

# FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

# SEMESTER 2 2020/2021

# BITU 3923 WORKSHOP 2

# FINAL REPORT

# PROJECT TITLE : GOLDEN YEAR CARE CONNECT

|  |  |
| --- | --- |
| **NAME** | **MATRIC NO** |
| NOR AMIRA HUSNA BINTI ABDUL RAHIM | B032310275 |
| NUR AQILAH BINTI ZAIDI | B032310148 |
| NUR AINA SOFEA BINTI AHMAD NAZZIB | B032310108 |
| MUHAMMAD ZULHELMI BIN NOOR AFENDI | B032310217 |
| MUHAMMAD ZULHUSNI BIN  MUHAMMAD ZAIMI | B032310206 |

**SUPERVISE BY DR. EMALIANA BINTI KASMURI**

|  |  |
| --- | --- |
| **CONTENTS** | **PAGE** |
| Chapter 1 : **INTRODUCTION**  1.1 Introduction  1.2 Problem Statement  1.3 Objective  1.4 Scope  1.5 Project Significant  1.6 Conclusion | 4 - 8 |
| Chapter 2: **SYSTEM METHODOLOGY**  **AND PLANNING**  2.1 Introduction  2.2 Agile Methodology  Methodology  2.3 Project Management  2.3.1. Project Milestones | 9 - 10 |
| Chapter 3 : **SYSTEM ANALYSIS**  3.1 Introduction  3.2 Current System Analysis  3.3 System To-Be Analysis  3.3.1 Context Diagram  3.3.2 Data Flow Diagram | 11 - 16 |
| Chapter 4 : **SYSTEM DESIGN**  4.1 Introduction  4.2 Conceptual Database Design  4.3 Logical Database Design  4.3.1 Data Dictionary  4.4 Physical Database Design  4.4.1 Data Definition Language | 17 - 37 |
| Chapter 5 : **INDIVIDUAL**  **APPLICATION MODULE**  **IMPLEMENTATION**  5.1 Introduction  5.2 User Interface  5.2.1 User Page  5.2.2 Module Flower Inventory  5.2.3 Module Ordering  5.2.4 Module Delivery  5.2.5 Module Feedback  5.3 Programming Technique  5.4 Error Handling  5.5 Conclusion | 38 - 63 |
| Chapter 6 : **DATABASE INTEGRATION**  **AND TESTING**  6.0 Introduction  6.1 Database Installation  6.1.1 Install XAMPP  6.1.2 Setup MySQLBackup  6.2 Database Administration  6.2.1 Backup  6.2.2 Restore  6.4 Conclusion | 64 - 76 |
| Chapter 7 : **CONCLUSION**  7.1 Introduction  7.2 Achievement  7.3 Project Limitation  7.4 Suggestion and Improvement  7.5 Potential Commercialization  4.6 Conclusion | 77 - 78 |

### CHAPTER 1 INTRODUCTION

#### 1.1 Introduction

The number of senior citizens is growing quickly around the world, including in Malaysia. Senior citizens, aged 60 and above, often face challenges such as health problems, difficulty moving around, and loneliness. This highlights the need for improved caregiving services to better support them.

Golden Years Care Connect is a caregiving system and mobile app designed to assist senior citizens and their families. It offers features such as personalized care plans, booking appointments for activities, medication reminders, activity tracking, and emergency alerts. The app also aims to help digitalize caregiving management, making it more organized and accessible.

This platform strives to simplify caregiving, improve management practices, and foster a safer and more supportive environment for senior citizens. Golden Years Care Connect seeks to enhance their quality of life and make caregiving more manageable for everyone involved.

#### 1.2 Problem Statement

1. The increasing number of senior citizens in Malaysia highlights the need for better caregiving services to address their unique challenges, such as health issues, mobility limitations, and social isolation.
2. Caregivers struggle to organize schedules, track activities, and handle emergencies without proper tools.
3. There is no centralized system to make caregiving easier and more efficient for seniors and caregivers.

#### 1.3 Objectives

1. To create a digital tool to help seniors manage daily tasks, medications, and appointments with reminders and tracking.
2. To support caregivers in providing care through simple digital tools.
3. To ensure user data is secure and make the system easy to use for both seniors and caregivers.

1.4 **Scope Of The Project**

### ● Authentication and User Management Module

This module allows users to log in and out securely. If a user forgets their password, they can contact the admin by email to reset it. Only authorized users, such as caregivers and admins, can access the system.

### ● User Management Module

This module helps admins manage caregiver profiles. Admins can add, update, soft-delete, and search for caregiver profiles. It also provides a summary of caregiver information to make managing them easier.

### ● In-App Module

Senior citizens can access the app through their guardian’s login (one-time login by their guardian). If a senior citizen is in a care centre, they don’t need to worry about logging in, as caregivers will handle updates on their behalf. Seniors can use the system freely to view information without needing to manage the account themselves.

### ● Services Management Module

### This module allows admins to manage the services offered by the company. New services can be added as the business grows or updates its offerings. This ensures the system can adapt to future needs.

### ● Care Plan Management Module

This module helps caregivers manage care plans for senior citizens. Caregivers can use the web app to add, update, or delete care plans based on client needs. It also makes it easy to search for specific care plans and view a summary of services provided to each client.

### ● Care Log Management Module

This module allows caregivers to update care logs with details of activities that are completed or cancelled. Admins can assign caregivers to clients manually, ensuring each client is paired with the right caregiver. The care log keeps a record of activities for monitoring and reporting.

### ● Alert and Notification Module

### This module sends notifications to keep everyone updated. Anomaly alerts, such as emergencies, are sent through push notifications, SMS, or WhatsApp of the senior citizen. Task and appointment reminders are also sent to ensure nothing is missed.

### ● Dashboard and Summary Module

This module gives admins and caregiver an overview of the system. It shows key metrics, like the number of clients, caregivers, and recent activities, such as new care plans. The dashboard helps admins monitor the system and make decisions more easily.

1.5 **Project Significance**

**The Golden Years Care Connect** project focuses on improving caregiving by digitalizing processes and offering an app-based solution. It supports senior citizens, caregivers, and families in the following ways:

1. **Caregiving from Home.** The app allows senior citizens to receive care from their own homes. Features like reminders for appointments, medications, and activities help them stay organized and supported.
2. **Simplifying Caregiving Processes**. By replacing manual methods with a digital platform, tasks such as managing care plans, updating activities, and tracking health become more efficient and accessible.
3. **Easy Access for Seniors.** Senior citizens can use the app to view their care information, such as schedules and reminders. For those in care centres, caregivers handle updates, allowing seniors to focus on using the app to stay informed.
4. **Emergency Alerts and Notifications**. The app provides real-time notifications for emergencies and sends reminders for medications and appointments. This helps reduce missed tasks and ensures quick responses when needed.

#### 

#### 1.6 Conclusion

In this chapter, the foundation of the Golden Years Care Connect project was established, highlighting its purpose, scope, and significance. The project addresses the growing need for efficient and accessible caregiving by digitalizing traditional processes and introducing an app-based solution. By focusing on senior citizens, caregivers, and families, the project aims to simplify caregiving tasks, provide real-time notifications, and enhance communication.

The system's modules, such as care plan management, reminders, and emergency alerts, ensure senior citizens receive personalized support whether at home or in care canters. Additionally, its scalability allows the platform to adapt to future caregiving needs. Overall, this project contributes to improving the quality of care and promoting a more connected and supportive caregiving experience.

### CHAPTER 2 PROJECT METHODOLOGY AND PLANNING

#### 2.1 Introduction

The Agile methodology is a software development and project management technique that prioritizes adaptability, teamwork, and customer focus. It is the most recent model that big businesses like Facebook, Google, and Amazon are using. This method iteratively develops the essential features, emphasizing feedback and ongoing enhancement.

The Agile Development Phase is essential in creating a high-quality, user-focused product like *Golden Years Care Connect*. This phase breaks down the development process into manageable, iterative cycles called **sprints**. Agile allows for flexibility, continuous feedback, and improvement, which is especially important when developing a caregiving app that must be reliable, secure, and user-friendly.

#### 2.2 Agile Methodology

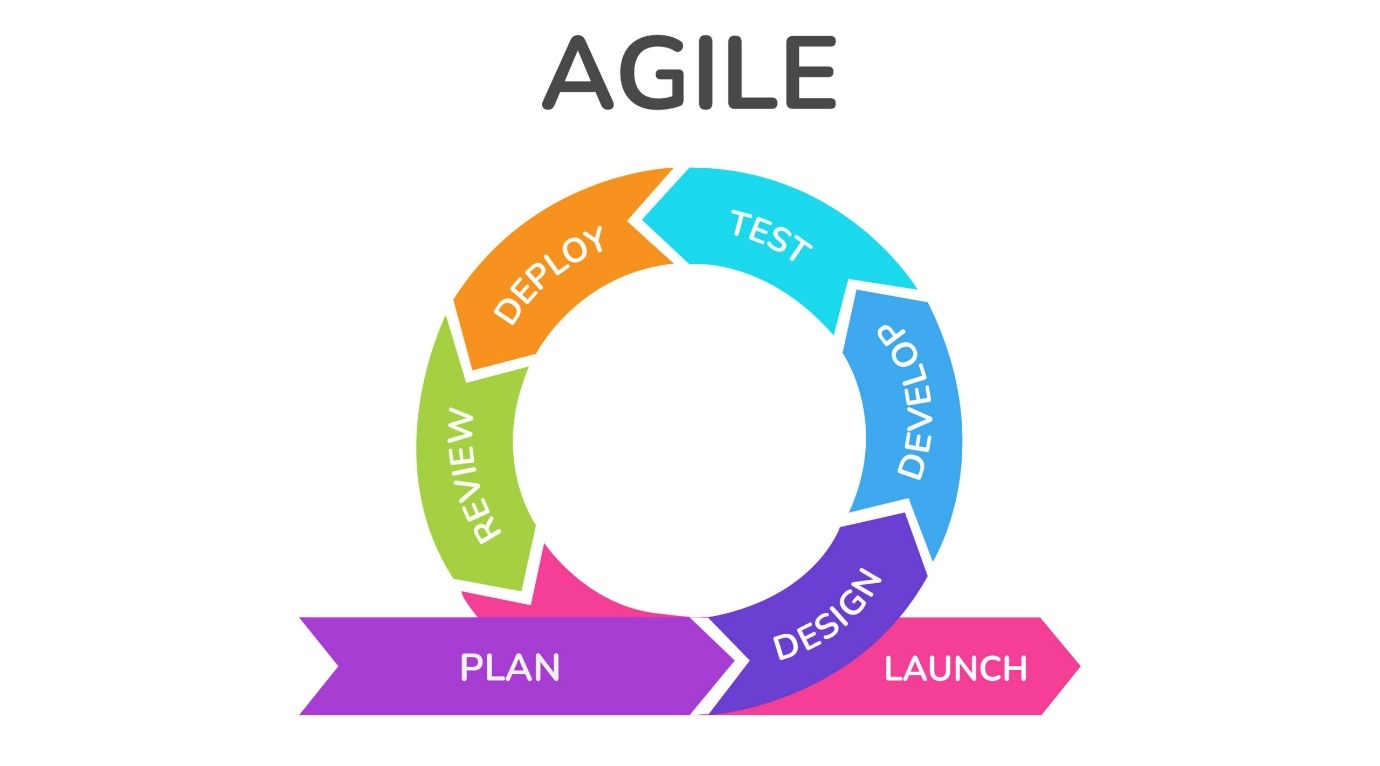


Figure 2. Agile Methodology

1. Sprint Planning

Sprint planning is a critical step in the development process, where the team defines what will be achieved in the upcoming sprint, typically spanning one to two weeks. The primary goal is to set clear, achievable objectives, focusing on a particular set of features to be developed. To achieve this, the chosen feature is broken down into user stories, which are concise, actionable objectives written from the user’s perspective. For instance, a user story might state: “As a caregiver, I want to receive an emergency alert notification so I can respond promptly.” Tasks are then assigned to team members based on their expertise and the sprint’s priorities.

1. Development and Testing

During the development and testing phase, developers work to build the selected features according to the requirements established in the sprint planning session. Code is regularly integrated into a shared repository to ensure compatibility and to catch any issues early. Each piece of code is rigorously tested for functionality and correctness. For example, an app’s emergency alert function would be tested to confirm that notifications are sent as expected. Automated tests are also utilized to quickly verify fundamental functionalities, such as app navigation or data entry forms, ensuring a stable and reliable product.

1. Regular Refinement (Backlog Grooming)

To maintain focus and adaptability, the team engages in regular refinement of the product backlog—a comprehensive list of all tasks, features, and potential improvements. This continuous grooming process ensures that the team prioritizes the most valuable and feasible items. It also helps the team understand the requirements and specifications for upcoming features, making adjustments based on new insights or lessons learned from previous sprints. If necessary, sprint goals may be modified to better align with evolving project needs.

1. Sprint Retrospective

At the end of each sprint, the team participates in a sprint retrospective. This reflective session provides an opportunity to review the sprint’s successes and challenges to improve workflows for future sprints. The team discusses successful aspects, such as efficient collaboration or features that received positive feedback, and identifies any obstacles encountered, such as technical issues or unclear requirements. Together, the team brainstorms solutions to address these challenges, ensuring continuous improvement in their development process.

The Agile Development Phase focuses on building *Golden Years Care Connect* iteratively, allowing the team to prioritize essential features and incorporate feedback regularly. Each sprint produces a functional increment of the app, bringing it closer to meeting the specific needs of seniors, caregivers, and families. By the end of Agile development, the team will have a robust, well-tested platform that reflects user needs, ensuring a successful product launch.

#### 2.3 Project Gantt Chart

#### 2.3.1 Project Milestones

There are specific schedules and milestone that have to be followed according to the committee during implementation of Workshop. This milestone is to ensure the project completely followed the duration given.

A screenshot of a computer

Description automatically generated  
Table 2.1 Workshop2 Milestones

### Chapter 3 DATABASE SYSTEM ANALYSIS

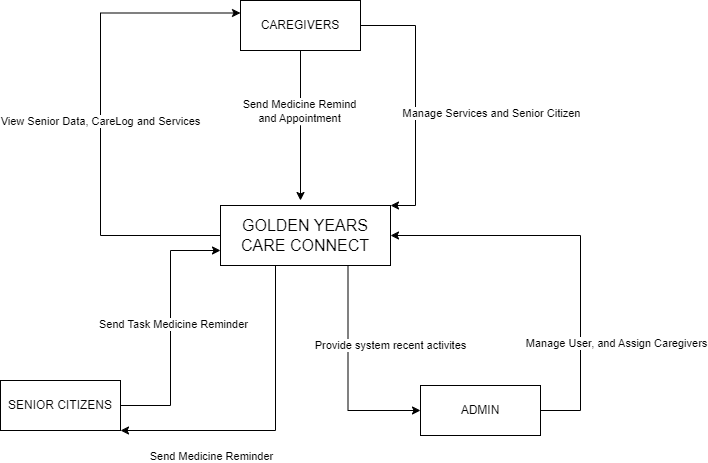
3.1 **Introduction**

The database system plays a critical role in managing and storing data for the application, ensuring data consistency, scalability, and efficiency. For this project, Firebase Realtime Database was chosen due to its cloud-hosted, NoSQL structure, which provides real-time data synchronization and seamless integration with Flutter, the framework used for building the application.

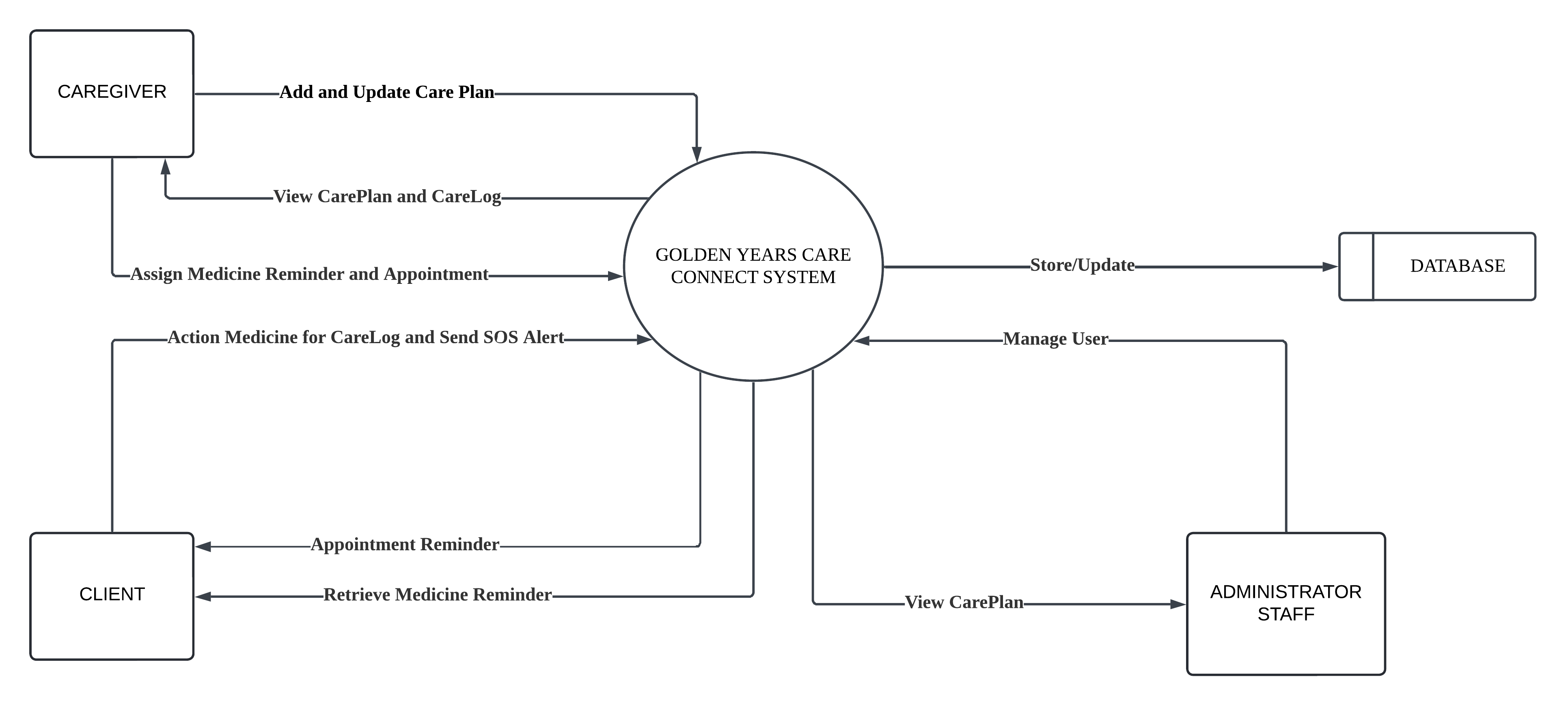
We use Firebase Realtime Database to manage and store data in real-time, ensuring smooth communication and reliable operations. The database structure supports key features like care plans, user profiles, reminders, care logs, and service management, all tailored to meet the needs of different users. By combining efficient data organization with real-time updates, the system ensures secure, scalable, and seamless operations across both web and mobile platforms.

3.2 **System Analysis**

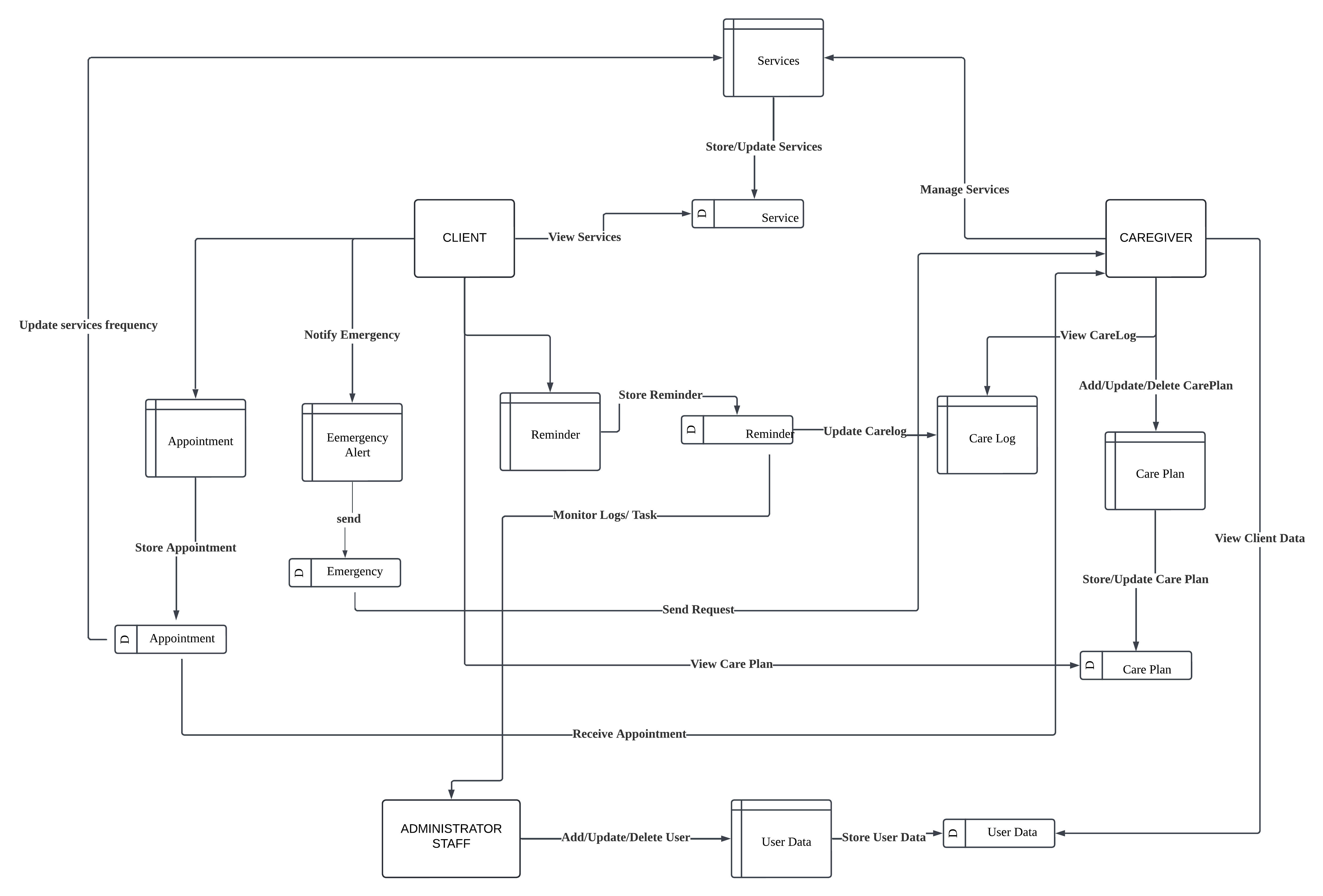
3.2.1 **Context Diagram**



3.2.2 Data Flow Diagram for Golden Years Care Connect System (Level 0)



3.2.2 Data Flow Diagram for Golden Years Care Connect System (Level 1)



**Chapter 4 DATABASE DESIGN**

4.1 **Introduction**

System design is a crucial phase in the development of the Golden Year Connect (GYCC) system. This chapter outlines the database design process, covering conceptual, logical, and physical database design. It ensures data integrity, consistency, and efficient data management.

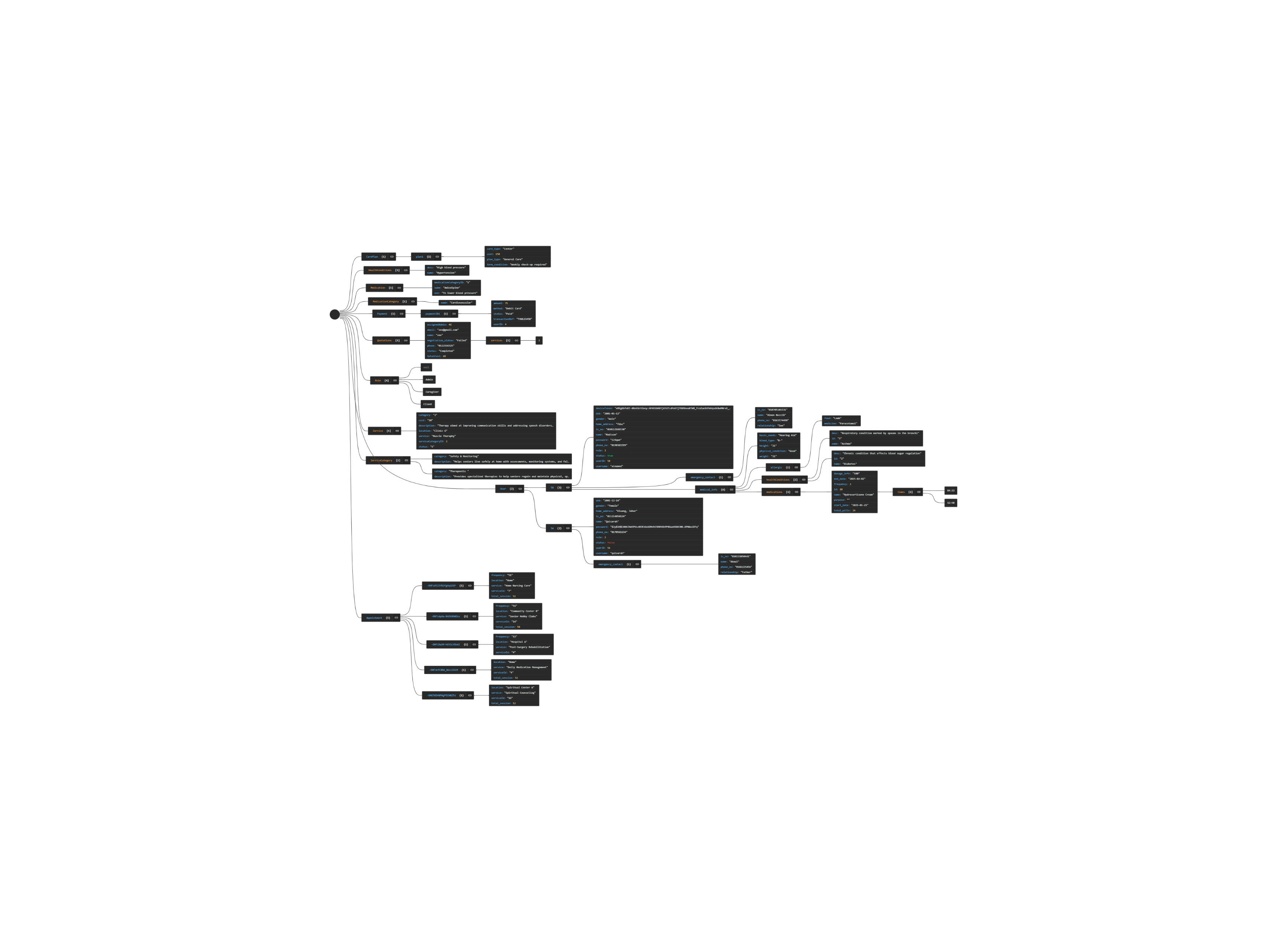
4.2 **Conceptual Database Design**

The conceptual database design represents the high-level structure of the database, defining key entities and relationships. Unlike traditional relational databases, the GYCC system utilizes Firebase Realtime Database, a NoSQL cloud-based solution that stores data in a JSON tree structure. This design ensures flexibility, scalability, and real-time synchronization across multiple users and devices.

Due to Firebase’s NoSQL nature, entities are structured hierarchically rather than relationally through tables. The system's primary entities include Users, Care Plans, Services, Appointments, and Payments, each represented as a top-level node in the JSON database. The relationships among these entities are managed through nested structures and unique identifiers, allowing efficient data retrieval while reducing redundancy.

**Key Entities in Firebase Structure:**

1. **Users :** Stores user details such as name, contact, and role (admin, caregiver, or client). Each user has a unique identifier (UID) from Firebase Authentication.
2. **Care Plans :** Defines in-center and residential care services available for patients, linked to services via their respective IDs.
3. **Services :** Represents various healthcare and support services provided under care plans, stored as separate child nodes.
4. **ServiceCategory :** Categorizes different types of services, such as medical, therapy, or home assistance, ensuring structured classification.
5. **Appointments :** Manages scheduled services between caregivers and patients, including date, time, and assigned provider. Appointments are stored under each user’s ID for quick access.
6. **Payments :** Tracks transaction details related to services provided, including amount, payment status, and method. Payment records are linked to the respective user and service.
7. **Medication :** Stores details about prescribed medicines, including name, dosage, and instructions for use, structured per user.
8. **MedicationCategory :** Categorizes medications based on their type or usage, such as pain relievers, antibiotics, or supplements, allowing better filtering.
9. **HealthConditions :** Maintains records of patients’ medical conditions, diagnoses, and relevant health history.
10. **Quotations :** Generates cost estimates for services or care plans before finalizing payments. Stored under each user's record.
11. **Role :** Defines user roles within the system, such as Admin, Caregiver, or Client, determining access permissions and system functionalities.



Picture 4.1 Firebase Structure

4.3 **Logical Database Design**

The logical database design translates the conceptual model into a structured schema that supports data retrieval and manipulation in Firebase. Since Firebase follows a NoSQL structure, data is stored in a hierarchical JSON format instead of tables.

4.3.1 **Data Dictionary**

The following Data Dictionary have been simplified from JSON format represents the Firebase database node and its property:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A specific health condition. Note: First element is null | Object |  | index starts at 1; if it is 'null', no condition specified. |
| desc | Description of the health condition. | String | "High blood pressure" |  |
| name | Name of the health condition. | String | "Hypertension" |  |

Table 4.1 : Data Dictionary For HealthConditions Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A medication item. Note: First element is null. | Object |  | index starts at 1; if it is 'null', no condition specified. |
| medicationCategoryID | Category ID of the medication. | String | “1” | Must match a MedicationCategory id |
| name | Name of the medication. | String | "Amlodipine" |  |
| use | Description of the medication's use. | String | "To lower blood pressure" |  |

Table 4.2 : Data Dictionary For Medication Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A medication category. Note: First element is null | Object |  | index starts at 1; if it is 'null', no condition specified. |
| name | Name of the medication category. | String | "Pain Relief" |  |

Table 4.3 : Data Dictionary For MedicationCategory Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A payment list item. Note: First element is null | Object |  | index starts at 1; if it is 'null', no condition specified. |
| amount | Amount of the payment | Number | 75 |  |
| method | Method of payment. | String | “Debit Card” |  |
| status | Status of the payment | String | “Paid” |  |
| transactionRef | Reference number for the transaction. | String | "TXN123458" |  |
| userID | ID of the user associated with the payment. | number | 4 | Must match a User key |

Table 4.4 : Data Dictionary For Payment Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A quotation entry. Note: First element is null | Object |  | index starts at 1; if it is 'null', no condition specified. |
| assignedAdmin | ID of the admin assigned to the quotation. | Number | 46 | Must match a User key with role = 1 |
| email | Email address associated with the quotation. | String | "[zex@gmail.com](https://www.google.com/url?sa=E&q=mailto%3Azex%40gmail.com)" |  |
| name | Name associated with the quotation. | String | "zex" |  |
| negotiation\_status | Negotiation status of the quotation. | String | "Failed" |  |
| phone | Phone number associated with the quotation. | String | "0112334325" |  |
| services | Array of service IDs associated with the quotation. | List (String) | ["2"] | Must match Service key |
| status | Status of the quotation. | String | "Completed" |  |
| totalCost | Total cost of the quotation | Number | 69 |  |

Table 4.5 : Data Dictionary For Quotations Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A user role | String | “Admin”, ”Caregiver”, “Client” | First element is null, index starts at 1 |

Table 4.6 : Data Dictionary For Role Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A service item. Note: First element is null | Object |  | index starts at 1; if it is 'null', no service specified. |
| category | Category of the service | Number | 2 | Must match ServiceCategory id |
| cost | Cost of the service. | Number | 10 |  |
| description | Description of the service. | String | "Therapy aimed at improving communication skills and addressing speech disorders." |  |
| location | Location where the service is provided. | String | "Clinic G" |  |
| service | Name of the service. | String | "Muscle Theraphy" |  |
| status | Status of the service. | Boolean | true or false |  |

Table 4.7 : Data Dictionary For Service Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [index] (Array item) | A service item. Note: First element is null | Object |  | index starts at 1; if it is 'null', no service specified. |
| category | Name of the service category. | String | "Safety & Monitoring" |  |
| description | Description of the service category. | String | "Helps seniors live safely at home with assessments, monitoring systems, and fall detection services." |  |

Table 4.8 : Data Dictionary For ServiceCategory Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node/Property | Description | Data Type | Example Value | Constrain/Rules |
| [userID] | Specific user object, where the key is the userID | Object |  | Each property is tied to a specific user |
| deviceToken | Device token for push notifications. | String | "d-FZfjtFR7-x6UR9vAoS4X:..." |  |
| dob | Date of birth of the user. | String | "1950-01-01" |  |
| email | Email address of the user. | String | "naqid@gycc.com" |  |
| gender | Gender of the user. | String | “male” or “female” |  |
| home\_address | Address of the user. | String | "789 Third St, Penang, Malaysia" |  |
| ic\_no | Identification card number. | String | “020708050123" |  |
| name | Name of the user. | String | "Naqiuddin Mohamad" |  |
| password | Password of the user. | String | "123qweasd" |  |
| phone\_no | Phone number of the user. | String | "60189196774" |  |
| role | Role of the user. | Number | 1 = admin,  2 = caregiver,  3 = client | Must be a value within the Role list |
| status | Account status. | Boolean | true or false |  |
| username | Username of the user. | String | "naqidrock" |  |
| --[Careplan] | Array of care plans for that user (first element is always null) | List (Object) |  | index starts at 1; if it is 'null', no careplan specified. |
| ----Services | Array of service objects associated with a specific plan | List (Object) |  | References Service node (serviceId) |
| -----frequency | Frequency of service provided within a plan | String | “365” |  |
| -----serviceId | ID of service provided within the plan | String | “7” | Must reference a valid ID from Service node |
| ---care\_type | Type of care provided within a plan | String | “In Center” |  |
| ---caregiverID | ID of caregiver assigned to plan | String | “27” | Must reference a valid ID from User node with role = 2 |
| ---cost | Cost of the care plan (RM) | Number | 5336 |  |
| ---start\_date | Start date of the plan | String | "2025-01-18" |  |
| ---end\_date | End date of the plan | String | "2026-01-19" |  |
| --emergency\_contact | Contact details for emergency |  |  |  |
| ---ic\_no | Emergency contact's IC number | String | "910705101530" |  |
| ---name | Emergency contact's name | String | "Arriya" |  |
| ---phone\_number | Emergency contact's phone number | String | "0192727727" |  |
| ---relationship | Emergency contact's relationship with user | String | "Brother" |  |
| --medical\_info | User's medical information | Object |  |  |
| ---basic\_needs | User's basic needs | String | "Wheelchair" |  |
| ---blood\_type | User's blood type | String | "A+" |  |
| ---height | User's height | String | "159" |  |
| ---weight | User's Weight | String | “50” |  |
| ---physical\_condition | User's physical condition | String | "Good" |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ----healthConditions | Array of health conditions (related to HealthConditions node) | List (Object) |  | References HealthConditions node (id) |
| -----desc | Description of health condition | String | "Hypersensitive reactions to certain substances" |  |
| -----id | ID of health condition | String | "6" | Must reference an ID in HealthConditions node |
| -----name | Name of health condition | String | "Allergies" |  |
| ----medications | Array of user's medications (first element is always null) | List (Object) |  | First element is always null, References Medication node(name) |
| -----dosage | Dosage of medication | String | "500" |  |
| -----frequency | Frequency of medication | String | "2 Times a day" |  |
| -----name | Name of medication | String | "Amlodipine" | Must match a medication name in Medication node |
| -----purpose | Purpose of taking the medication | String | "To lower blood pressure" |  |

Table 4.9 : Data Dictionary For User Node

4.4 **Physical Database Design**

The physical database design for Firebase involves structuring data to optimize reads and writes while ensuring security and scalability.

4.4.1 Firebase Security Rules

To enforce data integrity and security, the following Firebase security rules are defined:

|  |
| --- |
| {  "rules": {  "users": {  "$user\_id": {  ".read": "auth != null",  ".write": "auth.uid == $user\_id"  }  },  "appointments": {  ".indexOn": ["serviceId"],  "$appointment\_id": {  ".read": "auth != null",  ".write": "auth != null"  }  },  "payments": {  "$payment\_id": {  ".read": "auth.uid == data.child('user\_id').val()",  ".write": "auth.uid == data.child('user\_id').val()"  }  },  "User": {  ".indexOn": "username"  }  }  } |

This structure ensures that only authenticated users can access their own data, appointment data is indexed for efficient querying based on serviceId, and users can only modify their own payment information. Additionally, the User node is indexed on username for better query performance. Further indexing strategies and optimizations will be implemented to enhance database performance.

**Chapter 5 INDIVIDUAL APPLICATION MODULE IMPLEMENTATION**

**5.1 Introduction**

After obtaining the requirement and designing the component to show flow of the system, project implementation is where the project execution is taking place from the documented data into a real program. The focus of this chapter is to show the programming technique used to develop the system and the error handling. Programming techniques is the technique used to describe the development process, and the details of programming language code used.

**5.2 User Interface Design**

5.2.1 Web Application (Laravel Framework)

|  |  |
| --- | --- |
| 5.2.1.1 Quotation Management | A screenshot of a computer  Description automatically generated  A screenshot of a computer screen  Description automatically generated |
| 5.2.1.2  Authentication | A screenshot of a computer  Description automatically generated  Picture 5.2.1.2.1 Login Validation  A screenshot of a computer  Description automatically generated  Picture 5.2.1.2.2 Login Password encypt  A screenshot of a computer  Description automatically generated  Picture 5.2.1.2.3 Admin Support |
| 5.2.1.3  User Management Module (Administrator) | A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Home Page Admin    Picture 5.2.1.3.1 User View Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Update User Details Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Update User Details Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Update User Medication Section Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Register User Page Admin  (Detail fill automation for IC input can display date of birth and gender.)  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Validation Error Register User Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Validation Message Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 View Inactive User Page Admin  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Filter Seach By Name  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Filter Seach By Role |

|  |  |
| --- | --- |
| 5.2.1.4 Service Management Module ( Administrator ) | A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 View All Service  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Search Service By Service Name  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Search Service By Service    A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Update Service Details  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Create New Service |
| 5.2.1.5 Assign Caregiver Module (Administrator ) | A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 View All Care Plan  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 Update Client Caregiver  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 View Care Plan by Input Caregiver Name  A screenshot of a computer  Description automatically generated  Picture 5.2.1.3.1 View Care Plan by Input Client Name |
| 5.2.1.6 Client management Module (Caregiver ) | A screenshot of a computer  Description automatically generatedPicture 5.2.1.3.1 View All Client Assigned for The Caregiver  A screenshot of a computer  Description automatically generatedPicture 5.2.1.3.1 View CarePlan |
| 5.2.1.7 CarePlan management Module (Caregiver) | A screenshot of a computer  Description automatically generatedPicture 5.2.1.7.1 Register CarePlan  A screenshot of a computer  Description automatically generatedPicture 5.2.1.7.2 Register CarePlan (details section)  A screenshot of a computer  Description automatically generatedPicture 5.2.1.7.3 Register CarePlan (service and cost section)  A screenshot of a computer  Description automatically generatedPicture 5.2.1.7.4 View CarePlan |
| 5.2.1.8 CareLog Module (Caregiver) | A screenshot of a computer  Description automatically generatedPicture 5.2.1.8.1 View CareLog  A screenshot of a computer  Description automatically generatedPicture 5.2.1.8.2 View CareLog details |

**5.3.2 Mobile App (Flutter)**

|  |  |
| --- | --- |
| 5.3.2.1 Login | A screenshot of a login screen  Description automatically generated  Picture 5.3.2.1 Login |
| 5.3.2.2 Caregiver – Logout | A screenshot of a phone  Description automatically generated  Picture 5.2.3.2 Logout |

|  |  |
| --- | --- |
| 5.3.2.3 Dashboard (Caregiver and Client (Residential and Center)) | A screenshot of a phone  Description automatically generated  Picture 5.2.3.3.1 Caregiver  A screenshot of a phone  Description automatically generated  Picture 5.2.3.3.2 Client (Residential)  A screenshot of a phone  Description automatically generated  Picture 5.2.3.3.3 Client (Center) |
| 5.3.2.4 Caregiver – Assigned Client | A screenshot of a phone  Description automatically generatedA screenshot of a phone  Description automatically generated  Picture 5.2.3.2 Assigned Picture 5.2.3.2 Details  Senior Information Client |
| 5.3.2.5 Caregiver – Care Plan | A screenshot of a phone  Description automatically generatedA screenshot of a phone  Description automatically generated  Picture 5.3.2.4  Care Plan Type  Picture 5.3.2.4 Care Plan – List of Assigned Client |
| 5.3.2.6 Caregiver – Medicine Reminder | A screenshot of a phone  Description automatically generated  5.3.2.4 Assign Reminder for Client  A white background with black border  Description automatically generated  Picture 5.3.2.4 Reminder for Client (Center)  A white screen with a black and red text  Description automatically generated with medium confidence  Picture 5.3.2.4 Reminder on Client (Residential) |
| 5.3.2.7 Caregiver – Appointment | A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Assign Appointment for Client |
| 5.3.2.8 Caregiver – Care Log | A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Logs Medicine for All Assgined Senior  A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Filtered by Date  A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Filtered by Medicine Name  A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Filtered by Care Type  A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Filtered by Time  A screenshot of a phone  Description automatically generated  5.3.2.4 Filtered by Time |
| 5.3.2.8 Caregiver – Care Log |  |
| 5.3.2.9 Client (Residential) | A screenshot of a phone  Description automatically generated  Picture 5.3.2.4 Profile and Logout |
| 5.3.2.10 Client (Residential) | A white clouds in blue sky  Description automatically generated  Picture 5.3.2.9 SOS Alert with Siren  A screenshot of a contact form  Description automatically generated  Picture 5.3.2.9 Add Family Contact  A screen shot of a phone  Description automatically generated  Picture 5.3.2.9 Successful Add Family Contact |
| 5.3.2.11 Client (Residential) | A white background with pink and black lines  Description automatically generated with medium confidence  Picture 5.3.2.9 Medicine Reminder |
| 5.3.2.12 Client (Residential) | A screenshot of a phone  Description automatically generated  Picture 5.3.2.9 Services |
| 5.3.2.13 Client (Residential) | A screenshot of a phone  Description automatically generated  Picture 5.3.2.9 Health Condition |
| 5.3.2.14 Client (Center) | A white and pink gradient  Description automatically generated  Picture 5.3.2.9 Sos Alert with Siren |
| 5.3.2.15 Client (Center) | A screenshot of a phone  Description automatically generated  Picture 5.3.2.9 Services |
| 5.3.2.16 Client (Center) | A white background with pink and black lines  Description automatically generated with medium confidence  Picture 5.3.2.9 Health Condition |

**5.3 Programming Technique**

**5.3.1 Web Application (Laravel Framework)**

5.3.1.1 MVC Architecture

This project follows the Model-View-Controller (MVC) architecture, which separates the application logic into three interconnected components:

* **Model:** Represents the data and the business logic of the application.
* **View:** Represents the user interface and displays data to the user.
* **Controller:** Handles user input and interacts with the model to update the view.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

5.3.1.2 Blade Templating Engine

This project uses Laravel's Blade templating engine to create dynamic views. Blade provides a simple and powerful syntax for defining views and includes features like template inheritance, sections, and components.

Example :

A screen shot of a computer

Description automatically generated

5.3.1.3 Service-Oriented Architecture

The project uses a service-oriented architecture where business logic and data operations are encapsulated in service classes. This approach promotes separation of concerns and makes the code more modular and maintainable.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

5.3.1.4 Firebase Integration

The project integrates with Firebase for user data management. The FirebaseService class is used to interact with Firebase, providing methods to retrieve and manipulate user data.

A screen shot of a computer

Description automatically generated

5.3.1.5 Dependency Injection

The constructor of the UserController class uses dependency injection to inject instances of FirebaseService and StorageService. This technique promotes loose coupling and makes the code more testable and maintainable.

A screenshot of a computer

Description automatically generated

5.3.1.6 Validation and Error Handling

The project uses Laravel's built-in validation to ensure that the data submitted by users is valid. Validation rules are defined in the controller and applied to incoming requests.

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

5.3.1.7 Session Management

The view method in the UserController class checks for a user ID in the session to determine if the user is logged in. If the user is not logged in, they are redirected to the login page.

A screen shot of a computer code

Description automatically generated

5.3.1.8 Routing

This project uses Laravel's routing system to define routes for different endpoints. The routes are likely defined in the routes directory, which maps URLs to controller actions.

A screenshot of a computer screen

Description automatically generated

5.3.1.9 Form Handling and CSRF Protection

Forms in the project are handled using Laravel's form handling features, including CSRF protection to prevent cross-site request forgery attacks. The @csrf directive is used in forms to include a CSRF token.

Example:

A black and white screen with orange text

Description automatically generated with medium confidence

5.3.1.10 JavaScript and AJAX

The project includes JavaScript for client-side interactivity and AJAX requests. For example, the SweetAlert2 library is used for displaying alerts, and Cropper.js is used for image cropping.

Example : SweetAlert2 library

A computer screen shot of text

Description automatically generated



Example : Cropper.js

A screen shot of a computer program

Description automatically generated

5.3.1.11 Conditional Statements (if-else)

This technique is used throughout the project to handle various scenarios, such as validating input data and setting default values.

A screen shot of a computer code

Description automatically generated

5.3.1.12 Loops

This basic technique is used in the project to process arrays and objects, such as filtering users and adding dynamic form fields.

A computer code on a black background

Description automatically generated

5.3.1.13 CSS for Styling

This project uses CSS to style the user interface, ensuring a consistent and visually appealing design. A screen shot of a computer program

Description automatically generated

5.3.2 Mobile Apps (Flutter)

5.3.2.1 Stateful Architecture

* + - * 1. Direct Firebase Realtime Database Integration

We use Firebase Realtime Database to fetch, update, and save data directly in your application and DatabaseReference is used for all database interaction.

**A screen shot of a computer

Description automatically generated**

5.3.2.1.2 Asynchronous Programming

Asynchronous functions (async/await) are used to handle database queries and prevent blocking the UI thread.

A computer screen shot of code

Description automatically generated

* + - * 1. Validation and Error Handling

A computer screen with colorful text

Description automatically generated

A screen shot of a computer screen

Description automatically generated

5.3.2.1.4 Data Mapping and Transformation

Data fetched from Firebase is transformed into usable structures

A computer screen shot of a program code

Description automatically generated

* + - * 1. Conditional Statement

If…else Statement

A computer screen shot of a program

Description automatically generated

5.4.1 Error Handling (Web)

Error handling refers to the system's processes for responding to and recovering from error conditions. Its purpose is to ensure users understand the system's behaviour through messages or outputs, enabling the program to maintain its normal execution flow.

5.4.1.1 Sign in validation

Picture 5.4.1.1 shows that if user insert incorrect username or password, the error will display

A screenshot of a computer

Description automatically generated

Picture 5.4.1.1 Sign in validation

5.4.1.2 Empty input error handling (log in)

Picture 5.4.1.2 shows that if user left the field empty, the error would display.

A screenshot of a computer

Description automatically generated

Picture 5.4.1.2 Empty input error handling

5.4.1.3 Empty input error handling (register user)

A screenshot of a computer

Description automatically generated

Picture 5.4.1.3 Empty input error handling

5.4.1.4 Password input error handling (register user)

A screenshot of a error message

Description automatically generated

Picture 5.4.1.4 Password input error handling

5.4.1.5 Username redundancy error handling (register user)

A screenshot of a error message

Description automatically generated

Picture 5.4.1.5 Username redundancy error handling

5.4.1.6 Empty input error handling (register service)

A screenshot of a computer

Description automatically generated

5.4.1.6 Empty input error handling (register service)

5.4.1.7 Empty input error handling (register service)

A screenshot of a contact page

Description automatically generated

5.4.1.7 Empty input error handling (register service)

5.4.1.8 Empty input error handling (register careplan)

A screenshot of a computer

Description automatically generated

5.4.1.8 Empty input error handling (register careplan)

5.4.2 Error Handling (Mobile)

|  |  |
| --- | --- |
| A screenshot of a computer error  Description automatically generated  Picture 5.4.2.1 Error Handling Sign In | A screenshot of a computer error  Description automatically generated  Picture 5.4.2.1 Error Handling Wrong Email or Password |
| A screenshot of a computer  Description automatically generated  Picture 5.4.2.1 Error Handling Medicine Reminder (Caregiver) | A screenshot of a medical application  Description automatically generated  Picture 5.4.2.2 Error Handling Medicine Reminder (Caregiver) |

## 5.5 CONCLUSION

As for the summarization of this chapter, the implementation process is the core phase of the system development where the draft design becomes a reality. This includes the programming techniques used and error handling to ensure the normal flow of the system. The programming language used to develop the system is PHP, HTML, CSS and JavaScript. There are eleven (11) programming techniques which are IF Statement, IF Else Statement, IF While, IF While Else, Insert Statement, Join Statement and Aggregate function . Error handling helps in maintaining the normal flow of program execution by providing an error message or output. It is compulsory for every system to have an error handling so that there will be any problem regarding the database and would not interrupt the system flow.

**Chapter 6 DATABASE INTEGRATION AND TESTING**

## 6.0 INTRODUCTION

Database integration involves combining information from multiple sources into a unified database system, ensuring seamless access to organized data for users. In the **Golden Year Connect (GYCC+)** system, database integration facilitates efficient data management and retrieval, supporting core functionalities such as care plan management, user feedback, and service tracking.

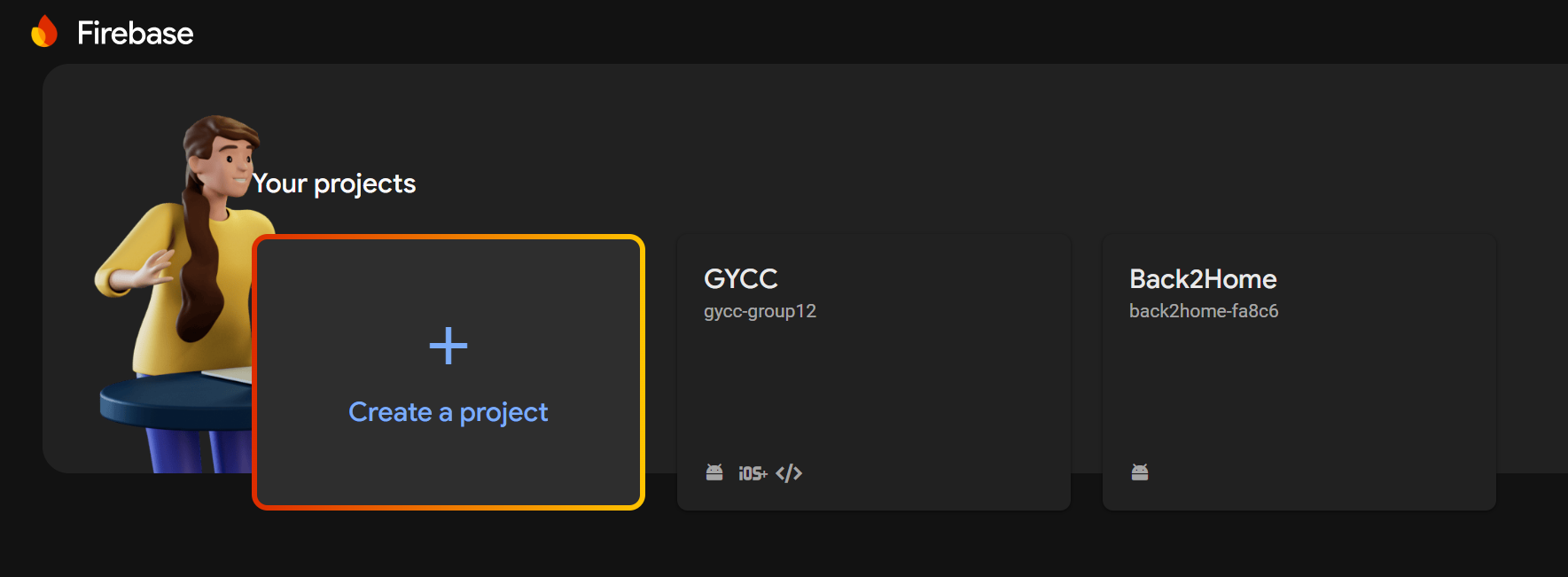
Database testing is a critical phase in the development process, focusing on the validation of schema, tables, relationships, and data consistency. This process ensures that the database performs reliably under various scenarios. It includes creating and executing queries to test for load, stress, and responsiveness, guaranteeing that the system remains robust and efficient in handling user demands.

### 6.1 DATABASE INSTALLATION

**6.1.1 FIREBASE REALTIME INSTALLATION**

**Step 1 : Create a Firebase Project**

Go to the Firebase Console ( <https://console.firebase.google.com/> ) and click on Add project and follow the setup wizard to create a new project.



Picture 6.1 Firebase Console

#### Step 2 : Add Firebase to Your Laravel Project

Once the project has been setup, install the kreait/laravel-firebase package, which is a Laravel package for Firebase. In Laravel project directory, run the following command:

|  |
| --- |
| composer require kreait/laravel-firebase |

#### Step 3 : Configure Firebase

After installation, publish the configuration file:

|  |
| --- |
| php artisan vendor:publish --provider="Kreait\Laravel\Firebase\ServiceProvider" |

This will generate a config/firebase.php file in your Laravel project.

#### Step 4 : Set Up Firebase Credential

Go to Firebase project’s **Project Settings** > **Service accounts** > **Generate new private key**. Download the JSON file containing the credentials.

#### Step 5 : Store Firebase Credentials

In Laravel project, place the Firebase service account JSON file in the storage/app directory. Rename the file to firebase\_credentials.json for simplicity.

#### Step 6 : Add Firebase Credentials to .env

In .env file, add the path to the Firebase credentials file:

|  |
| --- |
| FIREBASE\_CREDENTIALS=storage/app/firebase\_credentials.json |

#### Step 7 : Configure Firebase in config/firebase.php

Open the config/firebase.php file and set the path to the credentials in the 'credentials' array:

|  |
| --- |
| 'credentials' => env('FIREBASE\_CREDENTIALS'), |

**Step 8 : Using Firebase Realtime Database**

Now we can use Firebase’s Realtime Database in your application. For example:

|  |
| --- |
| use Kreait\Firebase\Database;  class FirebaseController extends Controller  {  protected $database;  public function \_\_construct(Database $database)  {  $this->database = $database;  }  public function saveData()  {  $reference = $this->database->getReference('users');  $reference->push([  'name' => 'John Doe',  'email' => 'john.doe@example.com',  ]);  }  public function getData()  {  $reference = $this->database->getReference('users');  $snapshot = $reference->getSnapshot();  $data = $snapshot->getValue();  return response()->json($data);  }  } |

#### Step 9: Testing Firebase Integration

Create routes to call the saveData and getData methods in web.php:

|  |
| --- |
| Route::get('firebase/save', [FirebaseController::class, 'saveData']);  Route::get('firebase/get', [FirebaseController::class, 'getData']); |

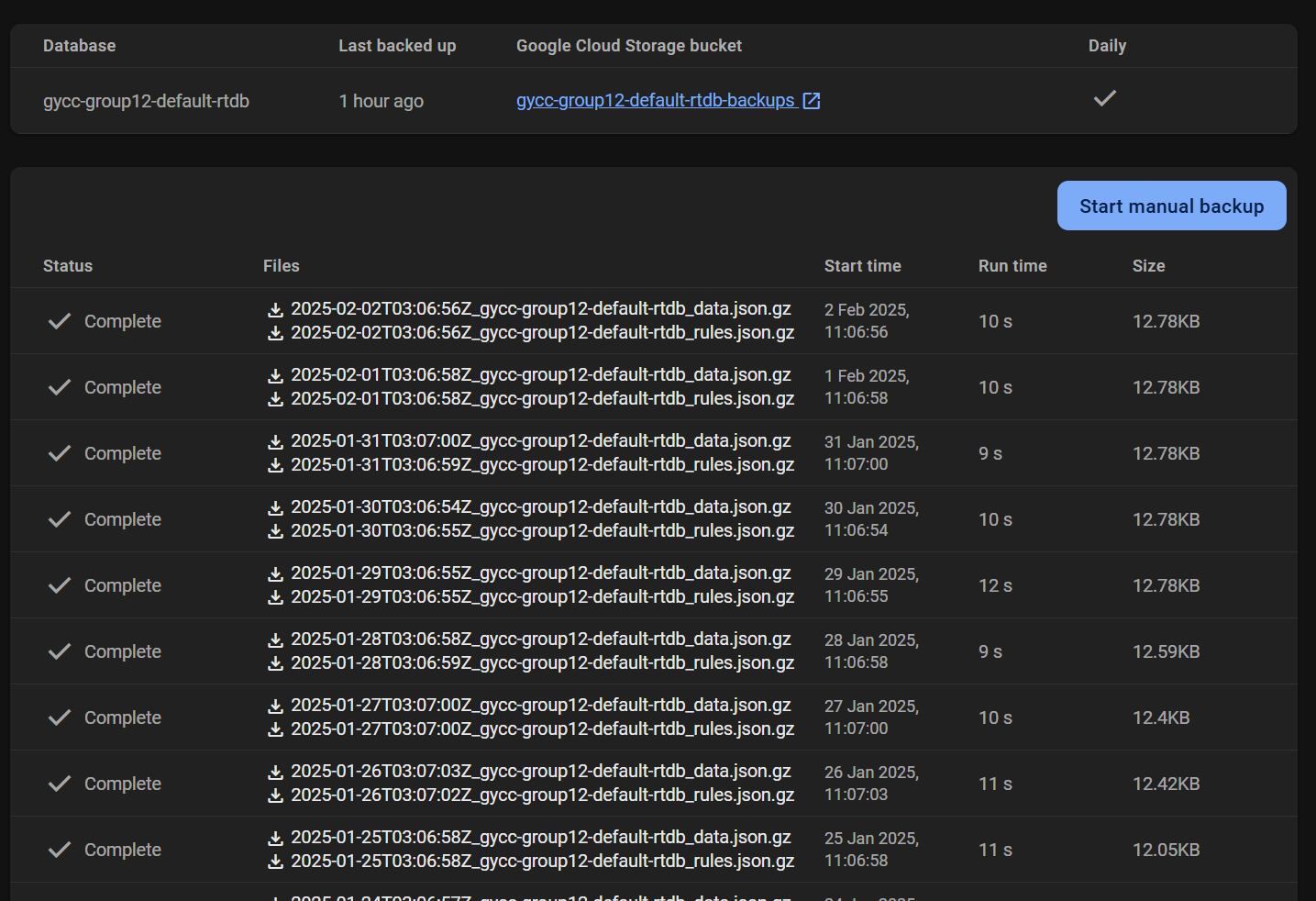
Then visit the route in the browser. Make sure that Laragon host has started.

### 6.2 DATABASE ADMINISTRATION

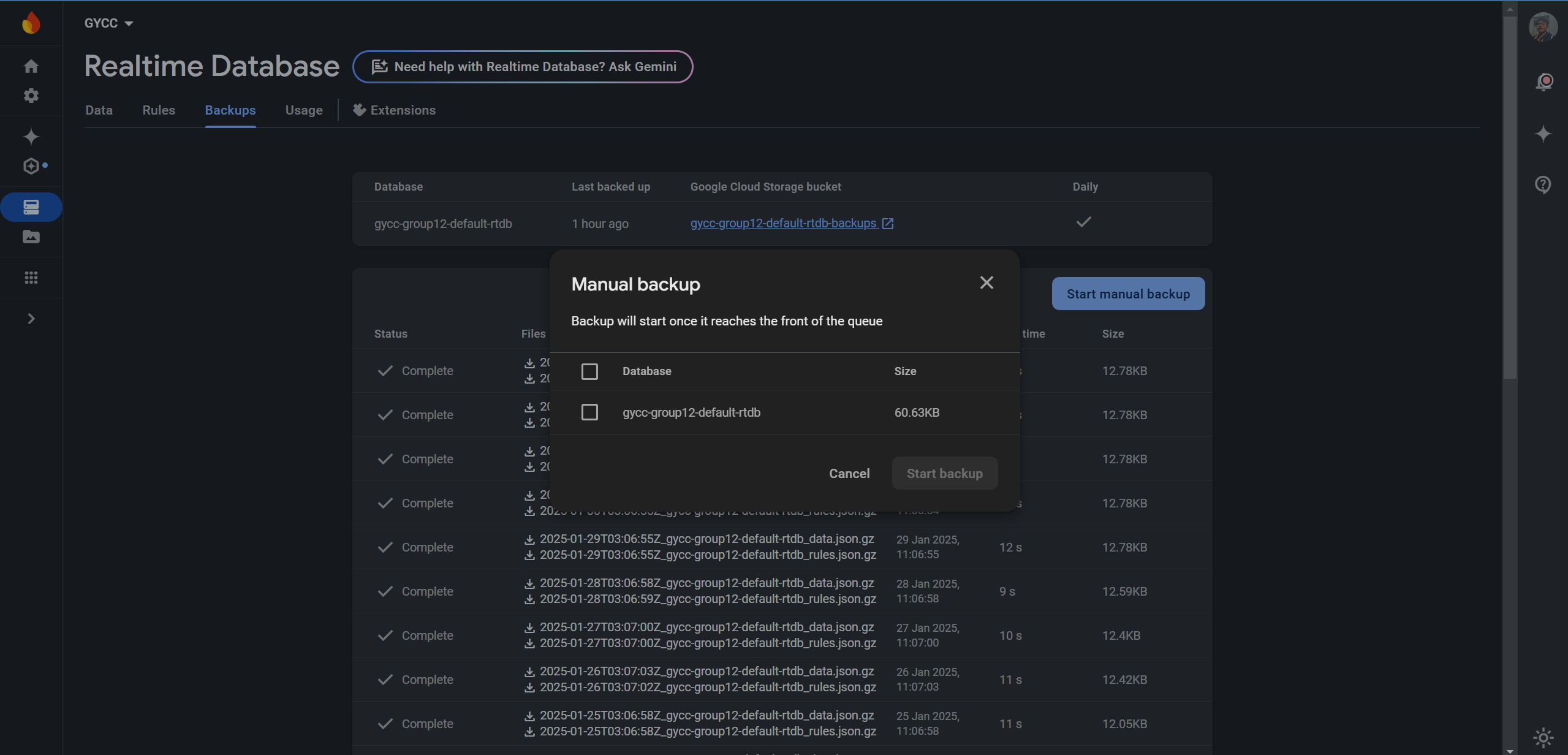
#### 6.2.1 BACKUP

#### For Firebase, the backup is automatically backup every day but if we want to backup manually, there is 2 ways that we can do this.

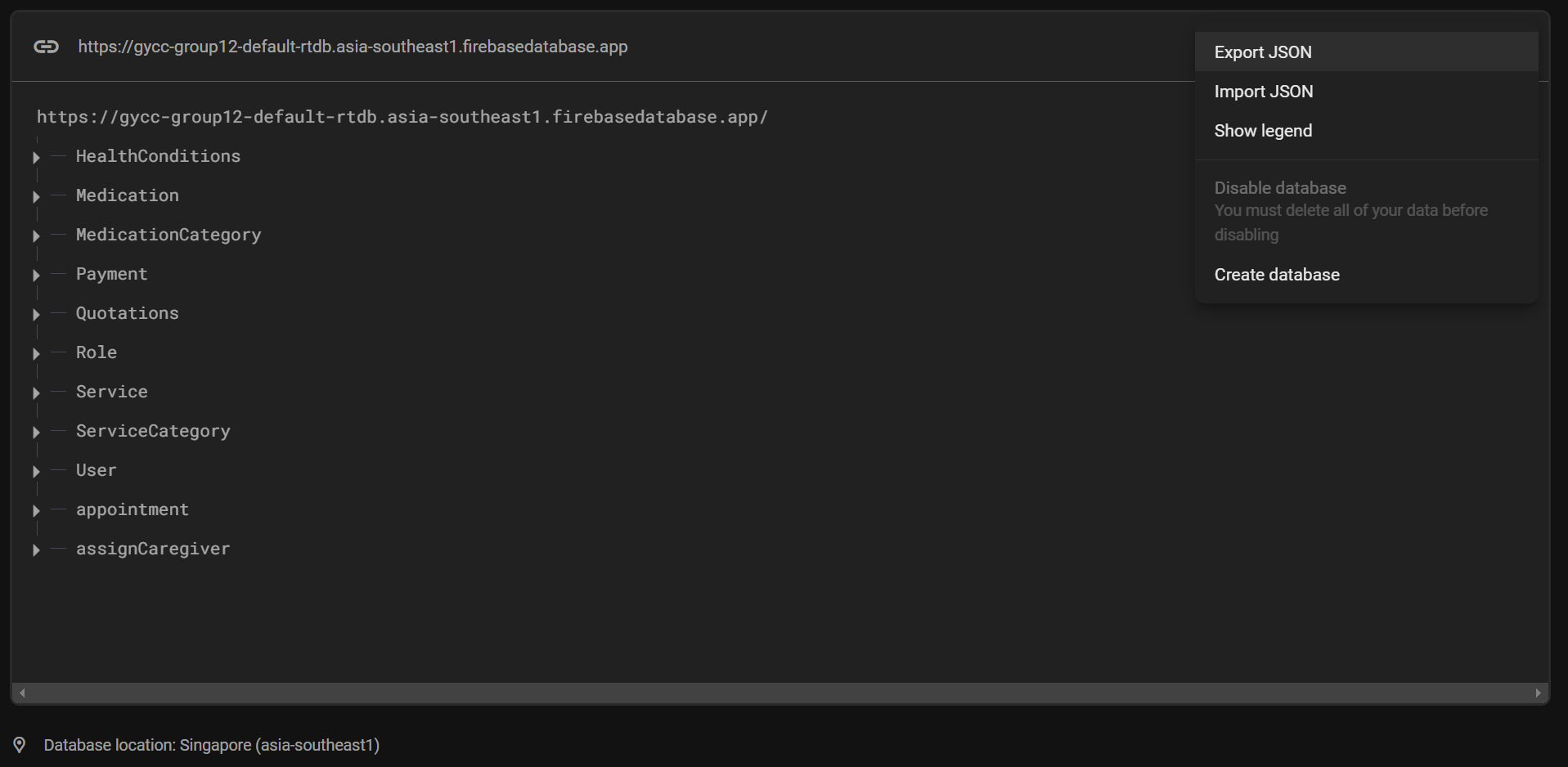
#### First step is backup on the server, which in Firebase itself and the other one is to store in local storage or .json file.



Picture 6.2.1 Daily backup by Firebase automatically



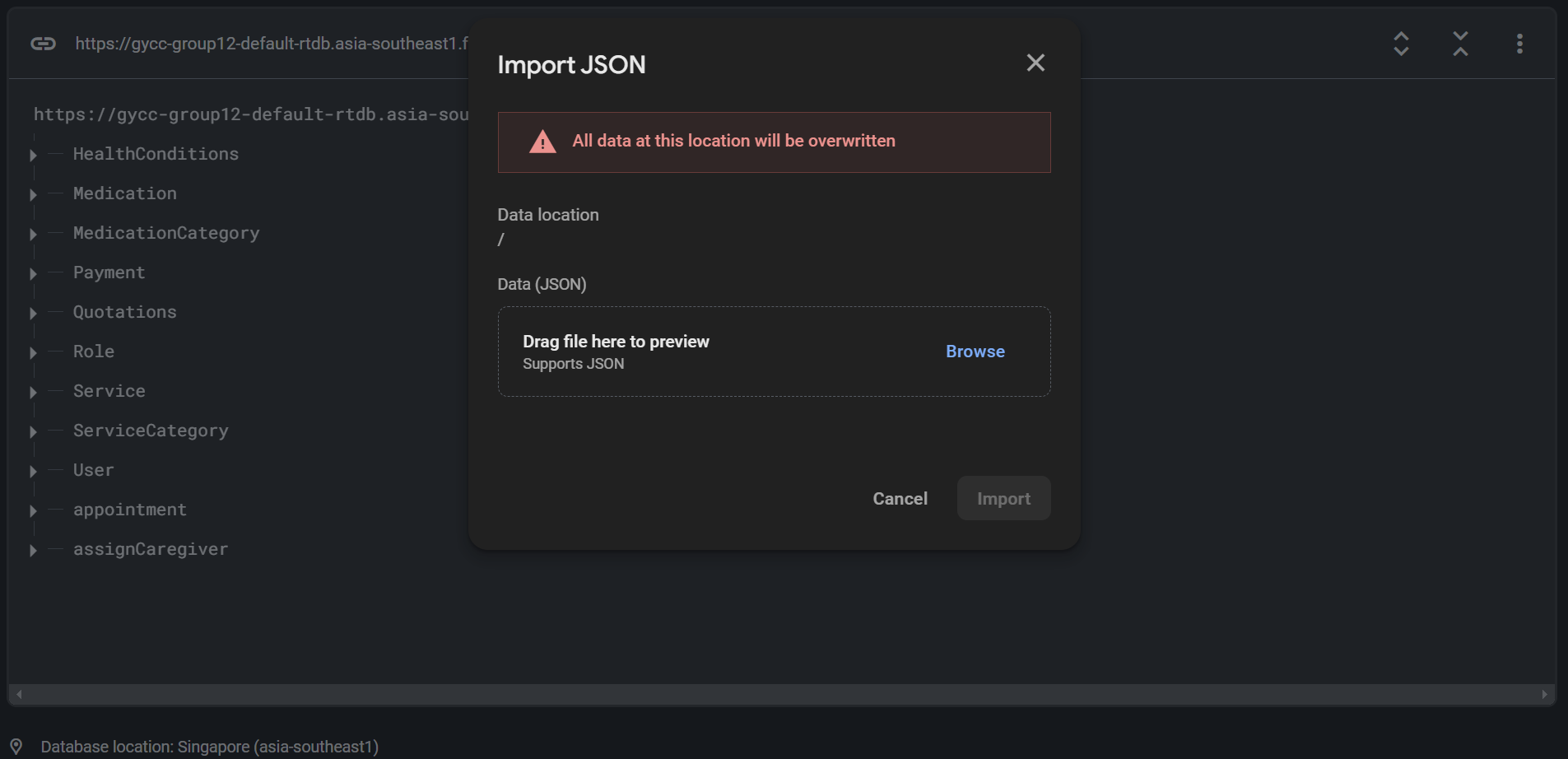
Picture 6.2.2 Manual backup in Firebase



Picture 6.2.3 Manual backup into JSON file

#### 6.2.2 RESTORE

For Restore the backup data, we can upload the backup file using the import function in the Firebase



Picture 6.2.4 Restore function using Firebase

**Chapter 7 : Conclusion**

##### 7.1 INTRODUCTION

In conclusion, the Golden Year Connect (GYCC+) system was developed to meet the outlined objectives, ensuring the integration of critical functions and modules in line with modern development standards. Each component was tested rigorously to achieve optimal performance and functionality. This report provides a detailed explanation of the system's implementation, with functional testing thoroughly documented. The individual modules are elaborated upon to enhance user understanding of the GYCC+ system. While challenges were encountered during development, the result is a system that aligns with the project goals. This system is designed to simplify access to care plans, providing seamless navigation and convenience for users.

##### 7.2 ACHIEVEMENT

From the inception of the project, we outlined several key modules for development, including Care Plan Management, Feedback Integration, User Authentication, and Service Pricing. All these modules were successfully implemented within the project scope, fulfilling the objectives set out in the initial phases. Extensive testing confirmed the system's functionality and readiness for deployment, ensuring a user-friendly experience.

##### 7.3 PROJECT LIMITATION

The GYCC+ system effectively addresses the need for better care management by enabling customers to access plans and services efficiently while storing and organizing user feedback and service usage reports. However, certain limitations exist. For example, administrators can only view individual user reports but lack the capability to generate comprehensive statistical analyses of service trends. Furthermore, some advanced personalization features, such as dynamic care plan recommendations, are not yet integrated.

##### 7.4 SUGGESTIONS FOR IMPROVEMENT

Future enhancements for **GYCC+** could include enabling users to provide more detailed feedback on care plans, along with a system for users to recommend improvements. Implementing a rewards program for feedback submissions could also increase engagement and help improve system adoption. Additionally, incorporating analytics to generate actionable insights on service usage trends and user behaviour could enhance decision-making for administrators. The other major improvement would be a more efficient medicine reminder on mobile application, such as push notifications, though it is presently plagued with errors and time limits for development. Improving the UI to make it easier to use and incorporating APIs and AI functionality would also greatly improve the platform.

### 7.5 POTENTIAL COMMERCIALISATION

Although the **GYCC+** system was developed as part of a workshop project, its features make it suitable for use by small to mid-sized care centers and home-based care providers. The system offers essential modules to manage care plans effectively on a modest budget. With additional development, including advanced analytics, scalability, and a more user-centric interface, the **GYCC+** system has significant potential for broader commercialization, catering to larger organizations or even franchised care centers.

**7.6 CONCLUSIONS**

The GYCC+ system underscores the importance of an efficient management platform in ensuring seamless operations for care providers. An unreliable system can disrupt operations and impact user satisfaction. In this project, we successfully met the predefined objectives, implemented planned modules, and conducted thorough testing to ensure reliable functionality. Overall, the **GYCC+** system is a success, laying a strong foundation for further development and broader adoption in the care management industry.