Bilkent University
Department of Computer Engineering

Senior Design Project
T2506
Hipograf

Final Report

Team Members:
Salih Furkan Göktaş, 22202620
Orhun Güder, 22202471
Şükrü Eren Gökırmak, 22203746
Ramiz Arda Ünal, 22202554
Artun Berke Gül, 22203316

Supervisor: Prof. Dr. Uğur Doğrusöz
Instructor: Mert Bıçakçı
Instructor: İlker Burak Kurt

04.05.2026

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

1.1 Purpose

Hipograf is a modern medical visualization tool that aims to help people interpret their medical data. The purpose of Hipograf is to assist both doctors and patients make informed medical decisions by providing them with straightforward yet holistic visualizations of their medical records. Users are able to display, compare and plot everything from blood test samples to their prescription medication details with an emphasis of necessary information being grouped together and interconnectedness for data points.

1.2 Scope

Our project, Hipograf, is a web application. It will visualize the medical data and history of the user, with various data visualization tools like charts, graphs, timelines, etc. Hipograf is intended to be useful to doctors when making a medical decision regarding their patient, by presenting them important information in a clear, intuitive way. The most important benefits of Hipograf are seeing the most important information in legible charts and facilitated filtering of desired information, which will result in more informed decision-making. Hipograf is intended for Turkish users, with a goal of being integrated with the Turkish medical system E-Nabız. Since the medical data format is different outside Türkiye, the app will be unavailable outside of Türkiye.

1.3 Design Goals

Our design goals for Hipograf largely align with our Non-Functional Requirements, which are elaborated on further in Section 2.2. These main four design goals are Reliability, Performance, Security and Usability.

Reliability: Hipograf is designed to be used in the medical sector, where access should be guaranteed for as long periods as possible. Utmost care should be taken to avoid presenting the user with erroneously incomplete information.

Performance: One of Hipograf's main objectives is to increase efficiency, being designed to be used by medical facilities across all of Türkiye. This means the system should be able to work under heavy load, letting users quickly access whatever representation of data they require. It should also function properly on the computers available at medical facilities.

Security: Secure handling of personal and private data is critical to Hipograf's success. No medical data will be kept permanently on site unless directly uploaded by the user (concerning data that does not have a one-to-one correspondence with equivalent data in existing external medical databases that Hipograf connects to). All external data that is

handled during the duration of a logged-in session will be removed immediately upon session termination.

Usability: As Hipograf's main purpose is to be an aid to medical information visualization, usability is one of the most important aspects of the system. The user interface should be both intuitive and accessible enough to maximize the number of people who can use it effectively.

2. Requirements Details

2.1 Functional Requirements

2.1.1 User Types

Hipograf has two types of users as its target audience: Medical practitioners and patients. Medical practitioners are expected to use the application to analyze their patients' medical information to make more informed decisions regarding their decision in treatment. Patients can use Hipograf to share their information, and view their own medical data for any reason, especially if they themselves are a particularly informed patient who would benefit from being able to access detailed visual representations of this data. Medical practitioners and their patients are somewhat interchangeable roles depending on the context, since one medical practitioner can be the patient of another. Medical practitioners who are also patients themselves can also view their own data.

2.1.2 Dashboard Page

There are two types of dashboard pages: One for the patient and one for the practitioner. In either account case, it will be the first page visible after the user logs in. This section will detail the patient dashboard. Information about the practitioner page can be found in Section 2.1.7.

The patient dashboard features various elements accessible to the user:

- An interactive 3d humanoid model that shows the relative anatomical locations of afflictions affecting the patient. For instance, if the user has a disease affecting their cardiovascular system, a colored orb will be visible above where the heart of the model would be. Clicking on these disease-designating orbs highlights them and creates a pop-up giving certain details about the disease. The user also has the option to view more information about the currently selected disease by selecting an option to view "Details" on the top right of the page. To bolster keyboard accessibility, the diseases can also be navigated through using the arrows found on the left side of the page.

- The choice of having either an activity timeline, agenda or calendar visible (these components will be elaborated on further in Section 2.1.4) on the page
- Access to a customizable selection of widgets that give the user condensed information about other components. These widgets include ones that:
    - Out-of-reference values in the most recent (chronologically) blood test
    - Current medications being taken by the patient
    - Overview of linked wearable data of the patient
    - Recent History of the events that the patient has undergone
    - Upcoming Schedule of the events that the patient will undergo

2.1.3 List Visualization Pages

Both Practitioners and Patients are able to access various pages that show user medical information in accessible list formats. These pages and their functionalities are detailed below:

Hospital Visits Page

List all the hospital visits that the user has undertaken in the past. Every hospital visit includes additional information, including the hospital it took place in, the department of the hospital it concerned, the doctor that attended to the patient and the date of the visit. If there is an attached diagnosis, information about the diagnosis is also present, including the name, ICD code, severity, affected body region and any relevant clinical notes left by the relevant doctor. Visits that contain diagnoses may also be linked to prescribed medications, in which case their name and consumption information will also be listed.

The list can be sorted with respect to date, in either ascending or descending order. They can also be sorted alphabetically with respect to hospital and doctor name in either order.

The user is able to search for hospital visits on the basis of the name of the hospital, doctor, department, diagnosis (if any) and prescriptions (if any). It is also possible to filter out only the visits that result in diagnoses.

Prescriptions Page

List all medications prescribed to the patient. On the main page, every prescription has its name, dosage, associated diagnosis, period of consumption and status (whether it's active or not) listed. Prescriptions can be filtered by selecting only ones that are currently active or already expired, or by searching for particular prescription/diagnosis names. Active prescriptions have green highlighting to differentiate them from expired ones.

Clicking on a particular prescription brings the user to an additional page with more detail about this particular prescription, containing the following information in clockwise order:

- The generic name, brand name, pharmaceutical class, absorption route, dosage and frequency
- Prescription ID, current status, start date and end date of the prescription period
- Linked hospital visit, including the hospital name, visit date,
- Linked diagnosis, including its ICD code, visit date, associated doctor

There are also other tabs on this additional page detailing the general usage instructions, safety precautions and potential side effects.

The information on the page can also be exported to a CSV file, where each row contains (in order) the medication brand name, generic name, dosage, frequency, diagnosis, prescribed by (doctor), start date, end date and current status.

Diagnoses Page

List all the diagnoses associated with the patient. In a list, each diagnoses' date, ICD code, diagnosis title, related hospital department and relevant doctor is listed.

The list can be sorted with respect to date, in either ascending or descending order. They can also be sorted alphabetically with respect to ICD Code, diagnosis name, department or doctor name in either order.

The patient can search for specific diagnoses using the diagnosis name, department name, ICD code, prescription names (if any).

A date range can be specified for the displayed lists. Presets for this range are also available, including options for the last month, last three months, last six months, last year, last 2 years and last 5 years. The date range can be modified dynamically after preset selection.

Displayed diagnoses can be filtered by severity: All, Mild, Moderate or Severe.

A particular diagnosis can be selected to reveal more information about it. This brings up three separate information boxes. The first box contains clinical details, including a natural language description of the diagnosis, the associated body region, common symptoms and common treatments (if applicable) and the severity. The second box shows the diagnosing physician, department, hospital, date and any clinical notes provided by the doctor. The final tab shows any and all prescriptions that are related to the diagnosis, including their name, dosage and frequency.

The information on the page can also be exported to a CSV file, where each row contains (in order) the date, ICD code, diagnosis name, disease severity, department name, doctor name and hospital.

Lab Tests Page

List a patient's medical tests, which mostly consists of their blood work tests. Each lab test is listed in chronologically descending order. The displayed selection can be filtered by a provided date range or by searching for a particular test or subtest name. Subtests refer to the individual data values reported within a particular test. By default, all tests are shown. Visible tests can be constrained by only showing the tests that contain exclusively normal results, only showing tests that contain at least one out-of-reference result, or only showing tests that contain at least one out-of-reference-result without showing the normal results at all (normal subtests are completely omitted).

Individual tests can be expanded to see the subtest results. Each subtest row contains the name of the subtest, the resulting value, the minimum and maximum normal reference range and the unit used for the measurement. A visual indicator of whether the subtest is normal or

out-of-reference is also provided. Patients have the option of viewing a specific subtest either on the timeline on the Timeline Page or to view a plot of them (for all time) on the Plot Page. The patient also has the option of expanding or collapsing all visible tests simultaneously

Users have the option of uploading their own lab test results. They can either do this entirely manually or upload a PDF which will be automatically scanned. The scan results will be presented to the user so that they can edit any mistakes that arose during the automated process, including the title of the test and the date it took place. Tests can also be edited or deleted after initial submission.

Imaging Data Page

List all the radiological images of a patient. The list of radiological images can be displayed in two different formats, either in a grid view or a card view with larger images. Each radiological image lists the name of the image itself, the hospital it was taken at and the date the image was taken alongside it (in both display formats). The search functionality can be used to search for images by either image name or hospital name.

A particular image can be selected to expand it and observe it more closely. The same information is provided here, alongside an option to download the image to the patient's local computer. Selecting a radiological image from the list takes the practitioner to a page that gives more detailed information.

Users are able to upload their own scans, providing the image name, hospital name, date and the image file itself. Various image file formats are supported.

Vaccinations Page

List vaccinations a patient has had administered in reverse chronological order. Vaccines are visible in a grid display format where each block in the grid contains the name of the vaccine, the date it was administered and the hospital it was administered in. The search feature can be used to narrow down results by either the name of the vaccination or the name of the associated hospital.

Users add their own vaccination records to the system by uploading the name, hospital and date of the vaccination.

Wearable Devices Page

Patients have the option of linking their wearable devices to the Hipograf ecosystem, including their mobile phones. Links are created through the mobile phone applet described in Section 2.1.9. If no links have been created, this page will contain instructions on how to create the link between Hipograf and the device. If there are existing links, the user will be shown an overview of the connected device including its type, name and when it was last synced to Hipograf. They will have the option to disconnect the device.

If a linked device exists, the relevant health data will be fetched periodically (every 5 minutes) and automatically displayed up to and including the last 30 days of activity. The displayed data is widely varied, including heart rate, steps, calories taken across distinct meals, calories burned and respiration rate.

2.1.4 Timeline Visualization Page

Hipograf will include a dedicated page containing a timeline, alongside an agenda and a calendar. The page serves as a centralized location for viewing all medical events the patient has undergone (in the past) and has scheduled (in the future) within a chronologically sorted manner. Visible elements include:

- Hospital appointments (and associated diagnoses), in blue
- Lab tests (including blood tests), in red
- Prescriptions (with the whole duration of the prescription displayed), in green
- Vaccinations, in purple

The user can filter these event types in any way they desire, including or excluding any combination of them using color-coded filtration buttons on the top right.

The timeline, agenda and calendar subpages offer different ways of viewing the same data with respect to the user's preference.

The timeline page lets the patient view this information on a sorted timeline. The timeline offers various levels of zoom, ranging from a particular day to a week, two weeks, a month and 4 months. Intermediate levels of zoom between the most and least specific options can also be achieved. The timeline is scrollable by either dragging the timeline itself by moving the horizontal scroll bar at the bottom of the timeline. The horizontal scroll bar scales with the timeline zoom level changes to provide a visual indication of the current zoom. The scroll bar can also get smaller if the user scrolls too far in either direction as this will cause the timeline itself to grow in horizontal length. It can be reduced back to the default length for the selected zoom level by clicking the 'Today' button on the top left, which will also bring the scroll bar back to its expected size.

The current day is indicated with a blue vertical line on the timeline. The visual display of the events themselves changes based on the zoom level. Prescriptions always use a green horizontal line ranging from their start date to their end date. Discrete events like tests (red), appointments (blue) and vaccinations (purple) show up as circular or rectangular icons depending on the zoom level. If too many discrete events that have occurred within quick succession of one another are included, they may be grouped together on the furthest-out zoom level. If so, the user will have to click on the group so the multiple events together. The grouping label will have color-coded indicators to indicate the types of events included within. Timeline events will be stacked vertically on top of each other where possible to indicate that they are on the same date. Events will also include a visual indicator of their type beyond the difference in color itself using an appropriate icon.

To bolster accessibility on the timeline, it is possible to use the keyboard to either change zoom level (Ctrl + Scroll Wheel) or scroll horizontally (Ctrl + Left/Right Arrow).

The agenda page lets the patient view their events on a weekly basis. A particular week is shown on screen at one time with options to move forwards or backwards and see subsequent or previous weeks respectively. Daily time slots are shown on the left, and events are placed accordingly. Events that have no particular time associated with them are

marked at the very top in a special row labeled “All Day”. The current day is marked by being highlighted in blue, and the current time is also highlighted with a blue horizontal line.

The calendar page lets the patient view their events on a monthly or yearly basis. By default, a particular month is shown on screen at one time, with the events that appear on particular days visible in a calendar format. Patients have the option of switching from this default month view to a year view instead for easier month selection. The year view shows a particular year on screen and contains the number of events that took place in a month for every month displayed on screen. The current day is marked by drawing a blue circle around the day’s number. In the year view, the month is highlighted in blue instead, including a tag for “Current”.

All timeline components are persistent across application pages, retaining their position and filtration state if navigated to another page and then returned to.

There is a final tab on the timeline page that shows a user’s event history and scheduled events, being functionally identical to the widgets described in Section 2.1.2.

2.1.5 Plot Visualization Page

The application will include a robust plotting system, where the user will have the option of choosing which plots to create based on which data they would like displayed. These plots will display data as a function of time. The data itself mostly consists of periodic data that is gathered on various discrete instances over time, such as the values acquired from medical blood tests. Various other events, like prescription durations, can also be plotted. Data obtained from wearable technology that the patient has linked appropriately can also be plotted.

The user is able to create plots on an interactive screen where one or more data types can be specified from a searchable dropdown menu, alongside a time range, to plot. The selections can consist of individual blood test values, grouping of all values included in an individual test, prescriptions and data gathered from the linked wearable device(s). These can all be searched for in the provided search menu. The time range selector allows the user to narrow down the plot that is about to be created to values that were taken in a specific time frame. It contains a calendar that has blue highlights for days that contain test values and green highlights for days when medications were taken. Pre-existing time ranges can also be selected, including the last month, last three months, last six months and last year.

A more powerful AI-powered search feature is available that automatically parses user input and gives the option of creating a plot that fits the provided criteria. For instance, a user might type in “kidney functions in the last 4 months” and it will automatically offer to create a plot of all the subtest related to kidneys with data points that have been measured in the last 4 months from today.

Created plots can be resized from a selection of being small, medium or large (in terms of vertical space). The data points in the plot can have their color, line style and point style changed. The data in the plot can also be automatically normalized.

When more than one plot has been created, the user can choose to either view them separately on the screen or combine them to produce plots plotting several data points at once. Plots can also be moved vertically relative to other existing plots to change their visual

ordering. Multiple y-axes will appear to accommodate the multiple data types and their different units. A combined plot can then also be split up back into the individual constituent plots. Ultimately, freedom will be given to the user on which datasets to combine on which plot according to what combinations they feel would be useful to observe. All existing plots on the page can be cleared using the clear button on the top right. Created plots can be exported as PNGs for external use.

Users will also have the option to save presets for the plots that are being displayed in order to recreate them at a later date. This specifically only includes the ways in which the data is displayed, not the data itself. For instance, the fact that a plot has been created over a certain time period that charts two different subtests together might be saved, which can then be brought back at any point. These presets can also be renamed or deleted.

All plots are persistent across application pages, retaining their position and state if navigated to another page and then returned to.

2.1.6 Practitioner Pages

Practitioners have access to a few additional pages that are not accessible to ordinary patient accounts, accessed when they login normally.

Upon logging in, practitioners have a separate dashboard that lists the patients that are linked to them alongside brief information for these patients.

They have a separate patient list page where they can choose to list all of the patients that are currently linked to them, similar to the dashboard. From this page, they can select a particular patient to enter a more detailed page for the selected patient. This page displays basic personal information (name, birthdate, city of residence, etc.), along with a brief overview of their existing medical information and relevant events.

From the detailed information page, practitioners can choose to “see” through the eyes of a practitioner, accessing Hipograf through their eyes and being able to use all the components described in Sections 2.1.2 (Patient Dashboard), 2.1.3 (List Pages), 2.1.4 (Timeline Page) and 2.1.5 (Plot Page) with the data of the selected patient. This state will be marked with a persistent indicator at the top of the screen (including the patient’s name), indicating that the current screen is using the data of the patient.

If a practitioner is also a patient themselves, they can also analyze their own data using a button in the bottom left of the screen.

2.1.7 Session/Account/Data Management Pages

All Hipograf features require user authentication. When a new user connects to Hipograf, they will first be greeted with a login page where they will be expected to input a correct email password tuple to continue. Erroneous login attempts will result in appropriate error messages being displayed. Users are not expected to provide their account type when logging in, it will be determined automatically from their existing account details.

Users can also choose to create an account, which will ask them for a name, username, email, password (alongside a password confirmation) and whether it will be a patient or a practitioner account. Creating accounts that share emails or usernames with existing accounts is forbidden and an error message will be displayed. The acceptable password criteria are elaborated in the Security Requirements portion of Section 2.2. After account creation, the user will be expected to log in with their newly created account.

Once logged in, the user is able to access a settings page from the bottom left of the page. The page includes several components, including granting users the ability to update their account information, change their password and manage their associated data. Users are able to update their username, display name and email. The username and email fields cannot be identical to the username or email of a different existing account for the update to be successful. Their password can also be changed, which also requires entering the current password. Users have the option of deleting all plot presets and uploaded tests associated with their account. They also have the option of deleting their account entirely, which will also delete all associated data alongside it.

Users authenticated by Hipograf are able to logout from their current session using the button on the bottom left of the page, returning them to the login page. Logging out will delete all existing persistent session data from the local computer being used to access HIpograf.

2.1.8 HIPPO Chatbot System

After logging in, users will see a persistent circular purple chat icon in the bottom right corner of the screen. This is a button used to access HIPPO, Hipograf's LLM-powered chatbot assistant. The keyboard shortcut Alt + H can also be used to access HIPPO. HIPPO has a wide knowledge-base of the Hipograf application and can give users guidance on how to access and use different pages of the application. User questions about how to use a specific feature of the application will also be answered. He is also capable of providing general medical advice, but all answers related to such matters include warnings to consult a medical professional.

2.1.9 Wearable Data Collection System

The wearable data is collected using a custom-made Android App as the mediator between the external device and Hipograf. Once installed, the android application asks the user to sign in with their existing Hipograf account, upon which they can sync the health data their phone has collected with Hipograf, viewing it on the Wearable Device List page explained in Section 2.1.3.

2.2 Non-Functional Requirements

Reliability Requirements

Hipograf is designed to be used in the medical sector, where access should be guaranteed for as long periods as possible. Utmost care should be taken to avoid presenting the user with incomplete information. Hipograf should notify the user that its knowledge base is incomplete when there is an issue with the amount of information it is able to access.

Hipograf will have an uptime of at least 95% on a daily basis, translating to at least 347 days a year of full availability. This includes weekends and holiday periods. Only the holiday periods of Türkiye will be considered in this definition, since the application is intended for use in Türkiye.

Any scheduled maintenance for server upkeep should not result in any outages exceeding 3 hours, with off-peak hours being preferred as possible, especially the early morning range from 3.00 AM to 6.00 AM GMT+3. Once again, since service is only provided within Türkiye, no consideration needs to be made for whether these hours would coincide with a different region's peak usage hours.

Any inability to fetch data from an API or otherwise, whether a complete or partial failure, must be immediately communicated to the Hipograf user in the form of an alert that blocks further input until acknowledged. This precaution will ensure that these users do not gain an unfound misunderstanding of an arbitrary patient's medical history with the limited available data.

Performance Requirements

One of Hipograf's main objectives is to increase efficiency, not take away from it; being designed to be a system that, once deployed, will be used by medical facilities across all of Türkiye. This means the system should be able to work under heavy load, letting users quickly access whatever representation of data they require.

Hipograf's login page should take less than 2 seconds (wall-clock time) to connect to a client with an internet connection (in other words, any potential slowdowns should not have Hipograf's server infrastructure as the common denominator. Naturally, we cannot account for subpar connection speeds that a user is experiencing when connecting to the internet).

After logging in, Hipograf's main overview dashboard should take less than 5 seconds to load.

Subsequent operations conducted to view medical data (such as moving/scaling/combining graphs, filtering selections, specifying affliction types to narrow data input, etc.) should feel responsive and fluid, with no input mouse/keyboard lag more than 200ms present on hardware comparable to the average work computer available at medical institutions in Türkiye.

Hipograf should still be fully accessible and operational when 500 users are simultaneously connected to it, without encountering any slowdowns that would violate the time constraints given in the previous paragraph.

Security Requirements

Secure handling of personal and private data is critical to Hipograf's success. No medical data will be kept permanently on site, and all data that is handled during the duration of a log-in session will be deleted immediately upon session termination.

Web-based access to Hipograf will be over an HTTPS connection using Transport Layer Security version 1.3, which itself will use the Advanced Encryption Standard in Galois Counter Mode with 256-bit keys.

Direct log-in to Hipograf will be enabled through user defined passwords that must meet the following requirements:

- At least eight (8) alphanumeric characters
- At least one (1) uppercase letter
- At least one (1) lowercase letter
- At least one (1) numeric digit
- At least one (1) special character

These passwords will be hashed using the Argon2 password hashing algorithm and stored in a local database that will not be directly web-facing.

After a user logs out of their current Hipograf session, no medical data will be allowed to persist on any layer of Hipograf's internal architecture.

Usability Requirements

As Hipograf's main purpose is to be an aid to medical information visualization, usability is one of the most important aspects of the system. The user interface should be easily understandable and usable, along with being accessible enough to keep the number of people that could use it effectively at a maximum.

All major operations will be possible by sole use of the keyboard, and care will be taken to avoid the presence of any keyboard traps (UI elements that keyboard input is unable to "escape" as focus cannot be redirected back to previously accessible page components) [1].

All visual content should have an explanatory textual description that is shown within the HTML tags and also when a visual component fails to load [1].

When navigating through to different components and pages of the site, the current position tab should be identified on the screen through a navigation bar [1].

There should be a color contrast of at least 4.5:1 between text and its background for optimal readability [1].

2.3 Pseudo Requirements

In addition to the functional and non-functional requirements, we impose additional requirements regarding the project.

- The frontend of Hipograf was developed using React Typescript, Tailwind CSS and additional UI libraries that can be found in the NPM package registry.

- The backend of Hipograf shall be developed using Python, using Flask as the web framework. Either pip (alongside a virtual environment) or uv can be used as the package manager. Additional packages shall be available from the PyPI repository.
- The database of Hipograf shall be a remote instance of MongoDB that all clients connect to, in order to ensure data parity across development clients.
- The android applet component shall be developed using Android Studio, using Jetpack Compose for the user interface and Kotlin for its own backend.
- All software used in the development of Hipograf shall be under permissive software licenses such as MIT license, Apache 2.0 license, etc [2], [3].

3. Final Architecture and Design Details

3.1 User Stories

General Management

US1 - Registration

User Story: As a medical practitioner or a patient, I want to register to Hipograf so that I can securely authenticate myself to the system in the future.

Acceptance Criteria:

- The user will provide a username and a password.
- The password will be checked to ensure it complies with the accepted standards.
- If successful, the user will be told that their registration was successful.
- The user will be redirected to the login page.

US2 - Login

User Story: As a medical practitioner or a patient, I want to log in to Hipograf so that I can use the application.

Acceptance Criteria:

- The user will provide a name and password.
- The username and the corresponding hash of the password will be checked to make sure it is equal to the one in the database.
- If the credentials are correct, the user will be logged into their account, and redirected to their home page.

US3 - Logout

User Story: As a medical practitioner or a patient, I want to log out of Hipograf so that I can close my session.

Acceptance Criteria:

- The user will click the “log out” button.
- If successful, the session of the user will end, and they will be logged out.
- If the server shuts down or experiences another issue, the valid cookies issued by the server will reset.

US4 - Account Deletion

User Story: As a medical practitioner or a patient, I want to be able to delete my account so that Hipograf servers will wipe my data.

Acceptance Criteria:

- The user will click the “Delete Account” button.
- On success, the information of the user, such as their login credentials and data viewing preferences, will be wiped from the Hipograf servers.
- On failure, a failure message will be shown, prompting the user to try again.

US5 - Password Change

User Story: As a medical practitioner or a patient, I want to be able to change my password so that I can change my login credentials.

Acceptance Criteria:

- While logged in, the user will click the “Change Password” button, and be prompted to enter their new password, and confirm it.
- On success, the password of the user will successfully be changed.
- On failure, a failure message will be shown, prompting the user to try again. Failure can happen due to network connectivity or an unsuitable password.

US6 - Account Information Update

User Story: As a medical practitioner or a patient, I want to update the information available on my account so that it stays up to date when my credentials change.

Acceptance Criteria:

- While logged in, the user will click the “Update Account Details” button, and be prompted with a page where they can update some of their credentials such as username, email, etc.
- On success, the account details of the user will be changed.
- On failure, a failure message will be shown, prompting the user to try again.

US7 - Reset Password

User Story: As a medical practitioner or a patient, I want to be able to reset my password in the login page, so that I can change it when I forget my password.

Acceptance Criteria:

- While logged out, the user will click the “Forgot password” button, and will be prompted to enter their registered email. After, a link will be sent to the user’s registered e-mail, where they can reset their password.
- On success, the user’s password will change and the user will be redirected back to the login page.
- On failure, a failure message will be shown, prompting the user to try again. Failure can happen due to the absence of an account related to the given email, or connectivity issues.

US8 - Clear Preferences Data

User Story: As a medical practitioner or a patient, I want to be able to clear my data preferences so that I can reset the information displayed in my dashboard to its default state.

Acceptance Criteria:

- While logged in, the user will click the “Clear preferences” button.
- On success, the data visualization preferences will be reset.
- On failure, a failure message will be shown.

US9 - Chat with HIPPO

User Story: As a medical practitioner or a patient, I want to be able to chat with HIPPO, so that I can be informed about the functionalities of Hipograf.

Acceptance Criteria:

- While logged in, the user will click the HIPPO icon, and either pick a predetermined chat option or type a query themselves.
- On success, HIPPO will answer the user’s query.
- On failure, a failure message will be shown.

Patient Selection System

US10 - List Patients

User Story: As a medical practitioner, I want to be able to list all patients appointed to me so that I can select a patient and access their medical history.

Acceptance Criteria:

- The system shall display a list containing the names of all patients who have granted me access to their medical records.
- Each patient name in the list must be selectable/clickable.

US11 - Select Patient Page

User Story: As a medical practitioner, I want to access the comprehensive, single set of pages for a selected patient, so that I can quickly review their entire medical history and current status.

Acceptance Criteria:

- Upon selecting a patient from the list, I shall be navigated to a detailed patient profile page.
- This page must clearly display Basic Personal Information (Name, Birthdate, City of Residence, etc.).
- This page must display a unified, filterable Medical Timeline compiling all major events.
- The page must include separate sections or widgets for Recent Blood Work, Medications, Hospital Visits, Diagnoses, Operations, Radiology Images, and Vital Signs data from wearables.

List Management System

US12 - List Blood Test

User Story: As a medical practitioner or a patient, I want to view and analyze all a patient's blood test results, so that I can monitor specific physiological changes and health trends over time.

Acceptance Criteria:

- I shall be able to view a list of all blood work instances and all radiology images.
- I must be able to filter blood work instances by criteria such as the presence of a specific parameter and the time/date the test was performed.
- I must be able to select a single blood work instance to view a detailed page showing the test date and all measured parameters.
- From the detailed blood work view, I must be able to click on a specific parameter to display a graph showing all historical measurements of that parameter across all blood work instances.

US13 - Upload Blood Test

User Story: As a medical practitioner or a patient, I want to upload blood test results to Hipograf so that I can use my blood test data points inside Hipograf.
Acceptance Criteria:

- I shall be able to upload a blood test file from an external API in PDF format to Hipograf, and get back the test results in table form.
- I must be able to see and edit the inferred test result data before submitting it.

US14 - List Diagnosis

User Story: As a medical practitioner or a patient, I want to view and filter a patient's past diagnoses, so that I can analyze their complete health profile and history of conditions.

Acceptance Criteria:

- I shall be able to view a list of all past and current diagnoses.
- I must be able to filter the diagnoses list using criteria such as whether the diagnosis is chronic or acute, the diagnosis's name/category, or the time frame when the diagnosis was given.

US15 - List Hospital Visit

User Story: As a medical practitioner or a patient, I want to list all of a patient's hospital visits and apply filters, so that I can review past acute care episodes and dates of admission.

Acceptance Criteria:

- I shall be able to view a list of all documented hospital visits.
- I must be able to filter the list of hospital visits by criteria, such as the date of admission.

US16 - List Radiological Images

User Story: As a medical practitioner or a patient, I want to view, filter, and access detailed information about a patient's radiological images, so that I can review the findings from medical imaging procedures.

Acceptance Criteria:

- I shall be able to view a list of all radiological images (X-rays, CTs, MRIs, etc.).
- I must be able to filter this list by the type of imaging (e.g., "CT Scan") and the date it was taken.
- Selecting an image from the list must navigate me to a page with detailed information about that specific radiological exam (e.g., full report, findings).

US17 - List Operations

User Story: As a medical practitioner or a patient, I want to list all of a patient's medical operations and apply various filters, so that I can understand their surgical history.

Acceptance Criteria:

- I shall be able to view a list of all recorded medical operations/surgeries.

- I must be able to filter the list of operations using various relevant criteria (e.g., date of operation, type of operation).

US18 - List Vaccinations

User Story: As a medical practitioner or a patient, I want to list all of a patient's past vaccinations so that I can be informed and avoid repeat doses of vaccines.

Acceptance Criteria:

- I shall be able to view a list of all past vaccinations.
- I must be able to filter the list of vaccines using the date of the vaccine administered, and the contents of the vaccine.

US19 - List Prescriptions

User Story: As a medical practitioner or a patient, I want to list all of a patient's past and current prescriptions so that I can be informed and avoid repeat doses or conflicting prescriptions.

Acceptance Criteria:

- I shall be able to view a list of all past and current prescriptions.
- I must be able to filter the list of prescriptions by date, and the active chemical within the prescriptions.

US20 - List Wearable Data

User Story: As a medical practitioner or a patient, I want to list all of a patient's wearable data, if they have opted in to list them, so that I can be more informed about my patient's medical status.

Acceptance Criteria:

- I shall be able to view a patient's past wearable data.
- I must be able to list the wearable data by date, and relevant wearable data class.

Timeline Management

US21 - Filter Event Category

User Story: As a medical practitioner or a patient, I want to be able to choose the categories of kinds of events shown in the timeline so that I can focus on events that are currently important to me.

Acceptance Criteria:

- The user will have access to a menu containing the currently available event categories on the same page as the timeline.
- The user will be able to toggle on or off these categories through the menu.
- The toggled off categories will be removed from the timeline.

US22 - Choose Timeframe

User Story: As a medical practitioner or a patient, I want to be able to choose a timeframe for the timeline so that I can focus on events that took place in that time window.

Acceptance Criteria:

- The user will be able to input a start date and end date for the timeline.
- These dates will decide the earliest and latest dates that will be shown on the timeline, respectively.

US23 - Edit Timeframe

User Story: As a medical practitioner or a patient, I want to be able to use the dedicated resize buttons or key combinations to change the length of the timeframe of the timeline so that I can prevent cluttering that can be caused by an accumulation of a high number of events in a timeframe.

Acceptance Criteria:

- The user will be able to zoom in and zoom out of the timeline.
- If the user zooms in, the earliest date shown on the screen will increase and the latest date shown on the screen will decrease, showing a narrower date interval on the screen.
- If the user zooms out, the earliest date shown on the screen will decrease and the latest date shown on the screen will increase, showing a wider date interval on the screen.

US24 - Scroll Timeline

User Story: As a medical practitioner or a patient, I want to be able to use the scroll wheel or a key combination to change the earliest and the latest dates shown on the timeline while keeping the length of the timeframe constant so that I can reach all events on the timeline even if I resized the timeline.

Acceptance Criteria:

- The user will be able to scroll the timeline horizontally towards either direction.
- If the timeline is scrolled to the right, both the earliest and the latest date shown on the screen will increase.
- If the timeline is scrolled to the left, both the earliest and the latest date shown on the screen will decrease.
- In both cases, the period of time shown on screen will be kept constant.

US25 - Obtain Event Information

User Story: As a medical practitioner or a patient, I want to be able to click on events shown on the timeline so that I can get detailed information about that specific event.

Acceptance Criteria:

- The user will click on a specific event stationed on the timeline.
- An info box connected to the chosen event will appear above or below depending on the circumstances of the timeline.
- This info box will contain detailed information related to the chosen event.

US26 - Save Preferences

User Story: As a practitioner or a patient, I want to be able to save the currently applied filtering and timeline viewing options as a preference so that I can retrieve it later and apply it instantly, preventing unnecessary burden.

- The user save will click to save their current preferences.
- They will be asked to give the preset a name.
- The currently applied customization choices will be saved under the given name.

US27 - Load Preferences

User Story: As a medical practitioner or a patient, I want to be able to load a preference set from a list I have previously saved before so that I would not need to reconfigure every option according to my needs everytime I login to the system.

Acceptance Criteria:

- The user opens the preset menu that lists all their presets.
- The user chooses the preset they would like to apply
- The timeline is adjusted according to the details of the chosen preset.

Graph Management

US28 - Create Graph

User Story: As a medical practitioner or a patient, I want to be able to create individual graphs that represent multiple different sets of data on a single graph so that I can compare and contrast their change relative to one another over time.

Acceptance Criteria:

- The user should be able to select multiple datasets when the graph is being created.
- The user should be able to select a timescale that fits for both of them on the same graph.

- The compounded data should be automatically normalized so that I can view the graphs with relative ease.

US29 - Search for Graph

User Story: As a medical practitioner or a patient, I want to be able to type out my desired graph set-up along with its relevant timeframe and have it appear for me.

Acceptance Criteria:

- The user should be able to type out, in normal English, their desired graph(s) along with the timeframes they remain relevant for.
- The user should then be able to immediately access these created graphs such that they can perform the rest of the normal graph operations on them.

US30 - Edit Graph

User Story: As a medical practitioner or a patient, I want to be able to edit, scale and move the graphs that represent my data so that I can gain a holistic understanding of the interaction of datasets.

Acceptance Criteria:

- The user should be able to edit graphs to change the timescale setting or the data being graphed.
- The user should be able to resize the graphs around the screen.
- The user should be able to move the graphs around on the screen without being limited to specific spots on the screen.

US31 - Recolor Graph

User Story: As a medical practitioner or a patient, I want to be able to recolor a graph so that I can have better color contrast with other present graphs.

Acceptance Criteria:

- The user should be able to change the color of any individual graph by picking from a color picker with several dozen color options

US32 - Combine Graph

User Story: As a medical practitioner or a patient, I want to be able to combine graphs to combine the data they are representing in a united space.

Acceptance Criteria:

- The user should be able to combine existing graphs to result in graphs that graph the data simultaneously.
- The user should see an overview that automatically normalizes graph data.

US33 - Split Graph

User Story: As a medical practitioner or a patient, I want to be able to split combined graphs to once again see them on their own.

Acceptance Criteria:

- The user should be able to split combined graphs to get the original graphs before the combination.

US34 - Save Preferences

User Story: As a medical practitioner or a patient, I want to be able to save my preferences for a graph so that I can refer to them at a later date.

Acceptance Criteria:

- The user should be able to save their preferences for time or data type, across single or multivariate graphs, in the local database.
- The user should not have the option of saving any medical data.

US35 - Load Preferences

User Story: As a medical practitioner or a patient, I want to be able to restore previously saved graph preferences so that I can receive a familiar overview without having to consistently reconfigure it.

Acceptance Criteria:

- The user should be able to reproduce the previously saved data onto the graphing interface.
- The user should be able to do this consistently, as many times as is required.
- The user should have the option of editing and deleting preferences in addition to restoring them.

3.2 Sequence Diagrams

A few select sequence diagrams capturing core behavioral functionality will be included.

Login

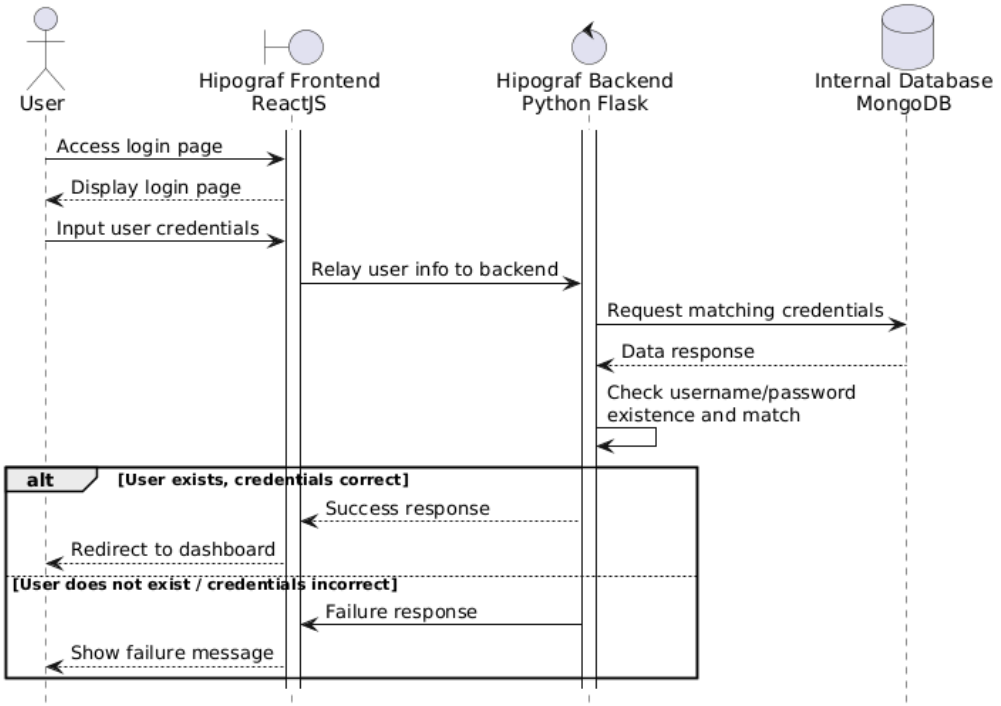

Fig. 1. Sequence Diagram for User Login.

Timeline Transformations

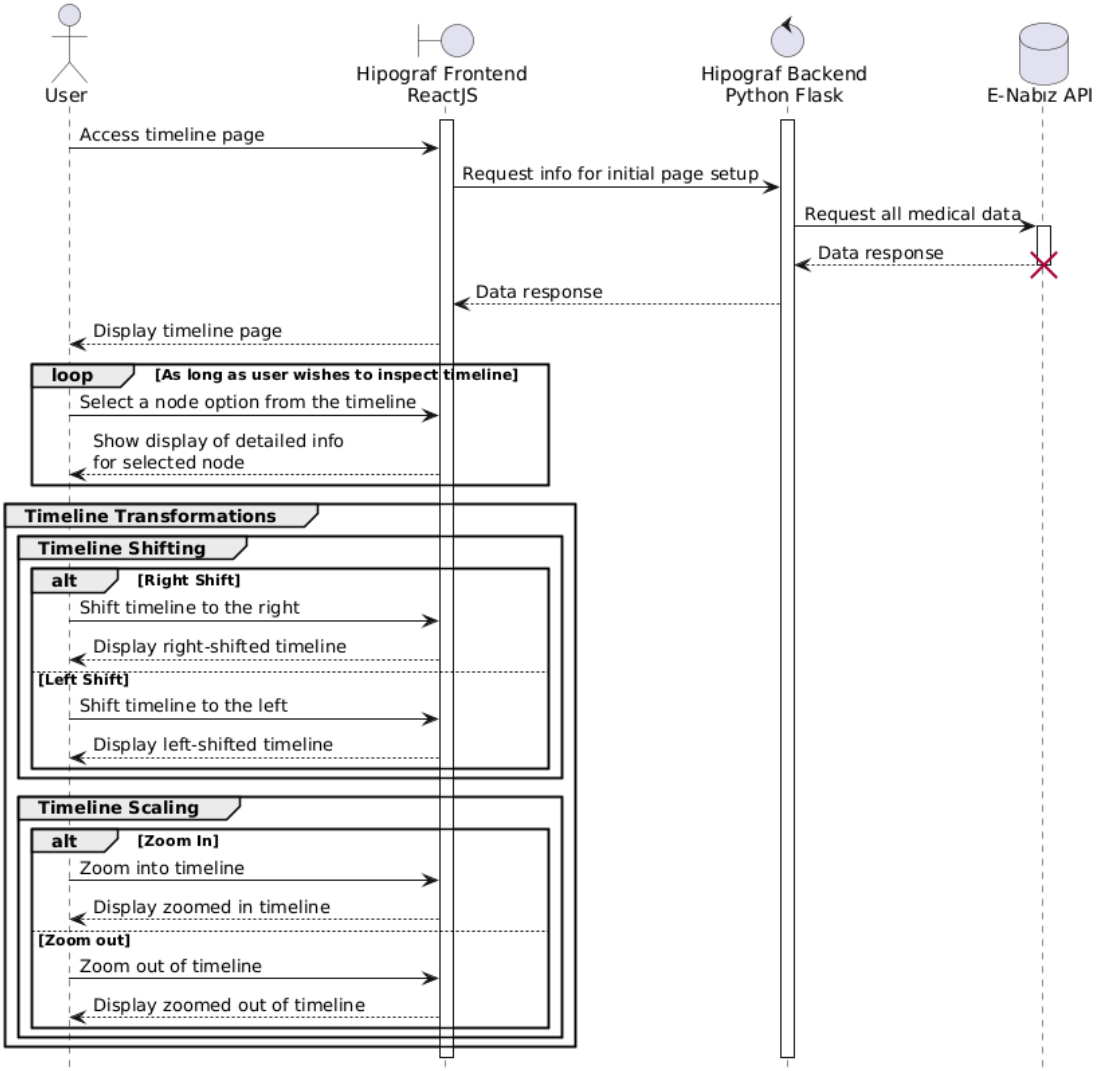

Fig. 2. Sequence Diagram for Transformations on the Timeline Page.

Graph Page Access

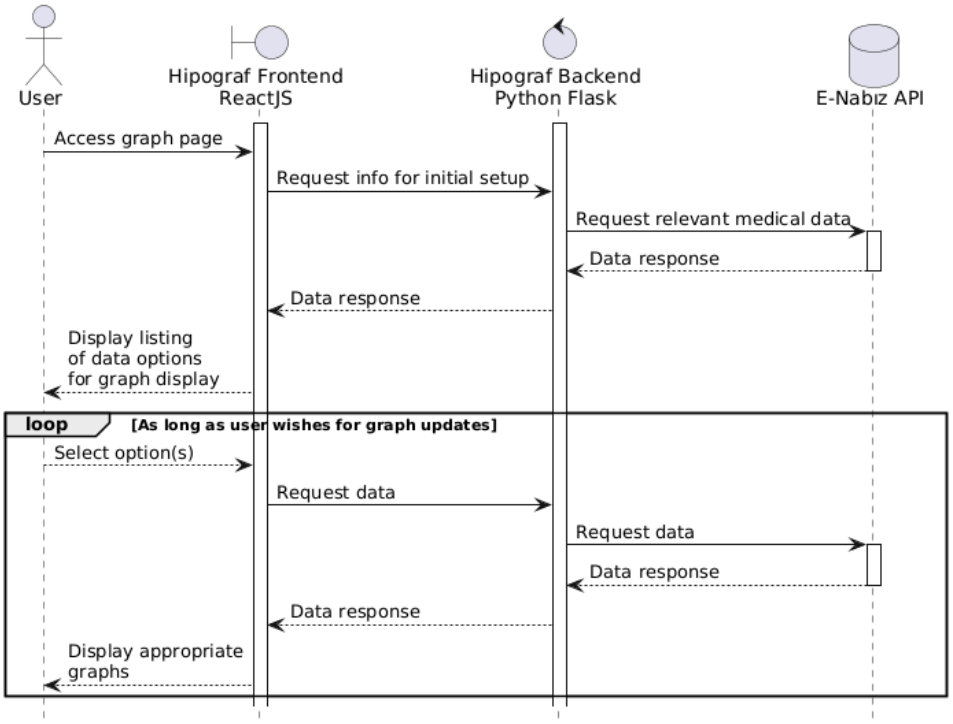

Fig. 3. Sequence Diagram for Initial Access of a Graph Page.

Graph Transformations

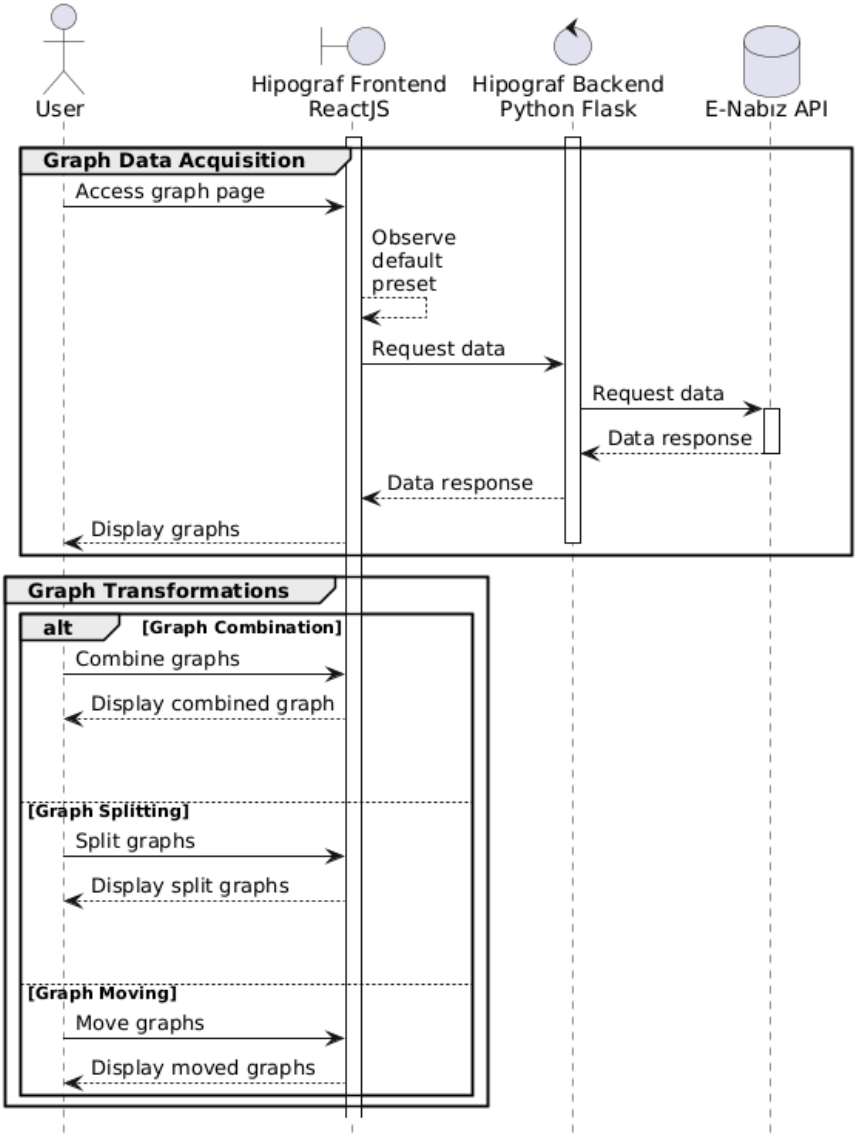

Fig. 4. Sequence Diagram Transformations on the Graph Page.

Graph Preset Saving

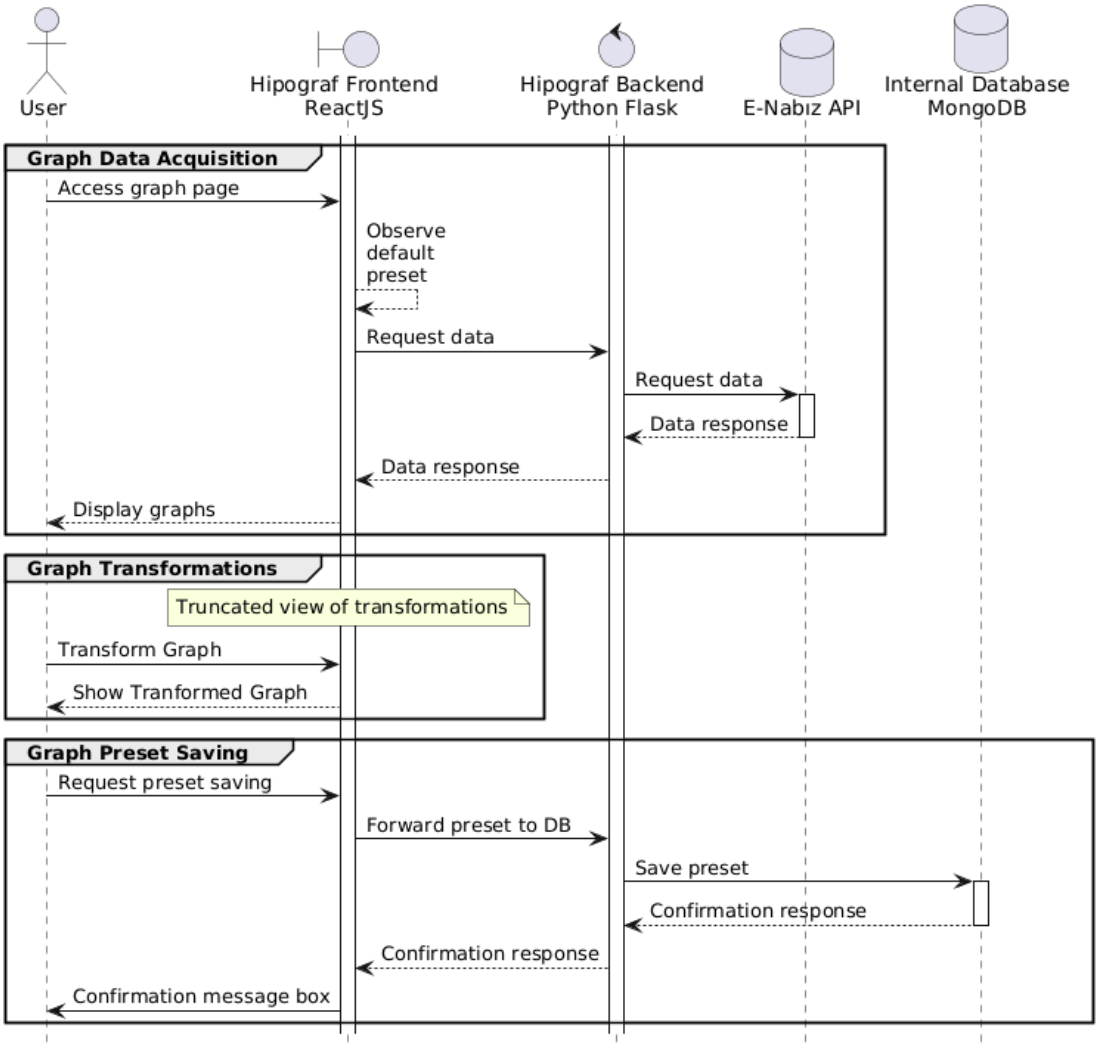

Fig. 5. Sequence Diagram Transformations Graph Preset Saving.

3.3 Activity Diagrams

A few select activity diagrams capturing core behavioral functionality will be included.

Register

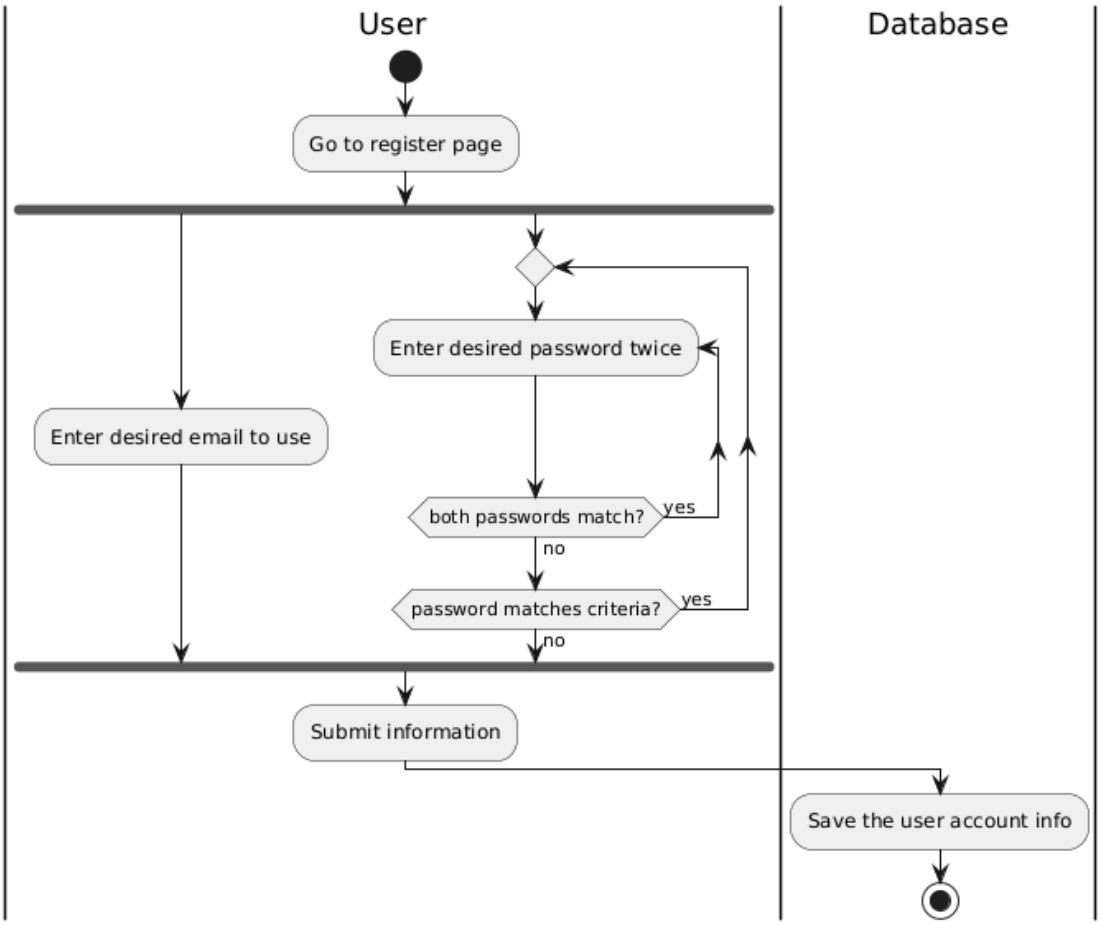

Fig. 6. Activity Diagram for Registration

Doctor Choosing a Patient

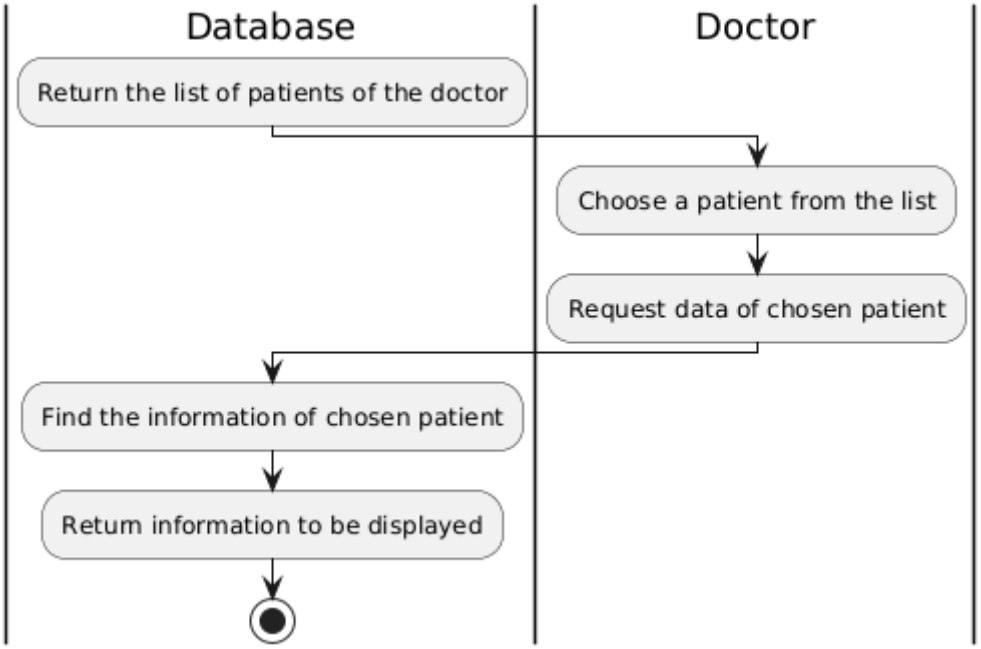

Fig. 7. Activity Diagram for Patient Selection

Visual Formatting of Timeline

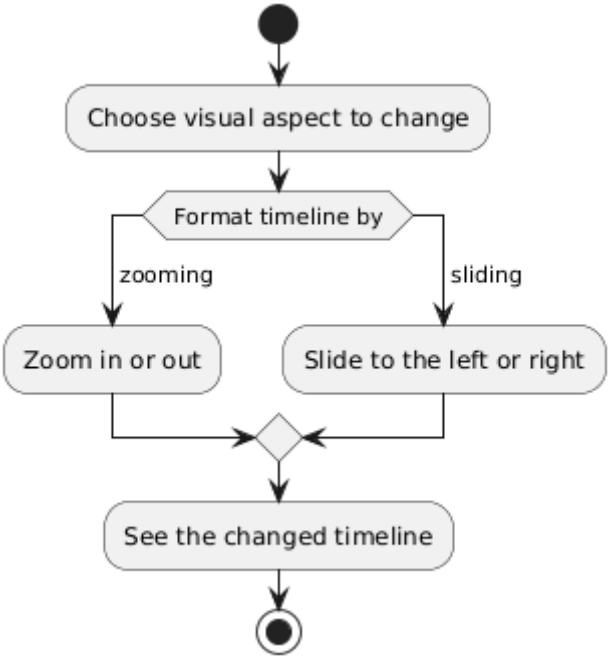

Fig. 8. Activity Diagram for Timeline Visual Formatting

Graph Merging and Splitting

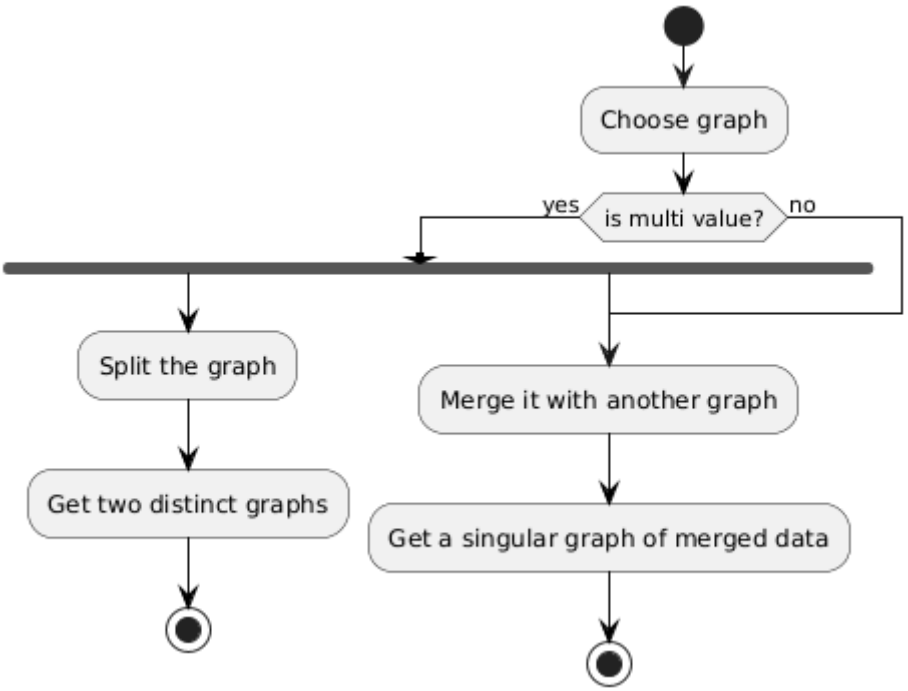

Fig. 9. Activity Diagram for Merging/Splitting

Generic List Page Usage

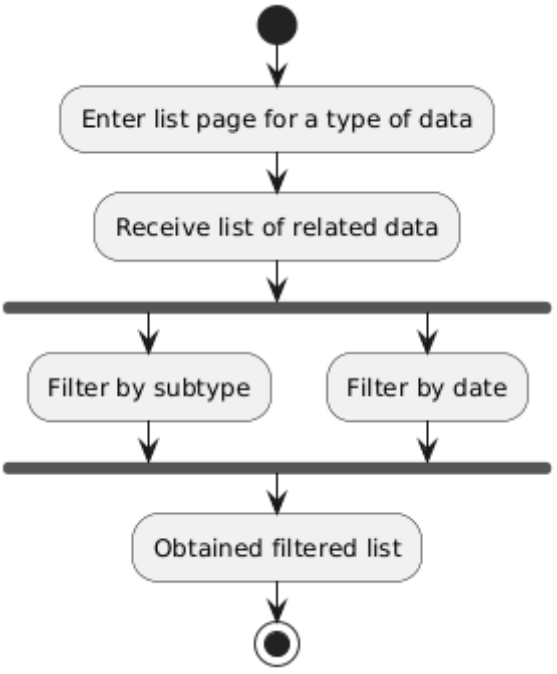

Fig. 10. Activity Diagram for List Page Usage

3.4 Object Diagram

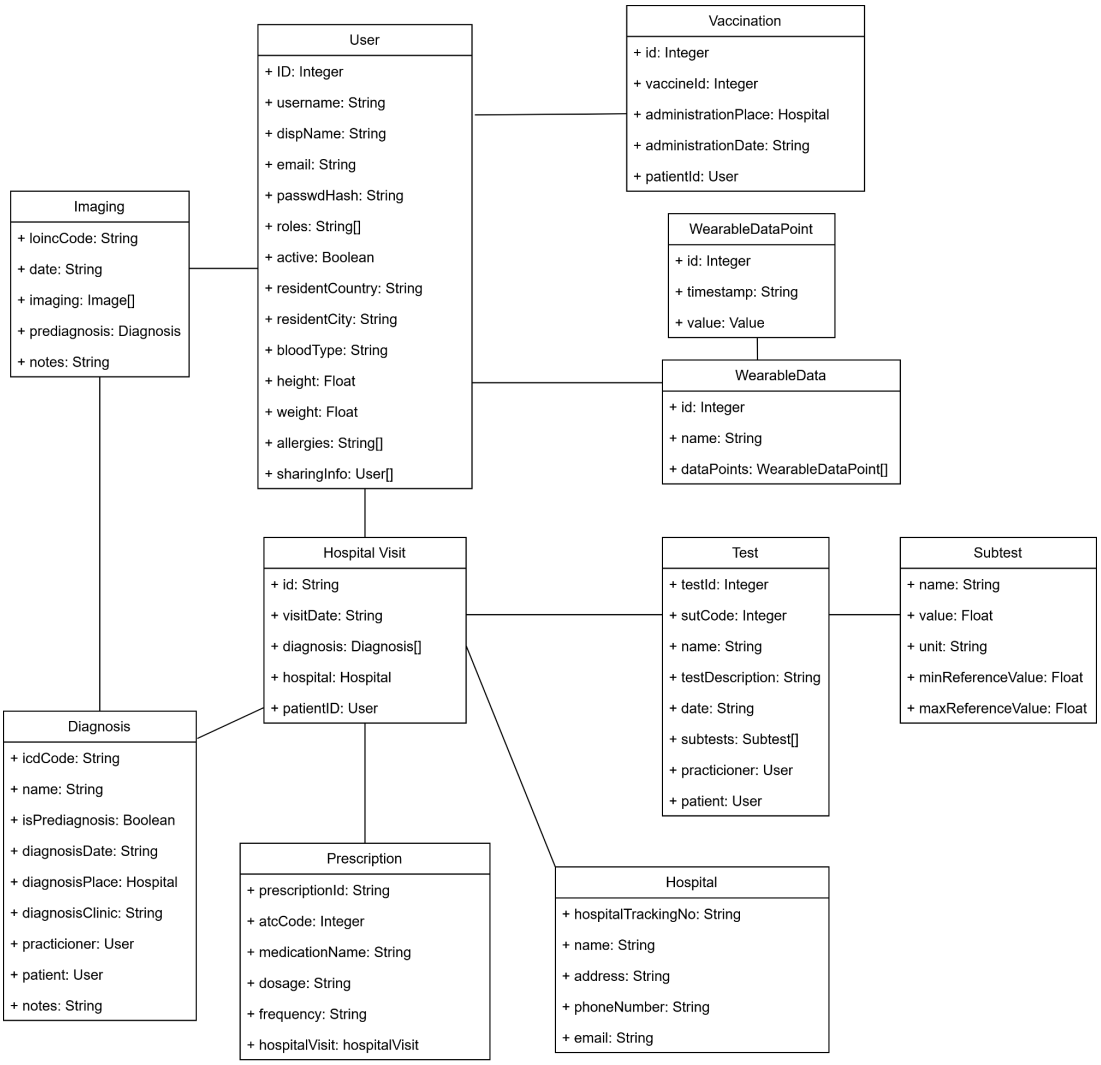

Fig. 11. Object Diagram

3.5 Class Diagram

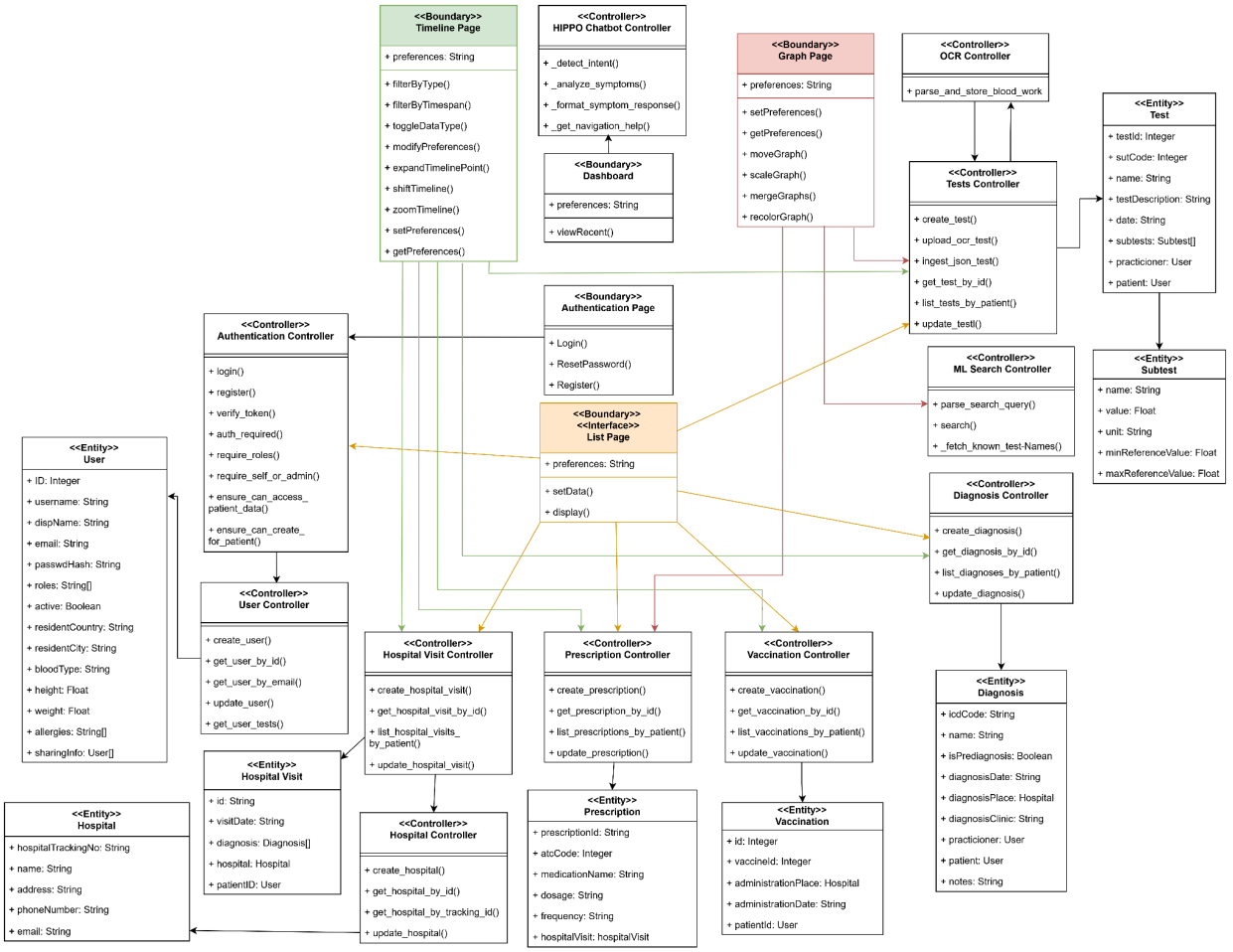

Fig. 12. Class Diagram for the Hipograf Backend

3.6 Subsystem Decomposition

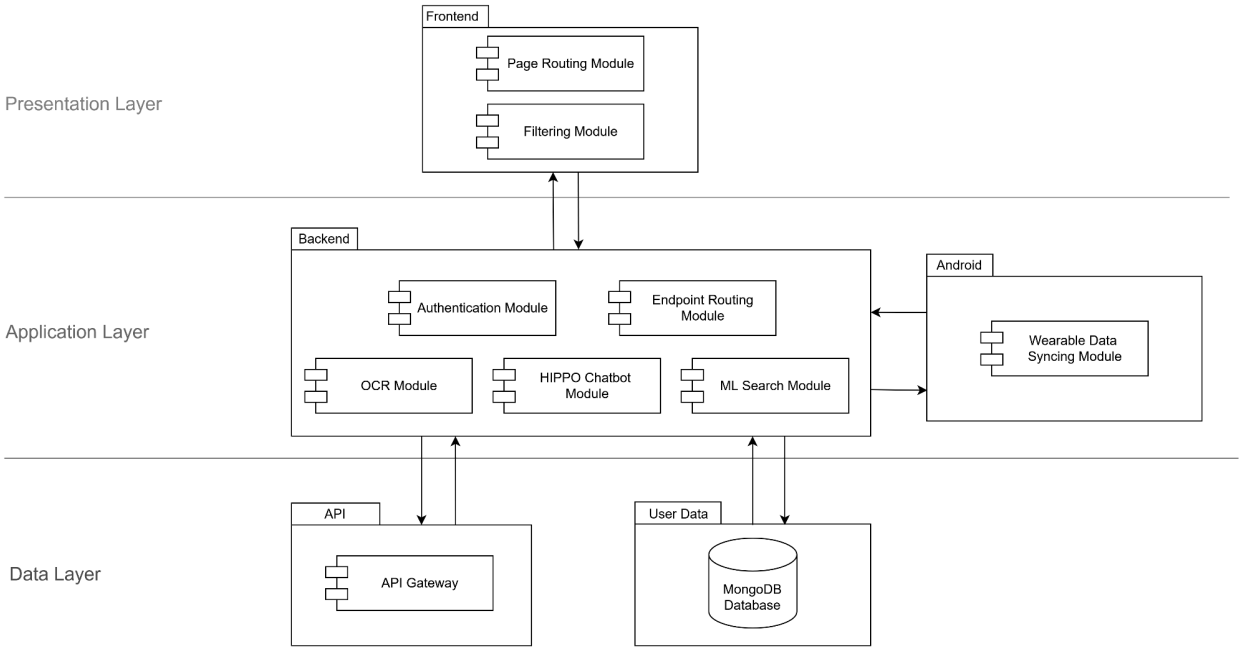

Fig. 13. Subsystem Decomposition Diagram for Hipograf

3.7 Hardware/Software Mapping

Hipograf will make use of traditional consumer computer hardware available in the medical industry in Türkiye. Its primary focus will be accessibility from any standard computer that is able to connect to the internet. To this end, it will be a web-based application that exists in the standard internet ecosystem.

Hipograf will make use of a variety of software products and packages. First and foremost, Hipograf functions by being connected to external APIs. After the user logs into their E-Nabız account and grants Hipograf access to their medical information, their relevant medical data will be retrieved from the relevant external APIs. As a web application, Hipograf is also reliant on the various software and packages used to build and host the website itself. In particular, it uses React TypeScript and Tailwind CSS. Hipograf will also be connected to a database for storing basic user information such as login credentials, data display preferences and additional user submitted information. This database will be an instance of MongoDB. The backend will link the database to the frontend. The backend of Hipograf will be built using the Flask web framework and will be written in Python.

3.8 Persistent Data Management

As Hipograf is only supposed to be a visualization solution, it is imperative that it does not modify any medical record from any system it is integrated with (i.e, any “write” operation). Its only role in relation to these systems is to fetch data (i.e, a “read” operation) and safely discard them from the backend and frontend afterwards. This data impermanence policy does not apply to application-specific data like the user-created format presets of plots. It also does not apply to anything that the user specifically decides to temporarily upload into the system.

Due to the fact that the application development must be done before integration with external APIs like e-Nabız, the application will have to store some medical data to construct the minimal viable product. This data will come from either the developer team's own medical information per their consent, or be generated data that does not belong to a real patient. After proper integration, this procedure would be avoided.

3.9 Access Control and Security

Access to Hipograf will be limited to logged-in users. All pages (and associated endpoints) in the application apart from the login, account creation and password reset will require standard web authentication. Different kinds of users will be available in the application. User roles include two normal roles, those being practitioners and patients. Practitioners and patients have similar access permissions, with the main difference being that practitioners also have a list of patients that they are responsible for.

There are several foundational choices that we made to bolster the security of Hipograf, especially during deployment. Requests to the backend API are reverse proxied by Apache web server serving the frontend. Since the requests are reverse proxied internally, the actual backend itself is not web-facing, reducing the number of external attack vectors. During the development process, Apache itself was configured to directly return a default error page and the 403 Forbidden HTTP Status Code if attempted to access outside of the Bilkent University subnet (139.179.0.0/16). Access to the remote MongoDB database storing the data was also IP limited, to the Bilkent Subnet (for local testing) and to the static IP address of the remote server containing the backend and frontend (for deployment). Another security measure was to ensure that the internal Linux users used to run the processes for the backend and the frontend did not have elevated privileges. All connections to Hipograf were secured through the TLSv1.3 protocol, with a signed Digicert certificate. Project code for the frontend and backend was sent to the remote server using the rsync protocol through a ssh tunnel with a public/private key pair generated using ssh-keygen.

4. Development/Implementation Details

4.1 Frontend

The frontend of the application is a React web application, written in Typescript. It is package-managed using npm. It is compiled using Vite. We use Recharts as our main visualization library, which acts as the backbone in several pages, such as the Plots page. The CSS styling is done through Tailwind CSS, which was chosen because of its modular nature, allowing fast development. Routing is done via React-Router. For icons used throughout the application, to save development time and effort, we chose Lucide React icons. For the 3D visualization found in the Dashboard page, we used the Three package, which is used for 3D rendering in Javascript.

4.2 Backend

The backend of the application is a Flask-based API, which does not serve any HTML/CSS/JS itself. It's used by the frontend for authentication, fetching user data, and processing API requests. We also used Flasgger, which allows for Flask to integrate with Swagger, to create easy-to-use OpenAPI compliant API documentation. The backend interacts with the database, in this case a MongoDB cluster, through the APIs provided in the PyMongo package. This connection requires an API key, which is stored in a .env file along with all the other secrets used in the application. Werkzeug is used as the primary Web Server Gateway Interface (WSGI) during development, which is switched for Gunicorn during deployment for production. Every API route featured in our backend is listed below with explanations.

Table I
Backend API Definitions (Route and Explanation)

PATCH /api/auth/change-password	Change the authenticated user's password.
POST /api/auth/login	Login with email or username and return an access token.
POST /api/auth/register	Register a new user and return an access token.
GET /api/auth/verify	Verifies a Bearer token and returns its payload.
POST /api/diagnoses	Creates a diagnosis record.
GET /api/diagnoses/<diagnosis_id>	Gets a diagnosis by ID.
PATCH /api/diagnoses/<diagnosis_id>	Updates a diagnosis record.
GET /api/diseases	Searches diseases by name or ICD code.
DELETE /api/graph-presets	Deletes all graph presets for a patient.

GET /api/graph-presets	Lists graph presets for a patient.
DELETE /api/graph-presets/<preset_id>	Deletes a graph preset by ID.
GET /api/graph-presets/<preset_id>	Gets a single graph preset by ID.
PUT /api/graph-presets/<preset_id>	Updates an existing graph preset by ID.
POST /api/hippo/chat	Sends a chat message to the HIPPO assistant.
GET /api/hippo/greeting	Gets HIPPO's greeting message.
GET /api/hospital-visits	Lists hospital visits using a patient ID query parameter.
POST /api/hospital-visits	Creates a hospital visit record.
GET /api/hospital-visits/<hospital_visit_id>	Gets a hospital visit by ID.
PATCH /api/hospital-visits/<hospital_visit_id>	Updates a hospital visit record.
GET /api/hospital-visits/by-practitioner	Lists visits for patients shared with the authenticated practitioner.
POST /api/hospitals	Creates a hospital record.
GET /api/hospitals/<hospital_id>	Gets a hospital by ID.
PATCH /api/hospitals/<hospital_id>	Updates a hospital record.
GET /api/hospitals/by-tracking-id/<tracking_id>	Get Hospital by Tracking ID Gets a hospital using its tracking ID.
GET /api/operations List	Lists operations using a patient ID query parameter.
POST /api/operations	Creates an operation record.
GET /api/operations/<operation_id>	Gets an operation by ID.
PATCH /api/operations/<operation_id>	Updates an operation record.
GET /api/prescriptions	Lists prescriptions using a patient ID query parameter.
POST /api/prescriptions	Creates a prescription record.
GET /api/prescriptions/<prescription_id>	Gets a prescription by ID.
PATCH	Updates a prescription record.

/api/prescriptions/<prescription_id>	
GET /api/data	Returns example data.
POST /api/echo	Echoes back posted data.
GET /api/hello	Returns a simple hello world response.
DELETE /api/tests	Deletes all tests for a patient.
GET /api/tests	Lists tests using a patient ID query parameter.
POST /api/tests	Creates a test record.
DELETE /api/tests/<test_id>	Deletes a single test record.
GET /api/tests/<test_id>	Gets a test by ID.
PATCH /api/tests/<test_id>	Updates a test record.
POST /api/tests/json	Saves a JSON payload as a test in the database.
POST /api/tests/ocr	Uploads a PDF and returns parsed blood-work results as JSON.
POST /api/tests/search/parse	Parses a natural-language bloodwork search query.
POST /api/users	Creates a new user from a JSON payload.
DELETE /api/users/<user_id>	Deletes a user account.
GET /api/users/<user_id>	Gets a user by ID.
PATCH /api/users/<user_id>	Updates a user's information.
GET /api/users/<user_id>/tests	Gets all tests for a user by ID.
POST /api/users/batch	Fetches multiple users by ID in one request.
GET /api/users/by-email	Gets a user by email address.
GET /api/users/by-role	Lists users filtered by role.
GET /api/vaccinations	Lists vaccinations using a patient ID query parameter.
POST /api/vaccinations	Creates a vaccination record.
GET /api/vaccinations/<vaccination_id>	Gets a vaccination by ID.
PATCH /api/vaccinations/<vaccination_id>	Updates a vaccination record.

4.3 Database

A remote MongoDB cluster is used for persistent data storage within Hipograf, accessed through the backend. We chose to avoid using locally managed separate databases during development to ensure data synchronisation. Having the data uploaded or modified by one person being reflected on everybody's Hipograf instance was important for efficiency during the development stage of the process.

4.4 Subsystem Services

This section presents a detailed overview of all the subsystems described in the subsystem diagram of the previous section, located in Section 3.6.

Presentation Layer

The presentation layer consists of the application web frontend that runs as a React TypeScript application that is built to be served as a static set of HTML, CSS and JavaScript files by the Apache httpd web server.

Page Routing Module

The frontend application uses React Router for the local page routing, which is handled internally and remains fully functional when the application frontend is built for deployment.

Filtering Module

Data fetched from the backend API endpoints largely arrive in their unfiltered form. Most of the data filtration takes place on the frontend, minimizing the amount of data processing that is required from the backend, instead leveraging the user's own browser (and by extension, computer) to perform these operations.

Application Layer

The application layer of our project consists of a web backend written in Python's Flask microframework. It is hosted with the Gunicorn Web Server Gateway Interface (WSGI) server, with the API calls to the backend being reverse proxied from the frontend's web server to avoid having a directly web-facing backend server.

Authentication Module

Hipograf's user accounts use standard and secure web authentication using localStorage based Bearer tokens and endpoints (both frontend and backend) secured behind authentication. The User Controller and Authentication Controller (visible in the Class Diagram in Section 3.5) are closely intertwined, and many backend endpoints are only visible/particular to certain user roles (further elaborated on within Section 3.9).

Endpoint Routing Module

Most of the backend endpoints are served as part of a standard data access API to the frontend. These endpoints, most requiring authentication, serve JSON-formatted data to only

the frontend of the Hipograf. It is on the same domain, existing behind the /api path. This is to prevent any potential Cross-Origin Request attacks.

OCR Module

The backend contains a detailed module for performing automated parsing of uploaded PDFs. Existing blood work reports, such as the ones available on e-Nabız, are submitted to the system. The module then proceeds to automatically identify the tested elements and their resulting values and couples it with the Hipograf system, whereupon the user can view this data on the List, Timeline, and Graph pages.

HIPPO Module

Another backend module is the HIPPO LLM-based chatbot component. On the frontend side, HIPPO is designated with a floating button that persists on the bottom left of the screen for authenticated web pages (i.e, everything visible after the user logs into the system). Users can ask HIPPO about the navigational structure of Hipograf, along with instructions on how to use a particular page's mechanisms.

ML Search Module

The final backend module is responsible for the advanced search mechanism present in the Graph page. This search mechanism allows the user to query, in natural language, for test parameters and date ranges and automatically retrieve plots associated with those parameters in the specified date ranges. The module uses an LLM to parse natural language and extract the relevant information from the provided query.

Wearable Data Syncing Module

Healthcare data available on a patient's mobile device can be synced with Hipograf using the module, provided they download the Android application. Once they sign in and authorize their account, they are able to 'sync' this data by interfacing with the Samsung Health API, resulting in them being able to see and interact with this data on the Hipograf backend. They have the option of cancelling this connection at any time, either from the normal Hipograf web backend or the Android application.

Data Layer

The Data Layer is split up into two main components. These are the external data fetched from external APIs, and also the MongoDB Database used to store certain components of application data (as detailed in Section 4.3).

API Gateway

The application models and data format has been designed with various pre-existing external APIs in mind, including platforms like e-Nabız. This is to ensure that external data can be efficiently fetched from these APIs and used in the Hipograf ecosystem.

MongoDB Database

The MongoDB Database is a MongoDB cluster used to store certain application and user data. The credentials required for authentication, and user customization/medical data used during development of the application are stored in this database. Connection with this database is made via the `pymongo` package, and is done through a secure authenticated channel. The backend includes several repository classes responsible for robustly performing operations on this database for each data model present.

4.5 Deployment

For deployment, we used a rented virtual dedicated server (VDS) with a static Internet Protocol Version 4 (IPv4) address assigned to it. We had previously purchased the domain name `hipograf.com.tr` that we've used for other matters related to the project. For instance, we've used the `info` subdomain available at https://info.hipograf.com.tr to share information about our project, including all of the reports we uploaded. During the development period, we used the `dev` subdomain in the form https://dev.hipograf.com.tr for testing the deployment of the website. The fully deployed website will be available on https://hipograf.com.tr.

The main web server serving the project is the Apache httpd web server. The Apache web server serves the application frontend directly from port 443 (default for HTTPS connections). HTTP connections are automatically upgraded to HTTPS by the web server configuration, disallowing unencrypted connections to the website.

The Apache virtual host configuration reverse proxies all requests to the domain starting with /api in their URI to the locally running backend, meaning that the backend itself is not directly web-facing. The backend also communicates with the remote MongoDB client since the IP of the VDS is whitelisted (more information about the security considerations can be found in Section 3.9)

5. Test Cases and Results

Test Case 1 - Patient Selection

Test ID	T2506-TC1	Category	Functional
Objective	This test case is to verify that the patient selection system is working as expected, allowing the application to fetch data correctly after the user is authenticated.		

Steps	1. A practitioner registers to the Hipograf System. 2. The practitioner accesses the list of patients they have access to. 3. The practitioner selects a patient from the list. 4. The practitioner brings up the patient's prescription page and observes the prescriptions.
Expected Outcome	The prescription page only displays the prescriptions of the selected patient.
User Stories Covered	US1 - Registration US10 - List Patients US11 - Select Patient US19 - List Prescriptions
Test Status - Date	PASS - 28.04.2026

Test Case 2 - Credential Changing

Test ID	T2506-TC2	Category	Functional
Objective	This test case is to verify elements of Hipograf's data persistence.		
Steps	1. The user logs in to the Hipograf system. 2. The user updates both their username and their password that was registered to their Hipograf account. 3. The user logs out and logs back in with their new password.		
Expected Outcome	The username and password have been changed correctly, allowing for the user to access the application after changing them.		
User Stories Covered	US2 - Login US6 - Account Information Update US5 - Password Change US3 - Logout		
Test Status - Date	PASS - 28.04.2026		

Test Case 3 - Blood Test Uploading

Test ID	T2506-TC3	Category	Functional
Objective	This test case is to verify that the upload blood test feature works as intended.		
Steps	1. Having forgotten their password, the user resets their Hipograf password and logs in to the system. 2. The user uploads several of their official blood tests as a PDF, which is then read and processed by the system. 3. The user checks the blood list page to ensure that the data has been uploaded correctly. 4. Finally, the user navigates to the timeline page to get a time-based perspective of their blood test data.		
Expected Outcome	The blood test data should be correctly displayed on the timeline, along with whatever other relevant data is fetched from the external API.		
User Stories Covered	US2 - Login US13 - Upload Blood Test US12 - List Blood Test US24 - Scroll Timeline		
Test Status - Date	PASS - 27.04.2026		

Test Case 4 - Chatbot Usage

Test ID	T2506-TC4	Category	Functional
Objective	This test case is to verify that a brand new user can use the AI chatbot, HIPPO, to successfully navigate the application.		
Steps	1. A new user registers to Hipograf. 2. The user accesses the HIPPO chatbot on the dashboard and asks it about the application. 3. The user asks HIPPO where it can find information about past radiological images and is directed to the radiological image page.		
Expected Outcome	Correct navigational responses by HIPPO should be provided.		
User Stories Covered	US1 - Registration US9 - Chat with HIPPO US16 - List Radiological Images		
Test Status - Date	PASS - 27.04.2026		

Test Case 5 - Account Deletion

Test ID	T2506-TC5	Category	Functional
Objective	This test case is to verify that account deletion works correctly.		
Steps	1. The user logs into the Hipograf system. 2. The user demands that all of their preference data pertaining to timeline and graph selections be deleted. 3. The user then proceeds to delete their account entirely. 4. Having been logged out automatically, the user attempts to reset their password.		
Expected Outcome	The password reset should fail (it is an expected failure, the ideal outcome).		
User Stories Covered	US2 - Login US8 - Clear Preferences Data US4 - Account Deletion US7 - Reset Password		
Test Status - Date	PASS - 28.04.2026		

Test Case 6 - Graph Preset Saving and Loading

Test ID	T2506-TC6	Category	Functional
Objective	This test case is to verify that previously saved graph presets are able to load correctly.		
Steps	1. The user logs into the Hipograf system. 2. The user navigates to the graph page and creates a simple graph setup of particular wearable data that exists in the system. 3. The user recolors one of the graphs and saves their presets, making sure that the 4. The user exits the application and logs back in. 5. The user navigates to the graph page and loads their previously saved graph preferences.		

Expected Outcome	The graphs that are loaded should match the previously saved graph.
User Stories Covered	US2 - Login US20 - List Wearable Data US28 - Create Graph US31 - Recolor Graph US34 - Save Preferences US35 - Load Preferences US3 - Logout
Test Status - Date	PASS - 28.04.2026

Test Case 7 - Timeline Preset Saving and Loading

Test ID	T2506-TC7	Category	Functional
Objective	This test case is to verify that previously saved timeline presets are able to load correctly.		
Steps	1. The user logs into the Hipograf application. 2. The user navigates to the timeline page and filters the visible events until only diagnoses in the past 6 months can be seen. 3. The user clicks on a particular diagnosis visible on the timeline for detail and ensures that they match the ones given on the diagnosis list page. 4. The user saves these timeline filtration preferences. 5. The user exits the application and logs back in. 6. The user navigates to the timeline page and loads their previously saved timeline preferences.		
Expected Outcome	The timeline that is loaded should match the previously saved timeline configuration settings.		
User Stories Covered	US2 - Login US14 - List Diagnosis US21 - Filter Event Category US26 - Save Preferences US27 - Load Preferences US25 - Obtain Event Information		
Test Status - Date	PASS - 24.04.2026		

Test Case 8 - List Page Consistency

Test ID	T2506-TC8	Category	Functional
Objective	This test case is to verify that information across various list pages are consistent with one another.		
Steps	1. The user logs into the application. 2. The user checks the hospital visit page for visits in the past year. 3. The user proceeds to check the operations and vaccinations pages in the same timeframe constraint.		
Expected Outcome	The vaccinations and operations listed should only have occurred on dates that have had hospital visits listed.		
User Stories Covered	US2 - Login US15 - List Hospital Visit US17 - List Operations US18 - List Vaccinations		
Test Status - Date	PASS - 24.04.2026		

Test Case 9 - Graph In-Depth Features

Test ID	T2506-TC9	Category	Functional
Objective	This test case is to verify various facets of the extended features of the graph page.		
Steps	1. The user navigates to the graph page. 2. The user creates a graph manually. 3. The user creates a graph using the natural language search functionality. 4. The user resizes both graphs and chooses different timeframes for the data they are presenting. 5. The user combines the two graphs into one. 6. The user splits the graphs again into their original forms.		

Expected Outcome	All graph operations conducted should give correct and appropriate results.
User Stories Covered	US28 - Create Graph US29 - Search for Graph US30 - Edit Graph US32 - Combine Graph US33 - Split Graph
Test Status - Date	PASS - 28.04.2026

Test Case 10 - Timeline In-Depth Features

Test ID	T2506-TC10	Category	Functional
Objective	This test case is to verify various facets of the extended features of the timeline page.		
Steps	1. The user logs into the application and navigates to the timeline page. 2. The user chooses a particular timeframe of the last year for the timeline. 3. The user changes the current timeframe to include only the last 6 months. 4. The user scrolls through the timeline, viewing segments before the selected timeframe. 5. The user filters the timeline so that only hospital visits will remain. 6. The user clicks on one of the hospital visit nodes to obtain a brief description about the event.		
Expected Outcome	All timeline operations conducted should give correct and appropriate results.		
User Stories Covered	US21 - Filter Event Category US22 - Choose Timeframe US23 - Edit Timeframe US24 - Scroll Timeline US25 - Obtain Event Information		
Test Status - Date	PASS - 28.04.2026		

Test Case 11 - Reliability

Test ID	T2506-TC11	Category	Non-Functional
Objective	Verifying the reliability of various components of the Hipograf architecture.		
Steps	1. Monitor the status of the frontend server for a period of three weeks using systemd's journalctl module. 2. Repeat the previous step for the backend server. 3. Check the statistics for the uptime of the MongoDB database from its administrative dashboard.		
Expected Outcome	An uptime of over 95% should be seen across the whole suite of systems tested.		
Test Status - Date	PASS - 27.04.2026		

Test Case 12 - Performance

Test ID	T2506-TC11	Category	Non-Functional
Objective	Verifying application performance using the online functionality provided by tools like Google PageSpeed Insights [4].		
Steps	1. Using Google PageSpeed Insights, measure the time taken to access the login page (after having emptied the browser's page cache) 2. Measure the time it takes for a majority of HTML objects to be loaded into the Dashboard using the same tool 3. Continue Step 2 for major application pages		
Expected Outcome	The planned maximum time to load the main content of Hipograf should be less than 2 seconds, and the input delay between subsequent operations should be less than 200ms.		
Test Status - Date	PASS - 26.04.2026		

Test Case 13 - Security

Test ID	T2506-TC11	Category	Non-Functional
Objective	Verifying that various aspects of Hipograf's application-wide security expectations on all three layers (Presentation /Application/Data) meet existing standards [5].		
Steps	1. Check through the web browser used to connect to Hipograf that the connection is an HTTPS connection 2. On the same medium, check the validity of the TLS certificate. 3. Check the MongoDB database to ensure that user passwords in the User model field 'passwdHash' are hashed correctly, as specified. 4. Ensure that the password length/content requirement checks are in-place in both the backend and the frontend.		
Expected Outcome	All manually conducted baseline security checks should be successful.		
Test Status - Date	PASS - 26.04.2026		

Test Case 14 - Usability

Test ID	T2506-TC11	Category	Non-Functional
Objective	Verifying the WCAG 2.1 accessibility guidelines for the application webpages concerning limited input vectors in scenarios where a mouse is not accessible or unusable [1].		
Steps	1. Navigate to the Login Page and verify that the elements of the page are all accessible by sole use of the keyboard. 2. Navigate to the Dashboard and perform the same verification. 3. Navigate to the Timeline Page and perform the same verification. 4. Navigate to the Graph Page and perform the same verification.		

	5. Navigate to the List Pages and perform the same verification.
Expected Outcome	All core functionalities of the application are accessible only by using the keyboard (without the mouse).
Test Status - Date	PASS - 26.04.2026

6. Maintenance Plan and Details

6.1 Regular Updates & Maintenance

The frontend of the project has its packages maintained through `npm`. This makes it very easy to make sure its packages are up-to-date using the command "`npm update`". Additionally, `npm` allows developers to check whether their project's packages have any known vulnerabilities through the command "`npm audit`". Similarly, the backend of the project is package-managed using `uv`. Packages can be updated using the "`uv sync --update`" command, and audited for security vulnerabilities using "`uv audit`".

The project should be regularly audited for out-of-date packages using these commands to ensure no external security vulnerabilities affect the project. In the case of packages that are deprecated, appropriate replacements for these packages should be found and integrated into the project. This is done to ensure the stability and security of the project as unmaintained packages are prone to introduce bugs and security flaws.

In the event that a user finds unintended behavior in the application, they should be able to disclose this information to the development team for its resolution. For this, a bug reporting platform such as GitHub Issues or a dedicated BugZilla instance should be set up.

6.2 Incident Plans

As Hipograf deals with sensitive personal information, incident plans must be prepared in advance.

6.2.1 Database Breach

In the event that unauthorized access to the database occurs, the following steps must be taken in quick succession:

1. Depending on the severity of the breach, the MongoDB cluster hosting the data of the application must be Paused or Terminated. Pausing the database terminates all existing connections and disallows further connections. Terminating the cluster results in the cluster being deleted, which allows for protection against more severe breaches, such as if a MongoDB database access API key is exposed.
2. All APIs that can access medical data in the backend must be closed.

3. All users with admin access to the backend or the MongoDB cluster must have their current credentials revoked and regenerated.
4. The root cause for the breach must be found and documented.
5. The steps taken must be well documented and disclosed to all affected parties within 72 hours from the first detection.

6.2.2 Breaking Change In E-Nabız API

Hipograf is able to integrate with the E-Nabız API through PDF files generated based on its API, which is defined out of this project's control. In the event that the format of these files change, the backend logic dealing with these files must also be updated.

7. Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

7.1.1. Standards

Throughout the course of the project, we have followed various standards in multiple different categories. These include engineering standards, legal standards and medical standards. Engineering standards specifically concern both the planning/design and the implementation of the project. Legal standards concern data privacy. Medical standards concern industry conventions for the labelling of various medical terms. Relevant definitions and the meaning of abbreviations can be found in Section 9.

Requirements Engineering and Documentation

For requirements engineering, we have sought standardization documents jointly published by the ISO, IEC and IEEE. These documents include ISO/IEC/IEEE 29148:2018 and ISO/IEC/IEEE 12207:2017. The first document gives an overview of requirements engineering for multiple different project types, including SRS which is the one that we are concerned with. The second document expands on this, also being cited in the first document, providing more detail on how various fields are expected to be filled out in the software case [6], [7].

Design Modelling

We refer to the UML 2.5.1 modelling standards published by the OMG for the system's design, encapsulated with the suite of structural and behavioral diagrams that are defined with UML including use case diagrams, activity diagrams, state diagrams and class diagrams [8].

Web Accessibility

In order to ensure that Hipograf remains accessible to as many people as possible, we intend to follow the WCAG 2.1, which are detailed web accessibility standards published by the W3C [1].

Privacy of Sensitive Data

In order to ensure that we are complying with national laws when dealing with patient data, we are referring to KVKK guidelines in dealing with sensitive data in a legally acceptable manner [9].

Disease Classification

The ICD is a detailed compendium of medical ailments and afflictions that is maintained by the WHO. Each disease or condition is provided with its own unique alphanumeric identifier code. Similarly classed or otherwise related conditions are given similar identifiers [10].

Disease Classification - Turkish Standard

The Turkish Ministry of Health keeps an online and publically accessible reference database for medical encodings and codes used in Türkiye, known as the 'Sağlık Kodlama Referans Sunucusu' (SKRS). One of these includes a translated and reformatted set of the aforementioned ICD-10 codes [11].

Treatment/Medication/Equipment Standards

The SKRS also contains other standards, including the SUT codes. This standard defines similar alphanumeric codes for various treatments, medication and equipment stationed at medical institutions around Türkiye [11].

7.1.2. Constraints

Rationale For Web Application

Hipograf will be primarily built for use on a standard personal use computer. This constraint exists because the devices accessing the application in a medical facility or similar setting are most likely to be a desktop or laptop computer, which entails comparable screen sizes and identical peripherals. Despite this expectation, we have not opted for a native desktop application. The issue is that a desktop application would make the application impossible to use from a mobile device without a direct port. We desired to keep the option of accessing Hipograf from mobile devices available even if the touchscreen reduces usability.

Security

Hipograf is designed to operate on private information in a state where it is computationally infeasible for an adversarial actor to gain malicious access. This requires that the system has proper authentication and authorization in place, which has been discussed in more detail in Section 3.9.

Public Health

As it is concerned with the health of the public, any changes to the general health of the national public has a great likelihood of affecting Hipograf. Health-related crises are likely to increase the load on Hipograf, requiring more performant servers.

Public Safety

Hipograf is not meaningfully influenced by the existing state of public safety. In the event that public safety is gravely threatened for unrelated reasons, Hipograf usage will be similarly affected.

Public Welfare

If a local populace's buying power was to be affected, it is possible that Hipograf would have to adapt to suit the new user base's computer specifications by further decreasing performance requirements.

Global Factors

Hipograf is only concerned with the E-Nabız system in Türkiye.

Cultural Factors

Hipograf is not influenced by cultural factors. It is designed to be used by medical entities that are themselves tied to state operations. A shift in culture will not change the usage of Hipograf.

Social Factors

Hipograf is not subject to social factors.

Environmental Factors

Hipograf is not subject to environmental factors.

Economic Factors

Hipograf will be a free-to-use software application. Its source code is under a permissive license and thus will be free-to-distribute.

7.1.3. Relevant Tables

Table II
Factors that can affect analysis and design

	Effect level	Effect
Public Health	High	More robust performance requirements
Public Safety	Very Low	Could reduce usage
Public Welfare	Low	More performant code
Global Factors	N/A	None
Cultural Factors	N/A	None
Social Factors	N/A	None
Environmental Factors	N/A	None

Economic Factors	N/A	None

Table III
Application Level Risk and Alternatives

Risk	Likelihood	Effect on the project	B Plan Summary
Integration Difficulties with E-Nabız	Medium	Patients can not automatically use their medical data stored in E-Nabız.	Release the software as a standalone application, allowing users to import their medical data semi-automatically.
Data Misrepresentation	Low	While a list is plain and unambiguous, visual representation gives a user much stronger intuition for pattern detection. There exists a risk that a mistake in how a graph plots data creates confusion in the representation of this data.	Re-evaluate the way in which the existing data is represented through the visual components and modify these components accordingly.
Data Leak	Low	The preferences of the users can be leaked, leading to reduced trust in the user base.	Create an incident report detailing the events that led to the database leakage, and improve the security based on the results of the report.

7.2 Ethics and Professional Responsibilities

Hipograf will operate on medical information of its users. Medical information is deeply personal and private. This makes it absolutely necessary that this data should be kept from malicious actors. Furthermore, the operation of the application must be compliant with KVKK [9].

Another point to consider is that this system will be used by medical practitioners to diagnose patients. This means that this application is partially responsible for all the diagnoses it's used in, including the errors in judgment that result from Hipograf's errors. For this reason, Hipograf must be implemented in a way where the information shown is ensured to be correct, and in the event of an error, the medical practitioner should be explicitly and clearly warned.

7.3 Teamwork Details

7.3.1. Contributing and functioning effectively on the team to establish goals, plan tasks, and meet objectives

Team Hipograf followed a hybrid development schedule wherein different team members work on different parts of the project at different times. This had two benefits: Everyone is familiar with every facet of the development process, and as an extension of this fact, members can be assigned to work on more urgent sections without significant delay.

The team was in regular contact with the project supervisor, where we usually planned during and after these meetings. The assignments of tasks to group members were done through GitHub Issues. This method allowed us to asynchronously work on tasks while keeping each other updated through issue tracking.

7.3.2. Helping creating a collaborative and inclusive environment

Creating a collaborative and inclusive environment is a very delicate process because everybody needs to be willing to do everything they can do to help, as the unwillingness of a single team member has the potential to hamper the efforts of the rest. In other words, everyone has to contribute equally. Thanks to our existing intra-group synergy, strong personal bonds and dedication to nurturing this environment amongst one another, everyone in Team Hipograf contributed to this equally. It wouldn't be possible without everyone giving it their all. As such, there is not much to say for this section other than the fact that we were able to create this collaborative and inclusive environment.

7.3.3. Taking lead role and sharing leadership on the team

Team Hipograf followed a somewhat decentralized leadership structure where people took up leadership positions as temporary leaders of hybrid work compartments like pieces of the frontend, backend, etc. A good overview of each member's primary leadership roles throughout the development of the project were as follows:

- Ramiz Arda Ünal took the lead role in frontend development and design.
- Salih Furkan Göktaş took the lead role in report writing and system modelling.
- Orhun Güder took the lead role in the OCR module backend subsystem.
- Artun Berke Gül took the lead role in the application layer interactions.
- Şükrü Eren Gökırmak took the lead role in general backend development.

It is important to note that these are primarily the areas where team members specifically lead others, not parts they worked on exclusively. Everybody contributed to every portion of the project.

7.3.4. Meeting objectives

Members of Team Hipograf were all active in efforts of meeting objectives decided during meetings in the team. To this end, each member of the team has been active in completing their assigned tasks to meet deadlines set, and helped others within the team to complete theirs. The team has achieved almost every deliverable objective set within the work

packages that were previously specified in the Analysis and Requirements Report of CS491, as detailed below.

Frontend Implementation, designated WP1, required for the team to integrate the existing and future UI mockups into Hipograf, and improve the existing frontend systems. Along with the outlined deliverables, which were to implement the Prescriptions page and the Wearables page, we have added additional pages and functionality, such as the Calendar and the Agenda components to the Timeline page. The existing frontend has also been revamped since then, improving the general look and the response time. Thus, we believe this Work Package has been completed in its entirety.

Backend Implementation, designated WP2, was focused on implementing the backend endpoints for frontend usage. It featured the tasks of converting the object diagrams into a MongoDB database schema, and using that schema to create a Flask API with the purpose of connecting the frontend to the backend. We have indeed converted the existing object diagrams into a database schema with minimal adjustments, and have implemented the Flask API for use in the frontend. We conclude that this Work Package has also been completed in full.

WP3 was related to Testing. Its objective was to implement a testing and verification process to ensure all facets of the application were maximally functioning. The tests outlined in the deliverables of WP3 consisted of Reliability Tests, Security Tests, Performance Tests, and Usability Tests. After setting these goals, we have successfully implemented and passed tests in these categories, and have documented them thoroughly in Section 5 of this report. Our testing covers each of the test categories in these deliverables. In addition, we have performed functional tests not outlined on WP3. Thus, we have completed every task in this Work Package.

Non-Required Feature Implementation, denoted WP4, was a Work Package we had designated previously as a potential one that may be looked at after the core functionalities of Hipograf were in place. Inside WP4 were 3 tasks, Implementation of the Sandbox Module, OCR Blood-Test Support, and Mobile Support. Of these three, we have implemented OCR Blood-Test Support with a different vision. We have looked at some PDF OCR solutions, and have found that it was unreliable, often getting values wrong in a way that did not provide the user any convenience. Thus, we have seen fit to change the blood test upload pipeline. Currently, users can submit a PDF of their blood test into Hipograf to get back their test results in a table they can freely edit and upload into Hipograf. This solution is much more reliable. For the implementation of the Sandbox Module, we have, again, seen fit to change our implementation. Instead of a Sandbox module that allowed users many functionalities in one page, we have instead distributed the functionalities into several other existing modules. We have changed our solution in order to make Hipograf easier to use. And finally, for the Mobile Support, we have decided that mobile would be a subpar choice to use Hipograf, as the small screen size would disallow the user from taking advantage of functionality like multiple plot visualization, and timeline visualization that is core to Hipograf. Thus, we have considered every task within WP4.

The E-Nabız API integration process, designated WP5, required for the project to be completed as a prerequisite due to the government regulations about said integration.

Additionally, it would mean that we would have to divert a significant amount of effort to make this integration possible. This additional effort would include security audits, correspondence with the authorities, and additional paperwork. This would slow the development process. At this stage of development, we decided to not move forward with this work package for the scope of the CS491/2 courses. Instead, we made sure to structure our backend data in line with what the E-Nabız API is able to provide, so that a possible integration in the future would be as smooth as possible.

7.4 New Knowledge Acquired and Applied

Developing a medical data visualization platform required us learning a modern, full-stack ecosystem. Throughout this project, our team acquired knowledge on a variety of industry-standard tools, frameworks, and libraries. Our primary learning approach was self-directed learning, mostly on official documentation and practical examples from the official repositories of these tools.

To construct the user interface, we utilized npm for package management and React to build a component-based architecture. We learned to manage application state and component lifecycles by studying React's official documentation. To create a pleasant UI for all users, we used Tailwind CSS. By referencing Tailwind's utility-first documentation, we were able to rapidly prototype and refine the styling without writing custom CSS files. The core of our application, creating integrated medical record viewing tools was achieved using Recharts.

On the backend, we managed our Python environment and dependencies using uv, which we adopted for its ease-of-use based on current Python packaging standards. We developed our core backend architecture using Flask. Using Flask's documentation and their selected examples, we learned how to structure a backend capable of serving sensitive data to the frontend. To handle the storage of medical records, we relied on MongoDB. We chose MongoDB as it would be safer to rely on a tested database provider. Reviewing MongoDB's documentation taught us how to design document schemas that could accurately and efficiently store medical records. For language model prompting, we learned to use the DSPy framework. Rather than relying on manual prompt construction we used DSPy's declarative approach. We acquired new knowledge in defining program signatures, which allowed us to build reliable data processing pipelines.

To give a particular example of us acquiring and applying new knowledge, the following example can be given: During development, there was a point where we thought of processing the uploaded PDF files using Tesseract, an optical character recognition (OCR) engine. After testing this approach, we found that it was not as reliable as expected. Afterwards, we realized that since the files we usually expected are not scanned. This prompted us to parse the PDF file directly and only using OCR as a fallback solution.

8. Conclusion and Future Work

Currently, the development of Hipograf is complete. In the development process, we have managed to complete most of the future tasks we have defined for Hipograf in the Analysis and Requirements Report, as elaborated on in detail in Section 7.3.4. The goals that we have set for ourselves during CS491 have been largely met and sufficiently tested to be production-ready. We have especially taken care to enjoy that the project is holistically connected and that every added feature makes sense and integrates well with the rest of the application.

Our future work is in-line with what we have described in Section 7.3.4. The most significant avenues of future research involve integrating Hipograf with the e-Nabız system, which would involve significant research and interaction with the relevant parties over a long period of time. Being out of scope for CS492, it can be considered for the continued development of Hipograf.

9. Glossary

Definitions

e-Nabız: A web-based application developed by the Türkiye Cumhuriyeti Sağlık Bakanlığı to give citizens a unified access portal for digitized personal medical data [12].

HIPPO: The LLM-powered chatbot deployed within Hipograf.

Abbreviations

IEC: Abbreviation for ‘International Electrotechnical Commission’, a standardization organization specialising in electrical engineering hardware [6].

ISO: Abbreviation for ‘International Organization for Standardization’, an international standardization organization that covers scientific, engineering and more general fields alike [6].

IEEE: Abbreviation for ‘Institute of Electrical and Electronics Engineers’, an organization aiming to further advancement and research in electrical engineering and related disciplines [6].

SRS: Abbreviation for ‘Software Requirement Standards’, a set of standards jointly published by the IEC/ISO/IEEE that states their expectations for requirements documentation of software projects [6].

W3C: Abbreviation for ‘World Wide Web Consortium’, a standardization organization specialising in the usage and guidelines of the internet [1].

WCAG: Abbreviation for ‘Web Content Accessibility Guidelines’, a series of recommendations published by the W3C to further the accessibility of the internet, especially for disability-based access [1].

OMG: Abbreviation for ‘Object Management Group’, a standardization organization specialising in data modelling [8].

UML: Abbreviation for ‘Unified Modelling Language’, a set of modelling standards that capture the structural and behavioural design of software systems [8].

WHO: Abbreviation for ‘World Health Organization’, an agency of the United Nations that concerns itself with the general livelihood, alongside physical and mental health of people worldwide [10].

ICD: Abbreviation for ‘International Classification of Diseases’, a globally accessible and standardized list of diseases monitored and maintained by the WHO [10].

ATC: Abbreviation for ‘Anatomical Therapeutic Chemical’, a classification system for medically prescribed drugs and medication maintained and updated by the WHO [10].

SKRS: Abbreviation for ‘Sağlık Kodlama Referans Sunucusu’, an online reference resource maintained by the Türkiye Cumhuriyeti Sağlık Bakanlığı that contains medical classification databases used in national hospitals and care facilities [11].

SUT: Abbreviation for ‘Sağlık Uygulama Tebliği’, a communique regularly published and updated by the Sosyal Güvenlik Kurumu detailing the pricing of medical services and equipment [13].

KVKK: Abbreviation for ‘Kişisel Verilerin Korunması Kanunu’, a set of legal limitations on the processing of information considered to be relating to a person’s private affairs [9].

10. References

[1] “Web Content Accessibility Guidelines (WCAG) 2.1”. https://www.w3.org/TR/2025/REC-WCAG21-20250506/. [Accessed: Nov 27, 2025].

[2] “The MIT License” https://opensource.org/license/mit. [Accessed: Dec 17, 2025].

[3] “Apache License, Version 2.0”. https://www.apache.org/licenses/LICENSE-2.0. [Accessed: Dec 17, 2025].

[4] “About PageSpeed Insights”. https://developers.google.com/speed/docs/insights. [Accessed: Nov 27, 2025].

[5] ”About Certbot”. https://certbot.eff.org/pages/about. [Accessed: Nov 27, 2025].

[6] “IEEE/ISO/IEC 29148-2018”. https://standards.ieee.org/ieee/29148/6937/. [Accessed: Nov 27, 2025].

[7] “IEEE/ISO/IEC 12207-2017”. https://standards.ieee.org/ieee/12207/5672/. [Accessed: Nov 27, 2025].

[8] “Unified Modelling Language, v.2.5.1”. https://www.omg.org/spec/UML/2.5.1/PDF. [Accessed: Nov 27, 2025].

[9] “Mevzuat Bilgi Sistemi”. https://www.mevzuat.gov.tr/mevzuat?MevzuatNo=6698&MevzuatTur=1&MevzuatTertip=5. [Accessed: Nov 27, 2025].

[10] “ICD-10 Version:2019”. https://icd.who.int/browse10/2019/en. [Accessed: Nov 27, 2025].

[11] “Sağlık Kodlama Referans Sunucusu”. https://skrs.saglik.gov.tr/. [Accessed: Nov 27, 2025].

[12] “e-Nabız V.2.1 Kullanım Kılavuzu 2025”. https://enabiz.gov.tr/document/User_Manual.pdf. [Accessed: Nov 27, 2025].

[13] “SUT NEDİR?”. https://old.peduro.org.tr/sut_nedir.php. [Accessed: Nov 27, 2025].