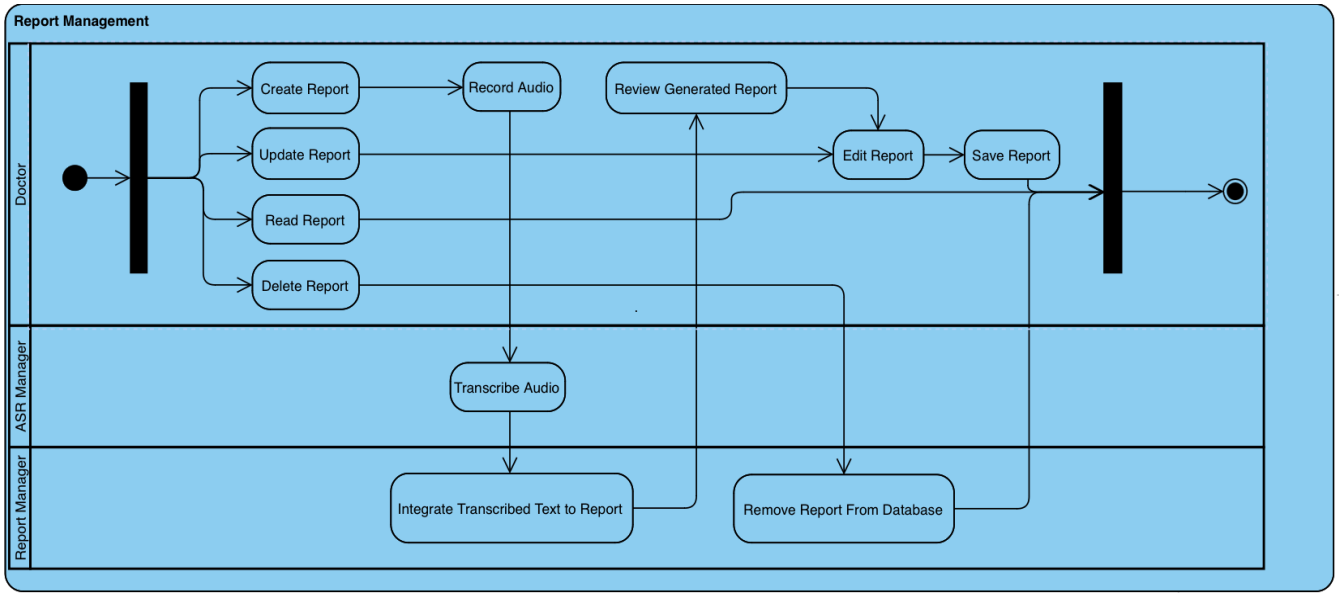

Fig. 13: Report Management Activity Diagram

3.5.4.3.1 Preliminary Diagnosis Suggestion

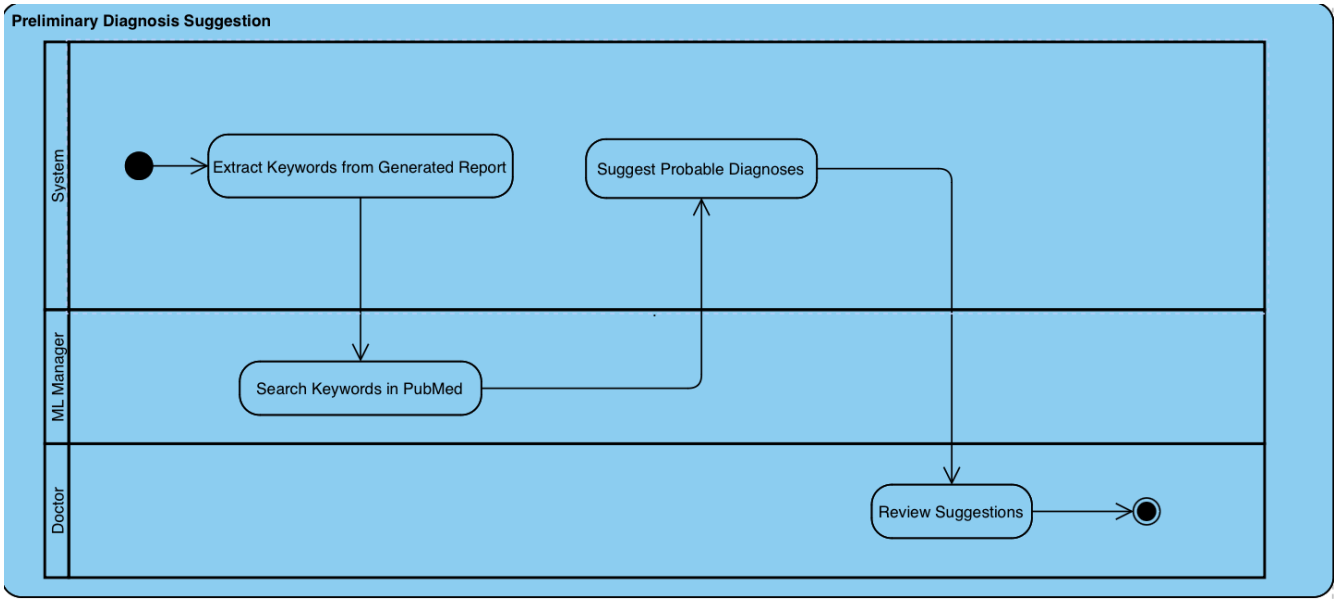

Fig. 14: Preliminary Diagnosis Suggestion Activity Diagram

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

3.5.5.1 Home Page

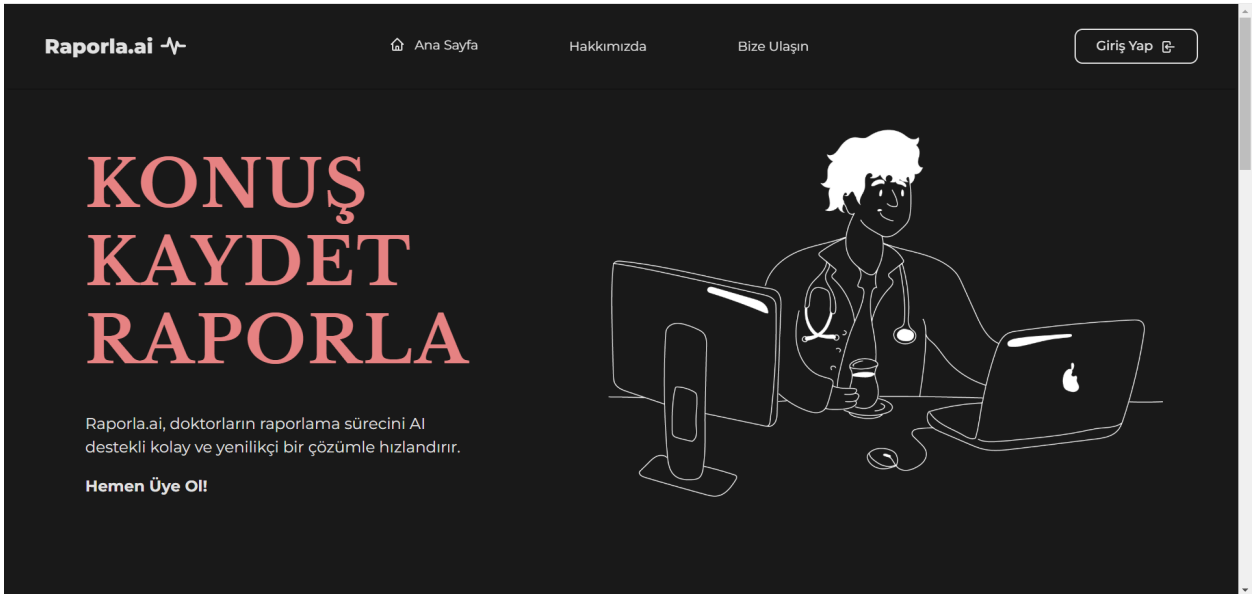

Fig. 15: Home Page Interface

3.5.5.2 Login

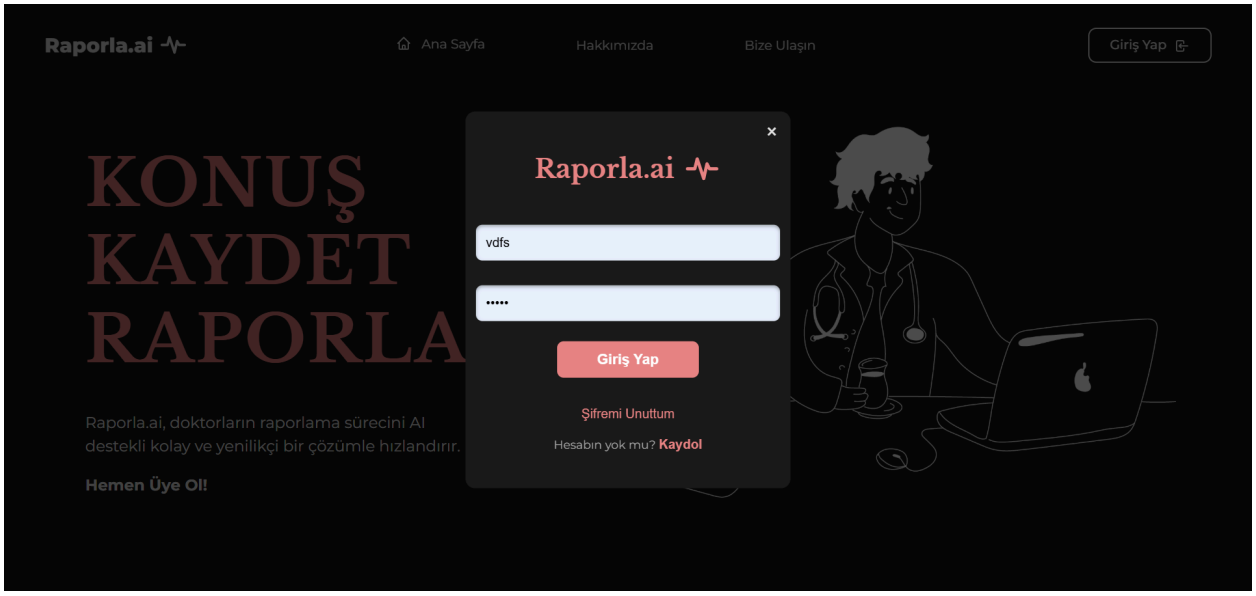

Fig. 16: Login Interface

3.5.5.3 Main Application Page (Initial)

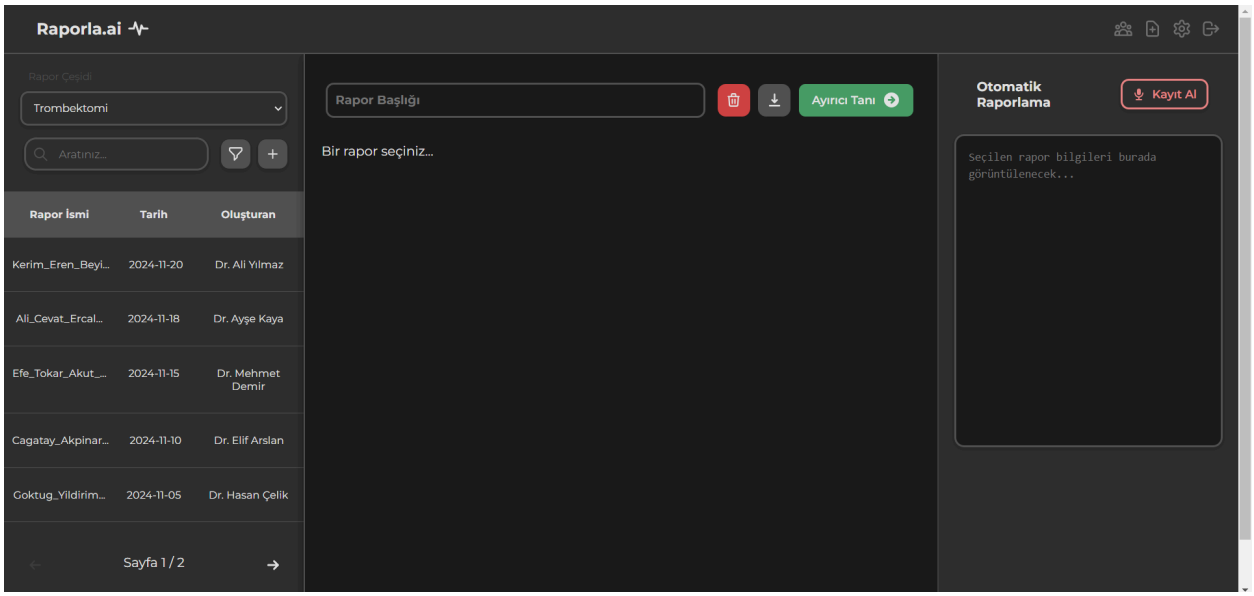

Fig. 17: Initial Application Page Interface

3.5.5.4 Main Application Page (Report Editing)

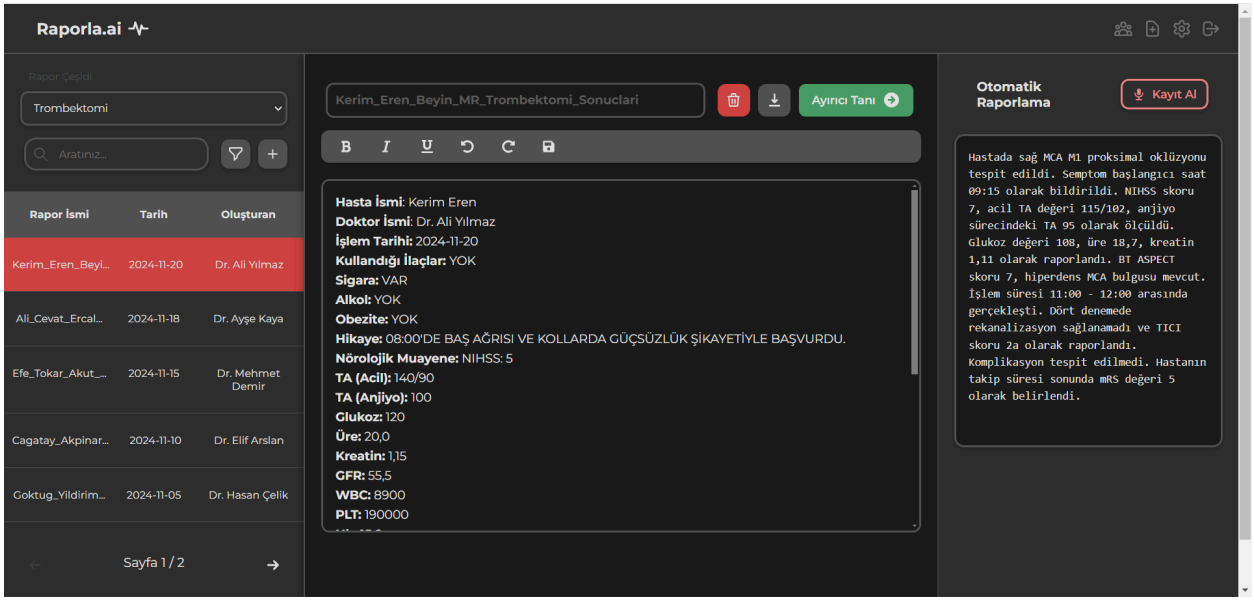

Fig. 18: Report Editing Interface

3.5.5.5 Main Application Page (Preliminary Diagnosis Recommendations)

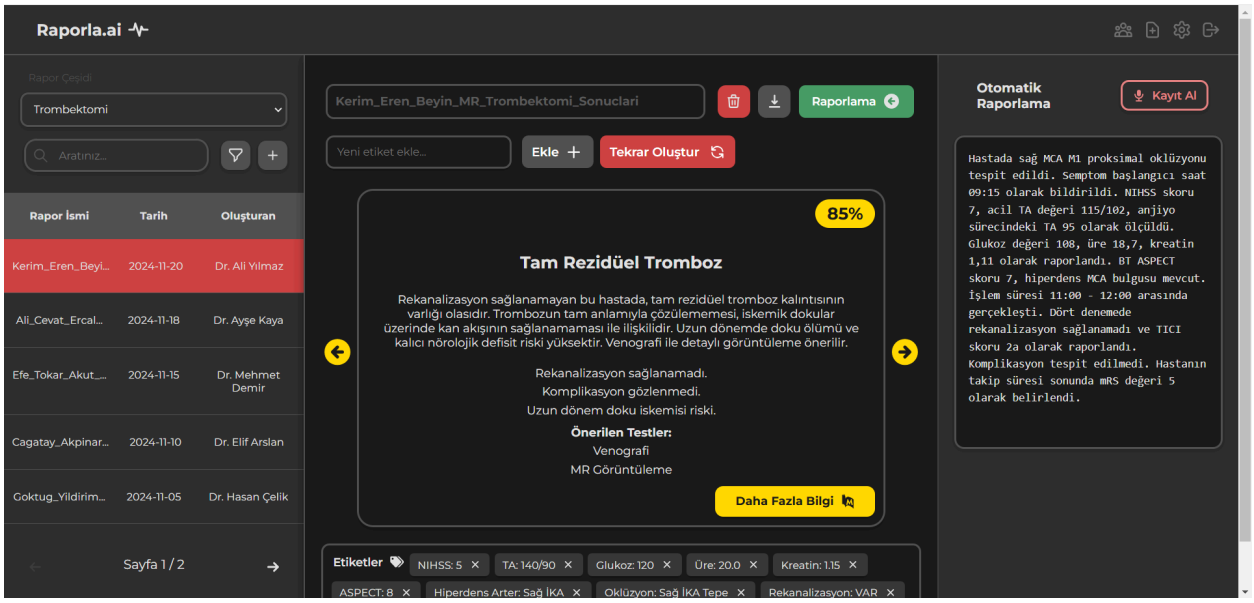

Fig. 19: Preliminary Diagnosis Suggestion Interface

3.5.5.6 Main Application Page (Create New Report)

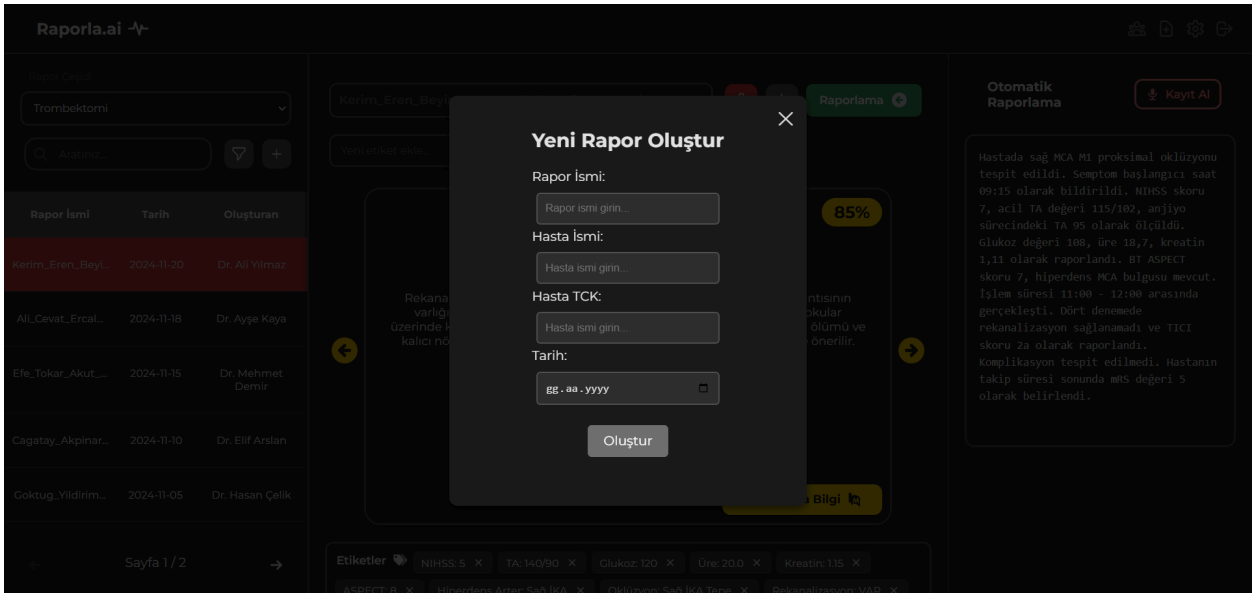

Fig. 20: Create New Report Interface

3.5.5.6 Logout

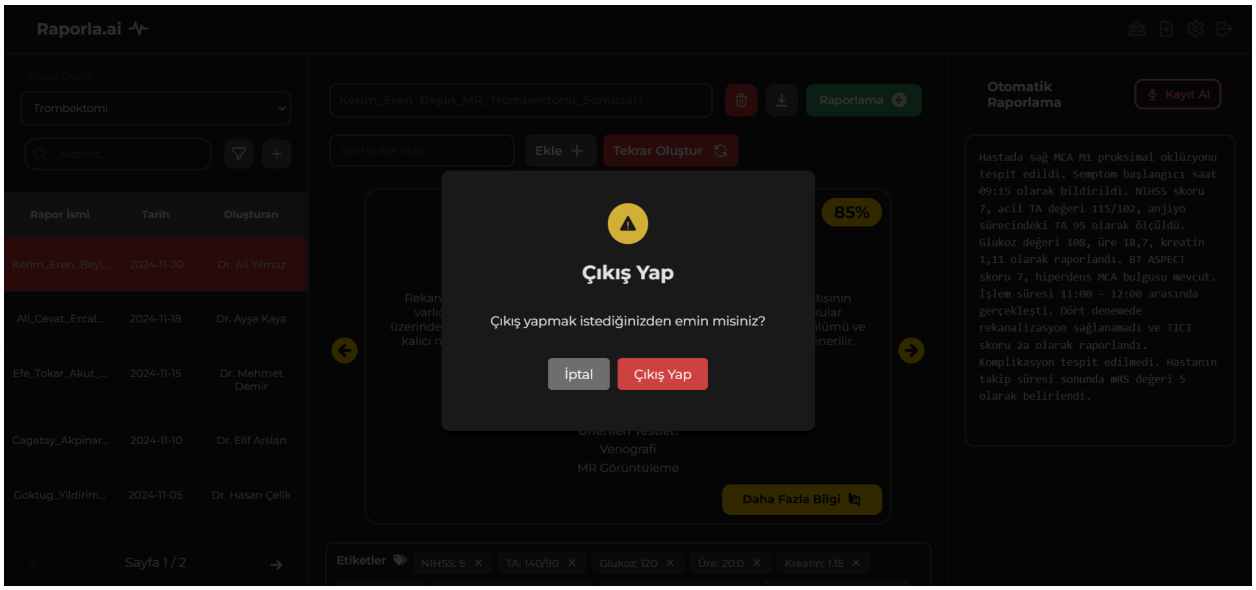

Fig. 21: Logout Interface

4. Other Analysis Elements

This section examines various external factors that may influence the design, implementation, and adoption of Raporla.ai. Given that this system aims to streamline medical reporting and assist Turkish doctors, understanding the broader context ensures that the solution is fit, ethical, and sustainable over time.

4.1 Consideration of Various Factors in Engineering Design

In developing Raporla.ai, several non-technical factors come into play, including public health, safety, welfare, global, cultural, social, environmental, and economic elements. By understanding these considerations early, the system can be optimized for social good, regulatory compliance, user acceptance, and long-term viability.

4.1.1 Public Health Considerations

Raporla.ai's primary goal is to improve the medical reporting process, potentially reducing reporting errors and allowing doctors to dedicate more time to patient care. More accurate and standardized reports contribute to better patient outcomes, while faster documentation may help doctors manage their workloads efficiently, improving overall public health standards. Ensuring patient data privacy and compliance with regulations like KVKK also indirectly supports public trust in health services.

4.1.2 Public Safety Considerations

Accurate and clearly documented medical reports are essential for safe patient care. By reducing transcription errors and ensuring consistency, Raporla.ai indirectly contributes to public safety. Misinterpretations or incomplete information in reports can lead to delays or mistakes in treatment. Thus, implementing strong error-checking and standardization promotes safer healthcare decisions.

4.1.3 Public Welfare Considerations

Although Raporla.ai does not directly influence public welfare policies, it can improve healthcare efficiency, making it easier and faster for patients to receive the correct diagnosis and treatment. This efficiency can positively impact the broader community by reducing wait times, improving patient satisfaction, and raising the overall quality of healthcare services.

4.1.4 Global Considerations

While the system initially focuses on Turkish healthcare contexts and language, it can later be adapted for global use. Global expansion would require addressing diverse medical standards, languages, and regulatory requirements. Ensuring multilingual support, compliance with various health regulations, and integrating internationally recognized medical datasets (e.g., PubMed) are considerations for future scalability.

4.1.5 Cultural Considerations

Medical practices and reporting expectations vary by region. As Raporla.ai specializes in Turkish medical literature, it respects local medical terminology, report formats, and communication styles. If adapted elsewhere, cultural factors such as different naming conventions, privacy attitudes, or doctor-patient communication norms would need to be considered for broader acceptance and utility.

4.1.6 Social Considerations

By easing the documentation burden, Raporla.ai may improve the work-life balance of doctors, potentially reducing burnout. This can enhance doctor-patient relationships and the overall healthcare environment. Moreover, standardizing reports can improve cooperation and understanding between medical professionals, leading to better teamwork and coordination in patient care.

4.1.7 Environmental Considerations

Digitizing and automating reporting can reduce the reliance on paper-based systems, indirectly lowering paper consumption and waste. Although the use of cloud-based ML models involves energy consumption, optimizing model size and computation times can help minimize the

system's environmental footprint. Over time, balancing performance with energy efficiency will remain an important consideration.

4.1.8 Economic Considerations

Efficient medical reporting can potentially reduce administrative costs by minimizing manual transcription and editing time. Hospitals and clinics may find the system economically beneficial as it streamlines documentation, speeds up patient turnover, and reduces error-related costs. Nonetheless, the initial investment in cloud infrastructure, machine learning development, and ongoing maintenance must be managed effectively to ensure affordability and accessibility for a wide range of healthcare institutions.

TABLE 1

THE EVALUATION OF VARIOUS FACTORS' EFFECTS MENTIONED IN SECTION 4.1

Factor	Effect Weight (out of 10)	Effect to the Application
Public Health Considerations	8	Improving report accuracy and standardization can enhance patient outcomes, reduce documentation errors, and build trust in healthcare services.
Public Safety Considerations	5	Clearer, more consistent reports can lower the risk of medical errors, indirectly increasing patient safety and promoting responsible healthcare practices.
Public Welfare Considerations	4	Efficient documentation may speed access to proper treatments, indirectly supporting overall community health and well-being through streamlined healthcare.
Global Considerations	7	Adapting to international standards, languages, and regulations facilitates broader adoption, requiring flexibility in language models and compliance frameworks.
Cultural Considerations	6	Variations in medical terminology and reporting expectations across regions necessitate customizable templates and NLP models respectful of cultural nuances.
Social Considerations	5	Reduced administrative burdens can alleviate physician stress, improving doctor-patient interactions and possibly fostering better healthcare relationships.

Environmental Considerations	3	Digital documentation reduces paper consumption, but computing resources incur energy costs, prompting optimization for efficiency and sustainability.
Economic Considerations	9	Balancing initial implementation costs against potential long-term savings in time and effort may influence stakeholder adoption and funding opportunities.

4.2 Risks and Alternatives

In this section, we will address the potential risks associated with the development and deployment of Raporla.ai and propose alternative solutions (Plan B) to mitigate these risks effectively.

4.2.1 Time Management

The development of Raporla.ai involves building a speech-to-text system specialized in Turkish medical literature, implementing standardized templates, and integrating a preliminary diagnosis suggestion feature. Given the limited time frame and the complexity of these tasks, there is a significant risk of not completing all functionalities within the semester. To address this, we will prioritize core functionalities, such as the transcription system and report generation features, while postponing the preliminary diagnosis system to a future phase. Adopting agile development practices with short iterative sprints will ensure continuous progress and provide flexibility to adjust priorities as needed.

4.2.2 Implementation

The implementation of Raporla.ai presents two main challenges: developing a highly accurate speech-to-text model tailored to Turkish medical terminology and integrating external medical datasets for the preliminary diagnosis system. Training and fine-tuning these models can be complex, especially given the team’s limited experience and the project’s computational constraints. To mitigate these risks, we will leverage pre-trained models, such as Whisper or Turkish-specific language models, fine-tuning only essential components. Additionally, if integrating medical datasets proves challenging, we will utilize static datasets or preprocessed medical terms as an interim solution. Conducting small-scale experiments and gradually scaling up will also help manage these risks effectively.

4.2.3 Work Distribution

The success of the project relies on efficient task distribution among team members. Unequal workload allocation or a lack of familiarity with certain technologies could lead to delays and inefficiencies. To address this, tasks will be distributed based on team members' expertise while providing resources and collaborative opportunities for learning unfamiliar technologies. Critical

tasks will be assigned to pairs of team members to ensure progress and reduce dependencies on individual efforts. Regular bi-weekly check-ins will be conducted to monitor progress and reallocate tasks as necessary to avoid bottlenecks and ensure timely completion of deliverables.

TABLE 2

SUMMARY OF RISKS, LIKELIHOOD, EFFECTS, AND ALTERNATIVES

Risk Name	**Likelihood**	**Effect on Project**	**Plan B Summary**
Time Management	Medium	Failure to deliver all functionalities	Prioritize core features; adopt agile sprints; postpone non-critical components.
Implementation	Medium	Suboptimal performance or incomplete features	Use pre-trained models and static datasets; focus on incremental improvements.
Work Distribution	Low	Delays or inefficiencies	Distribute tasks based on expertise; conduct regular check-ins to ensure balance.

4.3 Project Plan

WP 1: Ethics and Compliance Management			
Start Date: 17.10.24		**End Date: 26.01.25**	
Team Leader:	Göktuğ Serdar Yıldırım	**Members:**	All Team Members
Objectives: Ensure the project adheres to ethical, legal, and data privacy requirements, including KVKK and GDPR. Complete necessary permissions and approvals for using medical data.			
Tasks: **Task 1.1:** Prepare Ethics Committee Application ● **Subtask 1.1.1:** Research necessary documentation required for ethics committee approval. ● **Subtask 1.1.2:** Draft application forms and protocols for medical data usage. ● **Subtask 1.1.3:** Collaborate with Eskişehir Osmangazi University for formal approvals.			

Task 1.2: Data Privacy and Security Compliance

- **Subtask 1.2.1:** Ensure KVKK and GDPR compliance in data collection, storage, and processing.
- **Subtask 1.2.2:** Write data anonymization protocols to protect patient privacy.
- **Subtask 1.2.3:** Develop a consent form template for obtaining user data permissions.

Task 1.3: Ethical Considerations in AI Development

Task 1.4: Legal Documentation

- **Subtask 1.4.1:** Prepare a legal compliance document detailing patient data policies.
- **Subtask 1.4.2:** Draft Terms of Service and Privacy Policy for application deployment.

WP 1: Frontend Development			
Start Date: 13.11.24		**End Date: 13.05.25**	
Team Leader:	Efe Tokar	**Members:**	Göktuğ Serdar Yıldırım
Objectives: Develop the user interface and user experience for smooth experience			

Tasks:
Task 1.1: UI/UX Research and Design

- **Subtask 1.1.1:** Research medical app designs and identify best practices.
- **Subtask 1.1.2:** Design wireframes for Doctor and Admin pages.
- **Subtask 1.1.3:** Finalize mockups with Figma

Task 1.2: Implementation

Task 1.3: Frontend-Backend Integration

- **Subtask 1.3.1:** Set up API calls for speech-to-text and database services.
- **Subtask 1.3.2:** Validate responses and error handling.

WP 3: Backend Development	
Start Date: 13.11.24	**End Date: 13.05.25**

Team Leader:	Kerim Eren	**Members:**	Ali Cevat Erçal, Ömer Amir Akdağ

Objectives:
Develop the backend infrastructure for report generation, API integration, and user management.

Tasks:
Task 3.1: User Authentication

Task 3.2: API Development for Reports and Templates

- **Subtask 3.2.1:** Develop endpoints for report creation and customization.
- **Subtask 3.2.2:** Implement validation for missing fields.

Task 3.3: Integration of Speech-to-Text Models

- **Subtask 3.3.1:** Connect speech-to-text processing pipeline to the backend.
- **Subtask 3.3.2:** Optimize real-time transcription API.

Task 2.4: Secure Data Management

- **Subtask 3.4.1:** Encrypt and store sensitive patient data securely.
- **Subtask 3.4.2:** Ensure compliance with KVKK/GDPR.

WP 4: Database Implementation			
Start Date: 13.11.24		**End Date: 26.01.25**	
Team Leader:	Ömer Amir Akdağ	**Members:**	Ali Cevat Erçal, Kerim Eren

Objectives:
Design and implement a database for storing user data, medical reports, and templates.

Tasks:
Task 4.1: Database Schema Design

- **Subtask 4.1.1:** Design ER diagrams for users, reports, and templates.
- **Subtask 4.1.2:** Normalize database tables for performance and efficiency.

Task 4.2: Database Setup and Integration

Task 4.3: Data Security Measures

- **Subtask 4.3.1:** Apply encryption techniques for patient data protection.
- **Subtask 4.3.2:** Ensure compliance with KVKK and GDPR.

WP 5: Machine Learning Model Development			
Start Date: 13.11.24		**End Date: 13.05.25**	
Team Leader:	Ali Cevat Erçal	**Members:**	Kerim Eren

Objectives:

Develop and integrate machine learning models for speech-to-text transcription and preliminary diagnosis suggestions.

Tasks:
Task 5.1: Speech-to-Text Model Development

Task 5.2: Preliminary Diagnosis Module

- **Subtask 5.2.1:** Develop a keyword extraction system for medical reports.
- **Subtask 5.2.2:** Integrate PubMed or other medical datasets (to be decided) to generate diagnosis suggestions.
- **Subtask 5.2.3:** Evaluate accuracy and refine algorithms to minimize bias.

Task 4.3: Model Deployment and Optimization

- **Subtask 5.3.1:** Optimize model performance for real-time processing.
- **Subtask 5.3.2:** Deploy models on cloud servers for backend integration.

WP 6: Beta Release and Testing with Doctors			
Start Date: 17.12.24		**End Date: 10.02.25**	
Team Leader:	Göktuğ Serdar Yıldırım	**Members:**	All Members

Objectives:

Test system functionality, gather usability feedback from doctors, and address identified

issues.

Tasks:
Task 6.1: System Testing

- **Subtask 6.1.1:** Conduct unit and integration testing for system components.
- **Subtask 6.1.2:** Validate the backend, frontend, and ML models.

Task 6.2: Usability Testing with Medical Professionals

- **Subtask 6.2.1:** Organize user testing sessions with doctors.
- **Subtask 6.2.2:** Gather and analyze feedback on report generation and diagnosis features.

Task 6.3: Bug Fixes and Refinements

- **Subtask 6.3.1:** Identify and resolve bugs based on test results.
- **Subtask 6.3.2:** Implement refinements suggested during feedback sessions.

WP 7: Project Demo			
Start Date: 23.12.24		**End Date: 23.12.24**	
Team Leader:	Ali Cevat Erçal	**Members:**	All Members

Objectives:

Prepare and present a detailed demo of the project to showcase its functionality.

Tasks:

Task 7.1: Develop Demo Materials

- **Subtask 7.1.1:** Prepare a functional demo highlighting key features.
- **Subtask 7.1.2:** Create video recordings and slides for project walkthrough.

Task 7.2: Rehearse and Finalize Demo

- **Subtask 7.2.1:** Conduct internal rehearsals for presentation flow.
- **Subtask 7.2.2:** Refine and finalize demo content.
- features.

Task 6.3: Bug Fixes and Refinements

- **Subtask 6.3.1:** Identify and resolve bugs based on test results.
- **Subtask 6.3.2:** Implement refinements suggested during feedback sessions.

WP 8: Project Final Presentation			
Start Date: 13.05.25		**End Date: 13.05.25**	
Team Leader:	Kerim Eren	**Members:**	All Members
Objectives: Present the completed project, summarizing project outcomes, achievements, and challenges.			
Tasks: **Task 8.1:** Final Presentation Preparation ● **Subtask 8.1.1:** Create comprehensive presentation slides. ● **Subtask 8.1.2:** Highlight project progress, features, challenges, and test results. **Task 8.2:** Conduct Final Presentation ● **Subtask 8.2.1:** Present project findings to faculty and stakeholders. ● **Subtask 8.2.2:** Address feedback and summarize project impact.			

4.4 Ensuring Proper Teamwork

During the development of our project, we need to keep track of both technical and non-technical works. Since this is a group project, we should ensure each group member is assigned to both technical and non-technical tasks. We should also distribute these tasks fairly, keep track of their status, and update them if necessary. Also, for shared documents like articles, websites, and sample projects/reports, we should keep them in a place to refer to them when we need them. In addition, every member should be able to track others' work, at least the status of their work. To this end, we use two main applications: Notion and Github.

- **Notion:** We use Notion to keep track of our tasks, share necessary documents, and store the summaries of our meetings and decisions taken. We group the tasks based on the sub-projects (backend, frontend, model, etc.) Everyone can see a task's assignee, due

date, status, and subtasks if it exists. Also, we have folders to store the necessary documents that we should refer to regularly. Finally, for each of our meetings, we write meeting summaries, notes, and decisions taken in that meeting and store them in Notion pages.

- **Github:** We use Github for version control. It helps us to develop our tasks without disrupting others' work. Also, by adding reviewers, we can avoid unintended personal mistakes by allowing teammates to double check our work. Also, we can keep track of each member's contribution to the project by viewing their commits.

4.5 Ethics and Professional Responsibilities

As in all project development processes, a structured and respectful environment will be provided due to the requirements of teamwork during the development process of our project Raporla.ai. For the sake of the project, all team members will complete their task appropriately in the designated time interval as a professional. Weekly meetings will be held in order to stay in touch with each other and maintain our professional manner. Since the data we will use will carry important personal information, minimizing this security vulnerability is vital for the ethical part of the project. Any third party resource used for our project will be cited properly.

4.6 Planning for New Knowledge and Learning Strategies

Reports will be generated using Natural Language Processing, model fine-tuning, and training a model with datasets we generate; therefore, team members must become familiar with AI and NLP. Other team members learn about these technologies through online resources like tutorials, videos, and online courses, while others are already familiar with them. While some members are already familiar with model fine-tuning, others will learn it through practical experience and online resources. For the backend, we also intend to use the FastAPI framework, but the majority of the team is unfamiliar with this technology. This framework will be learned through online tutorials and resources, as well as by looking at related FastAPI projects.

5. References

[1] “RadMate AI,” radmate.ai. https://www.radmate.ai/ (accessed Nov. 20, 2024).

[2] “Amazon Transcribe Medical,” aws.amazon.com. https://aws.amazon.com/transcribe/medical/ (accessed Nov. 20, 2024).

[3] “Rad AI,” radai.com. https://www.radai.com/ (accessed Nov. 20, 2024).

[4] A. Papworth, “Non Functional Requirements: Quick Guide for the business analyst in 2024,” BusinessAnalystMentor.com. https://businessanalystmentor.com/non-functional-requirements/ (accessed Nov. 21, 2024).