CS 492 - Senior Design Project II

Raporla.ai

Detailed Design Report

Kerim Eren - 22103759

Ömer Amir Akdağ - 22102918

Ali Cevat Erçal - 22103341

Göktuğ Serdar Yıldırım - 22103111

Efe Tokar - 22103299

Table of Contents

1. Introduction......3
- 1.1 Overview......3
- 1.2 Design goals......3
  - 1.2.1 Usability......3
  - 1.2.2 Reliability......4
  - 1.2.3 Performance......4
  - 1.2.4 Supportability......4
  - 1.2.5 Scalability......4
  - 1.2.6 Marketability......4
  - 1.2.7 Security......4

2. Current Software Architecture......5

3. Proposed software architecture......5
- 3.1 Overview......5
- 3.2 Subsystem decomposition......6
- 3.3 Access Control and Security......7
  - 3.3.1 User Authentication Flow......7
    - 3.3.1.1 User Registration (Signup)......7
    - 3.3.1.2 User Login (Authentication)......7
    - 3.3.1.3 Token-Based Access Control......7
  - 3.3.2 Role-Based Access Control......8

4. Subsystem services......8

5. Test Cases......11

6. Consideration of Various Factors in Engineering Design......29
  - 6.1 Public Health Considerations......29
  - 6.2 Public Safety Considerations......29
  - 6.3 Public Welfare Considerations......30
  - 6.4 Global Considerations......30
  - 6.5 Cultural Considerations......30
  - 6.6 Social Considerations......30
  - 6.7 Environmental Considerations......30
  - 6.8 Economic Considerations......30

7. Teamwork Details......32
- 7.1 Contributing and functioning effectively on the team......32
- 7.2 Helping creating a collaborative and inclusive environment......32
- 7.3 Taking lead role and sharing leadership on the team......33

8. References......35

1. Introduction

1.1 Overview

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.

1.2 Design goals

1.2.1 Usability

- The interface should be easy for common usage and require minimal training for doctors to use effectively.

- The application should support medical terminology and the Turkish language. [1].

1.2.2 Reliability

- The system needs to function without malfunctioning or crashing under all circumstances.
- High accuracy speech-to-text conversion should be maintained even when there is background noise or vocal fluctuation [1].

1.2.3 Performance

- The application should process audio files and generate reports to create an ideal experience.
- Diagnosis suggestions should be retrieved and displayed within feasible and reasonable seconds after report generation [1].

1.2.4 Supportability

- The application should be easy to update for incorporating new medical datasets and templates [1].

1.2.5 Scalability

- The application must handle increasing numbers of users and simultaneous audio uploads without significant performance decline.
- The application should support the addition of new features like integration with hospital management systems [1].

1.2.6 Marketability

- The system should offer a unique value proposition compared to existing solutions in the medical speech-to-text and report generation market.
- The application should be easily adaptable to other languages and medical systems in the future.

1.2.7 Security

- All data processing should comply with KVKK (Turkey's Data Protection Law) and GDPR regulations to ensure legal security.
- The authentication system should use secure login mechanisms, including multi-factor authentication for users.

2. Current Software Architecture

In the AI-powered medical reporting market, there are several competitors and alternatives, each leveraging different technologies to address variety of needs in the market.

Competitors and Alternatives

1. **Rad AI**

   Rad AI specializes in automating radiology reporting using machine learning (CNNs for image analysis) and NLP for report generation.t integrates with PACS/RIS systems and emphasizes explainable AI (XAI) for transparency. Nevertheless, it is limited to radiology and primarily supports English [2].

2. **RadMate AI**

   RadMate AI focuses on speech-to-text technology for radiology report generation. It uses ASR models optimized for medical terminology and cloud-based infrastructure for real-time processing. Like Rad AI, it lacks support for Turkish and other medical fields [3].

3. **Amazon Transcribe Medical**

   Amazon Transcribe Medical offers high-accuracy speech-to-text services for healthcare. It uses advanced ASR models and cloud infrastructure but is not specialized for specific medical fields or languages like Turkish [4].

Existing solutions are primarily optimized for English and focus on specific fields like radiology. They lack support for Turkish medical terminology, standardized reporting, and preliminary diagnosis suggestions based on medical literature. Raporla.ai addresses these gaps by combining advanced NLP, speech-to-text models, and medical datasets to provide a comprehensive, multi-specialty solution tailored to Turkish healthcare professionals.

3. Proposed software architecture

3.1 Overview

Raporla.ai is structured into four main, interconnected parts to provide a complete medical reporting solution. The Client Layer provides the user interface for doctors and administrators, allowing them to interact with the system. The Communication Layer acts as a secure messenger, handling all data exchange between the user interface and the processing core. The Backend Layer is the "brains" of the operation, responsible for all the processing, including speech-to-text conversion, report generation, user management, and AI functionalities. Finally, the Storage

Layer securely stores all user data, generated reports, and the AI models that power the system. These four parts work together seamlessly to provide the functionality of our project.

3.2 Subsystem decomposition

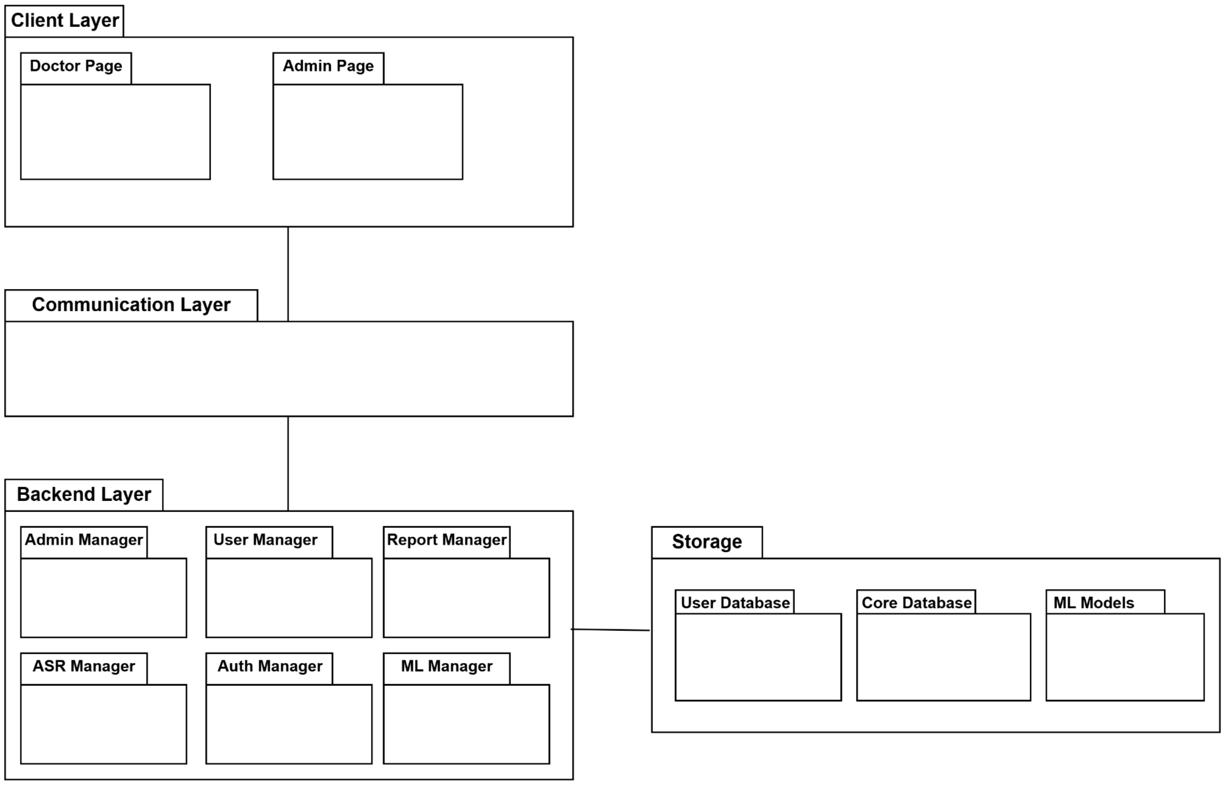

Figure 1: Subsystem Decomposition of our project

The **Client Layer** represents the user interface, tailored for both medical professionals and administrators. The Doctor Page provides a user-friendly environment for doctors to dictate reports via speech-to-text, review generated reports, and manage patient information, ensuring seamless integration into their clinical workflows. Conversely, the Admin Page empowers system administrators with tools to monitor user activity, manage roles and permissions, update system configurations, and access logs and analytics, enabling effective platform oversight.

The **Communication Layer** acts as a vital bridge between the client and backend systems. It manages data exchange, ensuring secure transmission for speech-to-text processing, report generation, and user management. This layer also facilitates API integrations, connecting client actions with backend services such as authentication, report generation, and machine learning tasks, thereby ensuring smooth and efficient data flow and service interaction.

At the core of the system lies the **Backend Layer**, the processing hub responsible for business logic and user request execution. This layer is modularized through several managers. The

Admin Manager oversees system-wide settings and configurations, maintaining system integrity and scalability. The User Manager handles user-related functionalities, including profile management, role assignments, and personalized system behavior. The Report Manager generates and manages medical reports, processing speech-to-text outputs and integrating patient data. The ASR Manager converts audio input into text using fine-tuned machine learning models, optimized for medical terminology. The Auth Manager ensures secure access through robust authentication and authorization processes. Finally, the ML Manager manages machine learning models, ensuring their efficient deployment and relevance.

The **Storage Layer** serves as the data management backbone, organizing and maintaining all essential data. The User Database stores user-specific information, while the Core Database acts as the central repository for system-critical data, including generated reports, system configurations, and operational logs. It also houses admin-specific data. The ML Models component contains trained machine learning models, facilitating version control and efficient deployment for AI functionalities such as Automatic Speech Recognition.

3.3 Access Control and Security

Our project uses a token-based authentication system implemented with FastAPI and JWT (JSON Web Tokens) to ensure secure access control.

3.3.1 User Authentication Flow

3.3.1.1 User Registration (Signup)

- During registration, passwords are securely hashed using bcrypt before being stored in the MongoDB database.
- The system prevents duplicate registrations by checking if an email is already registered.

3.3.1.2 User Login (Authentication)

- The system verifies the credentials against stored hashed passwords using bcrypt.
- If authentication is successful, the system generates a JWT access token and returns it to the client.
- This token serves as a secure session key for further API requests.

3.3.1.3 Token-Based Access Control

- Users must include the JWT token in the Authorization Header (as a Bearer Token) to access protected endpoints.
- If the token is invalid or expired, access is denied with a 401 Unauthorized response.

3.3.2 Role-Based Access Control

The system implements role-based access control to manage different levels of access:

- Standard users can only access their own reports.
- Doctors may have extended privileges to review additional patient data.
- Admins have full access to user management and system logs.

This ensures that only authorized users can access sensitive data, preventing unauthorized actions.

4. Subsystem services

4.1 Client Layer Services

- **Doctor Page Services:**
  - **Report Dictation and Speech-to-Text:** Provides an interface for doctors to dictate medical reports, leveraging the Communication Layer to send audio data to the Backend Layer's ASR Manager.
  - **Report Review and Editing:** Allows doctors to review the generated text, edit the report content, and apply formatting.
  - **Patient Information Management:** Enables doctors to view and manage patient data associated with the report.
  - **Report Template Management:** Provides functionalities to select and apply report templates. Supports integration with both default and custom templates.
  - **AI-Assisted Report Generation:** After generating voice recordings, users can fill the report with the use of AI.
- **Admin Page Services:**
  - **User Management:** Allows administrators to create, modify, and delete user accounts, roles, and permissions.
  - **Monitoring and Analytics:** Displays system performance metrics, user activity logs, and other relevant data for monitoring system health and usage.
  - **Access Control Management:** Allows administrators to define access control rules and policies to ensure data security and compliance.

4.2 Communication Layer Services

- **API Gateway:** Acts as a single entry point for all client requests, routing them to the appropriate Backend Layer service.
- **Authentication and Authorization:** Verifies user credentials and ensures that only authorized users can access specific resources.
- **Request Queuing:** Manages asynchronous requests, such as speech-to-text processing, to prevent overloading the Backend Layer.
- **Secure Data Transmission:** Encrypts data during transmission to protect sensitive information from unauthorized access.

4.3 Backend Layer Services

- **Admin Manager:**
    - **User Role Management:** Manages user roles and permissions, ensuring that users only have access to the resources they need.
    - **Logging and Auditing:** Records system events and user activity for auditing and troubleshooting purposes.
- **User Manager:**
    - **User Authentication:** Verifies user credentials and issues access tokens.
    - **User Profile Management:** Allows users to create, update, and delete their profiles.
    - **Role Assignment:** Assigns roles to users, granting them specific permissions.
- **Report Manager:**
    - **Report Generation:** Generates medical reports from speech-to-text output and patient data.
    - **Report Storage and Retrieval:** Stores and retrieves medical reports from the Storage Layer's Core Database.
    - **Report Formatting:** Applies formatting to medical reports, such as bold, italic, and underline.
    - **Template Integration:** Applies report templates to standardize report structure.
    - **PDF Generation:** Generates PDF versions of medical reports.

- **ASR Manager:**
    - **Speech-to-Text Conversion:** Converts audio input into text using fine-tuned machine learning models.
    - **Medical Terminology Processing:** Ensures accurate transcription of medical terms and abbreviations.
- **Auth Manager:**
    - **Authentication:** Provides secure authentication services for users.
    - **Authorization:** Enforces access control policies to protect sensitive data.
    - **Session Management:** Manages user sessions and access tokens.
- **ML Manager:**
    - **Model Deployment:** Deploys machine learning models for speech-to-text conversion, diagnosis suggestions, and other AI-powered features.
    - **Model Versioning:** Manages different versions of machine learning models.
    - **Paper Suggestion:** Provides related papers to the given report.

4.4 Storage Layer Services

- **User Database:**
    - **User Data Storage:** Stores user-specific information, such as usernames, passwords, and profile data.
    - **User Data Retrieval:** Retrieves user data for authentication, profile management, and other purposes.
- **Core Database:**
    - **Report Storage:** Stores generated medical reports, system configurations, and operational logs.
    - **Template Storage:** Stores default and custom report templates.
- **ML Models:**
    - **Model Storage:** Stores trained machine learning models.
    - **Model Version Control:** Manages different versions of machine learning models.

    - **Model Access:** Provides access to machine learning models for the Backend Layer's ML

5. Test Cases

Test Case ID: 01

Description: The user should be able to see the report list when the application loads.

Flow of Events:

1. Open the application.
2. Check if the report list container is visible.
3. Verify that the list items are displayed correctly.

Expected Result: The report list should be visible.

Priority: Critical

Test Case ID: 02

Description: The user should be able to select a report from the left panel.

Flow of Events:

1. Click on a report item in the left panel.
2. Verify that the editor container updates with the report content.

Expected Result: The selected report's content should be displayed.

Priority: Critical

Test Case ID: 03

Description: The user should be able to save a report.

Flow of Events:

1. Make a change in the editor container.
2. Click the "Save" button.
3. Check if a success message appears.

Expected Result: The report should be saved successfully, and a success message should appear.

Priority: Critical

Test Case ID: 04

Description: The user should be able to delete a report.

Flow of Events:

1. Select a report.
2. Click the "Delete" button.
3. Verify if a confirmation dialog appears.
4. Click the "Confirm Delete" button.
5. Ensure the report is removed from the list.

Expected Result: The report should be deleted and no longer appear in the list.

Priority: Critical

Test Case ID: 05

Description: The user should be able to download a report as a PDF.

Flow of Events:

1. Select a report.
2. Click the "Download PDF" button.
3. Check if the file download process starts.

Expected Result: The PDF should be successfully downloaded.

Priority: Major

Test Case ID: 06

Description: The user should be able to create a new report.

Flow of Events:

1. Click the "New Report" button.
2. Enter report details in the modal window.
3. Click the "Create" button.
4. Verify that the new report appears in the report list.

Expected Result: A new report should be created and appear in the list.

Priority: Critical

Test Case ID: 07

Description: The user should be able to start-stop audio recording and get a transcription.

Flow of Events:

1. Click the "Start Recording" button.
2. Click the "Stop Recording" button.
3. Verify that a success message appears after stopping and then a transcription according to that record appears.

Expected Result: The recording should start and stop successfully and the transcription should be displayed.

Priority: Major

Test Case ID: 08

Description: The user should be able to upload an audio file and get a transcription.

Flow of Events:

1. Click the "Select File" button and choose an audio file.
2. Click the "Upload" button.
3. Verify that the transcription appears.

Expected Result: The audio file should be successfully uploaded, and the transcription should be displayed.

Priority: Major

Test Case ID: 09

Description: The user should be able to apply bold formatting to text in the editor.

Flow of Events:

1. Select text in the editor.
2. Click the "Bold" button.

3. Verify that the selected text appears in bold.

Expected Result: The selected text should be bold.

Priority: Minor

Test Case ID: 10

Description: The user should be able to apply italic formatting to text in the editor.

Flow of Events:

1. Select text in the editor.
2. Click the "Italic" button.
3. Verify that the selected text appears in italic.

Expected Result: The selected text should be italicized.

Priority: Minor

Test Case ID: 11

Description: The user should be able to apply underline formatting to text in the editor.

Flow of Events:

1. Select text in the editor.
2. Click the "Underline" button.
3. Verify that the selected text appears underlined.

Expected Result: The selected text should be underlined.

Priority: Minor

Test Case ID: 12

Description: The user should be able to undo the last text action.

Flow of Events:

1. Type text in the editor.
2. Click the "Undo" button.
3. Verify that the last action is undone.

Expected Result: The last change should be reverted.

Priority: Minor

Test Case ID: 13

Description: The user should be able to redo the last undone action.

Flow of Events:

1. Type text in the editor.
2. Click the "Undo" button.
3. Click the "Redo" button.
4. Verify that the undone action is restored.

Expected Result: The undone action should be reapplied.

Priority: Minor

Test Case ID: 14

Description: The user should be able to switch between dark and light mode.

Flow of Events:

1. Click the "Toggle Theme" button.
2. Verify if the CSS class changes.

Expected Result: The theme should switch successfully.

Priority: Minor

Test Case ID: 15

Description: The user should see a "Not Saved" warning when changes are made.

Flow of Events:

1. Make changes in the editor.
2. Verify that the "Not Saved" warning appears.

Expected Result: The warning should appear when there are unsaved changes.

Priority: Minor

Test Case ID: 16

Description: The user should be able to filter reports using the dropdown menu.

Flow of Events:

1. Click on the category dropdown menu.
2. Select a category.
3. Verify that only reports matching the selected category are displayed.

Expected Result: The report list should be filtered correctly.

Priority: Major

Test Case ID: 17

Description: The user should be able to search for a report by name.

Flow of Events:

1. Enter a report name in the search bar.
2. Verify that only reports matching the search term appear.

Expected Result: Only matching reports should be displayed.

Priority: Critical

Test Case ID: 18

Description: The user should be able to navigate between report pages.

Flow of Events:

1. Click the "Next Page" button.
2. Verify that the next set of reports appears.
3. Click the "Previous Page" button.
4. Verify that the previous set of reports appears.

Expected Result: The user should be able to navigate between report pages.

Priority: Major

Test Case ID: 19

Description: The user should see an error when uploading an unsupported file type.

Flow of Events:

1. Click the "Select File" button.
2. Choose a non-audio file.
3. Click the "Upload" button.
4. Verify that an error message appears.

Expected Result: The system should reject unsupported file types.

Priority: Major

Test Case ID: 20

Description: The user should be able to generate a report using AI.

Flow of Events:

1. Select a report.
2. Click the "Generate Report" button.
3. Verify that the AI-generated content appears in the editor.

Expected Result: AI-generated content should be displayed in the editor.

Priority: Major

Test Case ID: 21

Description: The system should show a loading animation while AI is generating a report.

Flow of Events:

1. Click the "Generate Report" button.
2. Verify that a loading animation appears.
3. Verify that the animation disappears once the report is generated.

Expected Result: A loading animation should appear while AI is processing and disappear after completion.

Priority: Minor

Test Case ID: 22

Description: The user should be able to remove an uploaded audio file before transcription.

Flow of Events:

1. Click the "Select File" button and choose an audio file.
2. Verify that the file name appears in the UI.
3. Click the "Remove File" button.
4. Verify that the file is removed from the UI.

Expected Result: The selected file should be removed before uploading.

Priority: Minor

Test Case ID: 23

Description: The user should be able to search for default templates.

Flow of Events:

1. Enter a keyword in the search input under the "Report Templates" section.
2. Verify that only templates matching the search term appear.
3. Verify that unrelated templates are not displayed.

Expected Result: The template list should be filtered correctly.

Priority: Major

Test Case ID: 24

Description: The user should be able to navigate between pages of default templates.

Flow of Events:

1. Click the "Next Page" button under the "Report Templates" section.
2. Verify that a new set of templates appears.
3. Click the "Previous Page" button.
4. Verify that the previous set of templates appears.

Expected Result: The user should be able to navigate between template pages.

Priority: Major

Test Case ID: 25

Description: The user should be able to search for custom templates.

Flow of Events:

1. Enter a keyword in the search input under the "Custom Templates" section.
2. Verify that only custom templates matching the search term appear.
3. Verify that unrelated templates are not displayed.

Expected Result: The custom template list should be filtered correctly.

Priority: Major

Test Case ID: 26

Description: The user should be able to drag and drop a default template into the middle section.

Flow of Events:

1. Drag a template from the default template list.
2. Drop it into the middle section.
3. Verify that the template details and variables appear.

Expected Result: The template should be successfully added to the middle section.

Priority: Major

Test Case ID: 27

Description: The user should be able to drag and drop a custom template into the middle section.

Flow of Events:

1. Drag a template from the custom template list.
2. Drop it into the middle section.
3. Verify that the template details and variables appear.

Expected Result: The template should be successfully added to the middle section.

Priority: Major

Test Case ID: 28

Description: The user should be able to add a custom variable to the selected template.

Flow of Events:

1. Enter a new variable name in the "New Variable" input field.
2. Click the "Add" button.
3. Verify that the variable is added to the list of template variables.

Expected Result: The new variable should be added successfully.

Priority: Major

Test Case ID: 29

Description: The user should be able to delete a variable from the selected template.

Flow of Events:

1. Select a template.
2. Click the trash icon next to a variable.
3. Verify that the variable is removed from the list.

Expected Result: The variable should be deleted successfully.

Priority: Major

Test Case ID: 30

Description: The user should be able to edit an existing variable in the selected template.

Flow of Events:

1. Click the edit icon next to a variable.
2. Enter a new value in the input field.
3. Click the "Save" button.
4. Verify that the variable name is updated.

Expected Result: The variable should be updated successfully.

Priority: Major

Test Case ID: 31

Description: The user should be able to reorder variables within a template.

Flow of Events:

1. Drag a variable and drop it at a new position in the list.
2. Verify that the variables are reordered accordingly.

Expected Result: The user should be able to change the order of variables.

Priority: Minor

Test Case ID: 32

Description: The user should see a confirmation message when saving a template.

Flow of Events:

1. Make changes to the template variables.
2. Click the "Save" button.
3. Verify that a confirmation message appears.

Expected Result: The system should confirm the save operation.

Priority: Major

Test Case ID: 33

Description: The user should be able to set a report template back to its default settings.

Flow of Events:

1. Select a template.
2. Click the "Set as Default" button.
3. Verify that the template is marked as default and its variables and their orders are as the report's original version.

Expected Result: The selected template should be set as the default and return to its original form.

Priority: Major

Test Case ID: 34

Description: The user should be able to remove a custom template.

Flow of Events:

1. Select a custom template.
2. Click the "Delete" button.
3. Verify that the template is removed from the list.

Expected Result: The selected template should be deleted successfully.

Priority: Major

Test Case ID: 35

Description: The user should be able to create a new custom template.

Flow of Events:

1. Click the "Add" button in the "Custom Templates" section.
2. Enter a template name and variables.
3. Click "Save."
4. Verify that the new template appears in the list.

Expected Result: A new custom template should be created successfully.

Priority: Major

Test Case ID: 36

Description: The user should receive a warning before navigating away with unsaved changes.

Flow of Events:

1. Open a report in the editor.
2. Make a change to the report.
3. Attempt to close the tab or navigate to another page.
4. Verify that a confirmation dialog appears warning about unsaved changes.

Expected Result: The system should prompt the user to save or discard changes before leaving the page.

Priority: Major

Test Case ID: 37

Description: The system should restrict access to unauthorized users.

Flow of Events:

1. Attempt to access the application without logging in.
2. Verify that the system redirects to the login page.

Expected Result: Unauthorized users should not access the system.

Priority: Major

Test Case ID: 38

Description: The system should allow users to reset their password.

Flow of Events:

1. Click the "Forgot Password" link on the login page.
2. Enter a registered email address.
3. Click the "Reset Password" button.
4. Verify that an email with a reset link is sent.
5. Click the link in the email and enter a new password.
6. Log in with the new password to verify the reset was successful.

Expected Result: The user should be able to reset their password and log in with the new one.

Priority: Major

Test Case ID: 39

Description: The system should display an error message for incorrect login credentials.

Flow of Events:

1. Enter an incorrect username or password on the login screen.
2. Click the "Login" button.
3. Verify that an error message appears stating "Invalid username or password."

Expected Result: The system should prevent login with incorrect credentials and display an appropriate error message.

Priority: Major

Test Case ID: 40

Description: The user should be able to change their account settings (e.g., email, name).

Flow of Events:

1. Navigate to the "Account Settings" page.
2. Modify user information (e.g., email, name, or profile picture).
3. Click the "Save Changes" button.
4. Verify that a success message appears and changes are reflected.

Expected Result: The system should successfully update and display the modified account details.

Priority: Major

Test Case ID: 41

Description: The system should suggest related papers based on the content of the generated report.

Flow of Events:

1. Generate a report using text-to-speech or file upload.
2. Click the "Suggest Papers" button.
3. Verify that a list of relevant research papers appears.
4. Verify that each paper includes a title and a summary.

Expected Result: The system should display a list of papers relevant to the report's content.

Priority: Major

Test Case ID: 42

Description: The system should provide the user with an option to copy all the texts in the editor.

Flow of Events:

1. Click the "Copy" button.
2. Paste it to any other program to test.

Expected Result: The system should copy all the contents to the clipboard successfully.

Priority: Major

Test Case ID: 43

Description: The system should allow users to log out of their accounts.

Flow of Events:

1. Log in to the application.
2. Navigate to the "Settings" menu.
3. Click the "Logout" or "Sign Out" button.
4. Verify that the system redirects to the login page.

Expected Result: The user should be successfully logged out and redirected to the login page.

Priority: Major

Test Case ID: 44

Description: The system should handle unexpected server errors gracefully.

Flow of Events:

1. Simulate a server error.
2. Click the "Transcribe" button.
3. Verify that the system displays a user-friendly error message.

Expected Result: The system should display an informative error message without crashing or exposing sensitive information.

Priority: Major

Test Case ID: 45

Description: The system should lock a user’s account after multiple failed login attempts.

Flow of Events:

1. Open the login page.
2. Enter an incorrect password five times for a valid user account.
3. On the sixth attempt, enter the correct password.
4. Observe the login result.

Expected Result: After the fifth failed attempt, the system locks the user's account and displays an error message. The correct password should not allow login after the account is locked.

Priority: Critical

Test Case ID: 46

Description: The system should handle large audio file uploads appropriately.

Flow of Events:

1. Open the report creation page.
2. Click the "Upload Audio" button and select an audio file larger than 100 MB.
3. Click "Upload" and observe the response.

Expected Result: The system should either reject the file with an appropriate error message or successfully process it without crashing.

Priority: Major

Test Case ID: 47

Description: Only administrators should be able to create or delete user roles.

Flow of Events:

1. Log in as a standard user.
2. Attempt to access the "Role Management" section.
3. Attempt to create or delete a role.
4. Log in as an administrator and attempt the same actions.

Expected Result: Standard users should see an "Access Denied" error, while administrators should be able to manage roles successfully.

Priority: Critical

Test Case ID: 48

Description: The system should allow multiple users to edit different reports simultaneously without conflicts.

Flow of Events:

1. User A logs in and opens Report X.
2. User B logs in and opens Report Y.
3. Both users make edits and save their reports.
4. Observe if any data overwrites or conflicts occur.

Expected Result: Changes should be saved correctly for both users without overwriting each other's data.

Priority: Major

Test Case ID: 49

Description: The system should handle high concurrency for speech-to-text requests.

Flow of Events:

1. Simulate 50+ users initiating speech-to-text conversion simultaneously.
2. Monitor system response times and CPU/memory usage.
3. Observe if delays, failures, or crashes occur.

Expected Result: The system should process requests within acceptable performance limits without crashes.

Priority: Major

Test Case ID: 50

Description: The system should prevent duplicate variable names in a report template.

Flow of Events:

1. Open a report template.
2. Add a new variable named "PatientName."
3. Attempt to add another variable with the same name.

Expected Result: The system should display an error message and prevent duplicate entries.

Priority: Major

Test Case ID: 51

Description: User sessions should expire after a period of inactivity.

Flow of Events:

1. Log in to the system.
2. Remain inactive for the configured timeout period (e.g., 15 minutes).
3. Attempt to perform an action such as saving a report.

Expected Result: The system should display a “Session Expired” message and require the user to log in again.

Priority: Critical

Test Case ID: 52

Description: The system should handle external API failures gracefully when fetching preliminary diagnosis suggestions.

Flow of Events:

1. Simulate PubMed (or external API) downtime.
2. Attempt to use the "Preliminary Diagnosis Suggestion" feature.
3. Observe how the system responds.

Expected Result: The system should display a “Service Unavailable” message instead of crashing.

Priority: Major

Test Case ID: 53

Description: The system should enforce a maximum character limit for report fields.

Flow of Events:

1. Open a report template.
2. Enter a string exceeding the maximum allowed character length in a field.
3. Attempt to save the report.

Expected Result: The system should either truncate the input or display an error message preventing excessive input.

Priority: Major

Test Case ID: 54

Description: The system should handle multiple concurrent PDF exports without failure.

Flow of Events:

1. Open multiple reports in separate tabs.
2. Simultaneously trigger “Download PDF” actions.
3. Observe performance and potential errors.

Expected Result: Each PDF should generate correctly without performance degradation or failures.

Priority: Major

6. 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.

6.1 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

6.7 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.

6.8 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.

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

- **Ali Cevat Erçal** contributed to the backend part of the project. He worked on Docker container creation, MySQL and Alembic setup for database migration, and AWS account setup. He also implemented generating reports with transcripts, handled OpenAI API integration, fixed bugs on both frontend and backend, and performed cost analysis.
- **Kerim Eren** contributed to the backend design and infrastructure, making Docker updates for cross-platform features. He also implemented a text-to-speech feature, fixed bugs in backend and frontend, and managed the Docker environment. Additionally, he played a key role as a Scrum master and meeting coordinator.
- **Ömer Amir Akdağ** worked on report generation and backend functionalities. He created report templates, implemented API endpoints, and developed the PDF generation feature. He also conducted market and competitor analysis to support the project's business strategy.
- **Göktuğ Yıldırım** was responsible for frontend development. He implemented the main page and report customization page, designing the customization UI via Figma. Additionally, he worked on endpoint implementation and conducted risk analysis. Göktuğ is also the main name working on diagnosis/Pubmed label prediction.
- **Efe Tokar** focused on application design and communication with external parties. He handled the Ethics Board application process and coordinated with hospitals and doctors. He was also responsible for frontend development. He implemented the main page and report customization page, as well as designing the UI via Figma.

7.2 Helping creating a collaborative and inclusive environment

To create a collaborative and inclusive environment, **Ali Cevat Erçal** actively participated in discussions through communication channels like WhatsApp and online meetings. He ensured that technical challenges were communicated clearly and helped teammates troubleshoot issues. Additionally, he provided guidance on backend and deployment processes, making sure everyone felt comfortable asking questions.

Kerim Eren played a crucial role in maintaining team coordination as the Scrum master. He scheduled and facilitated meetings, ensuring all team members had a voice in discussions. He also encouraged knowledge-sharing sessions to keep everyone aligned on the project's progress and goals.

Ömer Amir Akdağ contributed to an inclusive work atmosphere by being proactive in discussions and decision-making processes. He helped teammates understand the backend and report generation aspects, making sure everyone felt involved in the implementation. He also encouraged open discussions about new ideas and improvements.

Göktuğ Yıldırım helped create a friendly and open environment by maintaining effective communication with both frontend and backend teams. He facilitated collaboration between developers and designers, ensuring smooth coordination. Additionally, he provided support in risk analysis and planning, ensuring transparency in decision-making.

Efe Tokar took an active role in communication and coordination, particularly in external relations with the Ethics Board and hospital representatives. Within the team, he contributed to discussions on UI/UX design, ensuring that all ideas were considered. He also helped create a welcoming environment by engaging teammates in both professional and casual conversations.

7.3 Taking lead role and sharing leadership on the team

Kerim Eren, as the Scrum master, played a crucial role in project management and coordination. He ensured that the team stayed on track by organizing meetings, planning sprints, and facilitating discussions. His leadership helped streamline the workflow and maintain clear communication between different team members.

Ali Cevat Erçal took the lead role in backend development, leveraging his expertise in MongoDB, database management, and API integrations. He guided the team through key technical decisions, resolving backend-related challenges and ensuring smooth deployment. His leadership in infrastructure and implementation made the development process more structured and efficient.

Efe Tokar led the frontend design efforts, focusing on UI/UX development and external communications. He was responsible for designing the application's interface via Figma and implementing frontend features. His leadership ensured that the user experience was intuitive and aligned with the project's goals.

Ömer Amir Akdağ took initiative in report generation and endpoint implementation, ensuring that the reporting system was well-structured and met the project's requirements. His leadership in this domain helped bridge the gap between data processing and user-facing reports.

Göktuğ Yıldırım played a key role in frontend implementation, focusing on the main page and customization features. He also facilitated the integration between frontend and backend, ensuring smooth communication between components. His leadership in UI development helped bring the application to life.

8. References

[1] 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).

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

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

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