



Project सत्र

Open-Source Clinic Patient Management

A PROJECT REPORT

Submitted by

Saksham Sharma (22BIS70051)

Brian Edwin Kihore (22BIS70024)

Sparsh Shandil (22BIS70013)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE



April 2024



CHANDIGARH
UNIVERSITY
Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report "**Project सत्र**" is the bonafide work of **Saksham Sharma, Brian Edwin Kihore, Sparsh Shandil** who carried out the project work under my supervision.

Signature

Aman Kaushik
Head Of Department
AIT-CSE

Signature

Neeru Bala
Supervisor
AIT-CSE

Submitted for the project viva-voce examination held on 30th April.

INTERNAL EXAMINER

EXTERNAL EXAMINER

Contents

Abstract	5
1. Introduction	6
1.1 Overview of the Project	6
1.2 Purpose and Objectives	7
1.3 Scope of the Project	8
2. System Architecture	9
2.1 Overview of the System	9
2.2 Technology Stack and Frameworks	9
2.3 System Components	10
3. Functional Requirements	12
3.1 User Roles and Permissions	12
3.2 User Interface Design	12
3.3 Patient Management	13
3.4 Medicine Management	14
3.5 Appointment Scheduling	15
3.6 Report Generation and Printing	16
4. System Implementation	18
4.1 Database Design and Schema	18
4.2 Data Storage and Management	19
4.3 Integration of Supabase for Online Database	19
4.4 Integration of PostgreSQL for Local Database	20
4.5 Windows UI Scheme Integration	21
5. System Testing and Validation	23
5.1 Test Plan	23
5.2 Unit Testing	24
5.3 Integration Testing	25
5.4 System Validation	26
6. Project Management	28
6.1 Resource Allocation	28
6.2 Risk Analysis and Mitigation	29
6.3 Project Documentation	30
6.4 Project Evaluation and Lessons Learned	31
7. Conclusion	32
7.1 Summary of the Project	32
7.2 Achievements and Limitations	32
7.3 Future Enhancements	33
8. Figures	34
8.1 Fluent UI	34
8.2 Custom Flutter Implementation	38
8.3 Database Formulation	40

• Patient Data	41
• Sample Symptom Notation	41
• Medications	42
8.4 Application Sneak Peeks	43
8.5 Code Showcase	46
8.6 Database Tools	46

Abstract

Project Sattrra presents a robust desktop application tailored for the efficient management of patient care and administrative tasks within small and medium-sized hospitals and clinics. Utilizing the Dart programming language and Flutter framework, the application delivers an intuitive user interface and advanced functionalities aimed at optimizing operational processes and enhancing patient care quality. With a focus on patient management, medication tracking, appointment scheduling, and report generation, Project Sattrra aims to streamline healthcare workflows and improve overall efficiency.

The system architecture adopts a client-server model, leveraging Supabase for online database storage and synchronization, alongside PostgreSQL for local data storage. This architecture ensures seamless data management and accessibility across different devices and locations, even in offline scenarios. The user interface design follows Fluent UI principles, offering a visually appealing and user-friendly experience for healthcare professionals.

The project emphasizes thorough testing and validation processes, encompassing unit testing, integration testing, and system validation, to ensure the application meets high standards of reliability and performance. Project management strategies focus on resource allocation, risk analysis, and comprehensive documentation, enabling effective coordination and successful project delivery.

Despite encountering challenges such as knowledge imbalances within the team and time constraints, the project achieves significant milestones in database design, UI development, and system integration. Future enhancements include the implementation of server database setups, user training initiatives, customization options, performance optimization measures, and community engagement efforts.

Project Sattrra emerges as a valuable solution for healthcare facilities, offering a comprehensive suite of features to enhance patient care delivery and streamline administrative processes. By addressing the needs of modern healthcare settings, Project Sattrra aims to contribute to improved patient outcomes and operational efficiency in the healthcare sector.

1. Introduction

1.1 Overview of the Project

Project Sattrra stands at the forefront of healthcare innovation, promising to revolutionize the way small and medium hospitals and clinics operate. Beyond being just a desktop application, it embodies a vision of efficiency, efficacy, and excellence in healthcare management. By amalgamating cutting-edge technology with a deep understanding of healthcare workflows, Project Sattrra offers a comprehensive solution that transcends traditional software boundaries.

At its core, Project Sattrra is a beacon of hope for healthcare providers burdened by the complexities of administrative tasks. Through its user-friendly interface and advanced functionalities, the application empowers healthcare professionals to navigate the intricate maze of patient information, medication records, and appointment scheduling with ease. By centralizing these critical components of healthcare management, Project Sattrra enables healthcare providers to make informed decisions, personalize patient care, and optimize operational processes.

Moreover, Project Sattrra is not just about efficiency—it's about elevating the quality of patient care to unprecedented heights. With automated medication tracking, stock management, and appointment scheduling, the application ensures that healthcare facilities operate at peak efficiency, maximizing resource utilization and minimizing wastage. By streamlining administrative tasks, Project Sattrra allows healthcare professionals to devote more time and attention to what truly matters: delivering compassionate and personalized care to their patients.

1.2 Purpose and Objectives

The purpose of Project Sattrra is to develop an intuitive and user-friendly desktop application that caters to the needs of healthcare providers in small and medium hospitals and clinics. It aims to empower these healthcare professionals to efficiently manage patient information, medication records, and appointment scheduling. By providing a comprehensive solution, Project Sattrra strives to streamline administrative tasks and enhance the overall healthcare experience for both healthcare providers and patients. The objectives of the project include:

- *Simplifying patient management:* The application will provide a centralized system for storing and retrieving patient details, medical history, and treatment plans. It aims to enhance the accuracy and accessibility of patient information, enabling healthcare professionals to make well-informed decisions.
- *Streamlining medication management:* Project Sattrra will facilitate the recording, tracking, and management of medicines within hospitals and clinics. It will allow healthcare providers to maintain a comprehensive database of medications, monitor stock levels, and generate alerts for reordering.
- *Efficient appointment scheduling:* The application will offer a convenient and automated appointment scheduling system. It will enable healthcare staff to create, reschedule, and cancel appointments, ensuring optimal utilization of resources and minimizing scheduling conflicts.
- *Report generation and printing:* Project Sattrra will include a feature to generate patient reports with relevant medical information. These reports can be easily printed for record-keeping purposes or shared with patients and other healthcare providers.

1.3 Scope of the Project

The scope of Project Sattrra extends far beyond the confines of traditional software development; it encompasses a holistic approach to healthcare management that encompasses the entire spectrum of administrative tasks and patient care processes. By leveraging the power of the Flutter framework, Supabase, and PostgreSQL, Project Sattrra aims to deliver a seamless and integrated solution that meets the diverse needs of modern healthcare settings.

From patient management to medication tracking and appointment scheduling, Project Sattrra offers a comprehensive suite of functionalities designed to streamline workflows, enhance efficiency, and improve patient outcomes. By centralizing patient information, medication records, and appointment schedules, the application ensures that healthcare providers have access to the information they need, when they need it, enabling them to deliver personalized and timely care to their patients.

Moreover, Project Sattrra's scope extends beyond just software development; it encompasses a broader vision of collaboration and innovation within the healthcare industry. By fostering partnerships with healthcare providers, industry stakeholders, and technology partners, Project Sattrra aims to create a vibrant ecosystem of innovation that drives continuous improvement and excellence in healthcare management. Through its visually appealing Windows UI scheme and seamless integration with online and local databases, Project Sattrra represents a paradigm shift in how healthcare facilities approach administrative tasks and patient care, setting new standards for efficiency, efficacy, and excellence in healthcare management.

2. System Architecture

2.1 Overview of the System

Project Sattrra's system architecture embodies a sophisticated and robust infrastructure designed to meet the multifaceted needs of healthcare management. At its core, the architecture follows a client-server model, facilitating efficient communication and data exchange between the desktop application and the backend servers.

The client-side, developed using Dart and Flutter, serves as the user-facing interface, offering healthcare professionals a seamless and intuitive platform to interact with. Meanwhile, the server-side infrastructure leverages Supabase for online database storage and synchronization, complemented by SQLite for offline storage, ensuring data availability and consistency across various scenarios, including intermittent internet connectivity.

Critical to the system's resilience is the seamless synchronization mechanism, which enables data to flow seamlessly between online and offline databases. This ensures that healthcare providers have uninterrupted access to vital patient information, medication records, and appointment schedules, regardless of their network environment.

2.2 Technology Stack and Frameworks

The technology stack and frameworks used in Project Sattrra are as follows:

- *Dart*: The primary programming language used for developing desktop applications.
- *Flutter*: A cross-platform framework that enables the development of rich and responsive user interfaces for the desktop application.

- *Supabase*: An open-source platform that provides online database storage and synchronization capabilities, ensuring data consistency across multiple devices and users.
- *PostgreSQL*: A robust relational database management system used for local data storage, enabling synchronization with the online database via Supabase.

2.3 System Components

The major components of the Project Sattrra system are as follows:

- *User Interface (UI) Module*: Serving as the interface between users and the application, this module is responsible for presenting information in a visually appealing and intuitive manner. It provides healthcare professionals with easy-to-use tools and features to navigate patient records, manage medications, schedule appointments, and generate reports.
- *Patient Management Module*: At the heart of healthcare administration, this module enables healthcare providers to efficiently manage patient information, including registration details, medical history, treatment plans, and demographic data. It empowers healthcare professionals with comprehensive insights into patient care, facilitating informed decision-making and personalized treatment strategies.
- *Medication Management Module*: Designed to streamline medication-related workflows, this module empowers healthcare providers to maintain accurate records of medications within

healthcare facilities. From inventory management to prescription tracking and automatic reordering, it ensures that medications are available when needed, minimizing disruptions to patient care, and optimizing resource utilization.

- *Appointment Scheduling Module:* This module automates the scheduling of appointments, allowing healthcare staff to efficiently manage their calendars and allocate resources effectively. With features such as appointment creation, rescheduling, and cancellation, it helps minimize wait times, reduce no-shows, and improve overall patient satisfaction.
- *Data Storage and Synchronization Module:* Integral to the system's reliability and resilience, this module manages the synchronization of data between online and offline databases. By ensuring seamless data replication, it enables healthcare professionals to access critical information regardless of their network connectivity, ensuring continuity of care and operational efficiency.
- *Reporting Module:* Catering to the need for comprehensive reporting capabilities, this module generates detailed reports containing relevant medical information. From patient summaries to treatment histories and appointment logs, these reports serve as valuable tools for record-keeping, communication, and decision support, empowering healthcare providers with actionable insights and facilitating collaborative care delivery.

3. Functional Requirements

3.1 User Roles and Permissions

Project Sattrra meticulously defines distinct user roles and corresponding permissions to ensure secure and efficient management of healthcare data.

- *Administrator Role:* As the custodian of system oversight, administrators wield comprehensive control over all functionalities within Project Sattrra. They have the authority to manage user accounts, assign permissions, and regulate access to sensitive data. Administrators can seamlessly navigate through different modules and perform administrative tasks such as system configuration, data backup, and user management.
- *Healthcare Provider Role:* Healthcare providers, including doctors and nurses, are equipped with functionalities tailored to patient care and treatment. They have access to patient management features, enabling them to input and update patient information, view medical histories, and prescribe medications. Additionally, healthcare providers can schedule appointments, generate reports, and collaborate with colleagues to ensure coordinated patient care. Their permissions are designed to optimize workflow efficiency while maintaining data integrity and patient privacy.

3.2 User Interface Design

The user interface Project Sattrra's user interface (UI) design embodies principles of usability, accessibility, and aesthetics to deliver an intuitive and engaging user experience.

- *Fluent UI Elements:* The UI incorporates elements from the Fluent Design System, Microsoft's comprehensive design language, to ensure consistency and familiarity for users accustomed to Windows-based applications. Elements such as ribbons, tabs, and context menus are seamlessly integrated to enhance navigation and usability.
- *Responsive Design:* The UI is designed to be responsive across various devices and screen sizes, enabling healthcare professionals to access and interact with the application seamlessly, whether on desktops, tablets, or mobile devices. Adaptive layout techniques ensure optimal presentation and usability across different form factors.
- *Accessibility Features:* Project Sattrra prioritizes accessibility by incorporating features such as high contrast modes, keyboard navigation support, and screen reader compatibility. These features ensure inclusivity and enable users with diverse needs to interact with the application effectively.

3.3 Patient Management

Project Sattrra's user interface (UI) design embodies principles of usability, accessibility, and aesthetics to deliver an intuitive and engaging user experience.

- *Fluent UI Elements:* The UI incorporates elements from the Fluent Design System, Microsoft's comprehensive design

language, to ensure consistency and familiarity for users accustomed to Windows-based applications. Elements such as ribbons, tabs, and context menus are seamlessly integrated to enhance navigation and usability.

- *Responsive Design:* The UI is designed to be responsive across various devices and screen sizes, enabling healthcare professionals to access and interact with the application seamlessly, whether on desktops, tablets, or mobile devices. Adaptive layout techniques ensure optimal presentation and usability across different form factors.
- *Accessibility Features:* Project Sattrra prioritizes accessibility by incorporating features such as high contrast modes, keyboard navigation support, and screen reader compatibility. These features ensure inclusivity and enable users with diverse needs to interact with the application effectively.

3.4 Medicine Management

Project Sattrra's medicine management module simplifies the process of inventory tracking, prescription management, and medication dispensing within healthcare facilities.

- *Centralized Medication Database:* The module maintains a centralized database of medications, including generic names, dosage forms, strengths, and indications. Users can easily search, add, or update medication records, ensuring accurate and up-to-date inventory management.

- *Prescription Workflow:* Healthcare providers can prescribe medications directly within the application, specifying dosage instructions, frequency, duration, and any special considerations. The module supports electronic prescribing (e-prescribing) and drug-drug interaction checks to prevent medication errors and ensure patient safety.
- *Stock Monitoring and Reordering:* Automated stock monitoring features track medication usage and inventory levels in real-time. When stock levels fall below predefined thresholds, the system generates alerts and prompts users to reorder medications to avoid stockouts and treatment delays. Integration with pharmacy suppliers and wholesalers streamlines the procurement process, ensuring timely replenishment of stock.

3.5 Appointment Scheduling

The appointment scheduling module streamlines the process of managing appointments in Project Sattrra.

- *Dynamic Scheduling Options:* Healthcare providers can schedule appointments based on patient preferences, provider availability, and facility resources. The module supports flexible scheduling options, including in-person visits, telehealth consultations, and group appointments, to accommodate diverse patient needs and preferences.
- *Automated Reminders and Notifications:* The system sends automated appointment reminders to patients via email, SMS, or

push notifications to minimize no-shows and late cancellations. Providers receive real-time notifications of new appointment requests, cancellations, or rescheduling requests, enabling proactive management of their schedules and patient lists.

- *Resource Allocation and Optimization:* Advanced scheduling algorithms consider various factors such as appointment duration, provider specialization, room availability, and equipment requirements to optimize resource allocation and minimize scheduling conflicts. Users can view real-time availability of providers, rooms, and equipment, facilitating efficient appointment booking and utilization.

3.6 Report Generation and Printing

The report generation and printing module empower healthcare providers to create, customize, and distribute comprehensive reports containing patient information, treatment summaries, and clinical insights.

- *Customizable Report Templates:* Users can choose from a variety of predefined report templates or create custom templates tailored to specific clinical scenarios, specialties, or organizational requirements. The module supports rich-text formatting, graphical elements, and data visualization tools to enhance the presentation and clarity of reports.
- *Data Export and Integration:* Reports can be exported in various formats such as PDF, Excel, or CSV for easy sharing, archiving, or integration with other systems. The module supports seamless integration with electronic health record (EHR) systems, practice

management software, and billing platforms to streamline documentation workflows and ensure data interoperability.

- *Compliance and Audit Trails:* Reports generated within Project Sattra adhere to regulatory standards and best practices for documentation, privacy, and security. Audit trails track the creation, modification, and distribution of reports, providing transparency and accountability for clinical documentation processes. Data encryption and access controls safeguard patient confidentiality and mitigate the risk of unauthorized access or data breaches.

4. System Implementation

4.1 Database Design and Schema

The database design of Project Sattrra revolves around a relational model, ensuring structured data storage and efficient retrieval. Here's an elaboration on the entities and attributes within the database schema:

- *Patient*: This entity encapsulates vital information about patients, including unique identifiers (ID), names, contact details, medical history, recorded symptoms, and ongoing treatments. The schema maintains relationships between patients and their associated treatments, enabling seamless tracking of healthcare interactions and outcomes.
- *Medicine*: The medicine entity manages details related to medications available within the healthcare facility. Attributes include unique identifiers, medicine names, dosage information, and current stock availability. The schema links medicine records to treatment instances, facilitating accurate medication administration and inventory management.
- *User*: Users represent individuals authorized to access the Project Sattrra system, each with a designated role determining their permissions and responsibilities. Attributes include unique identifiers, user names, roles, and references to the user who created the account. The schema enforces data integrity and access controls to safeguard system security and user privacy.

4.2 Data Storage and Management

Project Sattrra leverages Supabase as the primary online database solution, harnessing its robust features for secure data storage and efficient management. Key aspects of Supabase integration include:

- *Automatic Backups:* Supabase automates routine data backups, ensuring data resilience and mitigating the risk of data loss due to unforeseen circumstances such as system failures or hardware malfunctions.
- *Data Encryption:* Supabase employs advanced encryption techniques to protect sensitive data both at rest and in transit, safeguarding patient confidentiality and compliance with regulatory standards such as HIPAA.
- *Access Control:* Role-based access controls within Supabase regulate user permissions and restrict unauthorized access to sensitive information, enhancing data security and compliance with privacy regulations.

4.3 Integration of Supabase for Online Database

Supabase is seamlessly integrated into its architecture to enable real-time data synchronization and collaboration among users. Key aspects of Supabase integration include:

- *Real-Time Updates:* Supabase facilitates real-time synchronization of data changes across multiple devices and users, ensuring that all stakeholders have access to the most recent information and updates.

- *Conflict Resolution:* Supabase employs conflict resolution mechanisms to manage concurrent data modifications and prevent data inconsistencies or conflicts, maintaining data integrity and consistency across the system.
- *Offline Support:* Supabase offers offline support, allowing users to access and modify data even when disconnected from the internet. Changes made offline are synchronized with the online database once connectivity is restored, ensuring seamless data continuity and accessibility.

4.4 Integration of PostgreSQL for Local Database

PostgreSQL is integrated as the local database solution, providing reliable data storage and offline access to essential functionalities. Key aspects of PostgreSQL integration include:

- *Local Data Storage:* PostgreSQL stores critical data locally on users' devices, enabling seamless access to core functionalities even in offline environments. This local cache ensures uninterrupted workflow continuity and accessibility, enhancing user productivity and satisfaction.
- *Data Synchronization:* PostgreSQL synchronizes data changes with the online database via Supabase, ensuring that updates made locally are propagated to the central database once connectivity is restored. This bidirectional synchronization mechanism guarantees data consistency and coherence across distributed environments.

- *Performance Optimization:* PostgreSQL's robust performance optimization features enhance the responsiveness and efficiency of local data operations, ensuring smooth and seamless user interactions with the application even in resource-constrained environments.

4.5 Windows UI Scheme Integration

Project Sattrra has a user interface (UI) that embraces the Windows UI scheme, incorporating Fluent UI elements and design principles to deliver a cohesive and intuitive user experience. Key aspects of Windows UI integration include:

- *Fluent UI Elements:* The UI incorporates familiar Fluent Design System elements such as ribbons, tabs, and context menus, ensuring consistency and familiarity for users accustomed to Windows-based applications. These elements enhance navigation and usability, enabling users to intuitively interact with the application.
- *Visual Consistency:* Project Sattrra maintains visual consistency with the Windows UI scheme, adhering to design guidelines and conventions to create a seamless and cohesive user experience. Consistent styling, typography, and iconography enhance usability and reinforce brand identity, fostering user trust and engagement.
- *Responsive Design:* The UI adopts a responsive design approach, ensuring optimal presentation and usability across various devices and screen sizes. Adaptive layout techniques and fluid design elements dynamically adjust the UI layout and content to

accommodate different form factors, enhancing accessibility and usability for users on desktops, tablets, and mobile devices.

By integrating these elements into the system implementation, Project Sattrra ensures a robust, user-friendly, and efficient healthcare management solution tailored to the needs of healthcare providers and administrators.

5. System Testing and Validation

5.1 Test Plan

The test plan for Project Sattrra encompasses a comprehensive approach to ensure the quality and reliability of the application. Here's an elaboration on the key components of the test plan:

- *Unit Testing:* Unit testing involves testing individual units or components of the application in isolation to validate their functionality. Each function, class, and module will undergo rigorous testing to detect and rectify any defects or anomalies. Test cases will be designed to cover various scenarios and edge cases, ensuring thorough coverage of the codebase.
- *Integration Testing:* Integration testing focuses on verifying the interaction and interoperability between different modules or components of the application. This testing ensures that data flows seamlessly between interconnected parts of the system and that integration points function as expected. Test scenarios will be devised to assess the integration of modules such as patient management, medicine management, and appointment scheduling, validating end-to-end system behavior.
- *System Validation:* System validation aims to confirm that the application meets the requirements and expectations of its intended users and stakeholders. This process involves testing the application as a whole to assess its overall functionality, usability, and performance. Testers will evaluate the user interface for intuitiveness, navigation, and consistency with design standards.

Functional testing will validate that all features and functionalities work as intended, meeting the specified requirements and acceptance criteria. Additionally, performance testing will assess the application's responsiveness, scalability, and reliability under varying load conditions.

5.2 Unit Testing

In the unit testing phase, individual units of code within the Project Sattrra application will undergo rigorous testing to ensure their correctness and reliability. Key aspects of unit testing include:

- *Test Case Design:* Test cases will be designed to cover a wide range of scenarios, including normal use cases, boundary cases, and error conditions. Each unit of code will be tested independently to isolate defects and validate its behavior under different conditions.
- *Test Automation:* Automation tools and frameworks such as JUnit or flutter_test will be utilized to automate the execution of unit tests, streamlining the testing process and enabling frequent regression testing. Automated tests will be executed as part of the continuous integration (CI) pipeline to detect and address defects early in the development lifecycle.
- *Code Coverage Analysis:* Code coverage metrics will be used to assess the adequacy of unit testing and identify areas of the codebase that require additional test coverage. Aimed at achieving high code coverage, this analysis ensures thorough testing of critical code paths and reduces the risk of undiscovered defects.

5.3 Integration Testing

Integration testing in Project Sattrra focuses on validating the interaction and integration between different modules and components of the application. Key aspects of integration testing include:

- *Integration Test Scenarios:* Test scenarios will be designed to simulate real-world interactions between interconnected modules, ensuring that data flows correctly and integration points function as expected. Scenarios will cover various use cases, including data exchange, communication protocols, and error handling.
- *Component Integration:* Integration tests will verify the integration of modules such as patient management, medicine management, and appointment scheduling, validating end-to-end system behavior and functionality. Data consistency and integrity will be ensured across integrated components, with particular emphasis on boundary conditions and error scenarios.
- *Stub and Mock Objects:* Stub and mock objects will be used to simulate external dependencies or unavailable components during integration testing, enabling controlled testing of component interactions and behavior. This approach facilitates isolation of components and reduces dependencies, ensuring more predictable and reliable test results.

5.4 System Validation

System validation entails comprehensive testing of the Project Sattrra application to confirm its compliance with user requirements and fitness for purpose. Key aspects of system validation include:

- *User Acceptance Testing (UAT)*: UAT involves testing the application with real users to validate its usability, functionality, and alignment with user expectations. Testers representing diverse user roles and personas will execute predefined test scenarios and provide feedback on the application's usability, intuitiveness, and effectiveness in meeting user needs.
- *Functional Testing*: Functional testing verifies that all features and functionalities of the application work as intended and meet the specified requirements. Test cases will cover core functionalities such as patient management, medicine management, appointment scheduling, and report generation, validating their correctness, completeness, and reliability.
- *Performance Testing*: Performance testing evaluates the responsiveness, scalability, and reliability of the application under varying load conditions. Test scenarios will assess the application's response times, resource utilization, and throughput, ensuring optimal performance and stability under peak usage scenarios.
- *Security Testing*: Security testing will be conducted to identify and mitigate potential security vulnerabilities and threats within the application. Test scenarios will evaluate authentication mechanisms, data encryption, access controls, and vulnerability points such as input validation and session management.

ensuring the confidentiality, integrity, and availability of sensitive data.

6. Project Management

6.1 Resource Allocation

The project team for Project Sattrra consists of three key roles responsible for different aspects of development:

- *Backend Developer*: This role focuses on designing and developing the server-side components and APIs required for the application. They are responsible for implementing the database schema, integrating with Supabase for online storage, and handling data synchronization. The backend developer will also be involved in unit testing the backend components and ensuring their reliability and performance.
- *Frontend Developer*: The front-end developer is responsible for designing and developing the user interface of the application. They will utilize the Flutter framework and the Fluent UI scheme to create an intuitive and visually appealing interface. The frontend developer will work closely with the backend developer to integrate the frontend with the backend APIs and ensure seamless data flow between the user interface and the database.
- *Project Manager*: The project manager oversees the overall project management, including resource allocation, task coordination, and project documentation. They will ensure that the project stays on track, meets deadlines, and allocate resources efficiently. The project manager will also handle communication and collaboration among team members, stakeholders, and external parties.

Additionally, documentation and communication tasks will be managed by the backend and frontend developers as part of their respective responsibilities. They will maintain documentation for the backend APIs, frontend code, and user manuals/guides. The project manager will provide guidance and ensure that documentation is up to date.

6.2 Risk Analysis and Mitigation

The following is a risk analysis for Project Sattrra along with suggested mitigation strategies:

- Technical Risks:
 - Compatibility issues with different operating systems and devices.
Mitigation: Conduct thorough compatibility testing on various platforms during the development phase.
 - Performance issues due to scalability limitations.
Mitigation: Implement performance optimization techniques, such as caching mechanisms and load testing, to identify and resolve performance bottlenecks.
- Resource Risks:
 - Insufficient availability of skilled developers. Mitigation: Allocate dedicated time for knowledge sharing and training to enhance the skills of team members. Consider partnering with external consultants or hiring additional resources if required.
 - Risk: Dependency on external libraries or frameworks.
Mitigation: Stay updated with the latest releases and security patches of the used libraries or frameworks. Maintain a backup plan in case of any critical vulnerabilities or compatibility issues.
- Schedule and Budget Risks:

- Scope creep leading to delays and budget overruns.

Mitigation: Establish a well-defined scope and change control process. Conduct regular project reviews to monitor and control scope changes.

- Unforeseen technical complexities during development.

Mitigation: Perform thorough requirement analysis and feasibility studies before starting development. Allocate sufficient buffer time and resources to address potential technical challenges.

- Security and Privacy Risks:

- Data breaches or unauthorized access to sensitive patient information.

Mitigation: Implement robust security measures, including encryption of data at rest and in transit, access control mechanisms, and regular security audits. Adhere to industry best practices and compliance standards, such as HIPAA, to ensure data privacy and security.

6.3 Project Documentation

For effective project documentation, the following documents should be created and maintained throughout the project lifecycle:

- *Project Plan*: An overarching document outlining project objectives, deliverables, timelines, and resource allocation. It serves as a roadmap for the project and helps ensure all team members are aligned.
- *Requirements Documentation*: Detailed documentation specifying functional and nonfunctional requirements of the system, including user stories, use cases, and system specifications. It helps in guiding the development process and serves as a reference for future enhancements.
- *Design Documentation*: Documentation describing the system

architecture, database schema, API specifications, and UI/UX designs. It helps in understanding the system's structure and facilitates collaboration between backend and frontend developers.

- *Test Documentation:* Documentation detailing the test plan, test cases, and test results. It ensures comprehensive test coverage and enables efficient testing and debugging of the application.
- *User Documentation:* User manuals, guides, or tutorials explaining how to use the application. It provides step-by-step instructions for users to navigate and utilize the features of Project Sattrra effectively.

6.4 Project Evaluation and Lessons Learned

Upon completion of the project, a thorough evaluation and lessons learned process should be conducted. This includes the following:

- *Evaluation Criteria:* Define the criteria for evaluating the success of the project, such as meeting project objectives, adherence to budget and schedule, user satisfaction, and the overall performance and stability of the system.
- *Evaluation Process:* Utilize surveys, feedback sessions, and performance metrics analysis to evaluate the project against the defined criteria. Collect feedback from stakeholders, end-users, and the project team to gather valuable insights.
- *Lessons Learned:* Reflect on the successes, challenges, and areas for improvement throughout the project. Identify valuable lessons learned that can be applied to future projects, such as process improvements, technical insights, or communication enhancements.

By conducting a comprehensive evaluation and documenting lessons learned, the team can identify areas of improvement and apply best practices for future projects.

7. Conclusion

7.1 Summary of the Project

In conclusion, Project Sattrra stands as a pivotal open-source medical and patient management software solution tailored for clinics and small to medium-sized hospitals. With the primary objective of streamlining administrative tasks, enhancing operational efficiency, and elevating patient care standards, Project Sattrra delivers a comprehensive suite of user-friendly features and advanced functionalities. Leveraging the Dart programming language and the Flutter framework, the project has successfully developed a desktop application capable of running seamlessly across various platforms, including web, Linux, and macOS.

Throughout the project lifecycle, the team has diligently focused on key features such as patient management, medicine management, appointment scheduling, and report generation. The integration of Supabase for online database storage and PostgreSQL for local data storage ensures a robust and synchronized data management system. Moreover, the implementation of server database setup capability empowers healthcare facilities with complete control over their data, thus guaranteeing data privacy and security.

7.2 Achievements and Limitations

During the implementation phase, the project has achieved several notable milestones. The successful design and implementation of a relational database schema for patient and medicine management demonstrates the team's proficiency in database architecture. The integration of Supabase and PostgreSQL further enhances data storage and retrieval capabilities. Additionally, the development of the Fluent UI scheme has yielded a user-friendly and intuitive interface, catering to the diverse needs of healthcare

professionals.

Despite these achievements, the project has encountered certain limitations. The imbalance of knowledge within the team has potentially impacted the depth and breadth of implemented features. Furthermore, time constraints-imposed limitations on the inclusion of certain advanced functionalities, posing challenges during project execution.

7.3 Future Enhancements

Looking ahead there are several potential future enhancements that can be considered for Project Sattrra. These include:

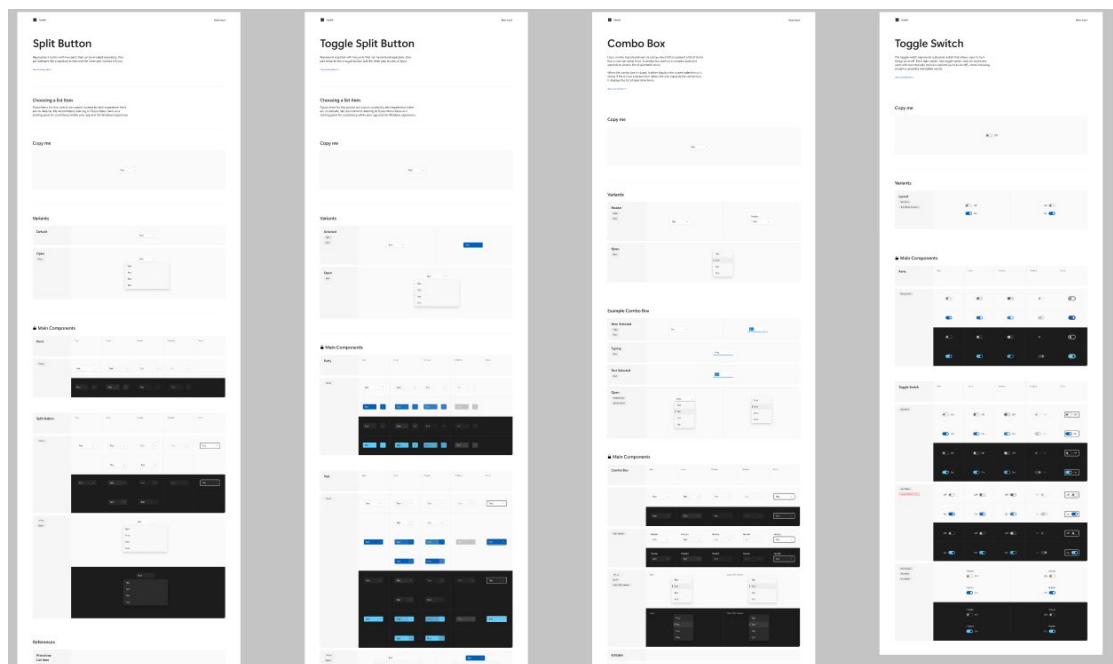
- *Server database setup*: Enhance the application to provide the ability for healthcare facilities to set up their own server database. This allows them to have complete control over their data and ensures data privacy and security.
- *User training and support*: Develop comprehensive user documentation and provide training resources to address the imbalance of knowledge within the team. This will enable healthcare professionals to effectively utilize the application's features and maximize its benefits.
- *Customization options*: Introduce customization options that allow healthcare facilities to tailor the application to their specific needs. This may include configurable fields, workflows, and reporting templates.
- *Performance optimization*: Continuously monitor and optimize the application's performance to ensure smooth and efficient operation, especially as the user base and data volume increase.
- *Community engagement*: Foster a community around the Project Sattrra open-source software, encouraging contributions, feedback, and

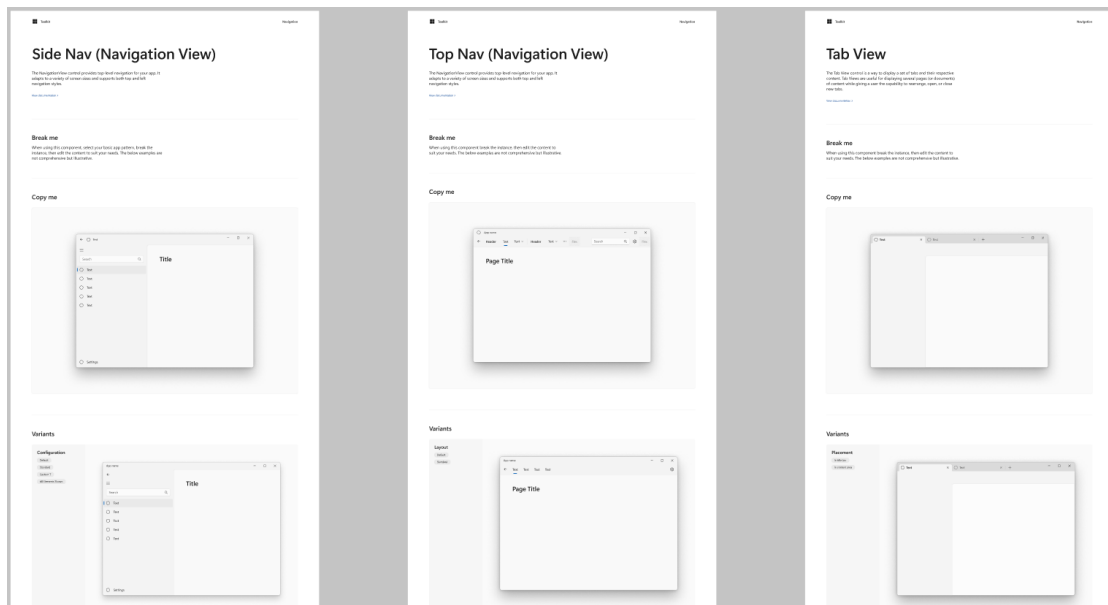
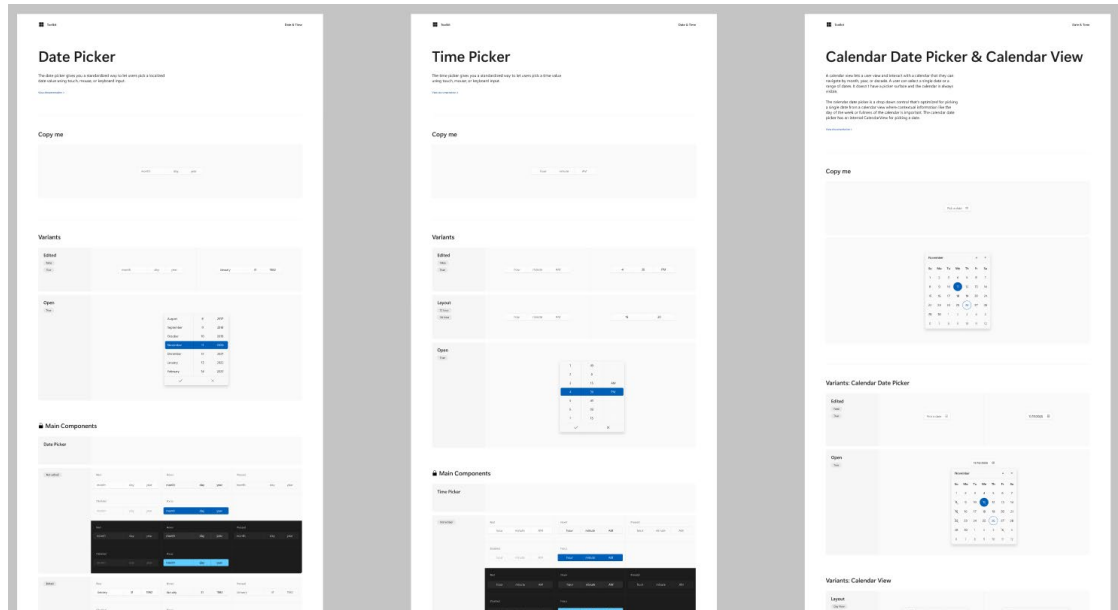
collaboration from developers and healthcare professionals. This can lead to ongoing improvements and wider adoption of the software.

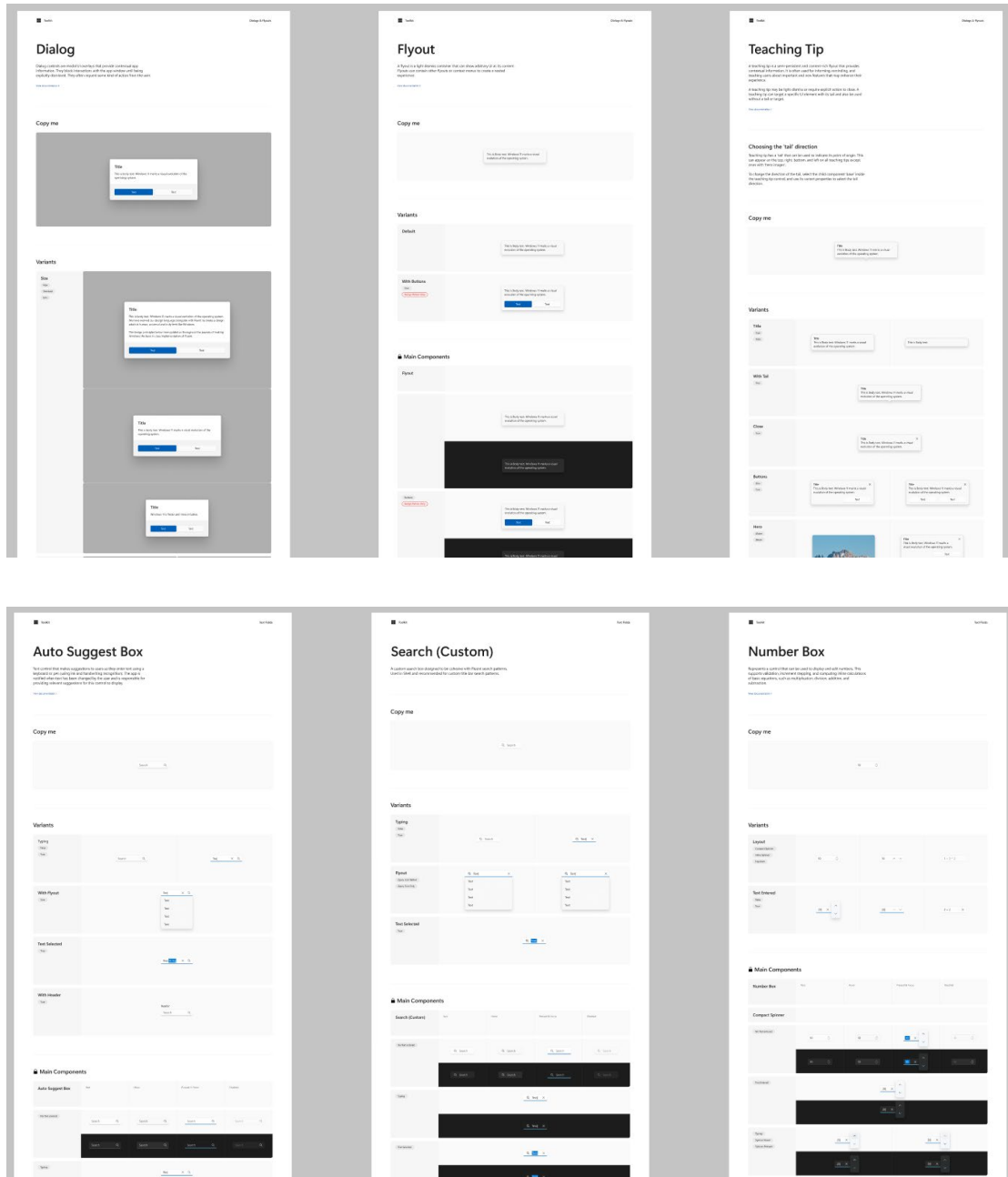
By considering these future enhancements, Project Sattrra can address the limitations faced during the project and continue to evolve as a valuable tool for clinics and small to medium-sized hospitals, ultimately improving patient care and operational efficiency.

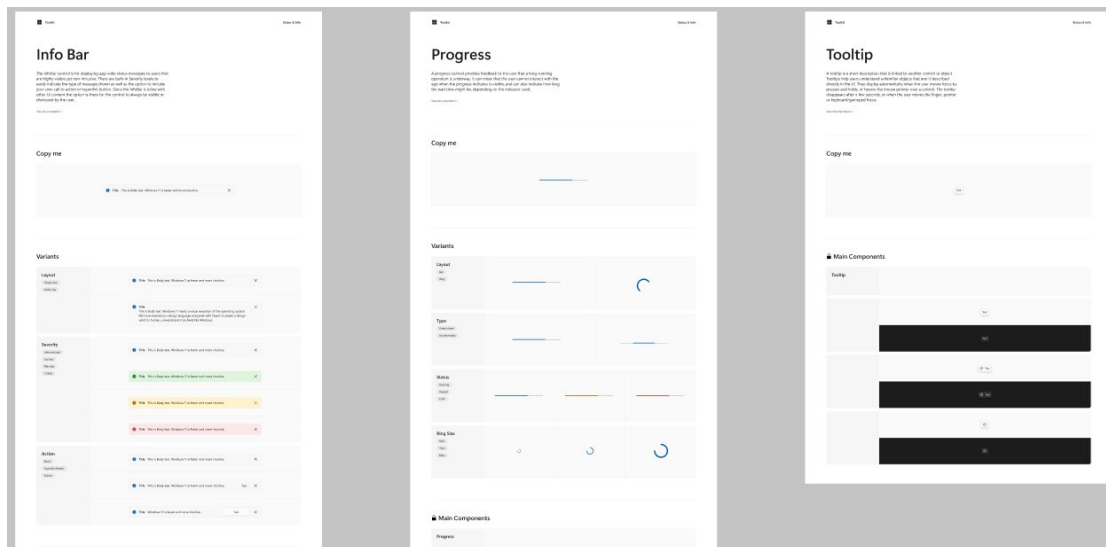
8. Figures

8.1 Fluent UI

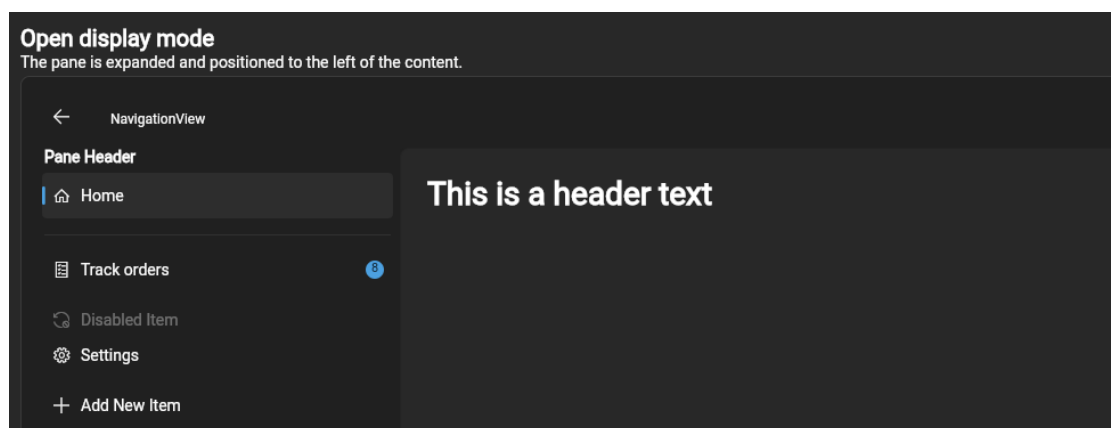
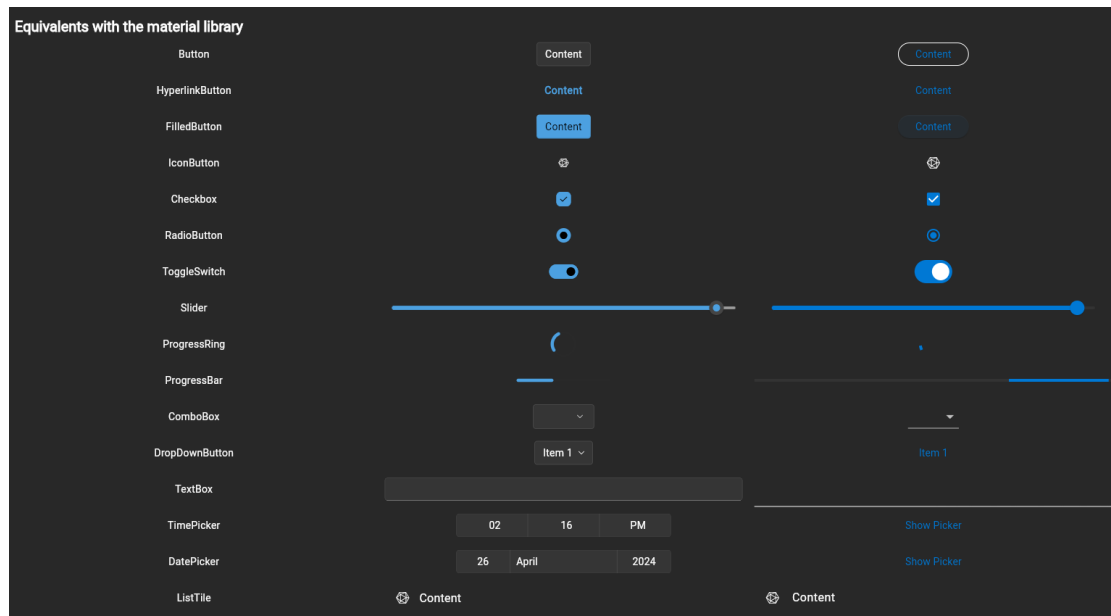


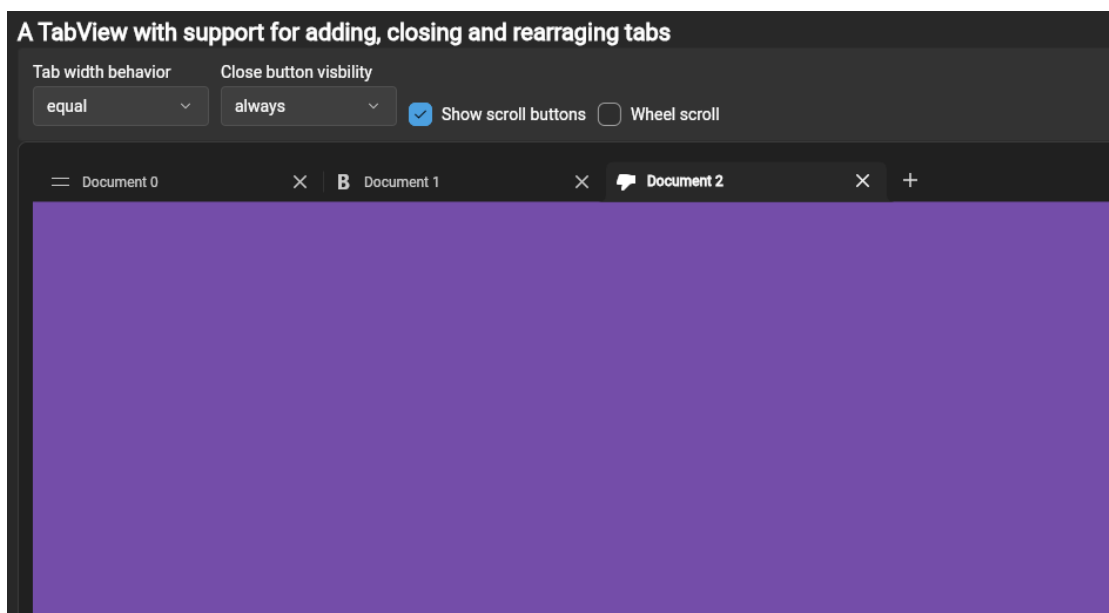
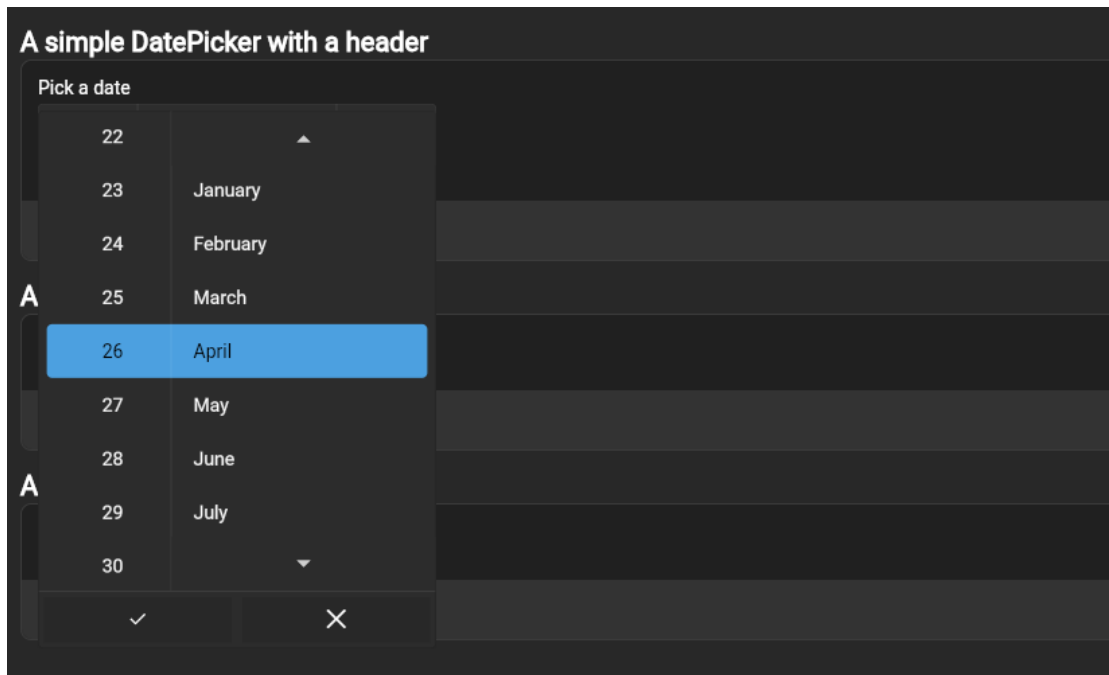






8.2 Custom Flutter Implementation





Simple expander
In this example, the trailing vanishes when the expander is open.

Choose your crost

☐ Classic ☐ Regular
☒ Whole wheat ☐ Thin
☐ Gluten free ☒ Pan
☐ Stuffed

Source code

Scrollable content

Open to see the scrollable text

aliquet id.

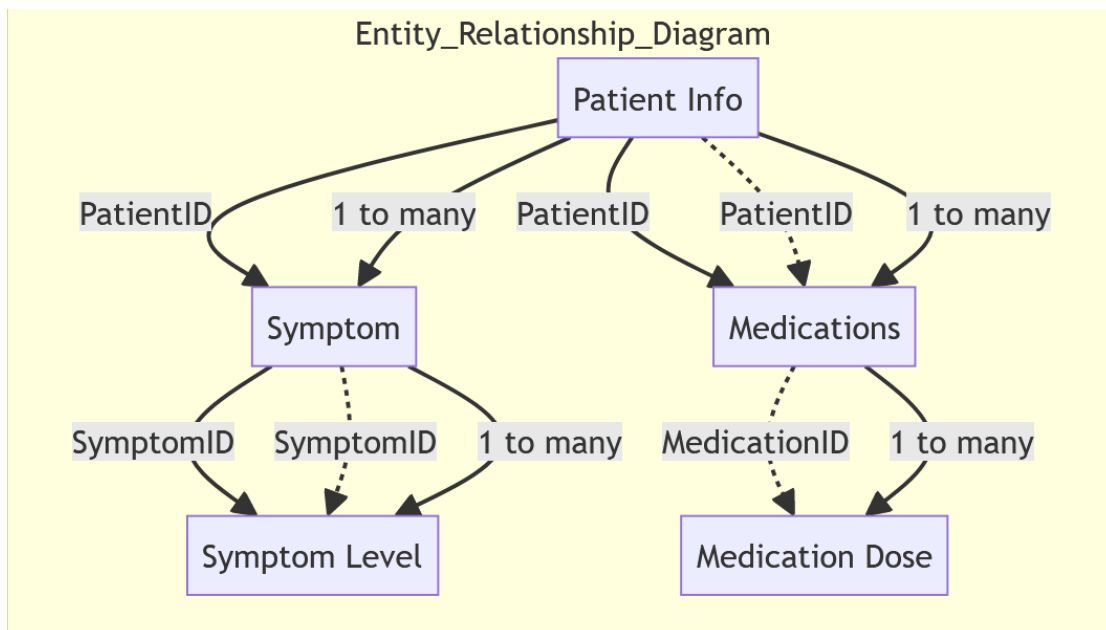
Fusce molestie quis augue vel eleifend. Praesent ligula velit, porta id diam sed, malesuada molestie odio. Proin egestas nisl vel leo accumsan, vel ullamcorper ipsum dapibus. blandit lacus. Suspendisse lacinia augue elit, sit amet auctor eros pretium sit amet. Proin ullamcorper augue nulla, sit amet rhoncus nisl gravida ac. Aenean auctor ligula in nib molestie vulputate diam, id rhoncus augue mattis vitae. Ut tempus tempus dui, in imperdiet elit tincidunt id. Integer congue urna eu nisl bibendum accumsan. Aliquam commo

Donec sit amet semper sem. Pellentesque commodo mi in est sagittis ultricies in ut elit. Donec vulputate commodo vestibulum. Pellentesque pulvinar tortor vel suscipit hend pharetra nisl, in volutpat sapien ipsum in velit. Donec gravida erat tellus, et molestie diam interdum sed. Pellentesque habitant morbi tristique senectus et netus et malesuada pellentesque molestie, tempor ut enim. Duis suscipit massa sed dolor suscipit mollis. In lobortis efficitur egestas. Integer blandit, dolor eu tristique mollis, lorem urna convallis vitae luctus urna. Nunc vehicula sagittis risus, vitae pretium lacus ornare semper. In ornare massa vitae odio consequat, eu lacinia mi imperdiet. Vivamus at augue diam. Fusce

Fusce tempor, dolor in porttitor porttitor, turpis leo ullamcorper urna, vitae ultrices lorem augue eget nulla. Nulla sodales venenatis tellus quis feugiat. Phasellus sit amet cond tincidunt volutpat ante. Fusce ultrices dui vel lorem tincidunt, in pellentesque ligula luctus. Morbi luctus est vitae eros blandit dictum. Quisque convallis diam sed arcu volutpa Praesent vestibulum viverra risus, nec sollicitudin mi mattis eu. Nulla vestibulum, nibh eget sagittis placerat, elit eros egestas libero, eu luctus justo ante eget tellus. Etiam qui

Fusce nunc neque, imperdiet id justo non, porttitor finibus massa. Ut quis risus quis tellus ultricies accumsan et et lorem. Nam pulvinar luctus velit, ut vehicula neque sagittis i justo feugiat leo, sit amet tempus est justo eu augue. Cras eget nibh ac enim bibendum lobortis. Sed ultricies nunc elit, imperdiet consectetur velit scelerisque eu. Aliquam sus

8.3 Database Formulation



- Patient Data

INFORMATION	DATA TYPE
NAME	Text
AGE	Number
SEX	Toggle
WEIGHT	Number
ADDRESS	Text
CHIEF COMPLAINTS	Text

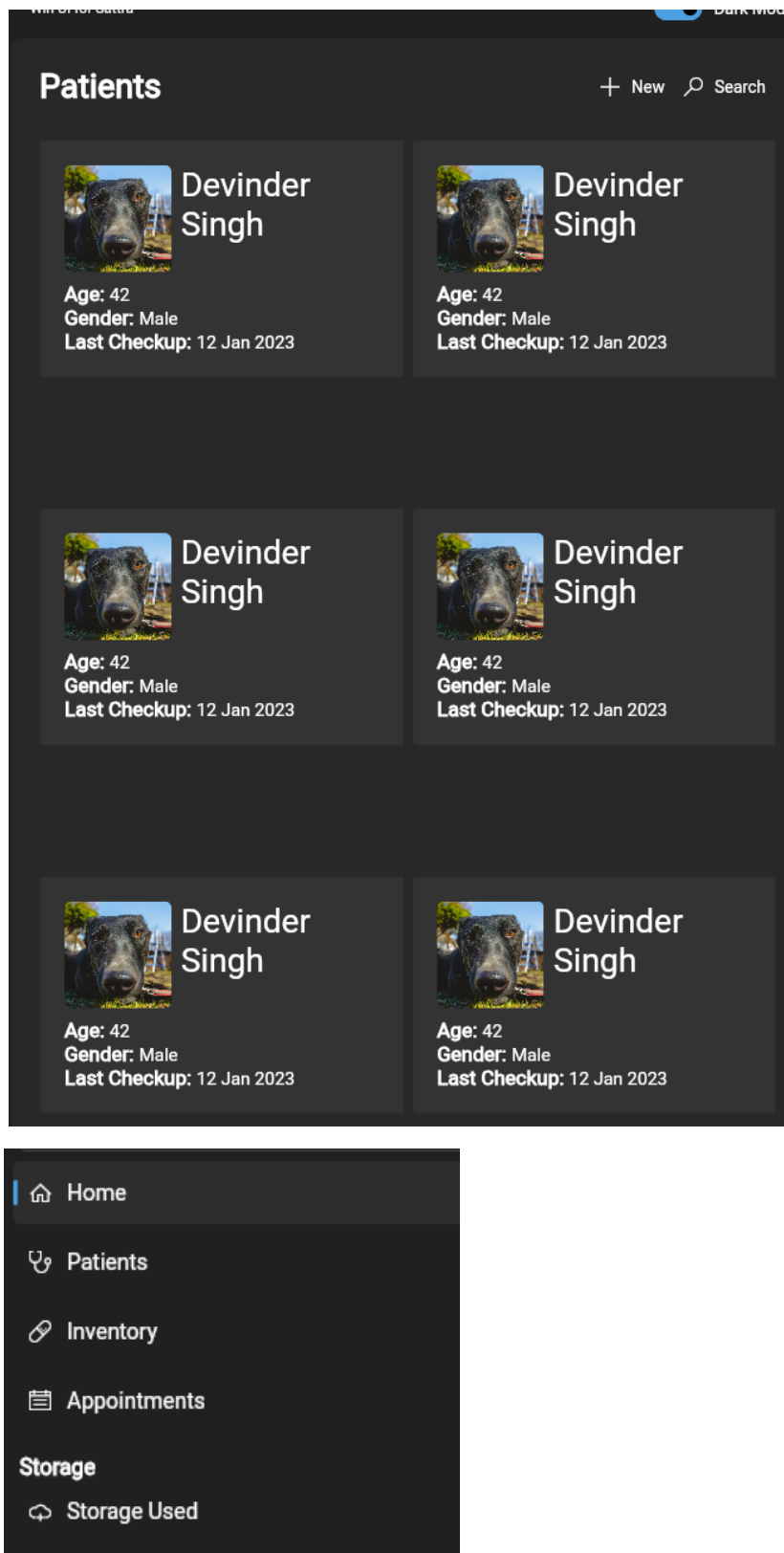
- Sample Symptom Notation

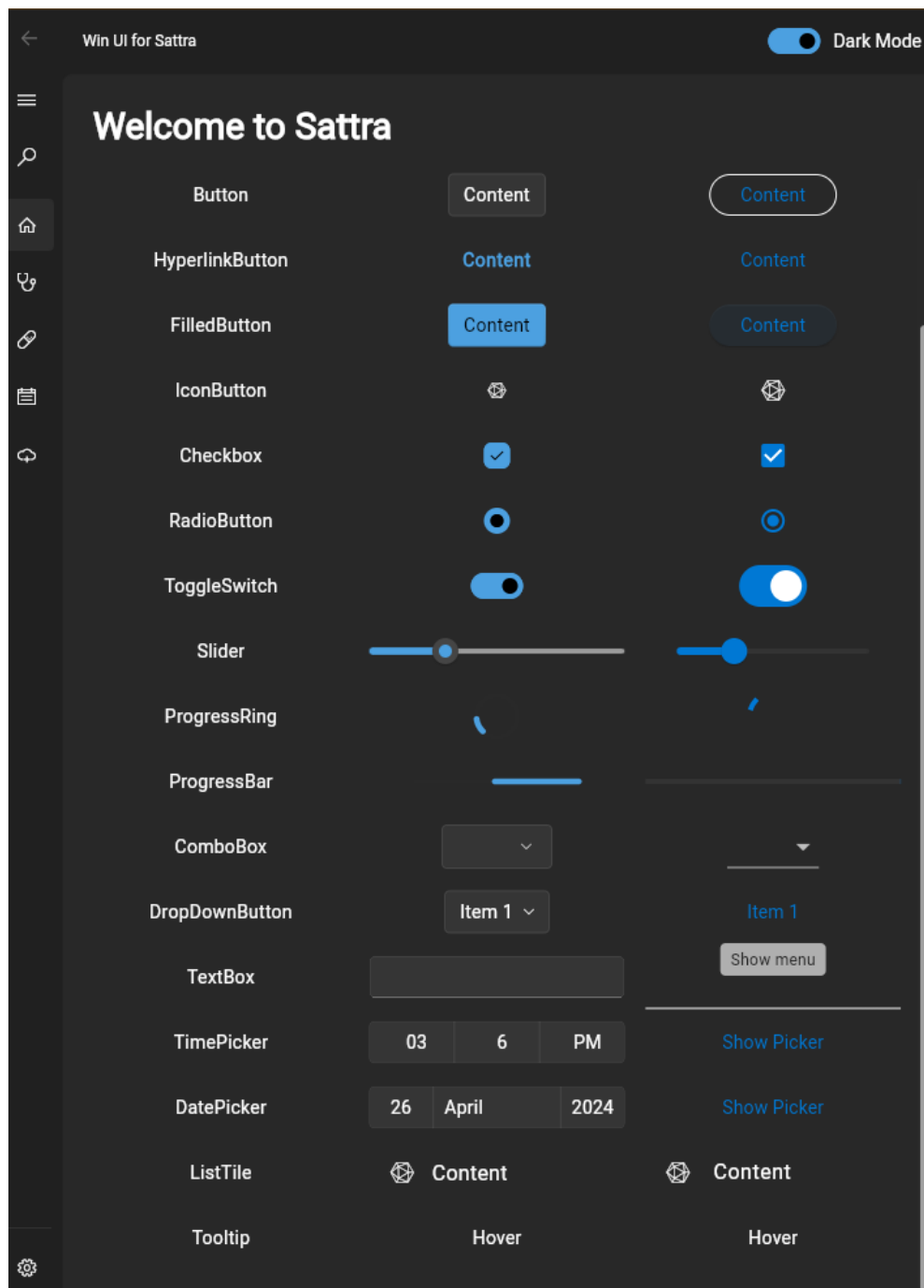
SYMPTOM	LEVEL
SLEEP	normal, disrupted, heavy
BOWEL	normal, constipation, frequent
APPETITE	normal, disrupted, heavy
DIGESTION	normal, Scanty, heavy
STRESS	normal, heavy, poor
MICTURITION	normal, frequent, low
TOLERANCE	normal, hot, cold
MENSTRUATION	normal, Scanty, heavy
LMP	Date

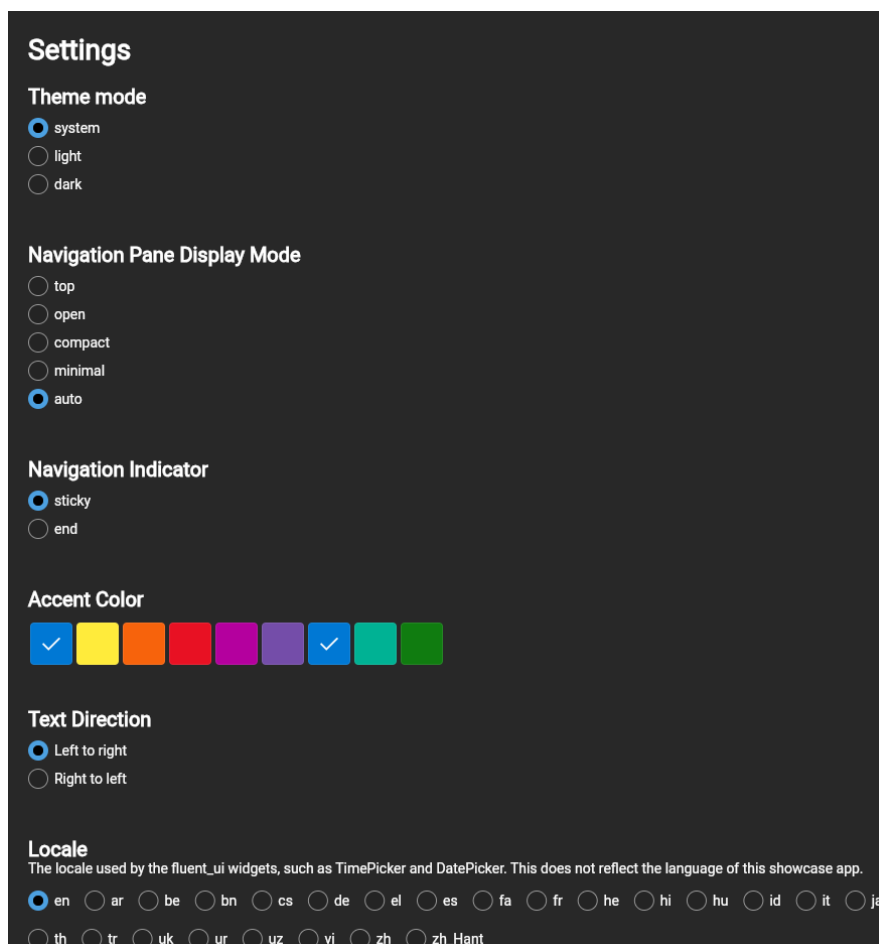
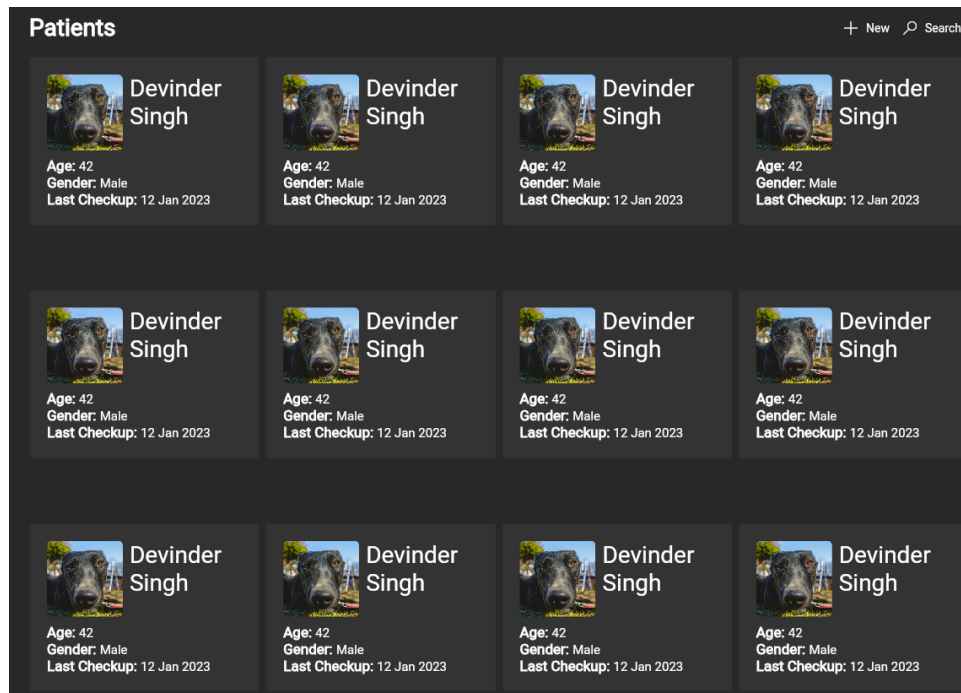
- **Medications**

MEDICATIONS	DOSE
ADHYA TABLET	OD, BD, Hs, TDS, EOD
AWIPATTI TABLET	OD, BD, Hs, TDS, EOD
MADHUPARNI TABLET	OD, BD, Hs, TDS, EOD
LEKHNIYA TABLET	OD, BD, Hs, TDS, EOD
VRAHKARNI TABLET	OD, BD, Hs, TDS, EOD
RAKTA SHODAK TABLET	OD, BD, Hs, TDS, EOD
MEDHYA RASAYAN TABLET	OD, BD, Hs, TDS, EOD
CALVERA TABLET	OD, BD, Hs, TDS, EOD
PUNARNAVA CAPSULE	OD, BD, Hs, TDS, EOD
MAARKHAV CAPSULE	OD, BD, Hs, TDS, EOD
CYSTOLVE TABLET	OD, BD, Hs, TDS, EOD
CYSTOLVE CAPSULE	OD, BD, Hs, TDS, EOD


8.4 Application Sneak Peeks







8.5 Code Showcase



```

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider.value(
      value: _appTheme,
      builder: (context, child) {
        final appTheme = context.watch<AppTheme>();
        return FluentApp.router(
          title: appTitle,
          themeMode: appTheme.mode,
          debugShowCheckedModeBanner: false,
          color: appTheme.color,
          darkTheme: FluentThemeData(
            brightness: Brightness.dark,
            accentColor: appTheme.color,
            visualDensity: VisualDensity.standard,
            focusTheme: FocusThemeData(
              glowFactor: is10footScreen(context) ? 2.0 : 0.0,
            ),
          ),
          theme: FluentThemeData(
            accentColor: appTheme.color,
            visualDensity: VisualDensity.standard,
            focusTheme: FocusThemeData(
              glowFactor: is10footScreen(context) ? 2.0 : 0.0,
            ),
          ),
          locale: appTheme.locale,
          builder: (context, child) {
            return Directionality(
              textDirection: appTheme.textDirection,
              child: NavigationPaneTheme(
                data: NavigationPaneThemeData(
                  backgroundColor: appTheme.windowEffect !=
                    flutter_acrylic.WindowEffect.disabled
                    ? Colors.transparent
                    : null,
                ),
              child: child!,
            ),
          );
        },
        routeInformationParser: router.routeInformationParser,
        routerDelegate: router.routerDelegate,
        routeInformationProvider: router.routeInformationProvider,
      );
    },
  );
}

```

8.6 Database Tools

postgres 3.2.0

Published 15 days ago Dart 3 compatible

[SDK](#) [DART](#) [FLUTTER](#) [PLATFORM](#) [ANDROID](#) [IOS](#) [LINUX](#) [MACOS](#) [WINDOWS](#)

👍 291

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

PostgreSQL client

 Dart passing

A library for connecting to and querying PostgreSQL databases (see [Postgres Protocol](#)). This driver uses the more efficient and secure extended query format of the PostgreSQL protocol.

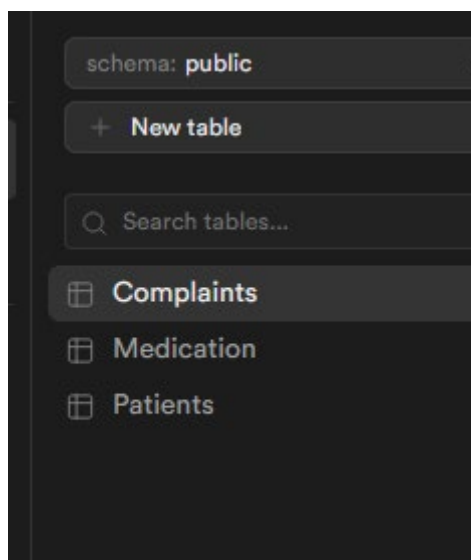
Usage

Create a `Connection`:

```
final conn = await Connection.open(Endpoint(  
  host: 'localhost',  
  database: 'postgres',  
  username: 'user',  
  password: 'pass',  
));
```

Execute queries with `execute`:

```
final result = await conn.execute("SELECT 'foo'");  
print(result[0][0]); // first row and first field
```



Project Settings

General settings

Project name

Proj Sattrra

Reference ID

erqxahxsdontoopatsx

Copy

Cancel

Save

Restart project

Your project will not be available for a few minutes.

Restart project

Pause project

Your project will not be accessible while it is paused.

Pause Project

Project usage statistics has been moved

You may view your project's usage under your organization's settings

View project usage

Database Settings

Connection string

Documentation

URI PSQL Golang JDBC .NET Nodejs PHP Python

☒ Display connection pooler

Mode: Session

Supervisor

Resolves to IPv4

PgBouncer pending removal

postgres://postgres.erqxahxsdontoopatsx:[YOUR-PASSWORD]@aws-0-ap-south-1.pooler.supabase.com:5432/postgres

Copy

How to connect to a different database or switch to another user

Connection parameters

☒ Display connection pooler

Mode: Session

Supervisor

Resolves to IPv4

PgBouncer pending removal

Host

aws-0-ap-south-1.pooler.supabase.com

Copy

Database name

postgres

Copy

Port

5432

Copy

User

postgres.erqxahxsdontoopatsx

Copy

Password

[The password you provided when you created this project]

#	oid	datname	datdba	encoding	datlocprovider	datistemplate	dataallowconn
1	5	postgres	10	6	c	f	t
2	1	template1	10	6	c	t	t
3	4	template0	10	6	c	t	f
4	16386	neondb	16385	6	c	f	t