OASIS Teledentistry

CO227 - Second Year Project Report

Project Supervisors:
Prof. Roshan G. Ragel
Dr. Isuru Navinne

Group 2 Members:
E/20/069 – Dilshan D.M.T
E/20/189 – Karunarathne K.N.P.
E/20/381 – Somathilaka A.M.T.
E/20/385 – Sriyarathna D.H.
E/20/420 – Wanasinghe J.K.

CONTENT

1. Introduction

The OASIS Teledentistry Mobile App project builds on the foundation laid by the E17 batch, who previously developed a web application as part of the OASIS research initiative. This mobile application aims to expand and enhance the functionalities of the original system, providing dentists with a more accessible platform for teleconsultations, patient entry management, and secure medical data storage.

The primary goal of this project is to enable dentists to perform consultations remotely while maintaining a comprehensive and user-friendly system for managing patient records. By transitioning to a mobile platform, the project seeks to meet the growing demand for on-the-go healthcare solutions. Developed with a Flutter-Dart front end and a Spring Boot backend, the app ensures reliability, scalability, and robust data security.

This report outlines the continuation of the OASIS research efforts, detailing the functional and non-functional requirements, key use cases, testing methodologies, and feedback received. By integrating advanced mobile technology, the OASIS Teledentistry Mobile App takes a significant step towards revolutionizing remote dental care and ensuring seamless usability for dental professionals.

2. Functional Requirements

1. User Defined Roles

- Admins are enabled to create roles and decide their authentication levels.
- After the sign up a user admin can assign the role which is created or already available.

2. Categorized Authentication Levels

- Authentication levels are based on user requirements and each category contains logical relationships within each other access.
- super admin can change the authentication level authorization.

3.Creation of a Patient Profile

- An authenticated user can add a patient to the system(if patient is not in the system)
- This contains a consent form of the patient.
- Patient is uniquely identified using its NIC.
- Created patient profile is only visible to the user who created the patient

4. Adding a Patient Query

- Users can add new patient queries in an existing patient profile.
- Users can add images of the Oral cavity(which are suspected to have cancerous infection), findings and relevant reports.
- Users can select the behaviors of habits already available or user can add new habits.

5. Sharing Patient Query

- Users are able to add multiple reviewers to the query of a patient created.
- Users can see the patients which they were assigned as a reviewer.
- Shearing a patient query will only share the exact query of the patient, not its profile.

6. Simple Secure Login and Signup Process

- Users should be able to undergo simple signup and Login process.
- Super admin should be able to approve the sign in requests.

3. Non-Functional Requirements

1. Security

a. Data Privacy

To ensure user data privacy and secure communication:

- **Encryption**: User credentials are securely stored using robust encryption techniques to protect sensitive information.
- **HTTPS Communication**: All data exchanges between the client and server are conducted over HTTPS, ensuring that communications are encrypted and safeguarded against interception.

b. Role-Based Access Control (RBAC)

The application employs role-based access control to restrict unauthorized actions and ensure appropriate permissions:

- **JWT Tokens for Authentication**: JSON Web Tokens (JWTs) are used as access tokens for verifying user sessions.
- **Refresh Token Mechanism**: A dedicated model was implemented to issue refresh tokens, enabling the generation of new access tokens once the current one expires.
- **Google Login**: Integration of Google Login simplifies user authentication and enhances security by leveraging OAuth 2.0 protocols.
- **Permission-Based Access**: The backend enforces permission-based access control by validating user permissions for each request. This ensures only authorized users can perform specific actions.
- **Dynamic Role Management**: A system to create roles and assign permissions dynamically provides flexibility in managing access levels for customers, shop owners, and admins.

2. Performance

a. Optimized Search and Filtering

To ensure quick and efficient retrieval of data:

- **Location-Based Queries**: Implemented optimized location-based queries to retrieve nearby products efficiently, enhancing user experience.

- **Efficient Indexing**: Products are indexed in the database by categories and shop locations, enabling fast filtering and search functionality.

b. Scalability

The system was designed to handle increasing loads without compromising performance:

- **RESTful APIs**: REST APIs were implemented to process a large number of requests efficiently, ensuring smooth operation under heavy traffic.

c. Response Time

The application maintains acceptable response times for key features such as product searches and notifications:

- **Code Quality Assurance**: Regularly used SonarLint to analyze the codebase for inefficiencies and ensure adherence to best practices.
- **Response Time Debugging**: Continuously debugged response times to meet acceptable performance thresholds.
- **Loose Coupling Architecture**: Followed a loose coupling approach in code design to minimize duplication and improve maintainability, indirectly boosting performance.
- **Database Optimization**: Utilized a NoSQL database to improve performance and scalability, especially for handling dynamic and large-scale data queries.

3. Usability

a. User-Friendly Interface

The application prioritizes ease of use for both customers and shop owners:

- **Intuitive Design**: The frontend interface is designed to be simple and user-friendly, ensuring that users can navigate effortlessly.
- **Clear Navigation**: Features such as browsing products and managing advertisements are labeled and organized for a seamless user experience.
- **One-Tap Login**: Integration of Google Login enables users to log in with a single tap, simplifying the authentication process.

b. Accessibility

Efforts were made to ensure accessibility for a diverse range of users:

- **Responsive Design**: The application is optimized for various screen sizes, providing a consistent and seamless experience on desktops, tablets, and mobile devices.
- **Teleconsultation Feature**: A dedicated teleconsultation entry point has been integrated to enhance usability, allowing users to access additional services conveniently.

c. Future Enhancements

- User Manual: A comprehensive user manual is planned to guide users in navigating the application and utilizing its features effectively.

4. Reliability

a. System Availability

The system ensures continuous availability and resilience to failures:

- Refresh Token Mechanism: Implemented a refresh token system to seamlessly reload expired sessions, enhancing user experience and maintaining reliability.

b. Error Handling

A robust error-handling framework is in place to assist with troubleshooting and improve usability:

- **Detailed Error DTOs**: Separate Data Transfer Objects (DTOs) were created to handle different types of errors. This approach provides clear and specific feedback to users, such as distinguishing between invalid inputs and network-related issues.
- **Comprehensive Logging**: The application generates detailed error logs, making it easier for developers to identify and resolve issues efficiently.
- **User-Friendly Error Messages**: Clear and actionable error messages help users understand what went wrong and how to address it.

5. Maintainability

a. Modular Architecture

The application follows a modular design to simplify updates and maintenance:

- **Backend Layer Separation**: The backend is organized into distinct layers:
  - **Module Layer**: Encapsulates specific functionalities for better modularity.
  - **Service Layer**: Manages business logic, ensuring separation of concerns.
  - **Controller Layer**: Handles requests and responses, promoting clean code practices.
    This structured approach allows for easy updates and debugging, enhancing overall maintainability.
- **Independent Components**: RESTful APIs enable independent development and updates to the front end, back end, and database without affecting other components.

b. Documentation

Comprehensive documentation ensures smoother maintenance and onboarding for future developers:

- **Swagger Documentation**: Detailed API documentation generated with Swagger provides a clear overview of backend endpoints, facilitating easier understanding and debugging.
- **Readable Code**: Code snippets are written with readability and clarity in mind, making them easier to understand and maintain.
- **Postman Testing**: Backend endpoints were rigorously tested using Postman, ensuring that they function as expected. This approach provides a reliable reference for developers during maintenance

4. Use Case Descriptions

1. User Sign Up

The sign-up process enables users, including Admins, Doctors, and Consultants, to create an account. Users authenticate using Google Authentication, which validates their basic details like name, SLMC, and email. When a user fills in those and requests to sign up, the admin gets an email including the user details. Then, the admin can accept and assign the user a role or decline the request.

2. User Sign In

Upon successful account creation or approval, a JWT token is generated and sent to the user for authentication. If validation fails, the system displays an appropriate error message, guiding the user to correct the issue. All user data is securely stored in the database, and role-based permissions determine access to app features.

3. Adding a Patient

The Add a Patient feature allows doctors to register new patients into the system. To add a patient, the doctor navigates to the "Add Patient" section and fills in the required details, including the patient's full name, age, gender, and contact information. Optional fields such as medical history can also be provided.

Once the doctor submits the form, the system validates the input to ensure all required details are complete and checks whether the patient already exists in the database. If the patient is new, the system saves their details in the database and generates a unique Patient ID to link all future entries and images to the patient.

If validation fails, the system displays an appropriate error message, prompting the doctor to correct any issues. In cases where a duplicate patient is detected, the system notifies the doctor and provides options to view or update existing records. Upon successful addition, the doctor can create detailed entries for the patient and upload images of oral cavities to these entries. All patient data is securely stored and easily accessible through the app for future reference.

4. Adding an Entry to a Patient Profile

The Add an Entry feature allows doctors to update a patient's profile with new information. To add an entry, the doctor first searches for the patient using their NIC (National Identity Card) number. Once the patient's profile is retrieved, the doctor navigates to the "Add Entry" section. Here, they can input details about the patient's condition, observations, treatment plans, or any other relevant notes.

The doctor also has the option to upload images of the patient’s oral cavity to accompany the entry. After filling in the necessary information, the doctor submits the entry. The system validates the input and associates the entry with the patient's profile using their unique Patient ID.

Upon successful addition, the updated entry is securely stored in the database and becomes available for future reference or sharing with consultants. If any issues arise, such as an invalid NIC or incomplete information, the system provides feedback to correct the errors. This functionality ensures that patient records are comprehensive and easily accessible for continued care.

5. Sharing Patients’ Entries

The Sharing Patient Entries feature allows doctors to securely share specific patient records with consultants. The doctor searches for the patient using their NIC, selects the desired entries, and chooses a consultant to share them with. The system notifies the consultant, granting them access to view the shared entries and associated images.

Access is restricted to ensure privacy, and consultants cannot edit the shared records. This feature enables streamlined collaboration for improved patient care.

5. Test Cases

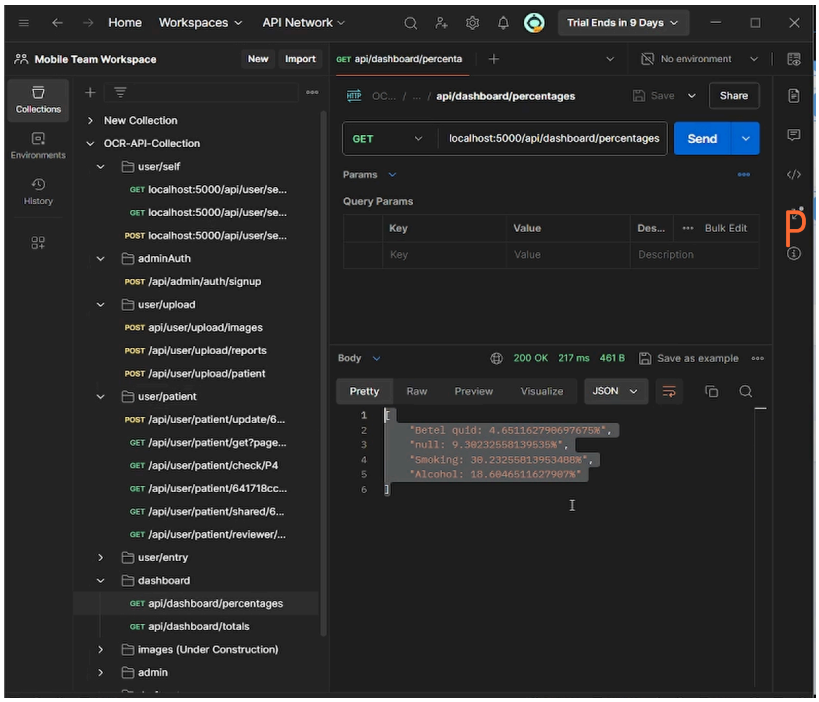
Testing is a critical phase in the software development lifecycle, aimed at ensuring the system meets both its functional and non-functional requirements. In the OASIS Teledentistry project, we conducted rigorous testing to validate the system's reliability, usability, and performance. This process helped us identify and rectify defects, ensuring that the final product delivered a seamless and user-friendly experience. By focusing on key test cases, we ensured that the core functionalities of the application operated as intended under various scenarios, paving the way for a robust and efficient solution.

Testings we conducted are as follows:

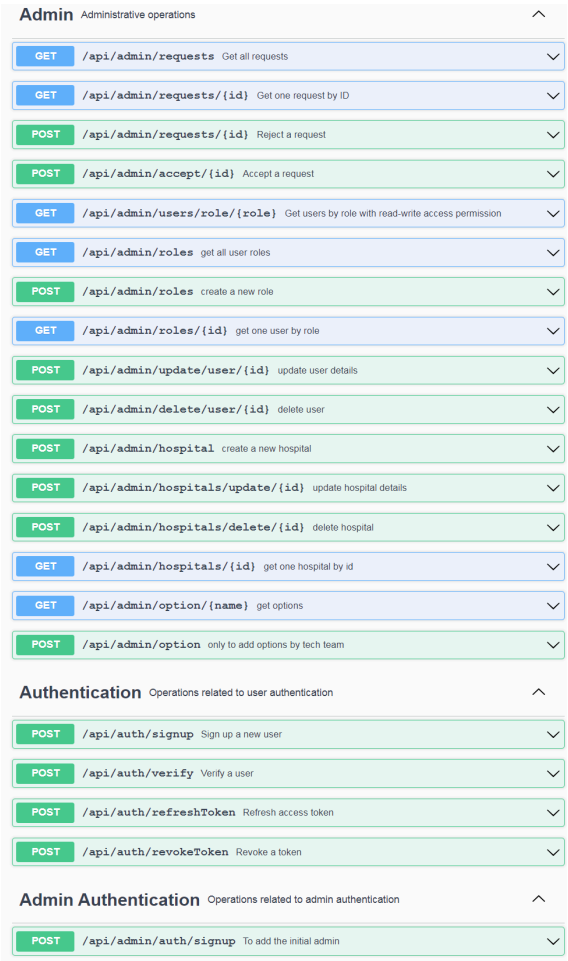
1. Manual Testing
2. Database Test with Simulated Data
3. Unit Testing (JUnit Testing)
4. Test the complete and integrated entire system as a whole.

1. Manual Testing

Manual testing was conducted upon the backend using tools such as Postman and the SWAGGER API UI. These tools ensured that each API call functioned correctly, returned the expected data types, and executed the corresponding backend operations accurately.



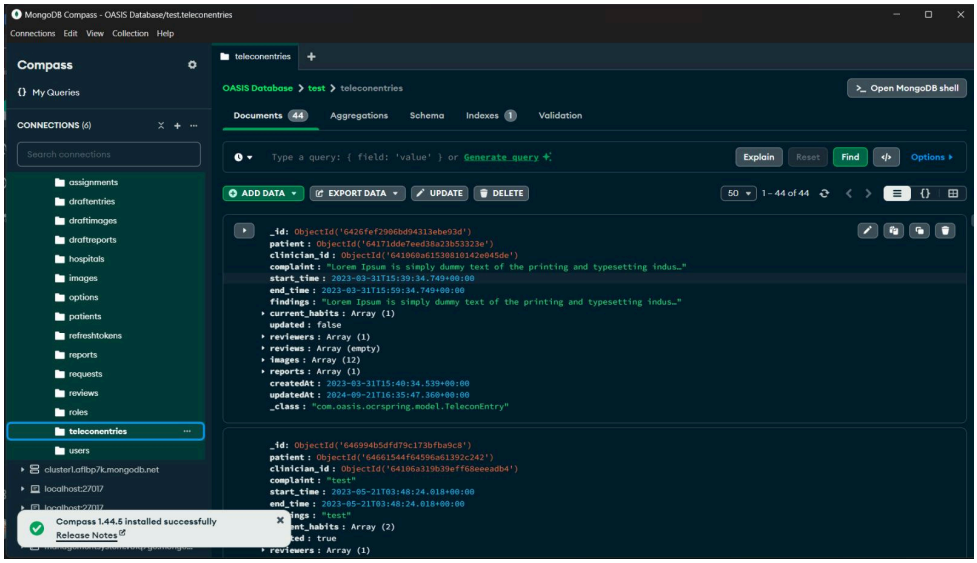
Testing a dashboard API call using postman



Part of the SWAGGER UI used for testing API Calls

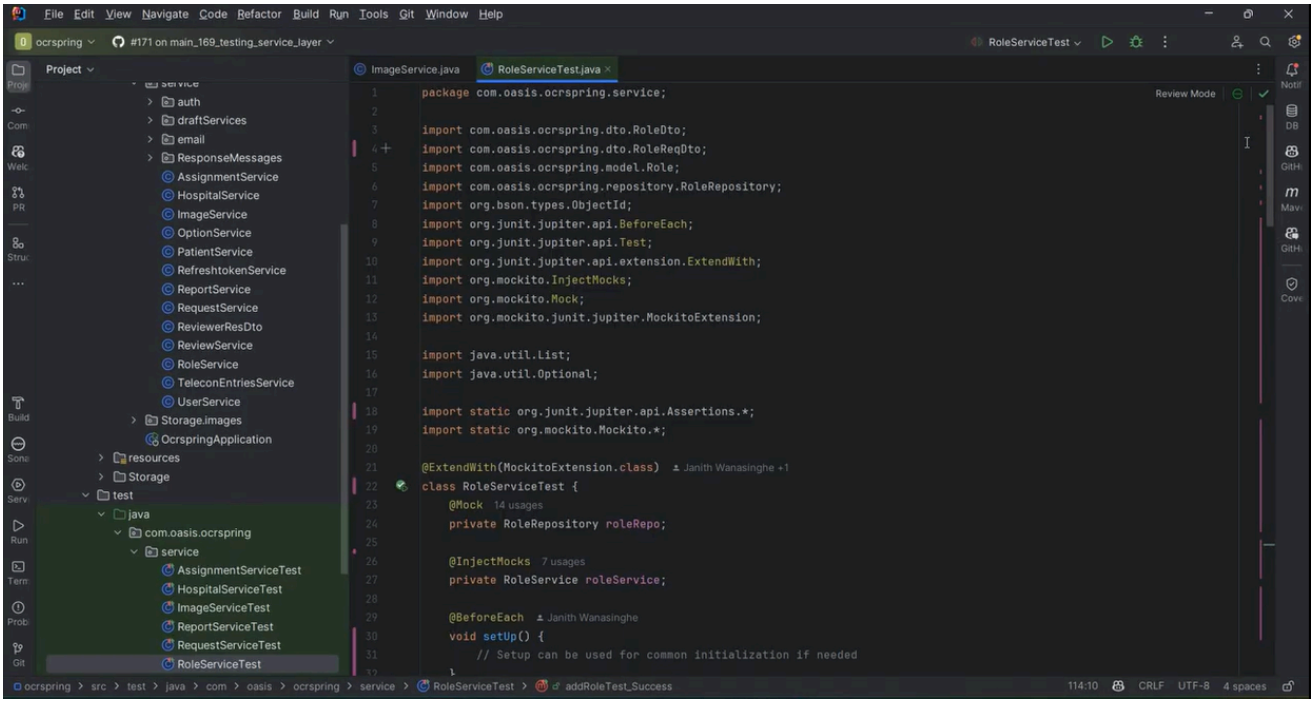
2. Database Test with Simulated Data

Database was tested with simulated data to see whether it is functioning as expected.

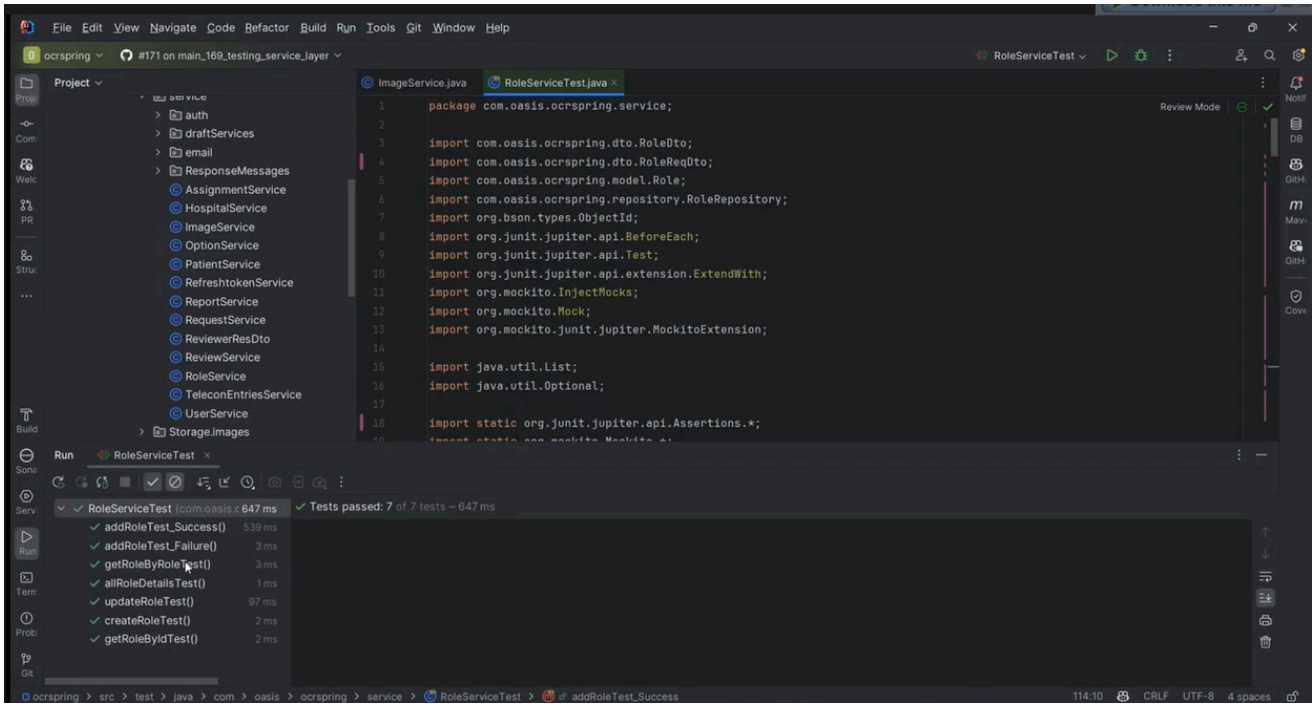


3. Unit Testing

JUnit testing was employed to validate the functionality of the service and controller layers. This testing ensured that individual components of the application operated as intended, with proper handling of inputs and outputs, and accurate interaction between the layers.

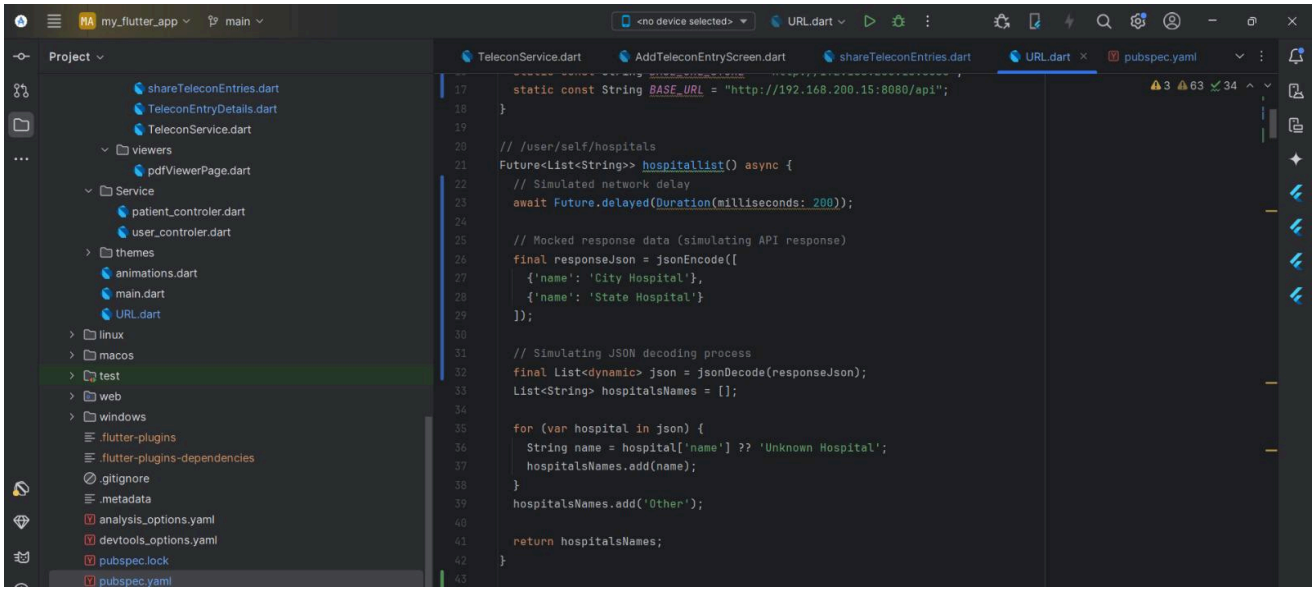


J-UNIT Testing for Service Layer (RoleService)

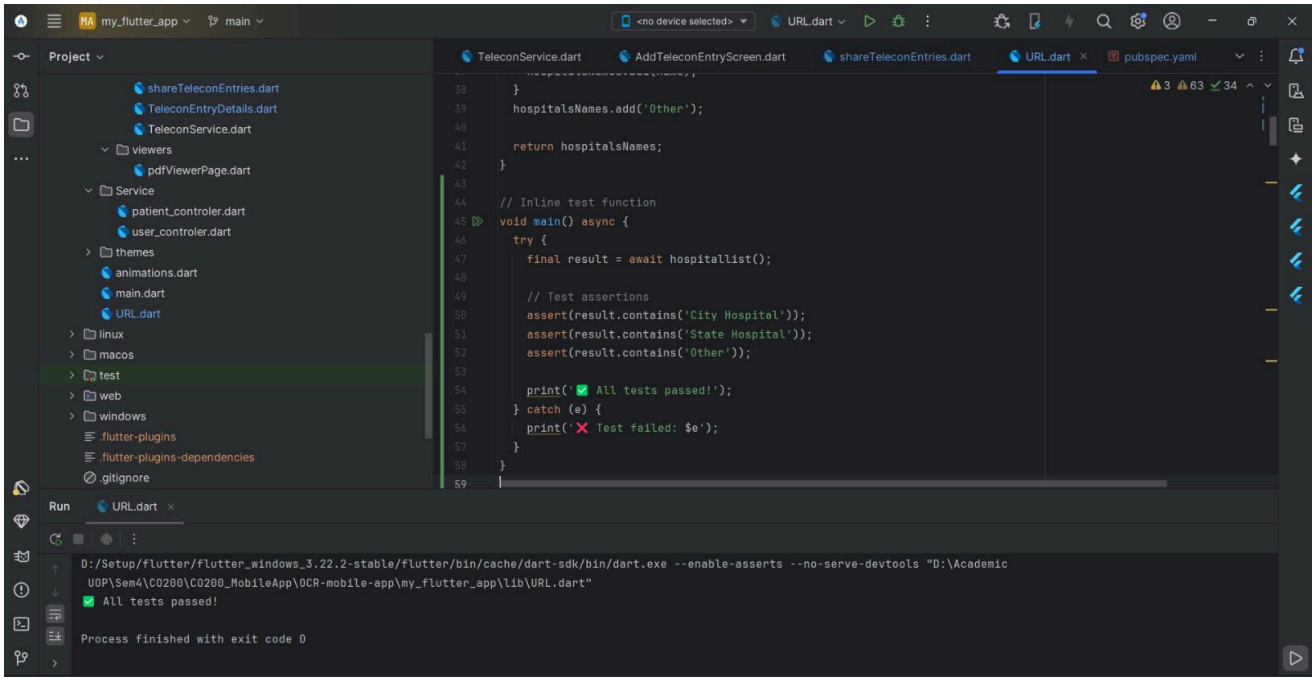


Role Service Test Being Performed

Front End Unit Tests were also conducted;



UNIT Testing for Front End



UNIT Testing for Front End Being Successful

4. Test the complete and integrated entire system as a whole

After integrating the front-end and back-end components, the complete system was tested as a whole. This phase involved both the supervisor and the team, who rigorously evaluated the application to ensure it delivered the required functionality and met the project objectives.

6. Project Management and Version Control

- For version control, the team used GitHub to efficiently manage the codebase, facilitate collaboration, and monitor project progress.
- The repositories for the "OASIS Teledentistry App" are available at the following links:

  Frontend Repository , Backend Repository.

  These repositories include the complete source code, documentation, and commit history, showcasing the project's development workflow.

  The following outlines our approach to project management and version control.

Project Management Approach

Our project followed the Agile methodology, specifically using the Scrum framework. We organized our development work into 2-week sprints, allowing for iterative progress and continuous feedback. This approach provided flexibility, improved collaboration, and ensured timely delivery of features.

Sprint Structure and Meetings

At the end of each sprint, we held review meetings with our mentors. These sessions allowed us to:

- **Evaluate progress**: Discuss tasks completed during the sprint.
- **Identify pending work**: Address tasks that weren't finished and understand the reasons.
- **Plan for the next sprint**: Prioritize upcoming tasks based on urgency and mentor feedback.

Mentor advice during these meetings was crucial in guiding our work, resolving issues, and ensuring alignment with project goals.

Task Management and Scrum Board

We utilized a **Scrum board** to organize and track tasks, ensuring clear visibility and avoiding confusion:

- Task Flow:
  - **Backlog:** All tasks were first created as issues and stored in the backlog.
  - **Ready:** When a task became a priority, we moved it to the "Ready" column.
  - **In Progress:** Tasks were transferred here once work began.
  - **Done:** Upon completion, tasks were moved to "Done.

This systematic approach kept the workflow organized and ensured that all team members understood task statuses at a glance.

Task Prioritization

Task prioritization was determined during sprint planning meetings with mentors and through team discussions. Urgent tasks received higher priority, ensuring that critical features or issues were addressed promptly.

Version Control Strategy

We used GitHub for version control, following a structured workflow:

- **Branching Strategy**:
  Each task had its own dedicated branch. This approach maintained isolation of features, reducing the risk of conflicts in the main branch.
- **Pull Requests (PRs)**:
  Upon task completion, the changes were submitted through pull requests. The person reviewing and accepting the PR resolved any conflicts, ensuring code quality and consistency.
- **Merging**:
  After approval, branches were merged into the main branch, maintaining a clean and stable codebase.

Conflict Resolution

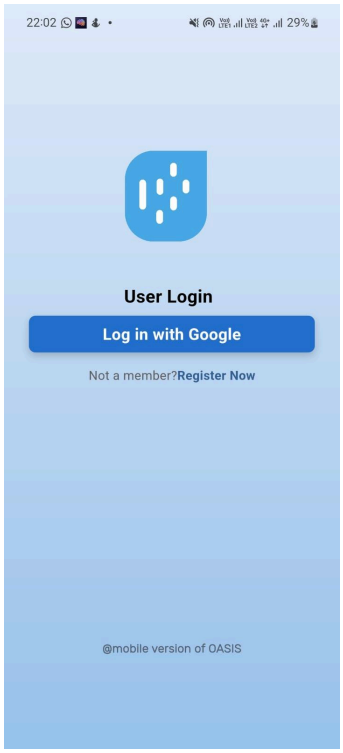
Conflicts arising during merges were handled by the team member responsible for reviewing the PR, ensuring that changes integrated smoothly without disrupting ongoing work.

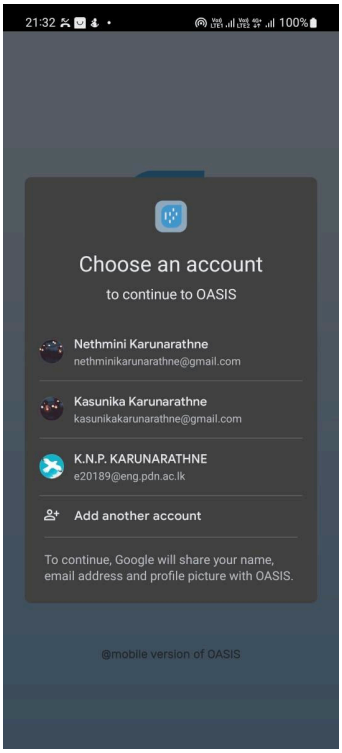
7. User Manual

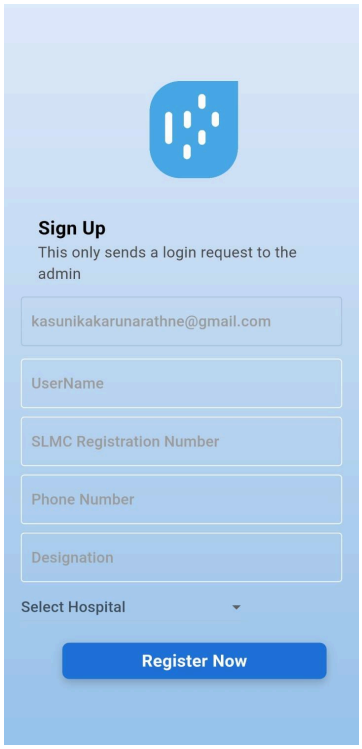
How to get Started…

01. Creating an Account:

1. Open the OASIS app.
2. Tap on the “Register Now” link displayed in the first image.
3. Sign in to your google account using Google Sign In.
4. You will be directed to the “Register Now” page.
5. Fill in the required details in the register now page.
6. Tap on the “Register Now” button.
7. You will be guided back to the login page.
8. Tap on the “Login with Google” button to login after the registration request is verified by the super admin .

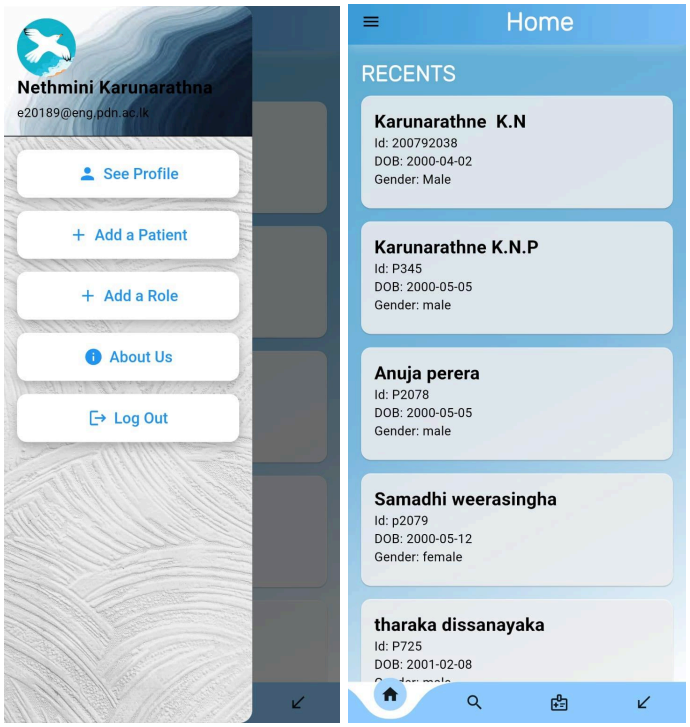






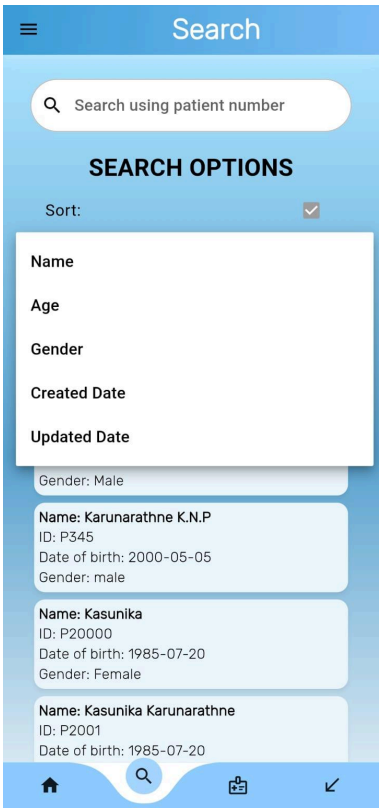
02.Add a patient

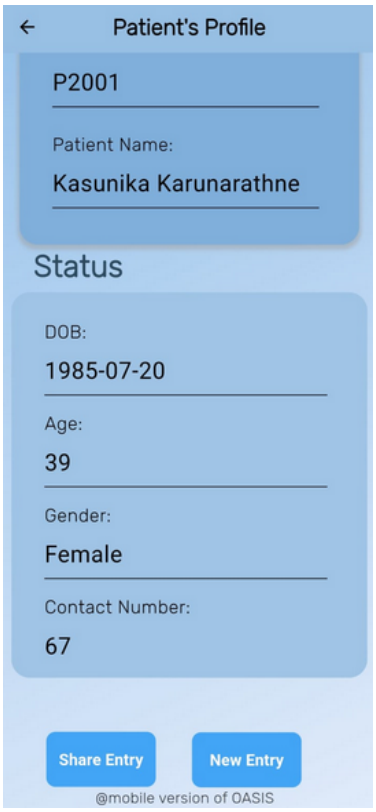
1. Log in to your account.
2. Tap the hamburger icon on the top left corner to go to the navigation menu.
3. Select Add patient in the navigation menu.
4. Fill in the required information and get it sign by the patient
5. Tap on generate consent form to complete



03.Search a patient

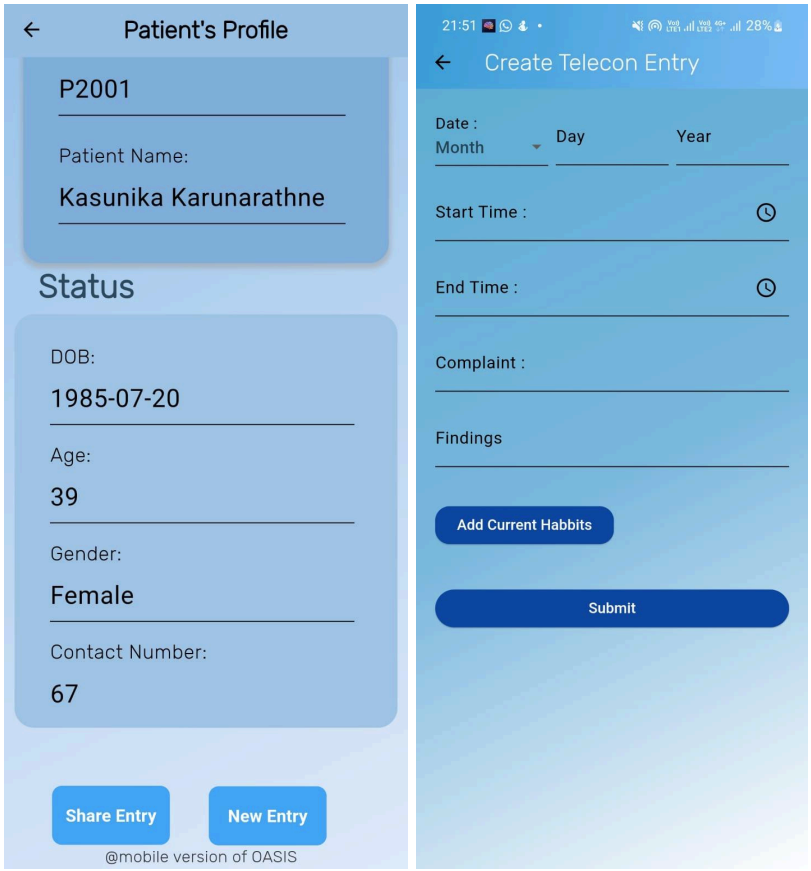
1. Log in to your account.
2. Select the search icon on the Bottom Navigation Bar.
3. Search the patient needed to access
4. Select the relevant patient
5. The patient details is displayed





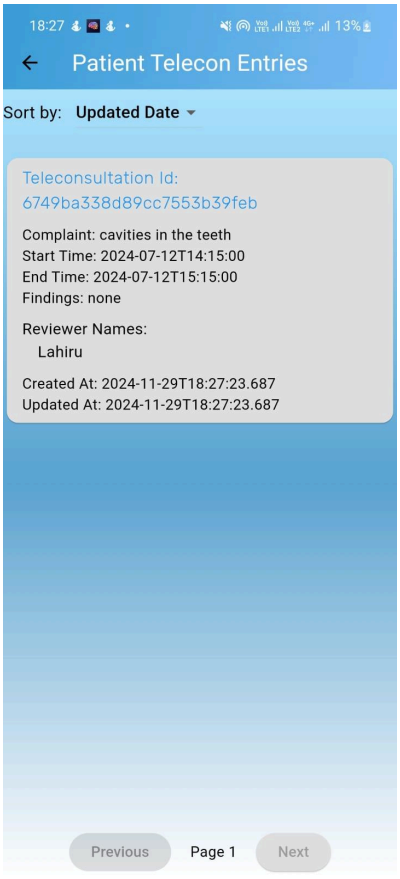
04.Create an Teleconsultation Entry

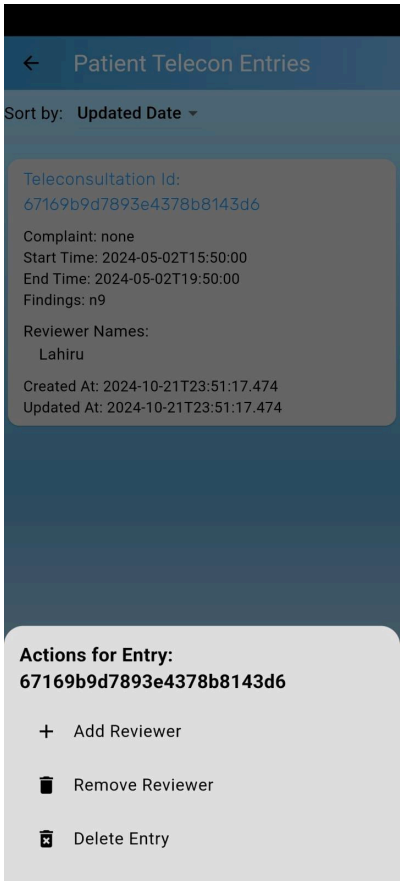
1. Log in to your account.
2. Select the search icon on the Bottom Navigation Bar.
3. Search the patient needed to access
4. Select the relevant patient
5. Select “New Entry” to create a new Telecon Entry.
6. Fill the details required and select “Submit ” to complete

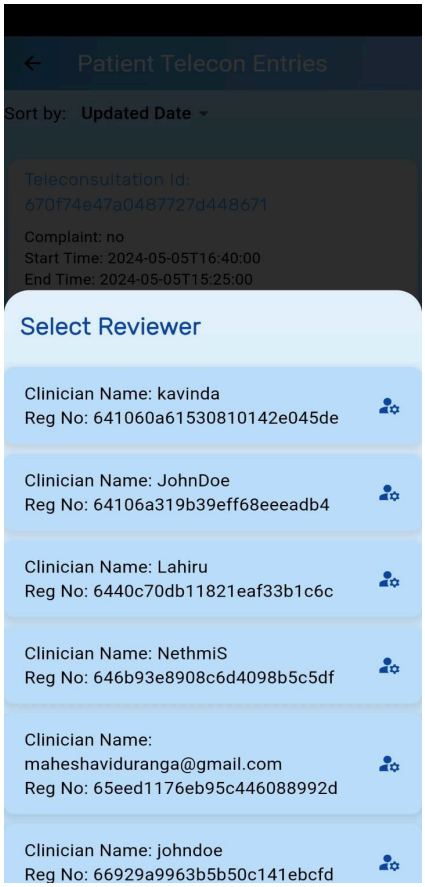


05. Share a Teleconsultation Entry of a patient

1. Log in to your account.
2. Select the search icon on the Bottom Navigation Bar.
3. Tap on the patient needed to access.
4. Select the relevant patient.
5. Tap on“Share Entry” in the patient profile to share the created Teleconsultation Entries.
6. Select the entry you want to share .
7. Tap on“Add Reviewer” to add Reviewers.
8. Assign a reviewer from the List.

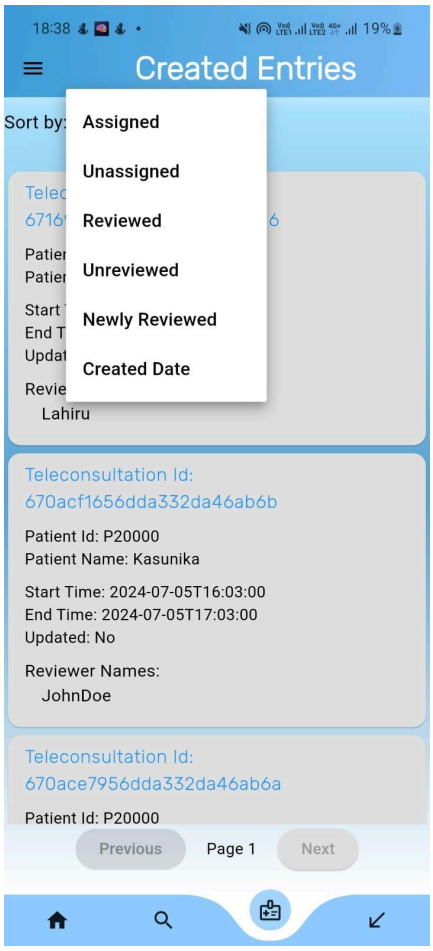






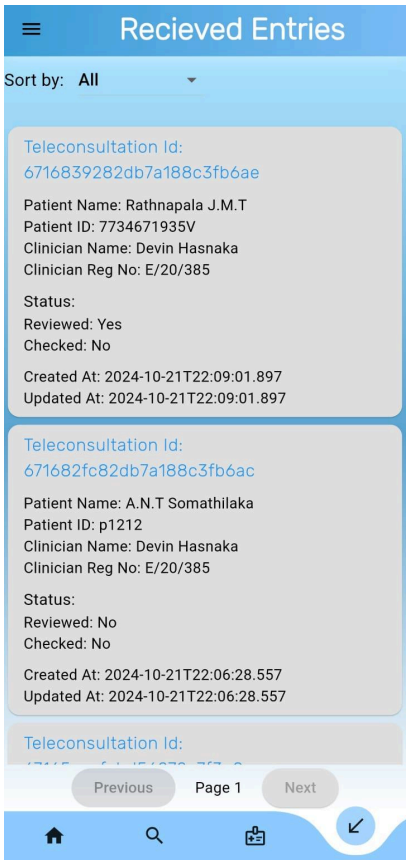
06.View created entries

1. Log in to your account.
2. Select the third icon from the left on the Bottom Navigation Bar.
3. You can sort the entries according to “created date”, “Assigned”,“Non Assigned”,“Reviewed”,“Unreviewed”



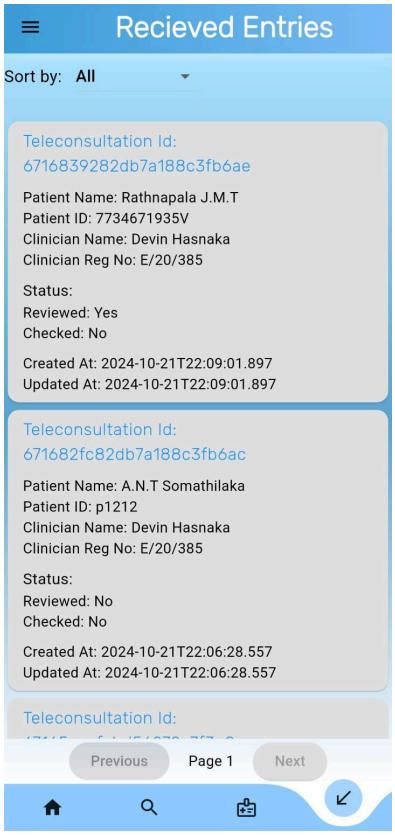
07. View received entries

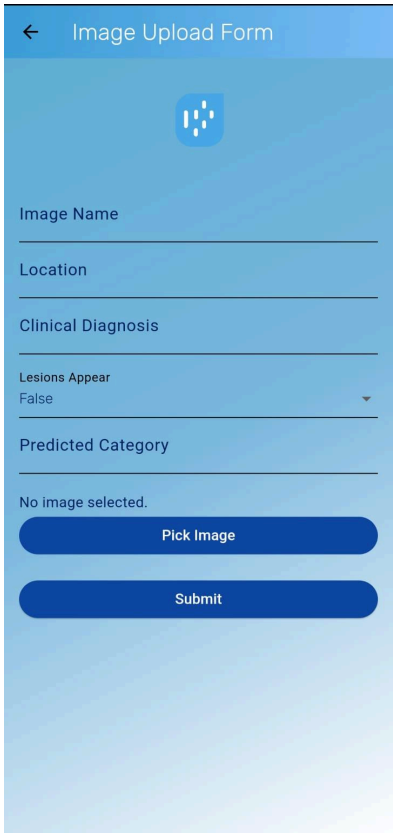
1. Log in to your account.
2. Select the fourth icon from the left on the Bottom Navigation Bar.
3. You can sort the entries according to “created date”,“All”
4. Tap on an entry to see the details



08. Add images to received entries

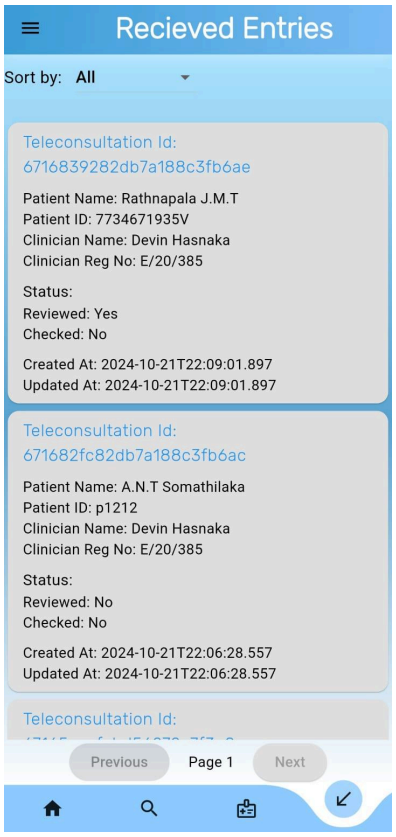
1. Log in to your account.
2. Select the fourth icon from the left on the Bottom Navigation Bar.
3. Select the required teleconsultation entry.
4. Tap on the “Upload Images ” button.
5. Fill the required details and select the “Pick Image” button.
6. Select the required image.
7. Tap on the “Submit” to complete

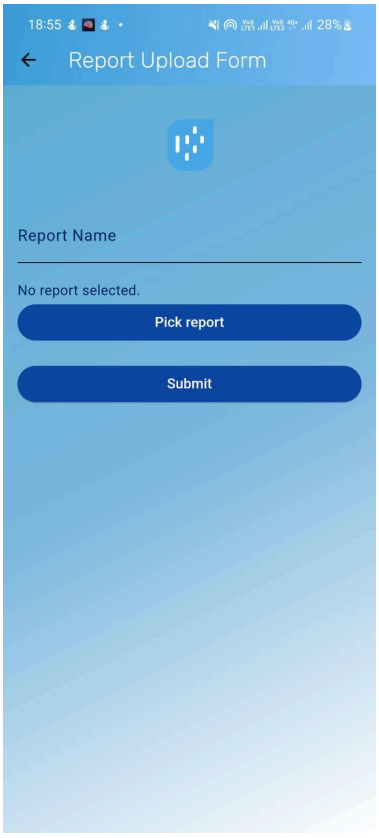




09. Add reports to received entries

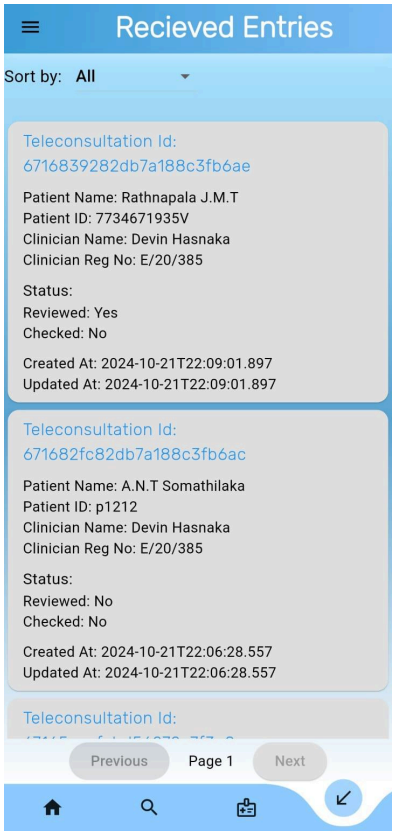
1. Log in to your account.
2. Select the fourth icon from the left on the Bottom Navigation Bar.
3. Select the required teleconsultation entry.
4. Tap on the “Upload Reports ” button .
5. Fill the required details and select the “Pick Report” button.
6. Select the required report.
7. Tap on the “Submit” button to complete.

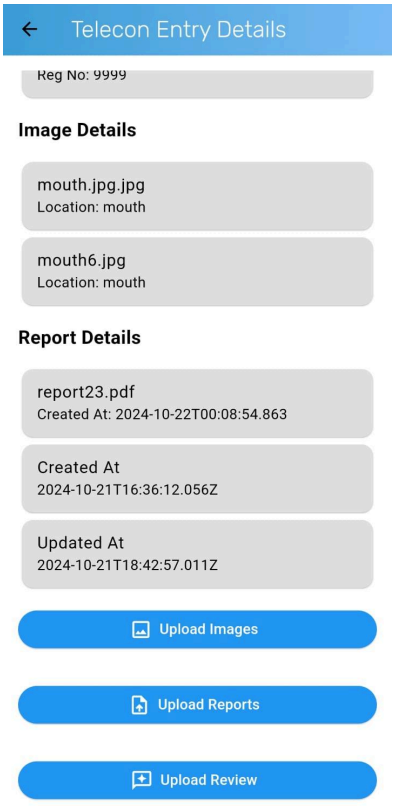


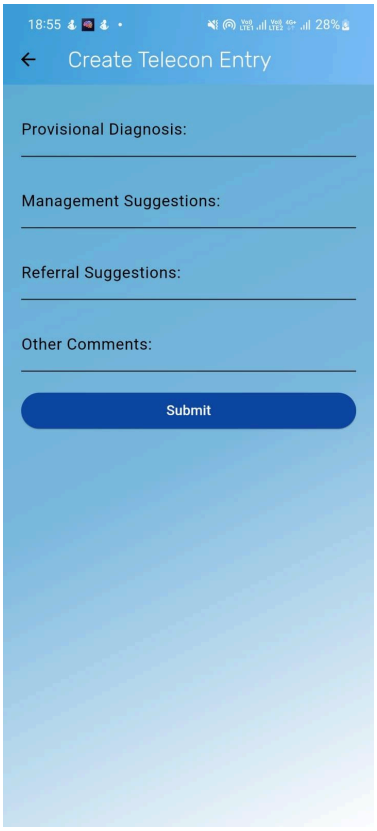


10.Add reviews to received entries

1. Log in to your account.
2. Select the fourth icon from the left on the Bottom Navigation Bar.
3. Select the required teleconsultation entry.
4. Tap on the “Upload Review” button.
5. Fill the required details and select the “Submit” button.







8. Feedback

Throughout the project, the E17 OASIS Teledentistry team, who had previously worked on the web application, provided valuable supervision and guidance. Their insights were instrumental in selecting suitable platforms, frameworks, and tools. Weekly progress meetings were conducted with the team to discuss improvements, address issues, and refine project guidelines. This collaborative feedback significantly contributed to the efficient organization and management of the project.

During the final presentation to the supervisor panel, Dr. Isuru Nawinne provided valuable suggestions to enhance the application's user experience (UX). He recommended improvements to the user interface (UI), including adjustments to color schemes, font selections, and other design elements, to ensure the application adhered to industry standards and was more user-friendly.

9. Challenges

1. Migration to New Platforms

To improve commercial feasibility, the previously developed MERN backend had to be transitioned to Spring Boot, while the frontend was migrated to Flutter/Dart. This required the team to learn new platforms and frameworks, presenting a significant challenge. However, with continuous learning and collaborative effort, the migration was successfully completed.

2. Ensuring Data Security

Given the sensitive nature of patient data and its transmission through hospital networks nationwide, implementing robust security measures was a critical challenge. To address this, JSON Web Tokens (JWTs) were utilized to provide secure and encrypted access control, ensuring that data remained protected throughout the system.

3. Simplifying Patient Sharing

Facilitating the sharing of patient information among dentists for further consultations and support posed another challenge, particularly in deciding what details should be shared. A teleconsultation entry system was introduced, simplifying the sharing process and ensuring smooth collaboration between dentists.

4. Team Collaboration and Integration

Initially, the team divided the work into frontend and backend tasks, which caused difficulties during integration and testing. To overcome this, the team adopted the AGILE methodology, with iterative cycles that enabled smoother collaboration and timely goal achievement. Peer-reviewed pull requests also helped resolve code conflicts effectively, leading to better integration.

5. Backend Complexity

Developing the backend became time-intensive due to the increasing complexity of APIs and schemas required for the growing functional requirements. To manage this, the team implemented a three-layer architecture (Controller, Service, Data Access) and documented APIs and schemas comprehensively using Swagger, which streamlined development and debugging.

6. Connectivity Issues on Campus

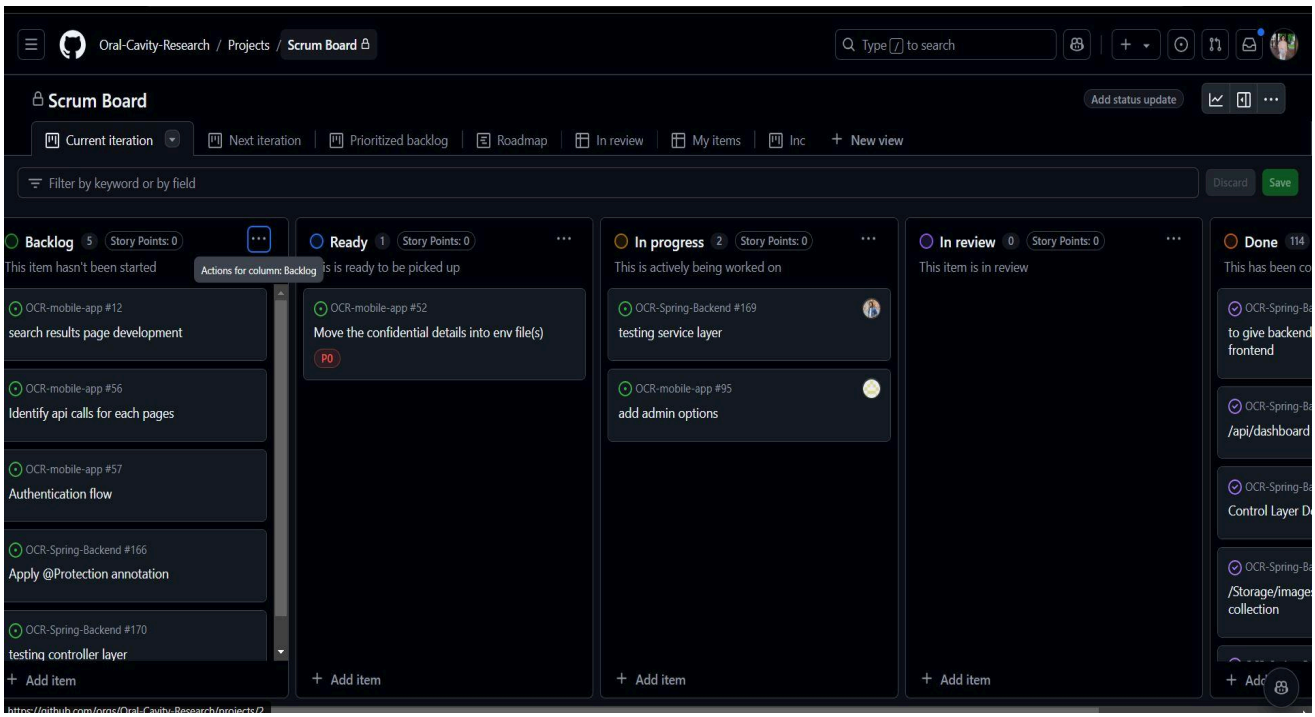
The university Wi-Fi network did not support the Amazon-hosted MongoDB server, leading to significant difficulties in software development and demonstrations on campus. Unstable hotspots further hindered progress. Despite attempts to resolve the issue with the NOC staff, a permanent solution could not be found, adding to the project's logistical challenges.

7. Compliance with International Laws and Data Storage

In commercial deployment, compliance with international laws regarding sensitive information, such as medical data, presents a significant challenge. According to these regulations, medical information cannot be stored outside the geographical boundaries of the country. This requires hosting databases within the country, which adds complexity to infrastructure planning. Additionally, since medical data cannot be deleted, storage requirements are expected to grow at a rapid rate, demanding scalable solutions and significant investment in storage infrastructure.

10. Work Task Division and Individual Contribution

Task Distribution



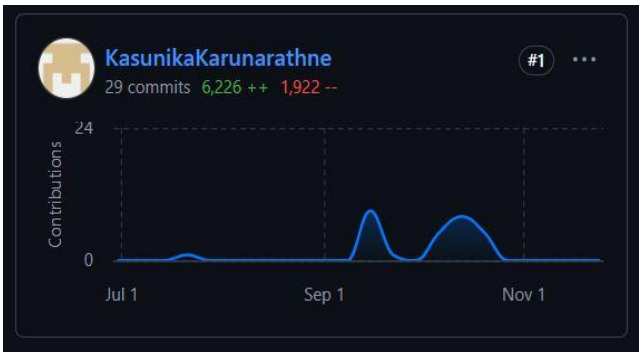
The screenshot displays the GitHub Scrum Board, which was used to efficiently distribute and track tasks among team members. Each task card is assigned to a specific member, ensuring accountability and clarity of roles. The board visually represents the progress of tasks within the sprint, facilitating collaboration and coordination across the team to achieve project goals.

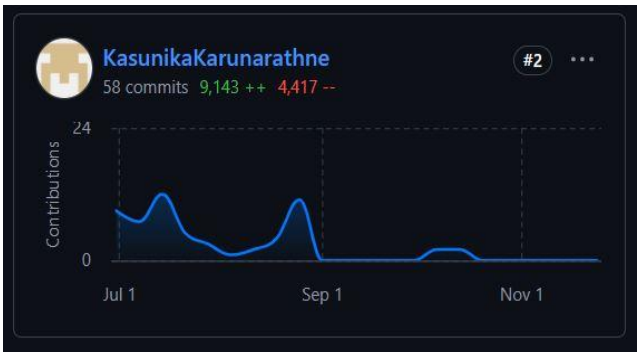
Contribution of group members

For Front-End

For Back-End

Karunarathne K.N.P. (E/20/189)





Contribution for the Front-end

Contribution for the Back-end

1. Backend Development

- Utilized Swagger documentation to design and document API endpoints.
- Implemented service layer, model, and controller layer for backend functionalities.Such as:
    - Image Upload Service
    - Report Upload Service
    - Teleconsultation Service
    - Draft Upload Service
    - Consent Form Generation Service
    - Assignment Service
    - Request Service
    - Admin Authentication Service

2. Frontend Development

- Designed and implemented essential UI components for various services, including:
    - Image Upload Service
    - Report Upload Service
    - Teleconsultation Service
    - Consent Form Generation Service
    - Review Upload service

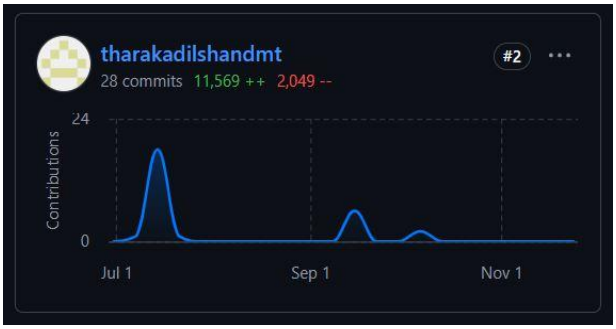
3. Integration

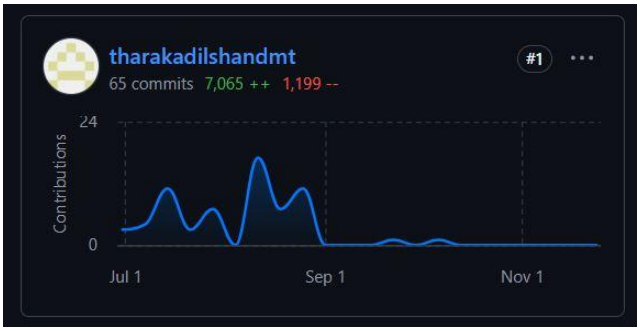
- Seamlessly integration between backend and frontend components for key services mentioned above

4. Testing

- Tested major backend endpoints using Postman.

Dilshan D.M.T. (E/20/069)





Contribution for the Front-end

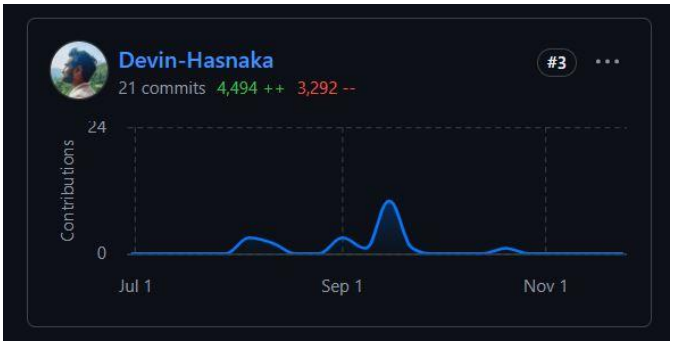
Contribution for the Back-end

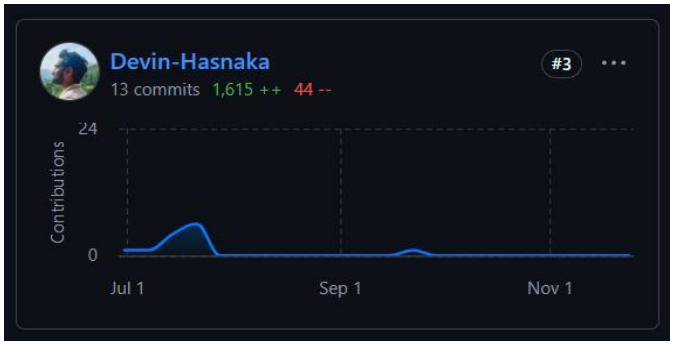
1. Backend Development
    - Completed main parts of the backend using Swagger documentation.
    - Tested major backend endpoints using Postman.
    - Implemented service layer, model, and controller layer for backend functionalities.
2. Authentication Features
    - Developed access token and refresh token generation using JWT.
3. Frontend Development
    - Designed key UI components using Flutter:
    - Patient details screen.
    - Home screen.
    - Login screen.

- File upload and download functionalities.
- Search engine.

4. Integration
   - Integrated backend & frontend, focusing on critical application functionalities.

Sriyarathna D.H. (E/20/385)





Contribution for the Front-end

1. Backend Development
   - Designing structure of the back-end
   - API call node connections
   - Implemented service layer, model, and controller layer for backend functionalities.
2. Frontend Development
   - Designed front end wireframes
   - UI development (Pages)
   - Front-end styling
   - Front-end API calls
3. Integration
   - Integrated backend & frontend, focusing on critical application functionalities.

Contribution for the Back-end

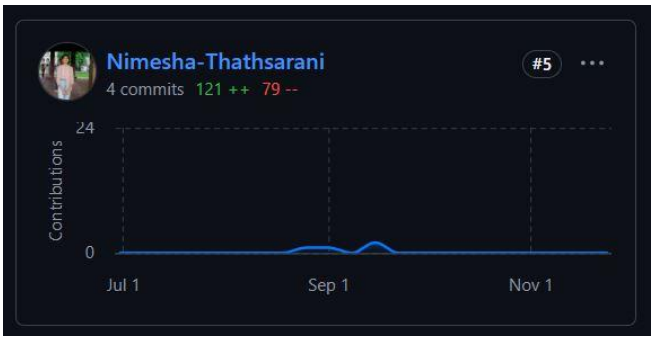
Wanasinghe J.K. (E/20/420)

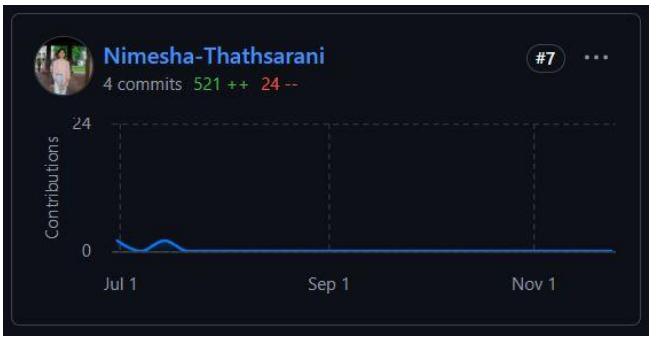
Contribution for the Front-end

Contribution for the Back-end

1. Backend Development
   - Code Quality Maintenance using Sonarlint
   - Refine Swagger Documentation
   - Lombok integration
2. Frontend Development
   - Code Quality maintenance using Flutter Fix
   - Search Page, Doctor Profile Page, about us page development
3. Testing
   - J-Unit Testing for service and control layer
4. Documents
   - Biweekly report preparation
   - Final report preparation

Somathilaka A.N.T. (E/20/381)





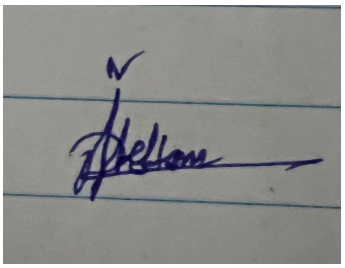
Contribution for the Front-end

Contribution for the Back-end

1. Backend Development
   - Swagger Documentation
   - Draft Entry Service Layer implementation
   - Image Control Layer implementation
2. Frontend Development
   - UI Development
   - Color theming for UX

11. Declaration

We, hereby declare that the particulars and work presented in this report are true and accurate to the best of our understanding and knowledge.

E. No.	Name	Signature
E/20/069	DILSHAN.D.M.T.	
E/20/189	KARUNARATHNE.K.N.P.	

E/20/381	SOMATHILAKA.A.N.T.	
E/20/385	SRIYARATHNA.D.H.	
E/20/420	WANASINGHE.J.K.