Acknowledgement

I would like to express my deep gratitude to my supervisor, Dr. Helena Wong, who has always been supporting me throughout my study and reviewing my ideas on this project. Her constructive advice and suggestions have provided me with enthusiastic encouragement throughout the project.

I would also like to acknowledge the resources provided by the City University of Hong Kong, including access to library resources and guidance from lecturers of the English course specified for this project.

Finally, I would like to thank my family and my friends, especially my mother, who has shared her valuable time and experiences that continuously bring me with different insights to improve the project.

Abstract

As the proportion of the population aged 65 and above in Hong Kong has been steadily growing. To accommodate the expected increase of residents in elderly homes in the future, a web-based elderly home management system is designed in this paper. A modern and robust full-stack technique named MERN stack is adopted to achieve the design, along with other web technologies and optimizations that ensure the performance and security of the project. As the system passes most of the test cases, the positive results indicate that this management system can be widely adopted in different elderly homes. However, there is still room for improvement. As the project would be focusing only on simplifying administration and documentation for staff users. One of the areas that can be further extended is the adoption of cloud services and Internet of Things (IoT) services. The system can also be expected to expand to different system structures such as mobile applications that help improve the user experiences and documentation performance for respective users.

Table of Contents

1. Introduction

1.1 Background information

A country in which 14% to 20% of the total population aged over 65 years old is considered an aged country by the definition from the World Health Organization (WHO, 2002), and Hong Kong had already been considered as an aged country since 2018 due to the fact that residents aged 65 years old and above made up 17% of the population in 2018, and it is projected to grow continuously to 33.7% in 2066 (Census and Statistics Department, 2017).

With the ageing problems in Hong Kong that will eventually lead to an increasing demand in elderly homes and its healthcare systems, an effective management system needs to be implemented in elderly homes to reduce the public financial burden and to accommodate the aged population in the future. Stepping into the information era, it is of great importance to build computerized information systems for the elderly health sector given the rapid development in information technology (Ahmadi et al., 2016), with the advantages of reducing cost and improving services efficiency (Aghayi et al., 2017; Chen et al.,2021) so that the traditional Yet, the adoption of electronic management systems in elderly homes is still lacking due to barriers in different aspects (Smiley, 2014).…

1.2 Problem statement

Nursing documentation is a part of daily routines carried out by caregivers in elderly homes. It is an invaluable repository of the health status of the resided elderly. Not only can caregivers access comprehensive health information of the respective elderly, but it also acts as an effective medium for communication among caregivers and assists them in the

decision-making process (Griffiths et al., 2010; Hector, 2010; Shrestha, 2016). It plays a vital role to facilitate the operational workflow and optimize the caring services (Kinnunen & Saranto, 2009). However, the documentation or other documents such as observation charts and admission forms are often manually recorded with pens and papers (Abejirinde et al., 2020). Electronic systems are commonly lacking in data storage and retrieval (Smiley, 2014). The paper-based documentation mode can adversely affect the performance of the caring services. Studies have shown that the time spent on documentation has taken 25% of caregivers' workday (Beachley et al., 2007). This is considerably time-consuming as most of the work is redundant, and more importantly the caring services will be hindered. Lack of standardization is also an issue as different caregivers may have different wordings or styles of documenting (PAHO, 2001). Not to mention that illegibility, documentation errors, modification or loss of data can happen in their daily routines, miscommunication or even fatal accidents can easily emerge.

1.3 Proposed solution

My proposed solution is to develop a web-based elderly home management system to address the aforementioned problems. Caregivers can use the system to facilitate their work routines in different aspects. With respect to documentation problems, electronic health records will be adopted (EHR). It is one of the modules to store the information of the resided elderly. Caregivers no longer need to search for respective paper files to conduct direct health care for the elderly, any modification or data update of the elderly record can be quickly processed after typing the elderly ID in the system. This is also a scalable and environmentally-friendly measure to cope with the vast amount of aged population in the future. More importantly, this can save time for caregivers to work with other duties. Other features which are not applicable

in paper-based documentation such as drop-down menus and predefined terminologies are provided in the system to shorten the documentation time and help standardize the records (Shrestha, 2016). Detailed feature descriptions and functions of other modules will be introduced later in the report.

1.4 Project scope and objectives

This project aims to provide services to elderly homes without a technical management system and improve the quality of services and management by achieving three objectives. First, to transform the paper-based management mode of elderly homes into organized and electronic management, which can be achieved by building a computerized web-based management system to replace manual and written records. Second, to reduce the potential human errors in documentation, a simple yet efficient system is constructed to cater for the fast working pace in elderly homes. Standardization of the record is also assured. Third, to provide optimal caring services to the elderly by reducing colleagues' workload, in addition to the aforementioned elderly electronic health records that can help caregivers take fast and appropriate health care to the elderly, other modules such as staff management, facility management and medical management are also developed for administrative colleagues. The workload of both of the positions can be lightened under systematic management. To conclude, it is hoped that the project can utilize administrative colleagues and caregivers' routine duties so that the caring performance brought to the elderly can be the best possible.

Due to the limitations of time scale and resources, this project only focuses on building a management system for elderly homes by investigating its purpose and methodology in the

literature review. Concepts of “smart elderly home” and other modern technology which can also support the caring system such as Internet of Things (IoT) or any real-time monitoring and tracking systems are not designed.

1.5 Report organization

This report will introduce the literature review in section 2, followed by the system design for the project in section 3. In section 4, implementation and testing methods are introduced, and lastly the results and evaluation of the project will be discussed in section 5.

2. Literature Review

In this section, the workforce structure and significant events in elderly homes will be discussed to better evaluate the needs and requirements of the proposed solution, followed by reviewing a conceptual design of the management system and currently existing management systems so that the design and implementation factors can be concluded. Lastly, different web technologies will be compared to choose the most appropriate technology for this project.

2.1 Structure and significant events in elderly homes

Understanding the structure and main activities in elderly homes can help gather users' requirements to implement more effective and tailored services. In the following, the structure of elderly homes, a conceptual design of an electronic management system for elderly homes and the evaluation of the design will be discussed.

2.1.1 Structure of elderly homes

Chew et al. (2021) defined elderly homes, in general, as a facility to provide 24-hour caring and medical services for resided elderly people who may not have the physical ability to take care of themselves with daily activities, or those who have declined health conditions. In terms of elderly home structure, an organizational workforce structure chart is displayed below (Fig. 1):

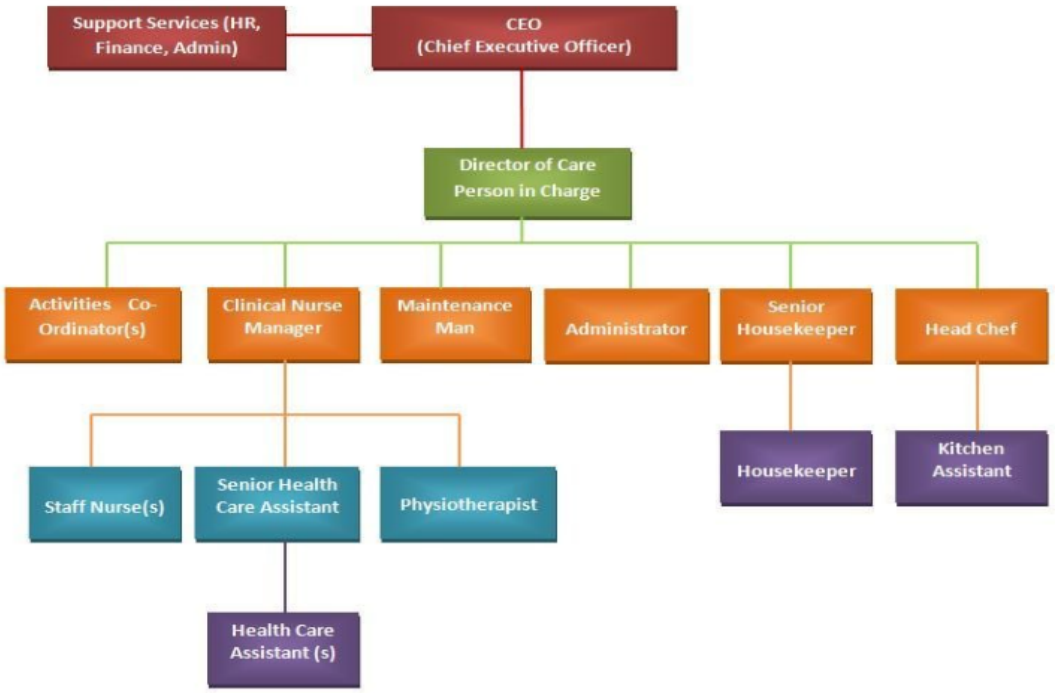
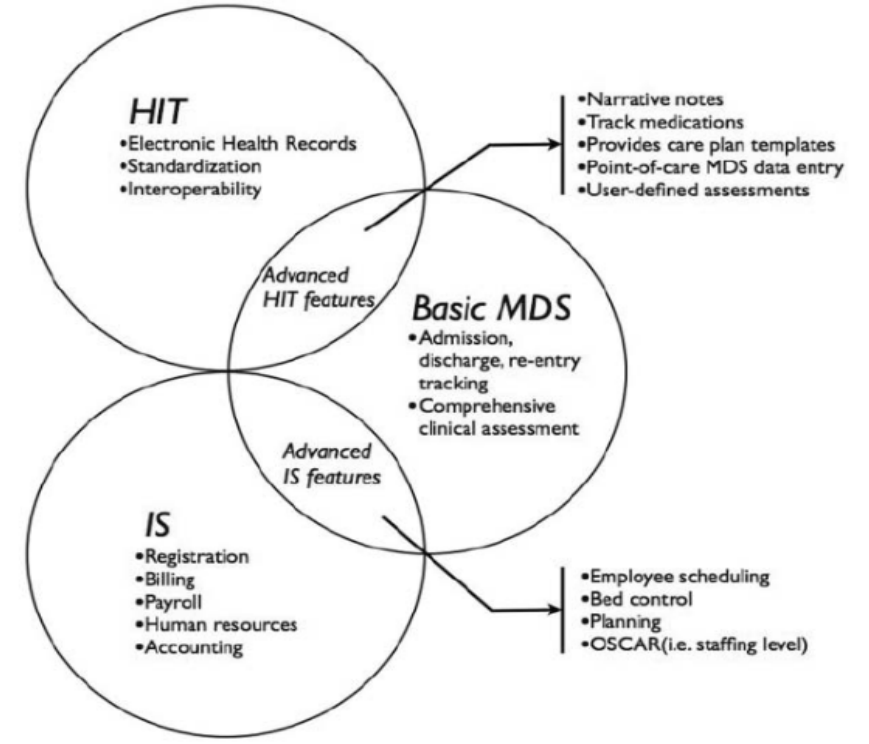


Figure 1 Organisation Chart of an Elderly Home

While an elderly home requires different individuals for the operation, it mostly relies on crucial members including administrators and frontline caregivers working with different responsibilities, including administration work and nursing services such as direct health care and assessment, nursing documentation and rehabilitation services respectively (Goh, 2019).

2.1.2 Conceptual management system design by Castle and Liu

The initial development of an electronic management system in elderly homes can date back to a study conducted by Castle and Liu (2008). In figure 2, a conceptual framework integrated with Health Information Technology (HIT), Information System (IS) and Basic MDS (Minimum Data Set) is displayed.



Note: HIT = health information technology; IS = information system; MDS = Minimum Data Set; OSCAR = On-line Survey, Certification and Reporting.

Figure 2 Conceptual Framework of HIT in elderly homes

The system consists of 3 components (Fig. 2). The first one is Health Information Technology (HIT). HIT can be adopted in various kinds of forms, ranging from hardware, software, to management systems with the same purpose of storage, processing and analysis of health information in an electronic environment (Jen et al., 2022; U.S. Department of Health & Human Services, 2020). Major components of HIT include electronic medical records and health record systems and also electronic prescribing (EMRs, EHRs, and PHRs). The second one is the information system (IS). An information system is similar to HIT, it collects and stores general information, not specifically health information, for the use of processing and dissemination (Bourgeois et al., 2014). The last component is MDS. It is a mandatory process for caregivers to assess a person's health status such as cognitive and psychological status,

which is helpful in formulating the care plan for the respective elderly (American Psychological Association, 2011; Centers for Medicine & Medicaid Services, 2021). The implementation of the above design can surely bring promising benefits including improvements in elderly health outcomes and provide information insights from data thanks to digitization and its relevant tools (Jen et al., 2022).

2.1.3 Evaluation of the conceptual design

Although the design can satisfy the basic operational requirement of elderly homes, there are a few issues that can be noticed. First, some of the important components that can affect the daily operation of elderly homes are not included in the design. For example, dietary management and medication management are the management components that Alwan (2009) had defined, but these are absent in Liu and Castle's preliminary design. The design also does not include the records of daily resident routine upon completion such as showering, shaving, and body temperature measurement. Alwan et al. (2009) claimed that daily personal care carried by caregivers should be recorded in an electronic management system.

In addition to the missing components, Liu and Castle's preliminary design integrates all three components into one single system, which might not be an appropriate and effective design as there are different types of colleagues in an elderly home, separation of systems would be a better approach for frontline caregivers and administrative colleagues whose duties are completely different. For example, frontline caregivers should not have the privilege of information system modules such as viewing the data in financial management or billing functions. Therefore, different user interfaces should be designed according to the colleagues' responsibilities respectively.

2.2 Existing system review

In this part, interfaces of three different elderly home management systems are displayed to evaluate their system design and user experiences so that limitations and improvements can be concluded for the project.

2.2.1 Text-orientated management systems

Figure 3.1 i-TechCare management interface (https://i-techsoft.com/patient-management-hk/)

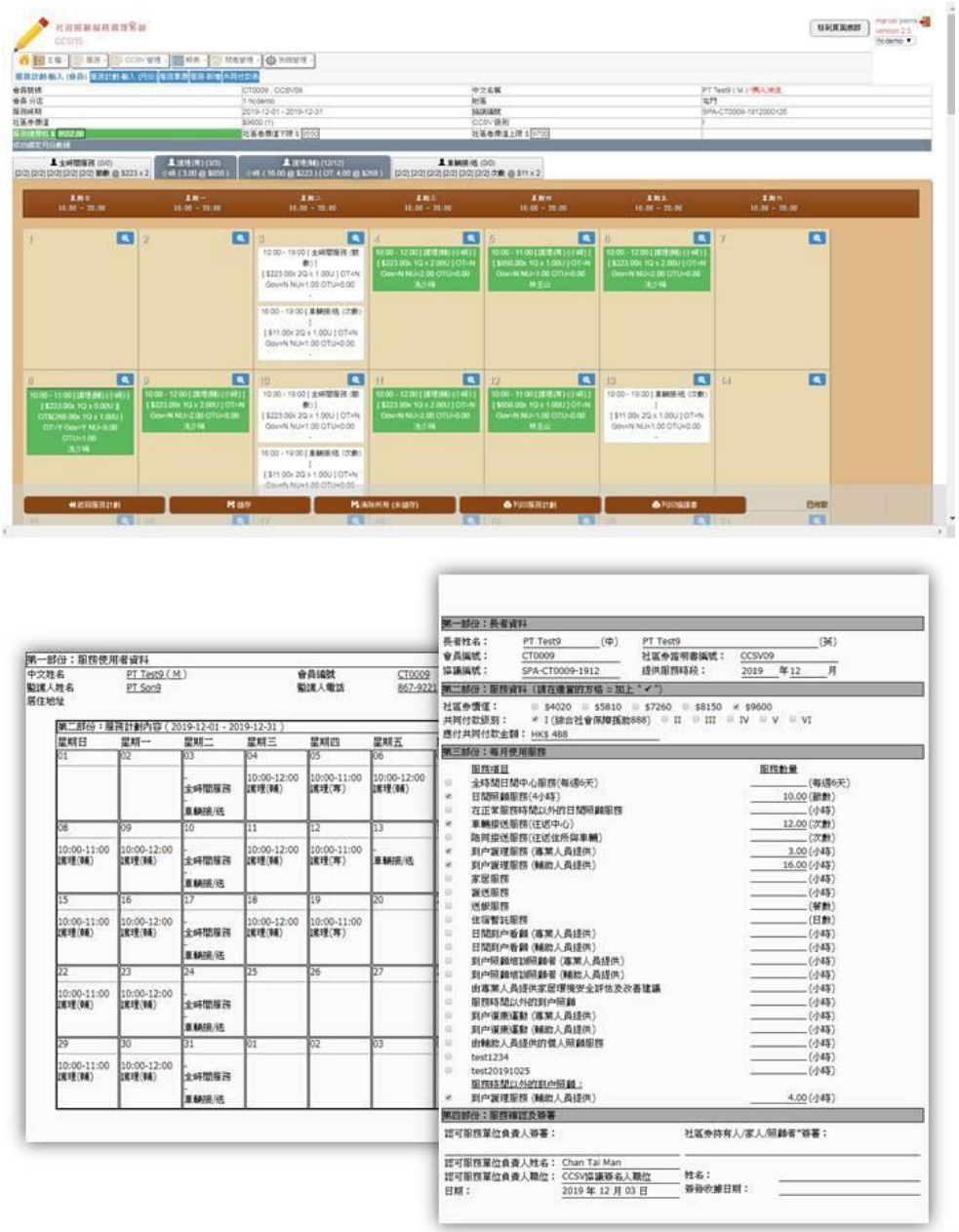


Figure 3.2 vDoCare management interface

Figure 3.1 and Figure 3.2 are the interfaces of management modules of i-TechCare and vDoCare. Although both of the systems display detailed personal information of the resided elderly, there are too many texts in the components and the components seem to be overwhelming. Visual elements such as colour, font differences, paddings and margins, or intuitive components such as buttons and icons that can distinguish the difference between

elements are lacking, which directly affects the system's usability and users' readability (Mandal et al., 2015). Besides, the documentation process is complex with multiple inputs of texts. For instance, in i-TechCare interfaces, rather than directly inputting the date of admission, it would be a better approach to choose a date directly from a virtual calendar, which simplifies the documentation process and prevents human error. Other effective designs including shortcuts or documentation templates also need to be developed to get rid of the above text-orientated system design.

2.2.2 Ungrouped routine categories

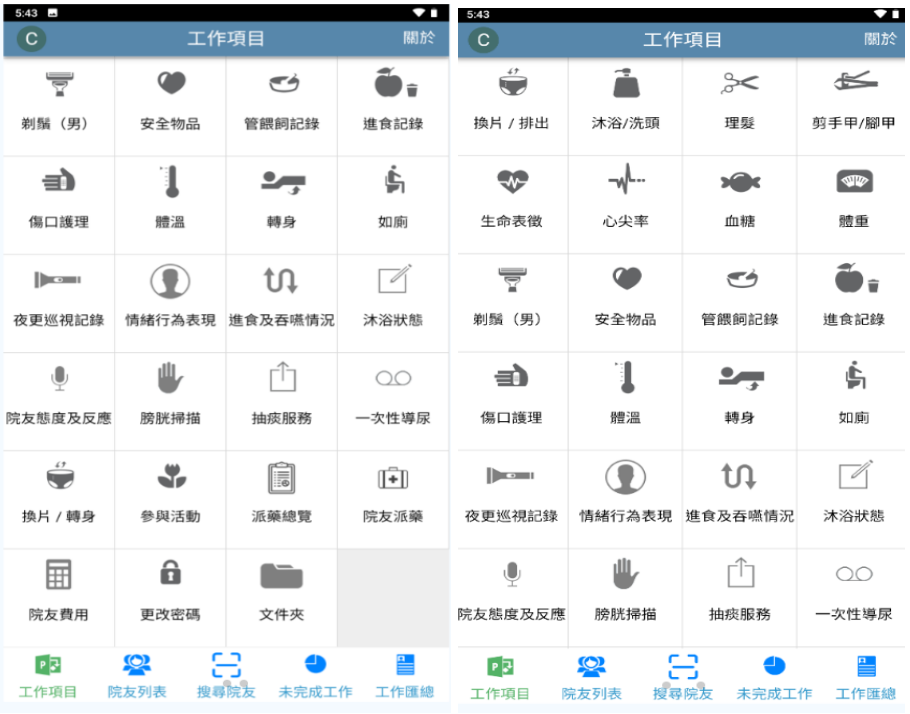


Figure 3.3 HMS management interfaces

Figure 3.3 displays the user interface for frontline caregivers. In the above figure, different direct daily care routines and other functions are listed in a 4x6 container on different pages. However, these items are not grouped and categorized. It may be time-consuming for caregivers to search for specific tasks, which may hinder their healthcare performance.

2.3 Existing web technology for implementation

After reviewing the users' requirements and system designs for the project, in this part, some of the current web technologies used in web-based applications, including web frameworks and databases will be discussed for project implementation.

2.3.1 Web framework

HTML, CSS and Javascript are the dominating technologies in the web development industry, which also act as the basic building blocks of a website. However, simply using them would not be sufficient for a complex application as building the website from scratch would be time-consuming and the complexity of the code and design increases gradually (Devi et al., 2020). Using a web framework for web application development can bring promising progress to developers as it helps shorten the software life cycles with a set of built-in libraries or components. It also helps in code standardisation as the project will be built on a well-defined structure, which boosts productivity in development and debugging (Deed, 2023)

2.3.2 Databases

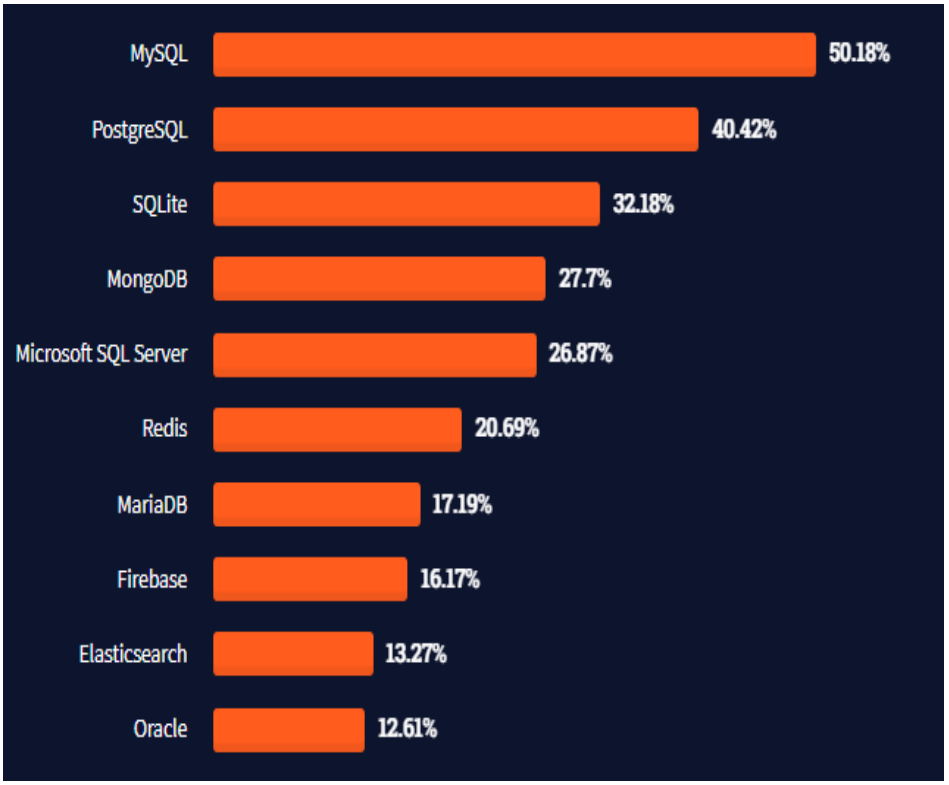


Figure 4 10 Most Used Databases in 2021

Database is an important tool for a management system to store and retrieve data. There are two types of databases, SQL databases and NoSQL databases. From Figure 4.2.2, MySQL, which is a SQL database, is ranked first by stack overflow communities and professional developers in terms of usage in 2021, followed by PostgreSQL (SQL), SQLite (SQL), and MongoDB (NoSQL). Although it is still a majority to use SQL databases based on statistics, the usage of NoSQL databases has grown steadily in the past few years with its exclusive advantages. For instance, the property of scaling out horizontally so that cheap commodity servers can be added for database expansion and data visualization such as chart generation. (MongoDB, n.d.)

Functions/ Features	Existing Solution (MySQL Databases)		**Proposed solution**
	MySQL	PostgreSQL	**MongoDB**
Data model	Relational	Relational	**Document**
Flexibility in schema	Fixed and predefined	Fixed and predefined	**Schemaless**
Scalability			✔
Indexing	✔	✔	✔
Horizontal scaling			✔
Faster queries for data searching			✔
Data analytics	✔	✔	✔
Data visualizations			✔

Figure 5.2 Function comparison of existing databases

2.4 Implementation factors conclusion

Some factors can be concluded regarding the implementation of the overall system. Acharya et al (2022) claimed that the amount of time spent on documentation in an electronic system can be similar to paper-based documentation as it heavily depends on the attitudes towards technology and the technical skills of caregivers. Smiley (2014) also claimed that the top HIT barrier is the lack of computer literacy of caregivers. Therefore, the electronic management system has to be simple yet detailed enough for colleagues to understand the system flow.

Second, the system should be an integrated system with different management modules, but separated into different user interfaces according to the position of the colleague. Another factor is the safety of the data, which is also one of the important components of HIT. A database needs to be constructed to store personal private data with the adoption of security measures. Running speed and user acceptance is also some issues that should be recognized (Shrestha, 2016; Yu, 2006).

3. System Design

In this chapter, the detailed system design of the web-based elderly home management system will be introduced. I will first discuss the system layer and architecture and the description of the proposed methodology with respect to my design. The functionality design and thorough descriptions of each function are introduced with the attached UML diagrams such as use case diagrams and sequence diagrams. The database design will also be covered.

3.1 Three-tier architecture

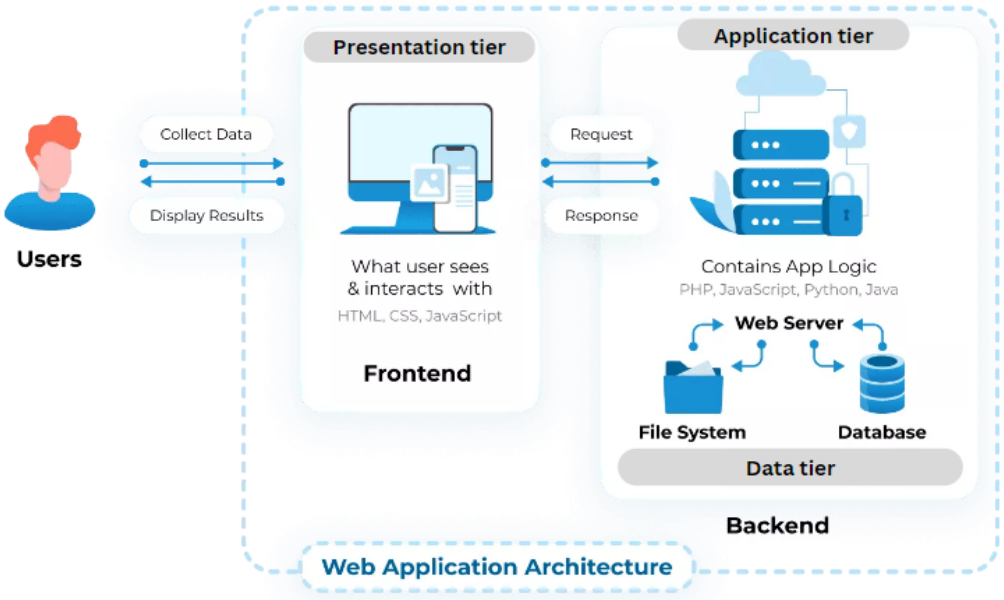


Figure 4 Web application architecture

The application flow can be described in the above figure (Fig. 3), when the users want to collect data, they can perform actions on the user interface and this will be noticed by the frontend system (presentation tier) and a request is made to the backend. The backend

receives the request by the web server (application tier) and accesses the database or file systems (data tier) to make a response, then the response can be displayed back to the frontend for users to notify. This type of architecture is the three-tier architecture, which includes the presentation tier, application tier and data tier, and it will be adopted into my design.

The presentation tier presents the users with information through client-side components such as the browser and UI/UX design and allows users to interact with the data layer. For the presentation tier in this project, three similar user interfaces will be designed and the target users are the administrative colleagues, frontline caregivers and the relatives of the resided elderly respectively. The application tier is the middle tier between the presentation and data tier, which acts as an important tier for supporting the application's functions with business logic and delivering requests and responses (TechTarget, 2021). For the data tier, a database will be developed for data storage. Create, read, update, and delete (CRUD) operations of data can be performed through the application layer.

3.2 Description of Major Technical Components

This project will be mainly implemented in the MERN stack, which is a full-stack development mainly comprised of Javascript with four modern and intertwined technologies, which are MongoDB, Express.js, React.js and Node.js. This method perfectly fits the abovementioned three-tier architecture in the project. Below will give a short description of each of the technologies.

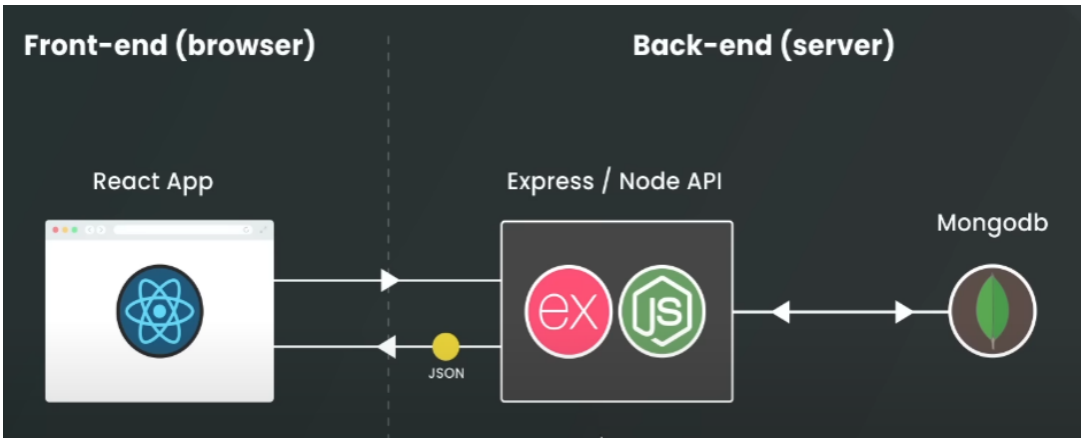


Figure 5 MERN stack architecture

3.2.1 Backend data storage MongoDB

For the data tier, MongoDB will be the best database for the project as it is mainly Javascript oriented, which fits the programming languages of the other technologies mentioned. It is a NoSQL database that is designed to store data in JSON format with keys and values (Couchbase, 2022). The adoption of MongoDB works well with the application so that the JSON data can flow from the frontend to the backend and back and forth to standardize the data flow and the system (MongoDB, 2022).

3.2.2 Middleware express and NodeJS server

Node.js and express.js will be responsible for the application tier. Node.js uses Javascript and has an asynchronous runtime environment. In addition to handling the HTTP requests, another feature of Node.js is the node package manager (NPM). NPM helps with the installation of node modules, which contain different packages that allow developers to require the packages for faster and more efficient development. Therefore, node.js supports both the frontend and the backend development. For express.js, it is one of the frameworks of Node.js, which can deeply enhance the advantages of Node.js. It is mainly responsible for responding to the HTTP request and responses as a middleware. It also has the advantage of

web application callback, route definition and handling and easier connection to the MongoDB (Sharma, 2022).

3.2.3 Front-end react application

For the presentation tier, the user interfaces will be developed with the Javascript library React.js. The adoption of a library can help develop clearer and more organised code, improving the development performance by accelerating the development and lowering the complexity with the use of implemented packages and libraries (Devi et al., 2020). React.js is also the most common and popular Javascript library for frontend development according to a study that investigates developers' satisfaction levels among different Javascript libraries (Adam, 2022).

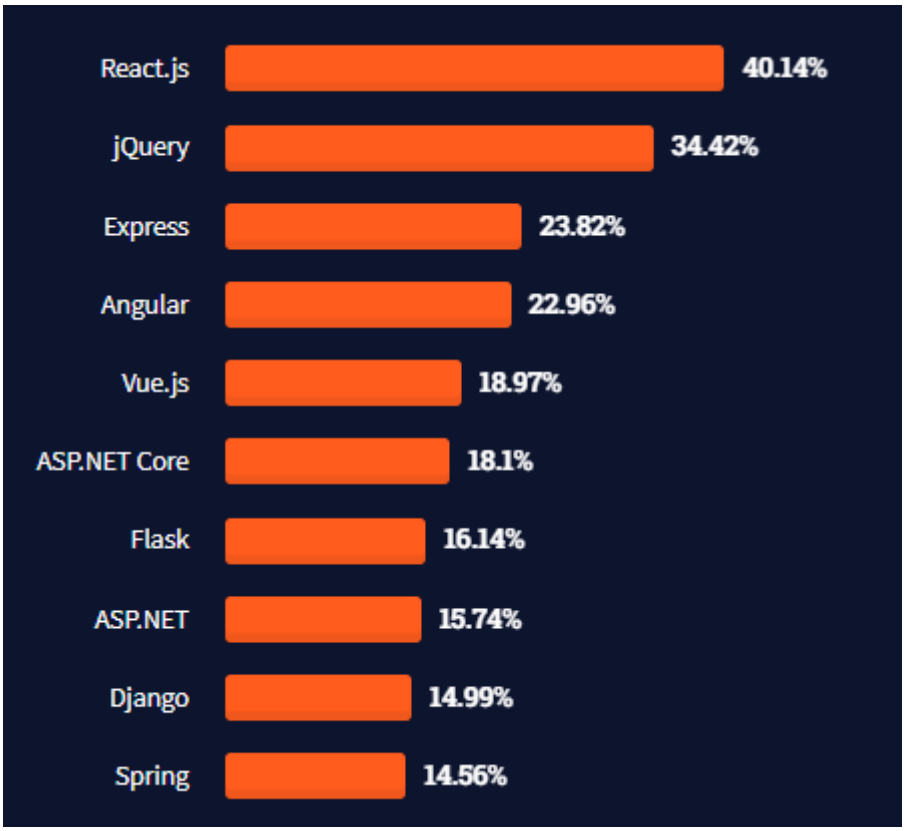


Figure 6 Top 10 most commonly used web framework

To conclude, with the considerations of the three-tier architecture and the major technical components, the proposed system structure overview is displayed in Figure 6.

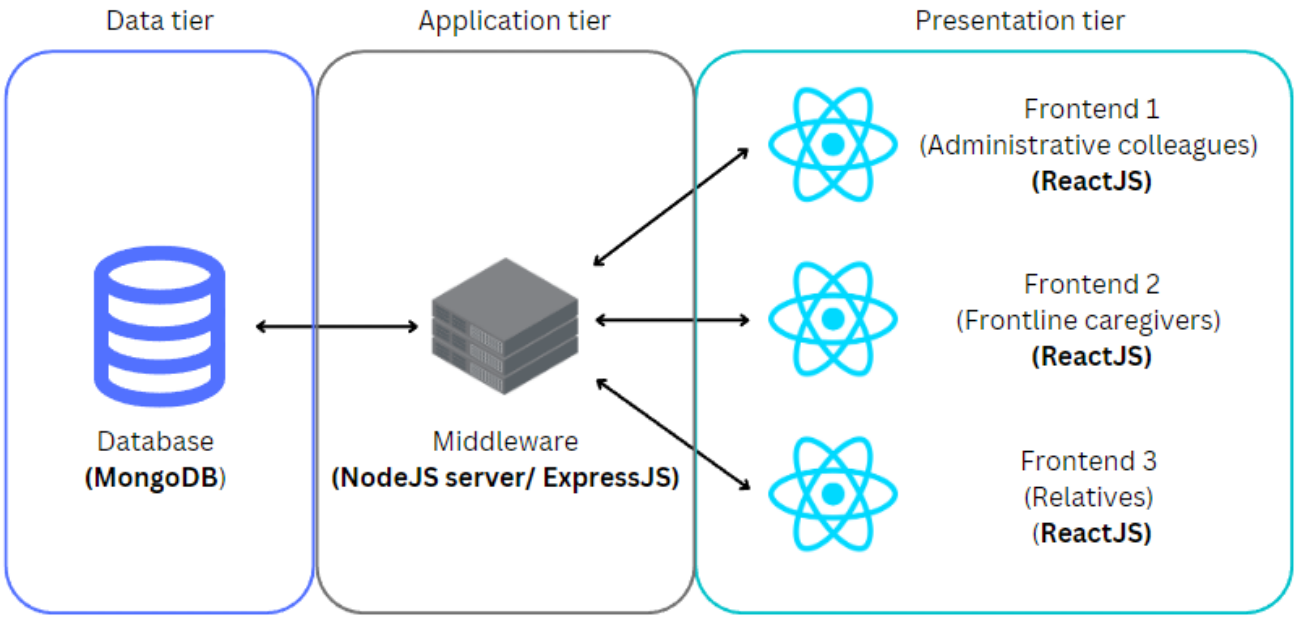


Figure 7 System overview

3.3 Functionalities design

The functionalities design will be discussed in this part, which includes the common features and the respective functionalities according to the user's identity.

3.3.1 Common features

Function	Expected Users	Description
Login	All	Log in to the system according to users' identity
Register	All	Account registration according to users' identity
Dashboard	All	Render the home page of the system

		according to users' identity
Notification	Admin + caregivers	Send notifications to specific admins and caregivers; View notifications
Search a user	Admin + caregivers	Get the profile of staff or elderly by inputting their ID or names
Notes	Admin + caregivers	Edit and view the notes as a reminder

Table 1.1 Common functionalities

3.3.2 Administrative colleagues

Function	**Description**
Document and reports	Provides documents and reports templates for users to download whenever written records are necessary; Stores the information of less important records
Planner	Plan today or future personal schedule
Calendar	View users' personal schedule in the form of a calendar
Management modules	
Staff management	Update and view the staff list; View staff profile by clicking on the item of the list
Residents management	Update and view the elderly list; View elderly profile by clicking on the item of the overview; View the records of elderly routine; Define default routines to the elderly
Activity management	Manage elderly home activities, including lists of activity information, match elderly to activities and activities scheduling

Financial management	Manage elderly home finance, including the admission and charges, residents account summary, cost services management and staff payroll
Medication management	Update and view the medicine list; Update and view the medical appointment for elderly; Manage elderly home medication, including medical records and define default medication to residents
Facility management	Manage elderly home facility (rooms and beds), including managing room status and its vacancy
Work management	View today's work records in both overview and detailed table; Schedule caregivers working areas
Others management	Manage less frequently used management such as diet management, update of notices and gallery

Table 1.2 Admin functionalities

3.3.3 Frontline caregivers

Function	Description
Today's duty	Displays a list of elderly for performing today's health care
Documentation	Update default or additional routine items/medication items status; Update elderly routine status upon completion
View routine records	View past routine records for selected elderly
View medication records	View past medicationrecords for selected elderly
Work review	Review the work records
Health assessment	Records the health questionnaire and any psychological meetings information with the elderly

Table 1.3 Caregivers functionalities

3.3.4 Relatives

Function	Description
Notices	List out the notices issued by elderly home
Gallery	Displays the activity photos based on whether the elderly has joined the respective activity
Account summary	Lists out the amount due, subscribed cost services items, payment status for the elderly account summary

Table 1.4 Relatives functionalities

3.4 Use case diagram

Below is the use case diagram for a better understanding of the overall design structure.

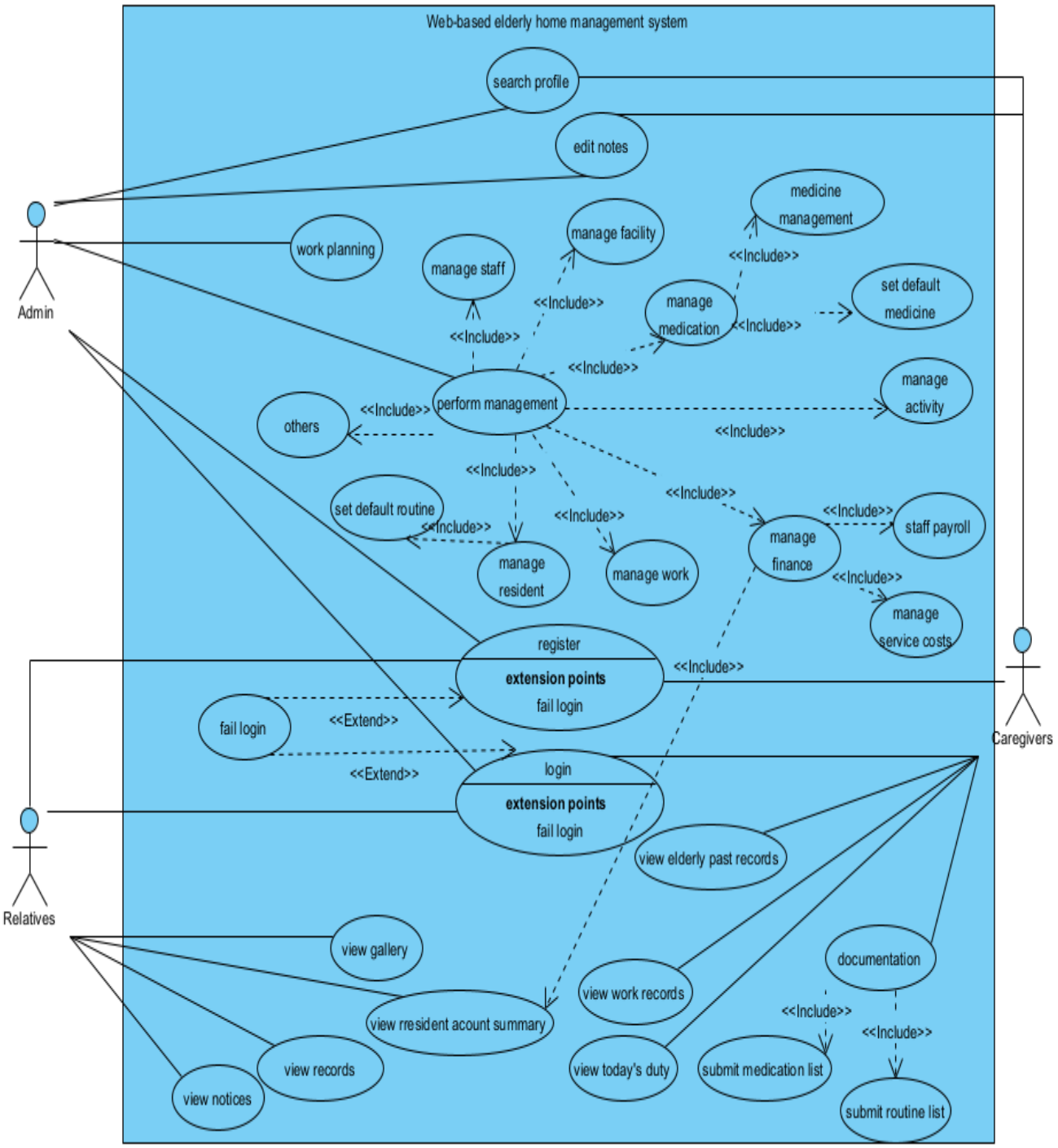


Figure 8 Use case diagram

3.5 Sequence diagram

Below is the sequence diagrams for the sequences of major events in the system.

3.5.1 User registration

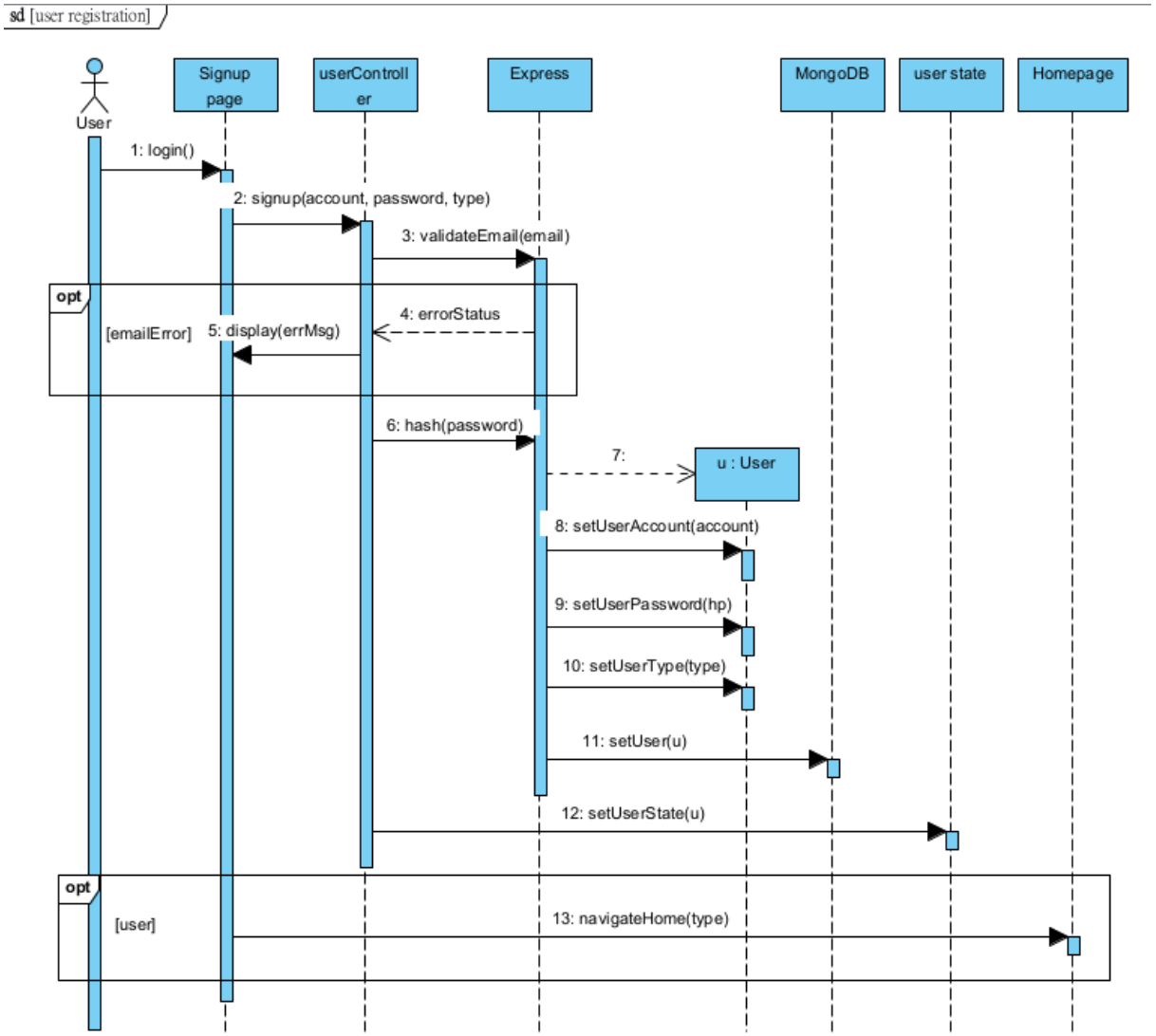


Figure 9.1 Sequence diagram for user registration

Sequence Diagram	Description
User registration	This sequence diagram is representing the procedure of account registration. First, when a user clicks the login button, it will direct them to the signup page. After the user

	inputs their account name and password and chooses the type of their account, userController will pass the request to express.js to perform validation and password hashing. Therefore, a user object is created and it is stored in MongoDB and updates the global state of the user. If it is added successfully, it will direct users to the homepage according to their identity.

3.5.2 Medication distribution

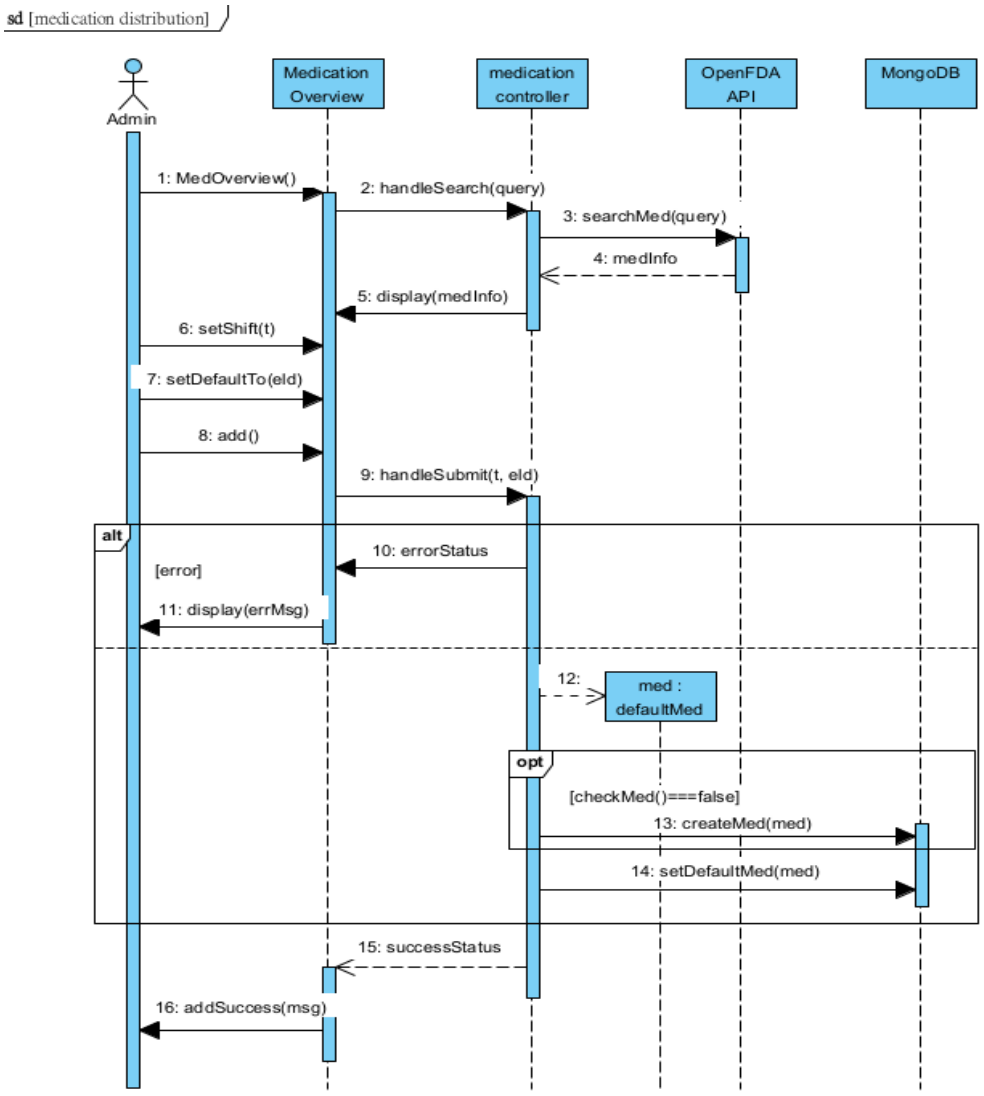


Figure 9.2 Sequence diagram for medication distribution

Sequence Diagram	Description
Medication distribution	This sequence diagram is representing the procedure of medication distribution performed by administrative colleagues. When clicking on the medication overview page, users can type in the search bar (e.g. drug names) to search for a specific medicine. The query is fetched to OpenFDA API and medical information is then returned to the medication controller. Then, users can set the time and medicine default to specific elderly. After clicking the add button, if there is an error (e.g. missing values), it will return error messages. Otherwise, the default medicine object is created and the medicine will also be created if the database does not store the same type of medicine. Finally, a success message will be displayed on the page to the user.

3.5.3 Documentation by caregivers

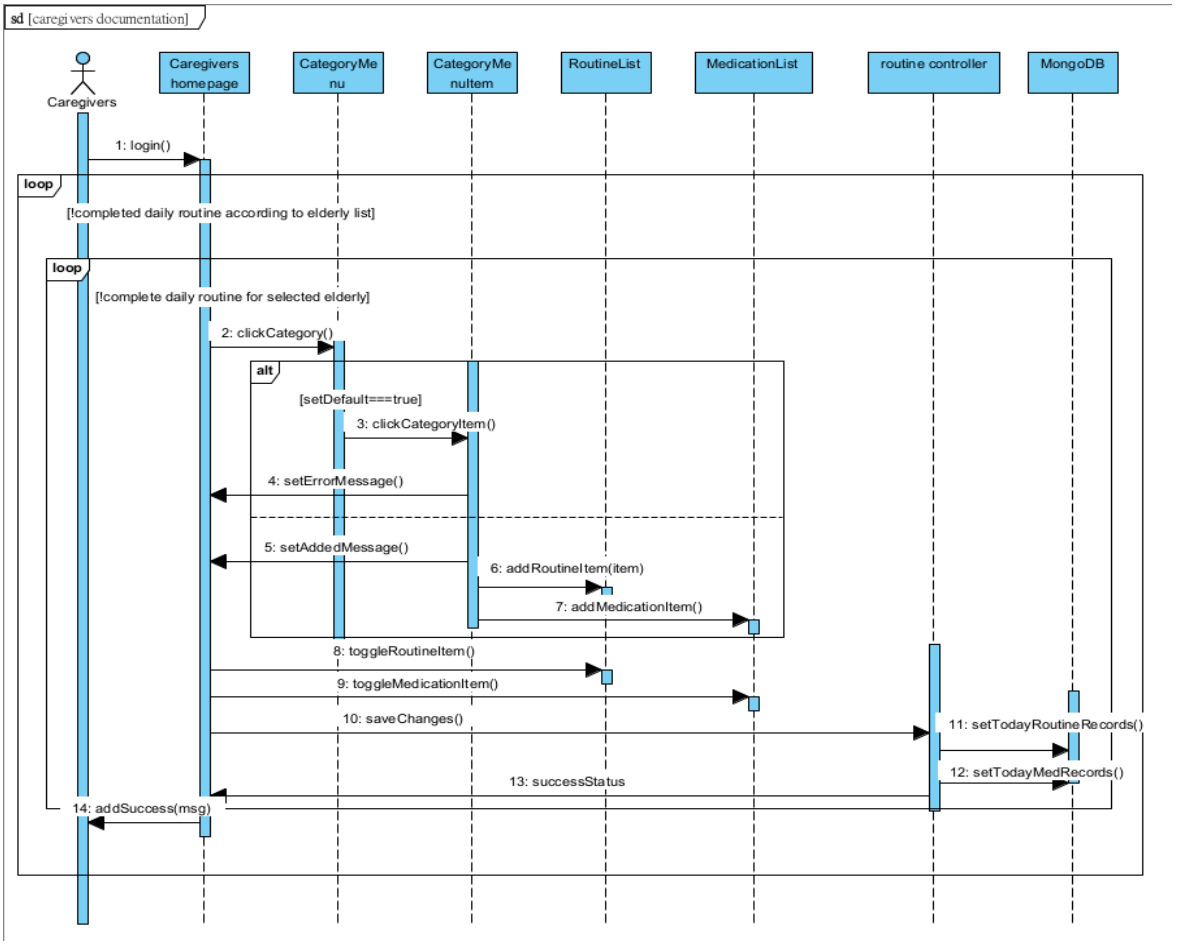


Figure 9.3 Sequence diagram for caregivers documentation

Sequence Diagram	Description
Caregivers documentation	This sequence diagram is representing the procedure of documentation performed by caregivers. After login as caregivers, users can view the "Today's duty" list with resident information to perform documentation. Users can click on routine categories to show category items. If the category item has already been set default, it will return an error message, else, the items will be added to the routine list or medication list respectively. Users can toggle the items in lists to change the status to true to indicate the completion of the routine items. After completing all the routine checking, users can submit the list and store both lists in MongoDB respectively. It will return a success message to the page if there is no error. The documentation is considered completed after performing routines for all the residents in the list.

3.6 Database design

A total of 14 collections have been created in MongoDB for this project, which can be considered as a group of BSON (Binary JSON) in which data are stored in key-value pairs. Figure 10.2 shows one of the documents in the resident account summary collection. Each document contains a primary key (_id) to clearly distinguish documents in the collections. By inspecting the data fields, the field 'itemSubscription' stores an array of objects. In MongoDB, nested key-value pairs are supported for complex data structures so that every operation on the data can be quickly processed by a given key, which is some of the exclusive features that traditional relational databases are not likely to achieve (MongoDB, n.d.). The feature also reduces the number of extra tables and data being created to store all these data.

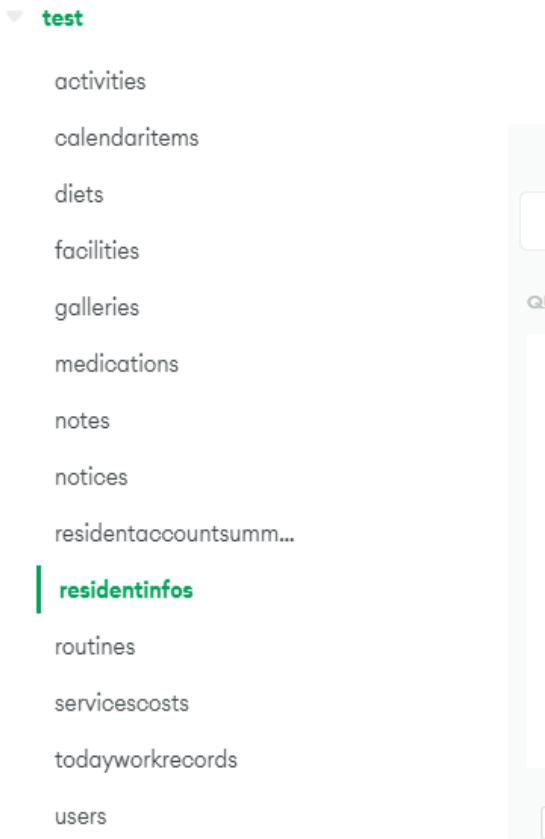


Figure 10.1 MongoDB collections

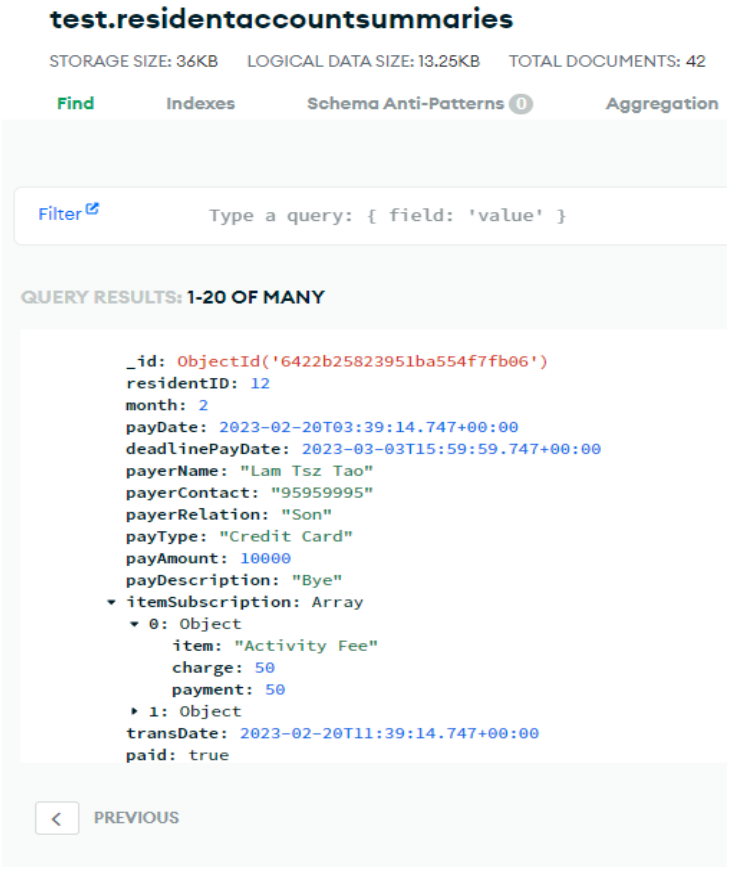


Figure 10.2 Document in resident account summary

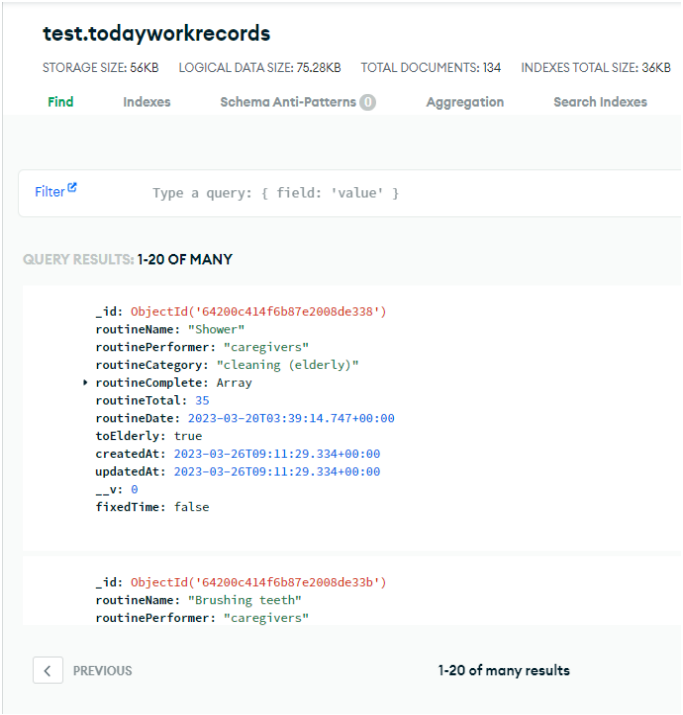


Figure 10.3 Document in today work records

4. Detailed Methodology and implementation

4.1 Class diagram

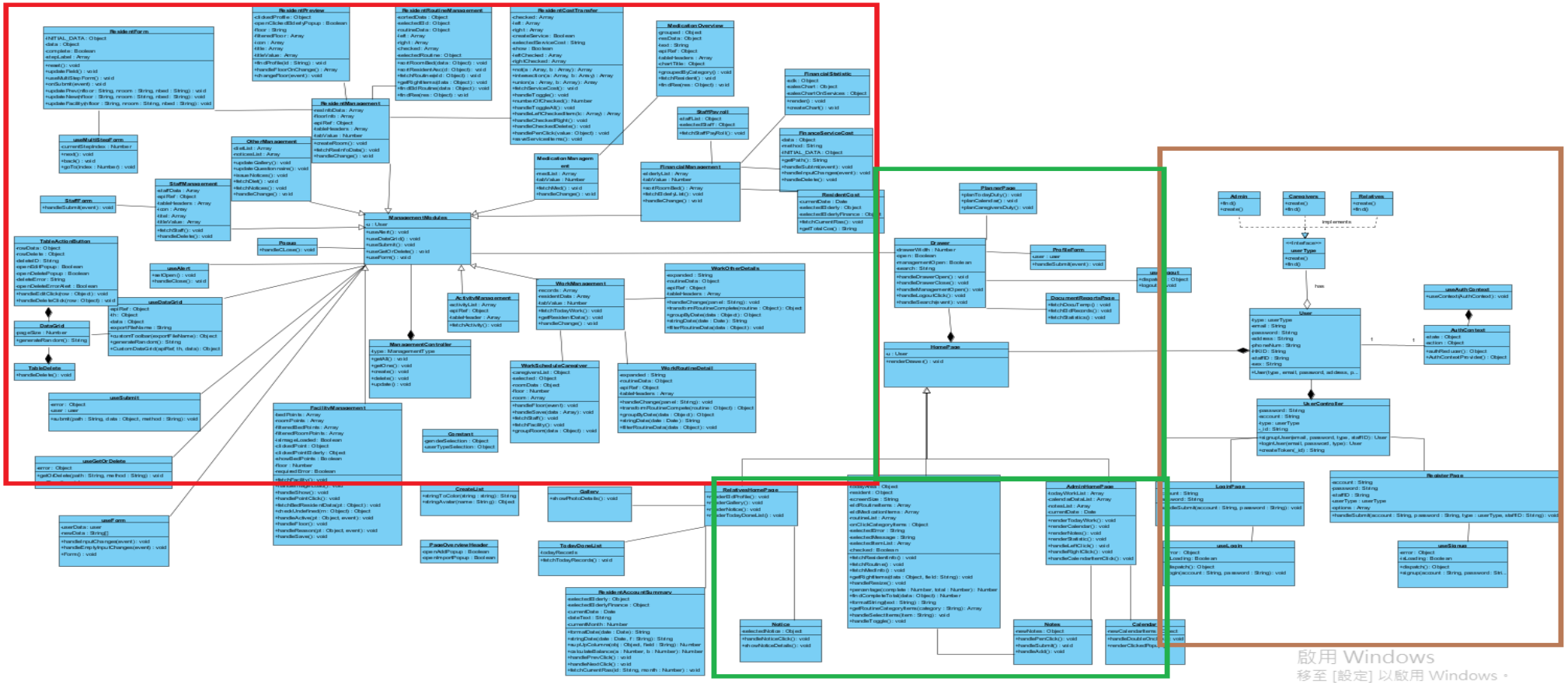


Figure 11.1 Class diagram overview

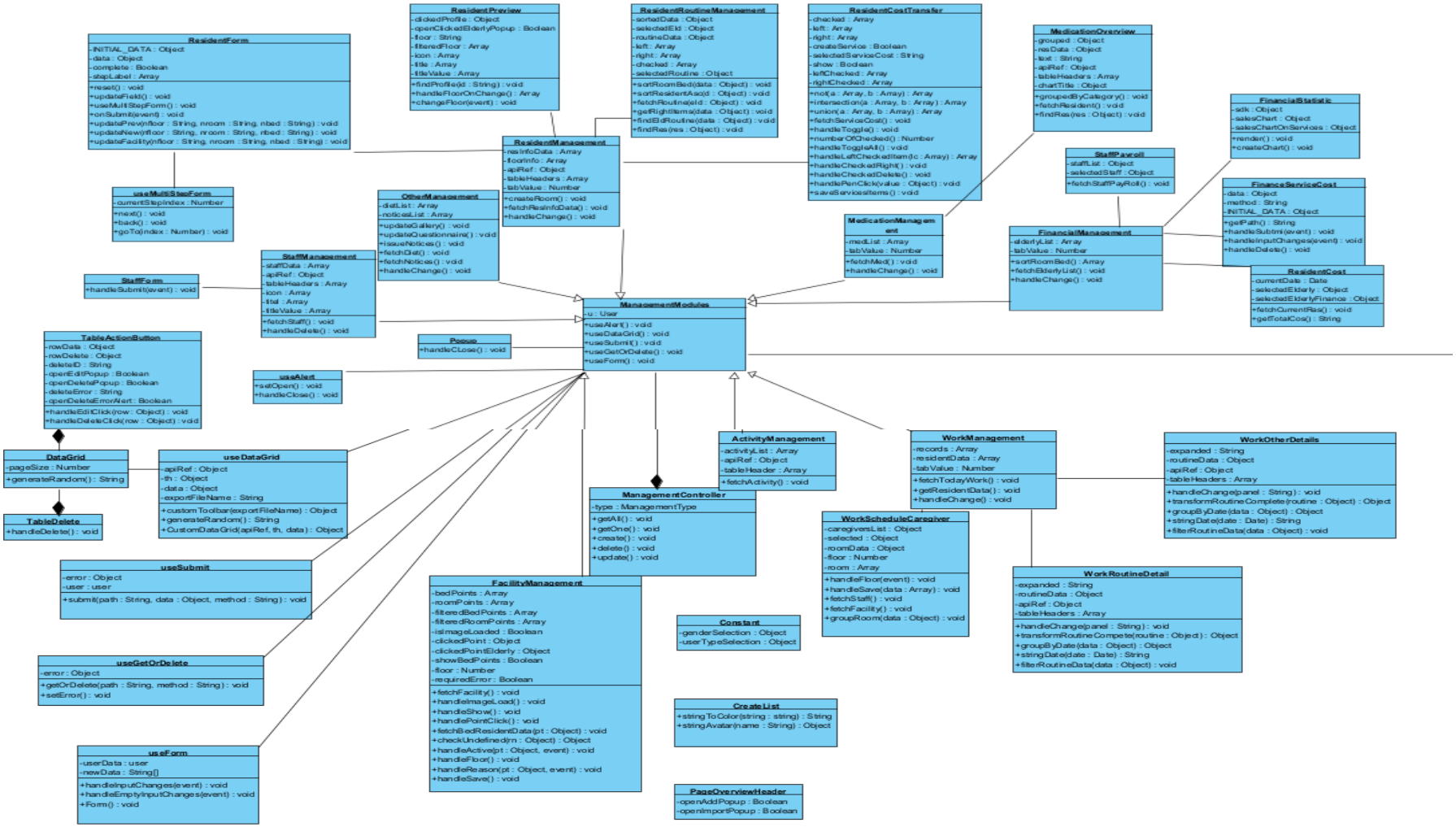


Figure 11.2 Class diagram for management modules

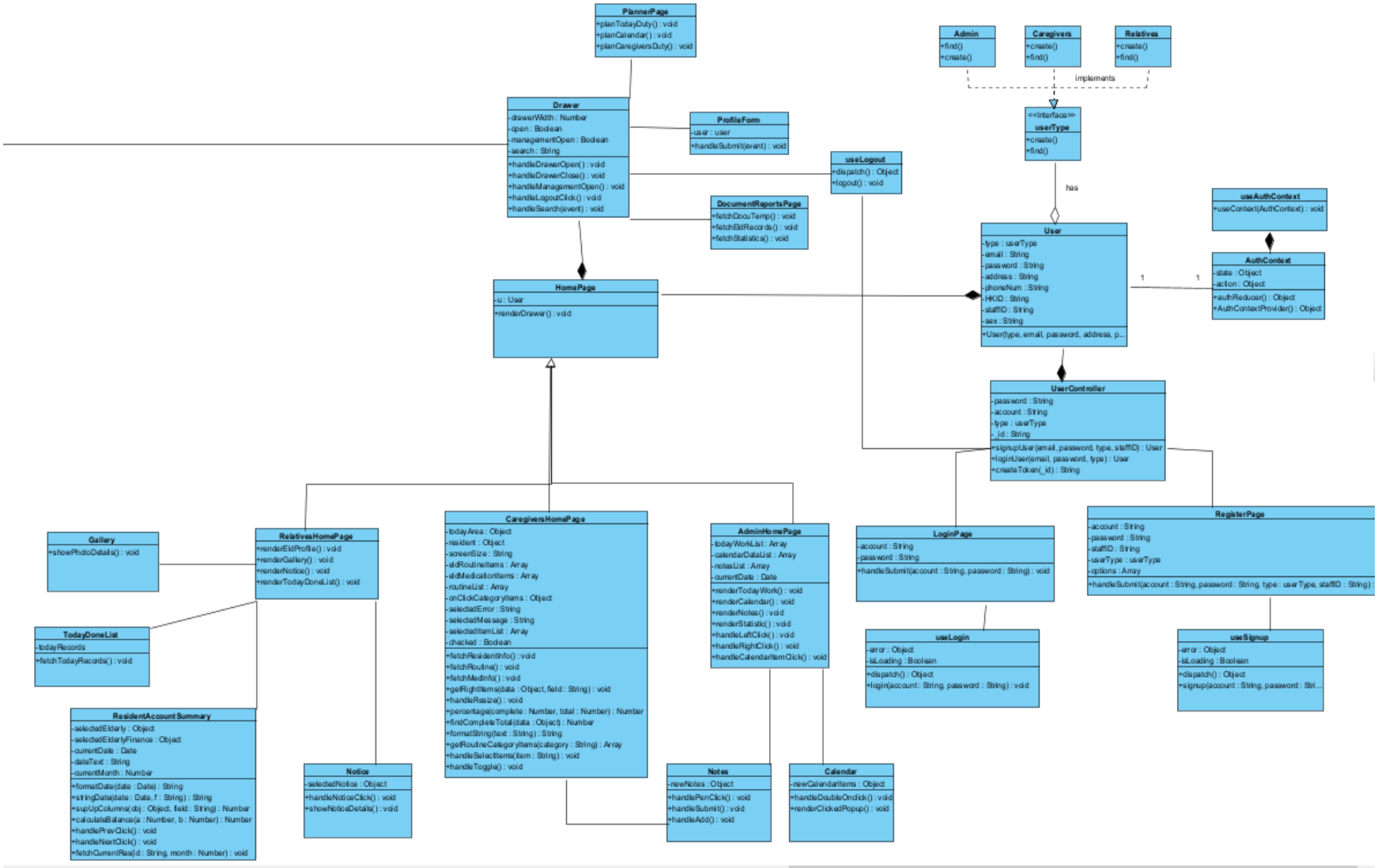


Figure 11.3 Class diagram for homepages and users identity

Classname	**Description**
User	
LoginPage	It is a page for users to login
RegisterPage	It is a page for users to signup
useLogin	It is a helper function for users to login (e.g. validation)
useSignup	It is a helper function for users to signup (e.g. validation)
useLogout	It is a helper function for users to logout
User	It is a model class that stores user personal and authentication information
AuthContext	It is a global state variable to check whether the user has logged in or logged out
useAuthContext	It is a helper function for AuthContext to be called in other components
userType	It is a superclass that records the type of user
Admin	It is a subclass that implements userType with the identity of admin
Caregivers	It is a subclass that implements userType with the identity of caregivers
Relatives	It is a subclass that implements userType with the identity of relatives
HomePage	
HomePage	It is a superclass that renders the drawer component and respective homepage information
AdminHomePage	It is a page that admin will be navigated after login
CaregiverHomePage	It is a page that caregivers will be navigated after login to perform documentation
RelativesHomePage	It is a page that relatives can view components of resident account summary, gallery, todayDoneList and notices
Calendar	It is a component that shows users' scheduled duties in a virtual calendar
Notes	It is a component for users' reminders
Notices	It is a component for users to issue notices
Gallery	It is a component that displays images of elderly
TodayDoneList	It is a component that displays todays medication and routine records of

	respective elderly
ResidentAccountSummary	It is a component that displays the account summary for respective elderly
Drawer	It is a component that acts as a combination of side navigation bar and top navigation bar, and links other components in management modules.
ProfileForm	It is a component at the top navigation bar to show personal information
DocumentReportsPage	It is a page that allow users to fetch and use different document and report templates
PlannerPage	It is a page that plan schedules in the calendar
Management Modules	
ManagementModules	It is a superclass that provides patterns for its relevant subclass, and helper functions can be used across different management
ManagementController	It is a helper class that handles the management request made by user
Staff Management	
StaffForm	It is a component to create a new staff
ResidentsManagement	
ResidentForm	It is a component to create a new resident
useMultiStepForm	It is a component that helps create resident by inputting information into the form in multi-step
ResidentPreview	It is a page that resident overview is displayed
ResidentRoutineManagement	It is a page that set default routines to residents and manage the routine items
MedicationManagement	
MedicationOverview	It is a page that shows the medicine distribution and its vacancy, and allows users to search for medicines from openFDA
WorkManagement	
WorkScheduleCaregiver	It is a page that schedule caregivers today's working areas
WorkRoutineDetail	It is a page that displays records of routine data
WorkOtherDetail	It is a page that display records of non-routine data
FinancialManagement	

FinancialStatistic	It is a page that shows financial charts
ResidentCost	It is a page that displays the component of resident cost transfer
FinancialServiceCost	It is a component that displays the information of services items
ResidentCostTransfer	It is a component that helps setting default routines
Staff payroll	It is a page that view and manage staff payrolls
Other components	
Popup	It is a component that pops a dialog
DataGrid	It is a component that stores data
TableActionButtpn	It is a component that contains a edit and delete button for datagrid
TableDelete	It is a component that handles deletion in data
useDataGrid	It is a helper function so that other management can import the function to use the data grid
useAlert	It is a helper function so that other management can import the function to use the alert to display success or error message
useSubmit	It is a helper function to submit data to database
useGetOrDelete	It is a helper function to fetch or delete data from databases
useForm	It is a helper function for other management to use form for creating data

Table 2 Class diagram description

4.2 Directory and file structure

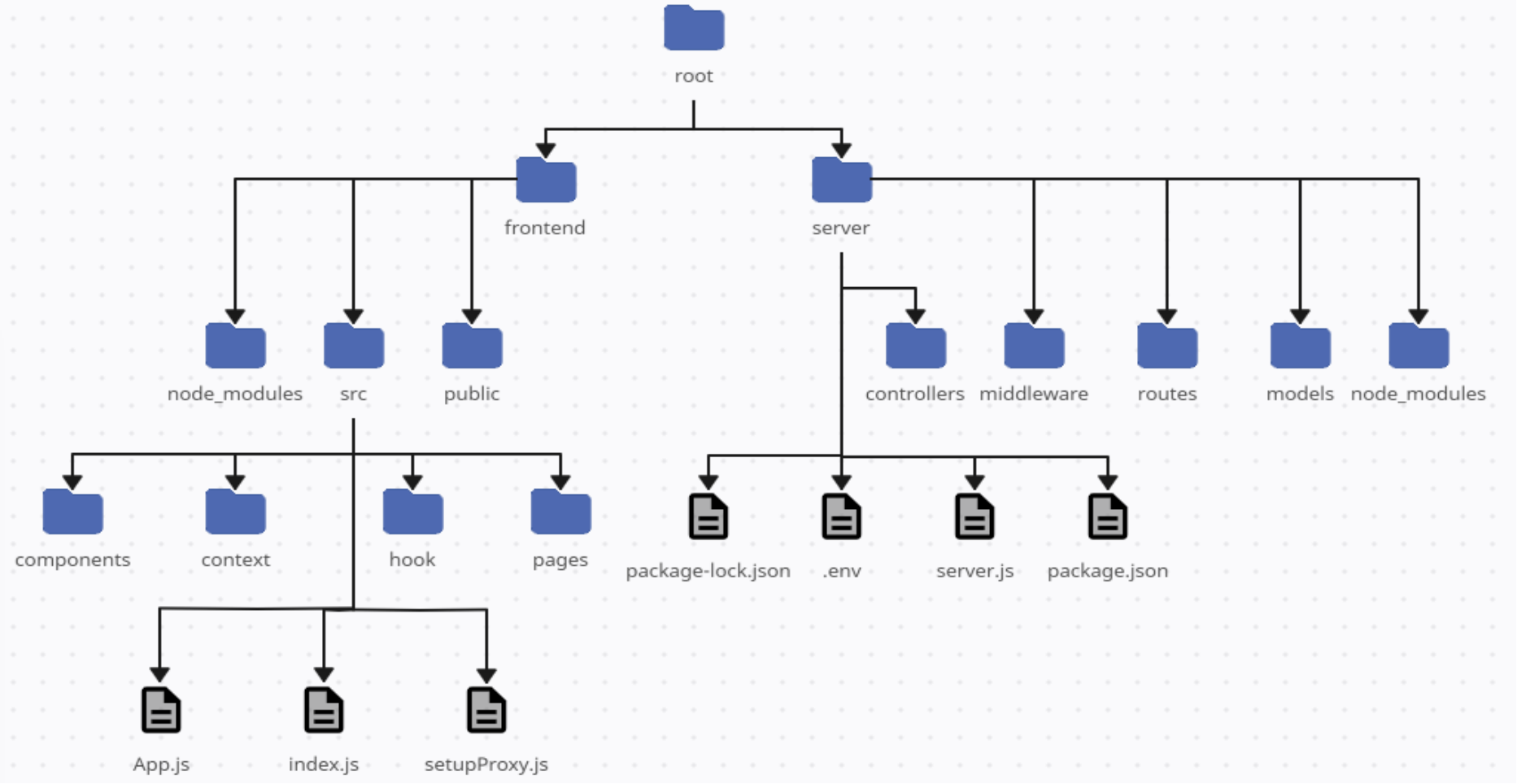


Figure 10 Directory and file structure

Folder Name	Description
node_modules	Itstores all the open-source reusable packages by node package manager
src	It contains most of the frontend development elements, including pages, components, hook, context and other javascript files
public	It contains some frontend development elements such as images and stylesheets
components	It stores and defines the elements that will be frequently reused in the project such as buttons and navbar
context	It stores and defines the logic of operations such as sharing of state to avoid prop drilling
hook	It stores all the self-defined function
pages	It stores all the frontend pages
controllers	It stores all the handlers for request handling
middleware	It stores all the middleware functions that define the app logic
routes	It stores all the routes that defines the app routing and logic
models	It stores all the schemas and schema logic for data storage and retrieval from MongoDB

Table 3.1 Folder description

File Name	Description
index.js	It creates the root element for app.js initialization
App.js	It initializes the system with the integration of app logic and components
setupProxy.js	It connects the frontend and backend proxy for localhost
package.json	It contains information about the project such as name, list of dependencies, package version to configure and describe the interaction with the project
package-lock.json	It contains information about the project such as name, list of dependencies, package version to configure and describe the interaction with the project
server.js	It contains the logic of connection to MongoDB and routing (express)
.env	It stores private variables to prevent being viewed in public

Table 3.2 File description

4.3 Major technical components

In this project, different software and modern web technologies are integrated. This section will first introduce how the MERN stack can implement in the design, followed by the User Interfaces (UI) implementation and methods for handling HTTP requests.

4.3.1 MERN stack

As the MERN stack stands for MongoDB, Express.JS, React.JS and Node.JS, the implementation of these technologies will be discussed in this session.

4.3.1.1 MongoDB

4.3.1.1.1 Database connection

The connection to MongoDB is necessary for data storage and retrieval. It will require the package “Mongoose”, which is one of the packages for Node.js. With the connection string (MONGO_URI) that contains the user name and password generated by MongoDB and the setup of the port number to bind and listen to the connections in the .env file, the connection can be made successful and starts the backend server. Figure 11.2 shows the successful connection to fetch data between the React application and the backend database.

```
const app = express();
// connect to db
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    //listen for requests once we connect to db
    app.listen(process.env.PORT, () => {
      console.log("Connected to db and Listening on port", process.env.PORT);
    });
  })
  .catch((error) => {
    console.log(error);
  });

```

Figure 11.1 Database connection

```
PS D:\Matthew\University\FYP\system\backend> npm run dev

> backend@1.0.0 dev
> nodemon server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Connected to db and Listening on port 4000
/api/management/residents GET
/api/management/residents GET
/api/management/residents GET
/api/management/residents GET
/api/management/medication GET
/api/management/medication GET
/api/management/residents GET
/api/management/residents GET
```

Figure 11.2 Data fetching for management modules

4.3.1.1.2 Database schema and static logic

In addition to the database connection, mongoose also allows users to define database schemas for collections inside source code files. It acts as a driver or a modelling tool for MongoDB to provide the creation of data to the database. Static logic and functions can also be defined for application logic. In Figure 12.1 &

Figure 12.2, the schema for users, which stores users' authentication and personal information, and user registration logic can be both integrated inside the same file.

```
const userSchema = new Schema({
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    address: {
        type: String,
        //required: true
    },
    phoneNum: {
        type: String,
        //required: true
    },
    sex: {
        type: String,
        //required: true
    },
    HKID: {
        type: String,
        //required: true
    },
    staffID: {
        type: String,
        //required: true
    }
})

```

Figure 12.1 Database schema

```
// static signup method
userSchema.statics.signup = async function (email, password) {

    // Validation
    if(!email || !password){
        throw Error('All fields must be filled')
    }

    if(!validator.isEmail(email)){
        throw Error('Email not valid')
    }

    // usually User.findOne..., but not declared, simply use this.findOne
    const exists = await this.findOne({email})

    if(exists){
        throw Error('Email already in use')
    }

    // Hashing (mern authentication #3)
    const salt = await bcrypt.genSalt(10)
    const hash = await bcrypt.hash(password, salt)

    const user = await this.create({email, password: hash, address: '', phoneNum:'', sex: '', staffID:'

    return user
}
```

Figure 12.2 Schema static logic

4.3.1.1.3 Trigger

Database triggers are useful for management systems as they can help automate the tasks that require our manual intervention to save time and prevent unexpected errors. In Figure 13, a scheduled trigger (cron job) have been set up so that at the start of every month, a new resident account summary will be automatically generated for each resident. The trigger not only could save time in generating the reports manually, but also help prevent unexpected human error.

Function

```
exports = function() {
  const mongodb = context.services.get("FYP")
  const ras = mongodb.db("test").collection("residentaccountsummaries")
  const res = mongodb.db("test").collection("residentinfos")

  const currentDate = new Date();
  const prevMonth = currentDate.getMonth()
  const currentMonth = currentDate.getMonth() + 1;
  const currentYear = currentDate.getFullYear();
  const lastMonthSecond = new Date(currentYear, currentMonth, 0, 23, 59, 59, 999)


  function previousRas(id, month, callback) {
    ras.findOne({$and: [{'residentID': id}, {'month': month}]}), function(err, result) {
      if (err) {
        return console.error(err);
      } else {
        console.log(id, result)
        return result ? result.itemSubscription : []
      }
    }
  }

  return res.aggregate([
    // all residents
    {$match: {}}
  ]).toArray()
    .then(residents => {
      residents.forEach(resident => {
        const accountSummary = {
            residentID: resident.residentID,
            month: currentMonth,
            payDate: "",
            deadlinePayData: lastMonthSecond,
            payerName: resident.relativesName,
            payerContact: resident.relativesPhone,
            payerRelation: "family",
            payType: "",
            payAmount: "",
            payDescription: "",
            // default: previous month item subscription
            itemSubscription: previousRas(resident.residentID, prevMonth),
            transDate: "",
            paid: false,
            received: false
        }
        ras.insertOne(accountSummary)

      })
    })
    .catch(
      (err=> console.error("Failed to generate report for resident: ${resident.residentID} ", err))
    )

};
```

Figure 13 Triggers on resident account summary

4.3.1.1.4 MongoDB charts

Business charts can also be generated in MongoDB through the use of different visualization tools and libraries. These charts can be powerful repositories for staff users to formulate their plans or make certain significant actions. In MongoDB, complex data structures can be aggregated into categories and values and transformed into charts. Data can then be visualized through charts and import to

the React application for data analytics. Figure 14 shows the financial statement of the total amount of charges and payments of routine items in 2023 based on the categories of month and item.

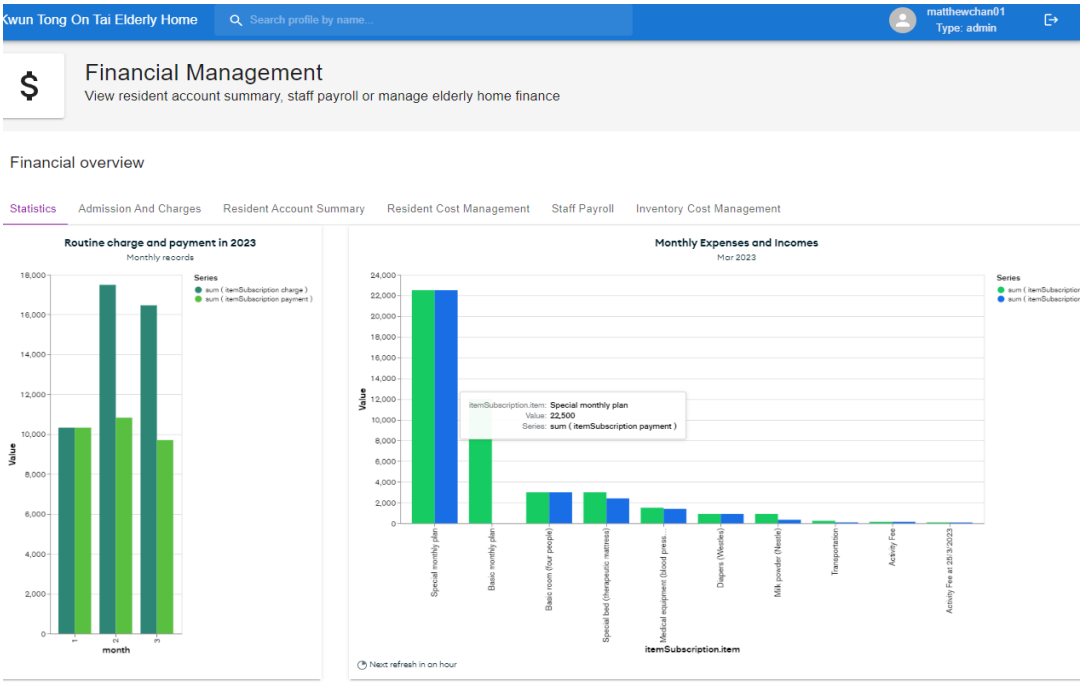


Figure 14 MongoDB charts in financial management

4.3.1.2 Express.JS

The main feature of Express.JS would be path routing. Express routing defines how the application's endpoints handle requests. Figure 15 shows a router that gathers all the routes and in each route is holded with a function. When a request is made to the directory that belongs to the router, it can handle the request and performs suitable actions of that function.

```
const router = express.Router()
// user needs to be authenticated to use these following functions
router.use(requireAuth)

//GET all records
router.get('/', getWorks)

//GET single record
router.get('/:id', getWork)

//POST a new record
router.post('/', createWork)

//DELETE a record
router.delete('/:id', deleteWork)

//UPDATE a record
router.patch('/:id', updateWork)

module.exports = router
```

Figure 15 Routing in Express.JS

4.3.1.3 React.JS

4.3.1.3.1 React-router

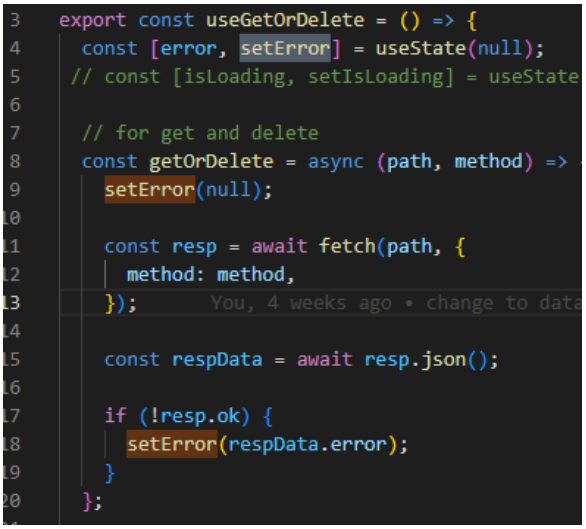
React-router is a library for managing routing in the frontend React application. With its client-side routing properties, when making a request to navigate between different pages, with the use of react-router, the web application is no longer fully reloaded and only fetches the data from the server upon changes and updates. It is expected that the system can provide low latency and seamless experiences to users with this feature. Figure 16 demonstrates the application of BrowserRouter and Routes from react-router to handle page changes.

```
<BrowserRouter>
  <div className="pages">
    <Routes>
      <Route path="/home" element={user ? userHomePage(user.userType): <Navigate to="/login"/>}></Route>
      <Route path="/login" element={!user? <Login />: <Navigate to="/home"/>} />
      <Route path="/signup" element={!user ? <Signup/>: <Navigate to="/home"/>} />
      <Route path="/profile" element={user? <Drawer main={<Profile/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/staff" element={user? <Drawer main={<StaffManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/residents" element={user? <Drawer main={<ResidentsManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/work" element={user? <Drawer main={<WorkManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/finance" element={user? <Drawer main={<FinanceManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/medication" element={user? <Drawer main={<MedicationManagement/>}/>: <Navigate to="/login"/
      >}/>
      <Route path="/management/activity" element={user? <Drawer main={<ActivityManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/facility" element={user? <Drawer main={<FacilityManagement/>}/>: <Navigate to="/login"/>}/>
      <Route path="/management/others" element={user? <Drawer main={<OthersManagement/>}/>: <Navigate to="/login"/>}/>
    </Routes>
  </div>
</BrowserRouter>
div>
```

Figure 16 React browser router

4.3.1.3.2 State management

State is the most important element in React application. It helps determine the data flow inside the system and the internal changes of state can make huge UI changes when users making interaction with the system. As React applications are built with components, the difficulty of managing states that need to be shared across different components would only become larger when the system becomes more complex. Therefore, state management is crucial for React applications. In this project, different state management techniques will be integrated with the system to achieve smooth and organized data flow between components. For example, hooks are one of the state management techniques which will be frequently used in this project, the following table will describe the effects of different hooks used.

State management	Description
useState	useState is similar to variable initialization. In Figure 17, error is defined as null initially, once the setError function is called, the error will become the content of the parameters in the setError function  *Figure 17 error with useState*
useEffect	useEffect helps perform side effects after render. The functions passed into the useEffect hook will trigger once the DOM update is finished. One of the typical usages of useEffect hook would be data fetching. In Figure 18, the useEffect is used to call fetchCurrentRas() whenever there is an update of selectedElderly and currentDate.  *Figure 18 fetch data with useEffect*
useMemo	useMemo prevents functions from needlessly running by

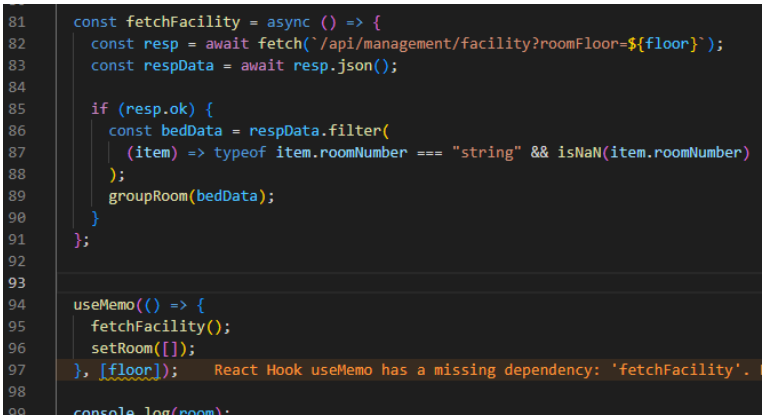
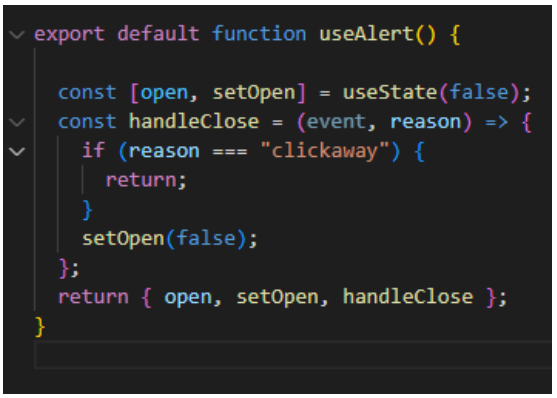
	memorizing the value in the dependencies, if one of them changes in the next rendering, it will cause the useMemo to invoke the functions inside. In Figure 19, if the floor is changed, it will invoke the fetchFacility() again to fetch room data of the floor  *Figure 19 fetch data with useMemo*
custom hook	Custom hooks are reusable and self-defined logic can help simplify the application logic. In Figure 20, the useAlert function is called throughout the application to display the messages after any data update.  *Figure 20 displays messages with useAlert*

Table 4 State management

4.3.1.4 NodeJS

NodeJS is the runtime environment for Javascript which helps both the frontend and backend developments in different aspects. The node package manager (NPM) contains different node module files and software packages that help boost development. For backend development, mongoose is one of the NPM packages, other packages such as bcrypt and validator are also implemented into the system for database logic. For frontend development, the react-router-dom package is important for page navigation and packages such as react-calendar can be useful for component development.

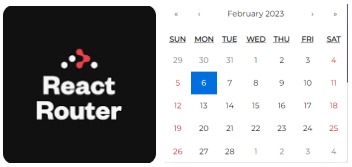
Development	**Packages**
Backend	
Frontend	

Table 5 NodeJS packages in system

4.3.2 User Interfaces Implementation

To enhances users' usability and readability to the system, Material UI will be used for user interface (UI) development in this project. Material UI is a UI component library for React with built-in and customizable components for developers to boost their development. In Figure 21, different Material UI components are integrated to enhance users' UI and UX experiences, which include Grid (Blue), Tab (Green), Drawer (Yellow), Stack (Red), Avatar (Brown) and DataGrid (Brown).

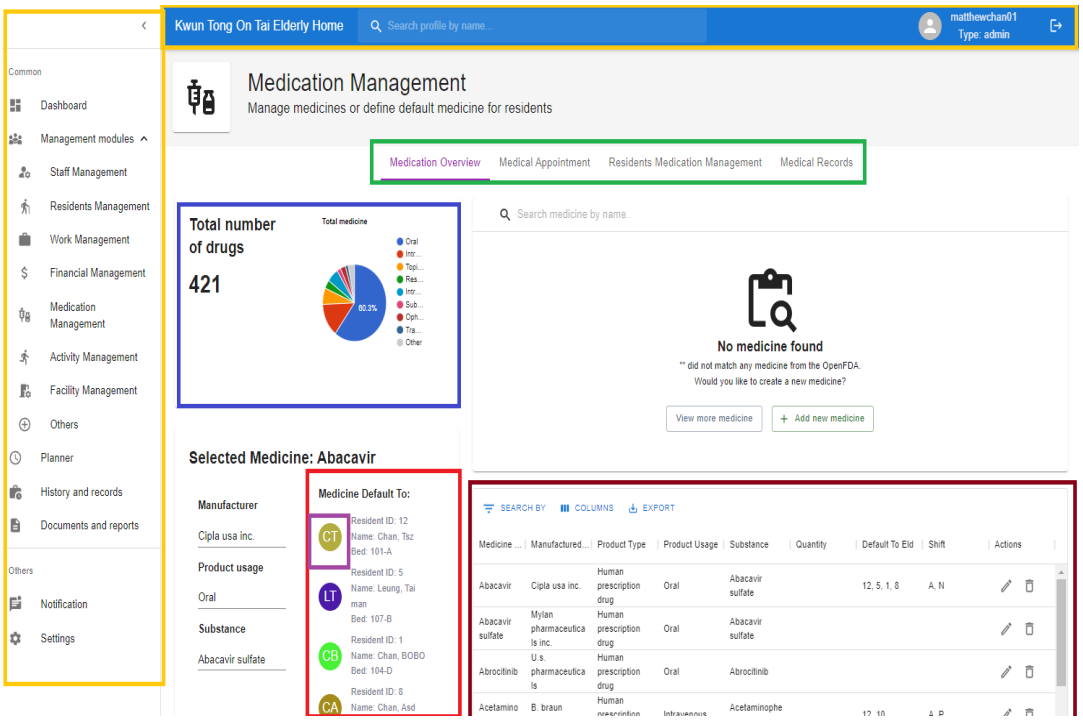


Figure 21 Components in Material UI

In addition to the variety of components for UI design, the most valuable feature in Material UI would be achieving responsive web design. Responsive means the application needs to know how to resize automatically when the user is browsing on different devices. It is of great importance to achieve the design in any elderly home applications, especially for caregivers who need to carry their tablet devices

around to perform documentation and direct health care. In Figure 22.1, it shows the normal display when users log in to the application as caregivers on a desktop. The responsive web design is achieved when using an iPad in Figure 22.2. The shrunk grid items (today's working area, routine progress) begin to grow. The Duty Today list also separates into two columns to fill up the empty spaces. All these responsive features improve UI/UX experiences for users.

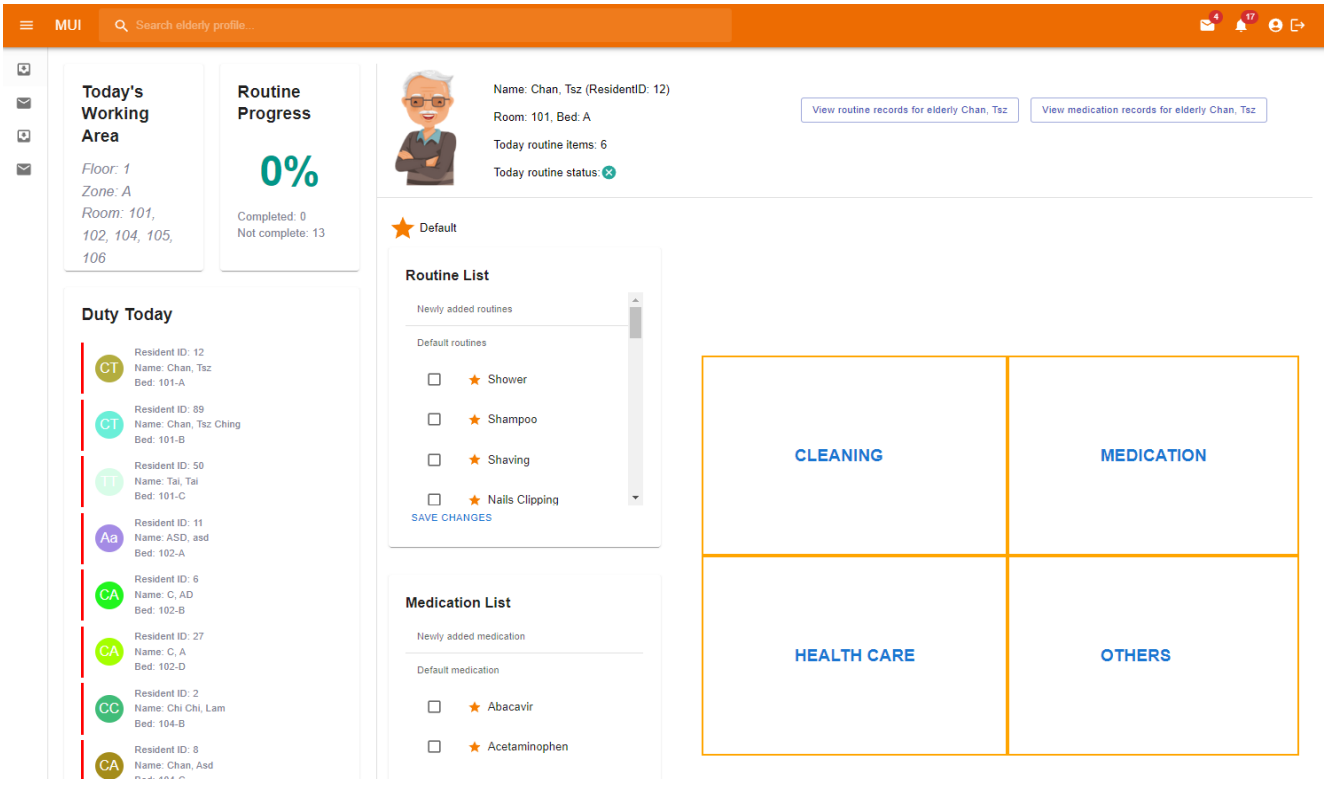


Figure 22.1 Normal display on desktop

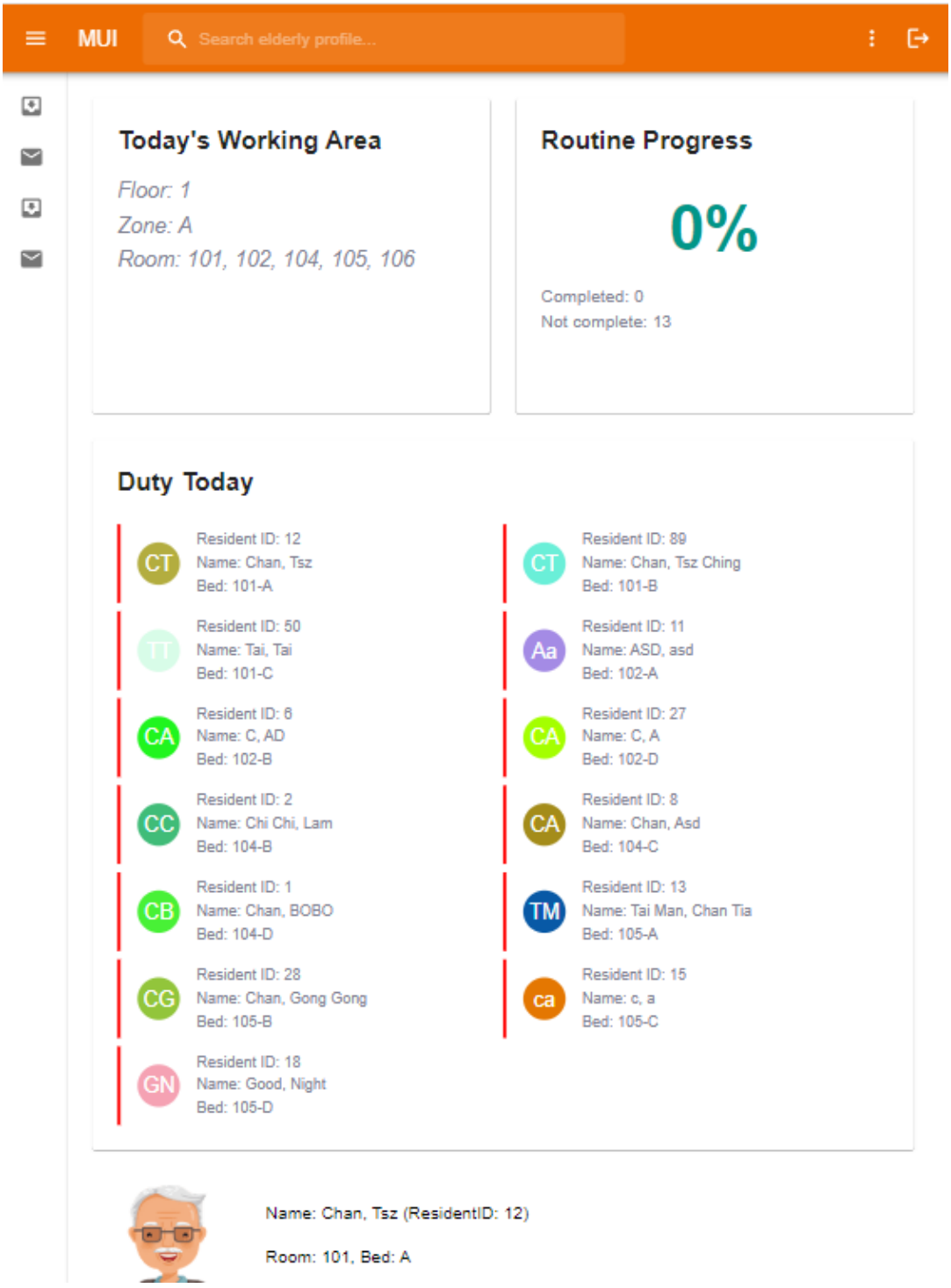


Figure 22.2 Responsive display on tablet devices

4.3.3 RESTful API

RESTful API uses HTTP request methods, which are a collection of request methods that handle requests and enables communications between server and client. The most common HTTP request methods are GET, POST, PATCH and DELETE, which corresponds to the CRUD (create, read, update, delete) operations respectively. If the client sends out a POST request, the fetch API will also be used to define the directory of requests and responses. Figure 23 shows a POST request to create an account with email and password and the destination directory is the signup function, if the request is made successful, responses will be sent back to the application in JSON format.

```
const resp = await fetch("/api/user/signup", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ account, password, userType, staffID }),
});
```

Figure 23.1 POST request


```
{
    "account": "matthew77",
    "password": "$2b$10$HTpw58Ly7SSgNKNelQkyYeMiHohC7bTHCHW9SpTZN2gUOki1ZNW.i",
    "userType": "Admin",
    "active": true,
    "email": "matthew.ctn56@gmail.com",
    "_id": "64250176b60b9ea3e77ad8c0",
    "createdAt": "2023-03-30T03:26:46.659Z",
    "updatedAt": "2023-03-30T03:26:46.659Z",
    "__v": 0,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NDI1MDE3NmI2MGI5ZWE
}
```


Figure 23.2 POST request response in Postman

4.4 Algorithm design and implementation to problems

In this part, the detailed algorithm for functions and its implementations will be discussed. The following are the solutions and its algorithm to address the problems defined in the problem statement, or the implementations of the proposed solutions.

4.4.1 Searching for paper files

One of the concerns is that it can be time-consuming for staff to search for specific paper files to view resident records. In this management system, a resident overview is designed (Figure 24.1). Residents are grouped in rooms so that staff can quickly find out the residents. The overview can further minimize by selecting a specific floor or inputting the resident's names to filter out the resident. Staff can then click on the icons to view residents' personal data (Figure 24.2)

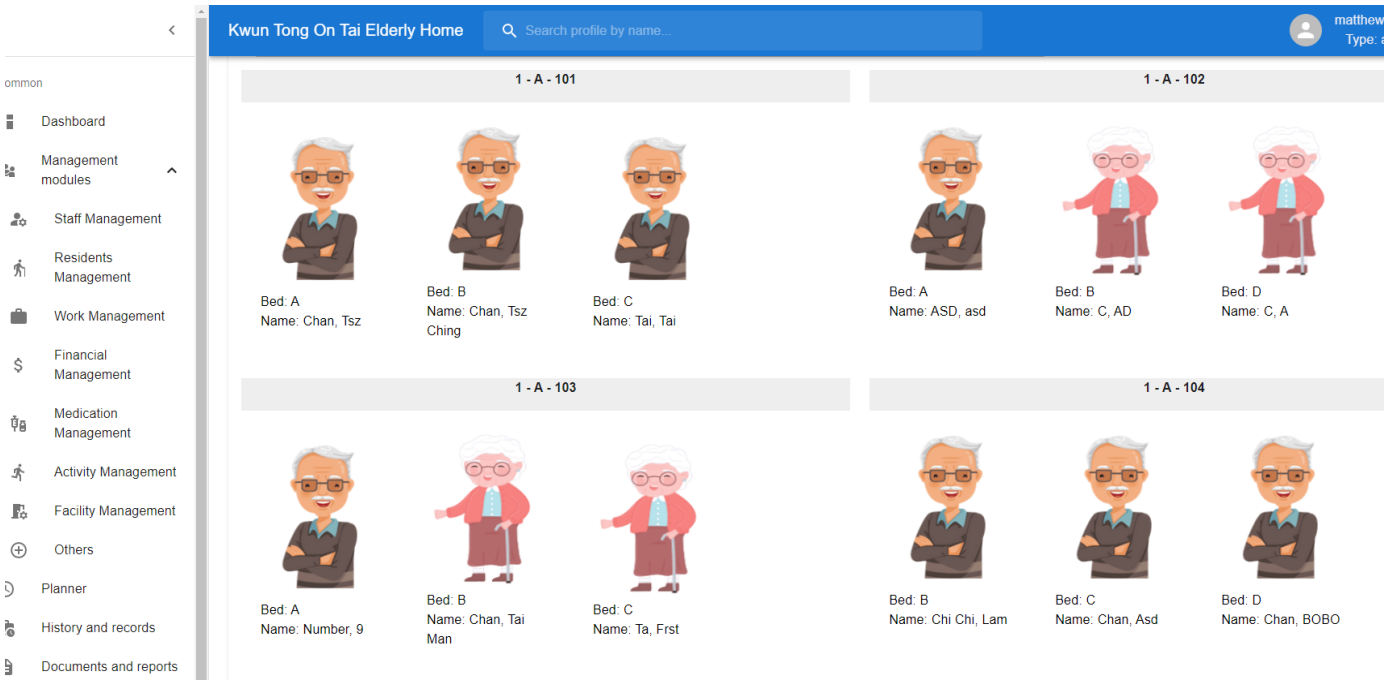


Figure 24.1 Resident Overview

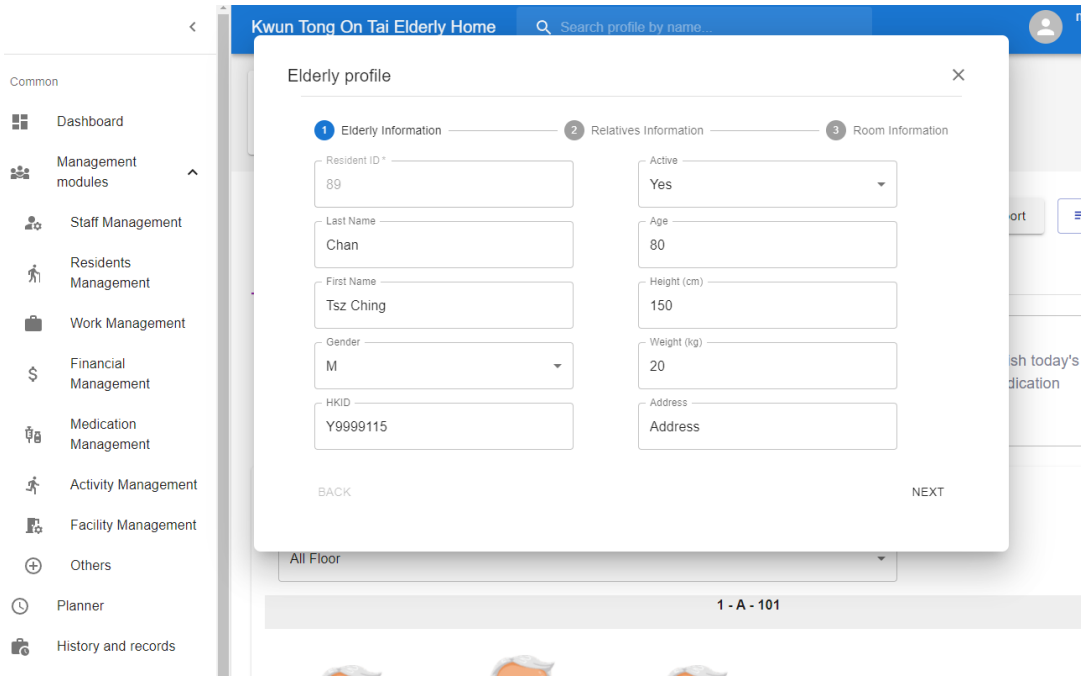


Figure 24.2 Resident multiform data

The resident overview is made successful based on the algorithm on createRooms() (Figure 24.3 & Figure 24.4) and handleFloorChanges() (Figure 24.5 & 24.6). When data fetches, for each room, it creates a new array so that residents can be pushed into the respective room array if their room has been created. In default, the resident overview will accept all rooms as Figure 24.4 shows a full list of residents. If the floor changes, it will invoke the handleFloorChanges() so that the data will be filtered out and the rooms only on that floor will remain. Figure 24.6 shows the array of rooms on floor 3. Therefore, only 3 residents will be shown on resident overview. The algorithm helps gather data and filter data if necessary to search for the elderly as fast as possible.

```
198  useEffect(() => {
199    const fetchResInfoData = async () => {
200      const resp = await fetch("/api/management/residents");
201      const respData = await resp.json();       You, 3 weeks ago • finish add an r
202
203      if (resp.ok) {
204        setResInfoData(respData);
205        createRoom(respData);
206      }
207    };
208
209    const createRoom = (data) => {
210      const uniqueStringsSet = new Set();
211      const roomElderly = {};
212
213      for (let i = 0; i < data.length; i++) {
214        const { floor, zone, room, bed, lastName, firstName, sex, residentID } =
215          data[i];
216        const roomStr = `${floor} - ${zone} - ${room}`;
217        if (!uniqueStringsSet.has(roomStr)) {
218          uniqueStringsSet.add(roomStr);
219          roomElderly[roomStr] = [];
220        }
221
222        roomElderly[roomStr] = [
223          ...roomElderly[roomStr],
224          [bed, lastName, firstName, residentID, sex],
225        ];
226        roomElderly[roomStr].sort();
227      }
228      setFloorInfo(roomElderly);
229    };
230
231
```

Figure 24.3 Algorithms to create rooms after data fetching

```
▼{1 - A - 104: Array(3), 1 - A - 103: Array(3), 1 - B - 115: Array(4), 1 - A - 107: Array(3), 1 - A - 102: Array(3), …}
  ▶1 - A - 101: (3) [Array(5), Array(5), Array(5)]
  ▶1 - A - 102: (3) [Array(5), Array(5), Array(5)]
  ▶1 - A - 103: (3) [Array(5), Array(5), Array(5)]
  ▶1 - A - 104: (3) [Array(5), Array(5), Array(5)]
  ▶1 - A - 105: (4) [Array(5), Array(5), Array(5), Array(5)]
  ▶1 - A - 107: (3) [Array(5), Array(5), Array(5)]
  ▶1 - B - 114: (4) [Array(5), Array(5), Array(5), Array(5)]
  ▶1 - B - 115: (4) [Array(5), Array(5), Array(5), Array(5)]
  ▶2 - A - 201: [Array(5)]
  ▶2 - B - 218: (2) [Array(5), Array(5)]
  ▶3 - A - 301: (3) [Array(5), Array(5), Array(5)]
  ▶3 - A - 302: [Array(5)]
  ▶3 - B - 313: [Array(5)]
  ▶[[Prototype]]: Object
```

Figure 24.4 Results of createRooms()

```
35  const handleFloorOnchange = () => {
36    if (floor === null || floor === "all") {
37      setFilteredFloor(floorInfo);
38    } else {
39      const keys = Object.keys(floorInfo);
40      const filteredKeys = keys.filter((k) => k.charAt(0) === floor);
41
42      const filtered = Object.keys(floorInfo)
43        .filter((key) => filteredKeys.includes(key))
44        .reduce((obj, key) => {
45          obj[key] = floorInfo[key];
46          return obj;
47        }, {});
48
49      setFilteredFloor(filtered);
50    }
51  };
52
53  const changeFloor = (e) => {
54    setFloor(e.target.value);
55  };
56
57  useEffect(() => {
58    handleFloorOnchange();
59  }, [floor]);    React Hook useEffect has a missing dependency: 'hand
60
```

Figure 24.5 Algorithms when handling floor changes to filter rooms

```
▼ {3 - A - 301: Array(3), 3 - B - 313: Array(1), 3 - A - 302: Array(1)}
  ▼ 3 - A - 301: Array(3)
    ▶ 0: (5) ['A', 'LN', 'FN', 16, 'F']
    ▶ 1: (5) ['B', 'dha', 'akudhasd', 23, 'M']
    ▶ 2: (5) ['D', 'TAIOHSID', 'AOIHDIA', 99, 'M']
      length: 3
    ▶ [[Prototype]]: Array(0)
  ▼ 3 - A - 302: Array(1)
    ▶ 0: (5) ['A', 'LN', 'Fn', 20, 'M']
      length: 1
    ▶ [[Prototype]]: Array(0)
  ▼ 3 - B - 313: Array(1)
    ▶ 0: (5) ['A', 'Lam', 'Tai Man', 17, 'M']
      length: 1
    ▶ [[Prototype]]: Array(0)
  ▶ [[Prototype]]: Object
```

Figure 24.6 Result of handleFloorChanges()

4.4.2 Long documentation process

Another concern is about the documentation process as it almost takes a quarter of caregivers' working time. Therefore, there are multiple features to address the concerns, including the rendering of the elderly list in caregivers' homepages to let caregivers know who are the targeted elderly, default medication and routine management to shorten the searching of routine items or any repetition works.

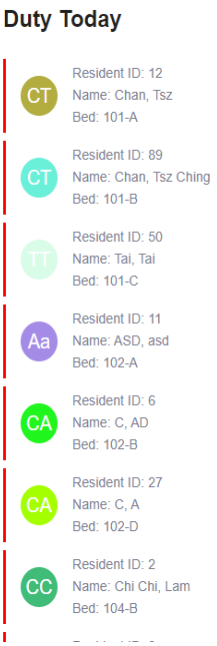


Figure 25.1 Rendering Duty Today List in caregivers' homepage

The default routine and medication management allow caregivers or admins to decide residents' default routine and medication. In Figure 25.2, users can toggle the selected items on the left to add items to the right (default routine items list for the elderly), which will be shown on the caregiver's homepage. Therefore, caregivers do not have to click and search for the category items every day if the routine is on a daily basis, which can help speed up the documentation process.

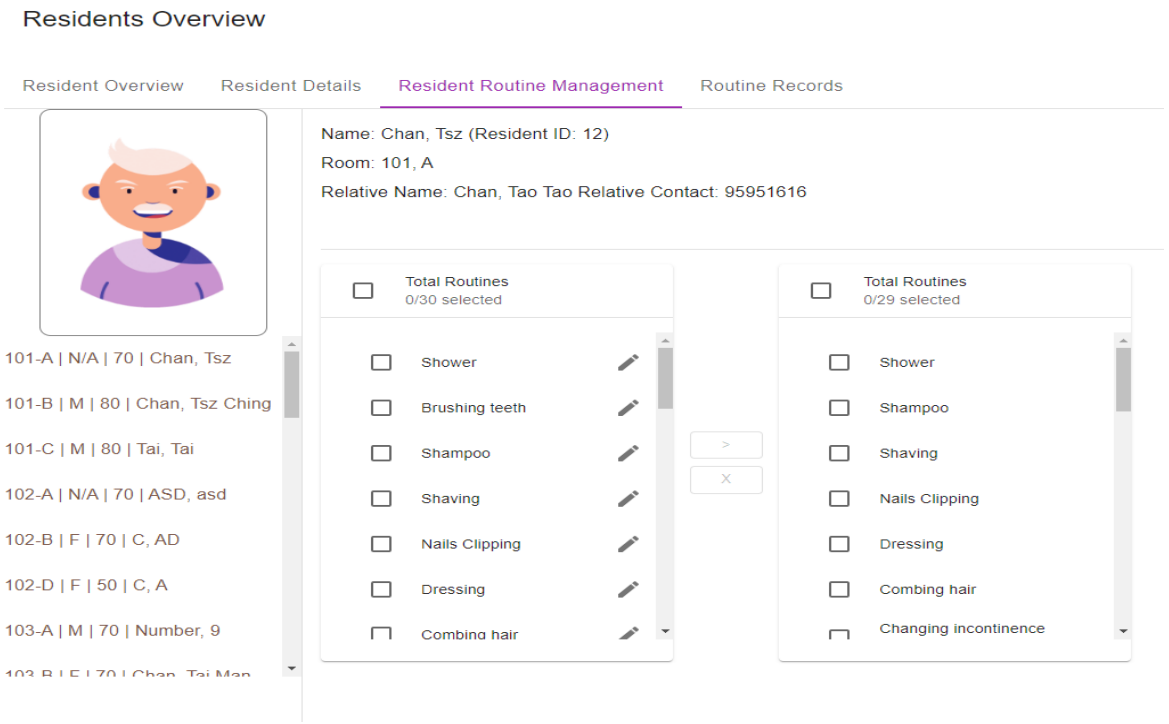


Figure 25.2 Default Routine Management by elderly

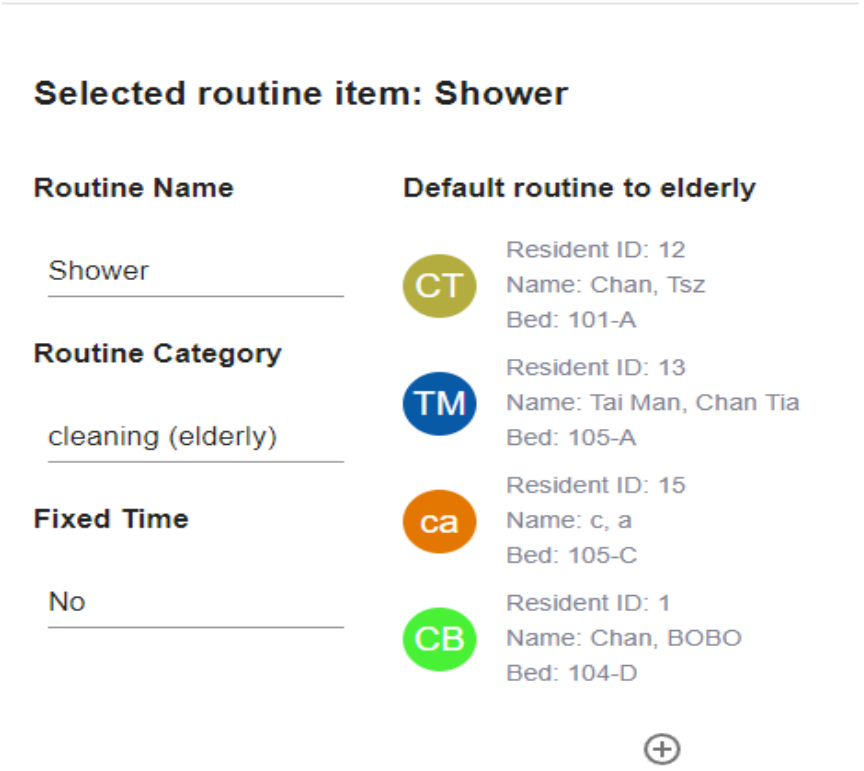


Figure 25.3 Default routine management by items

4.5 Other implementations

4.5.1 Resident Account Summary

Figure 26 shows the resident account summary for the current month. On top of the account summary tables, it shows the amount due and paid status. In the table, there are payment information and account summary information, which includes the subscribed items (services) for the current month. The table will also be shown on relatives' homepages so that they can know about the costs and financial status.

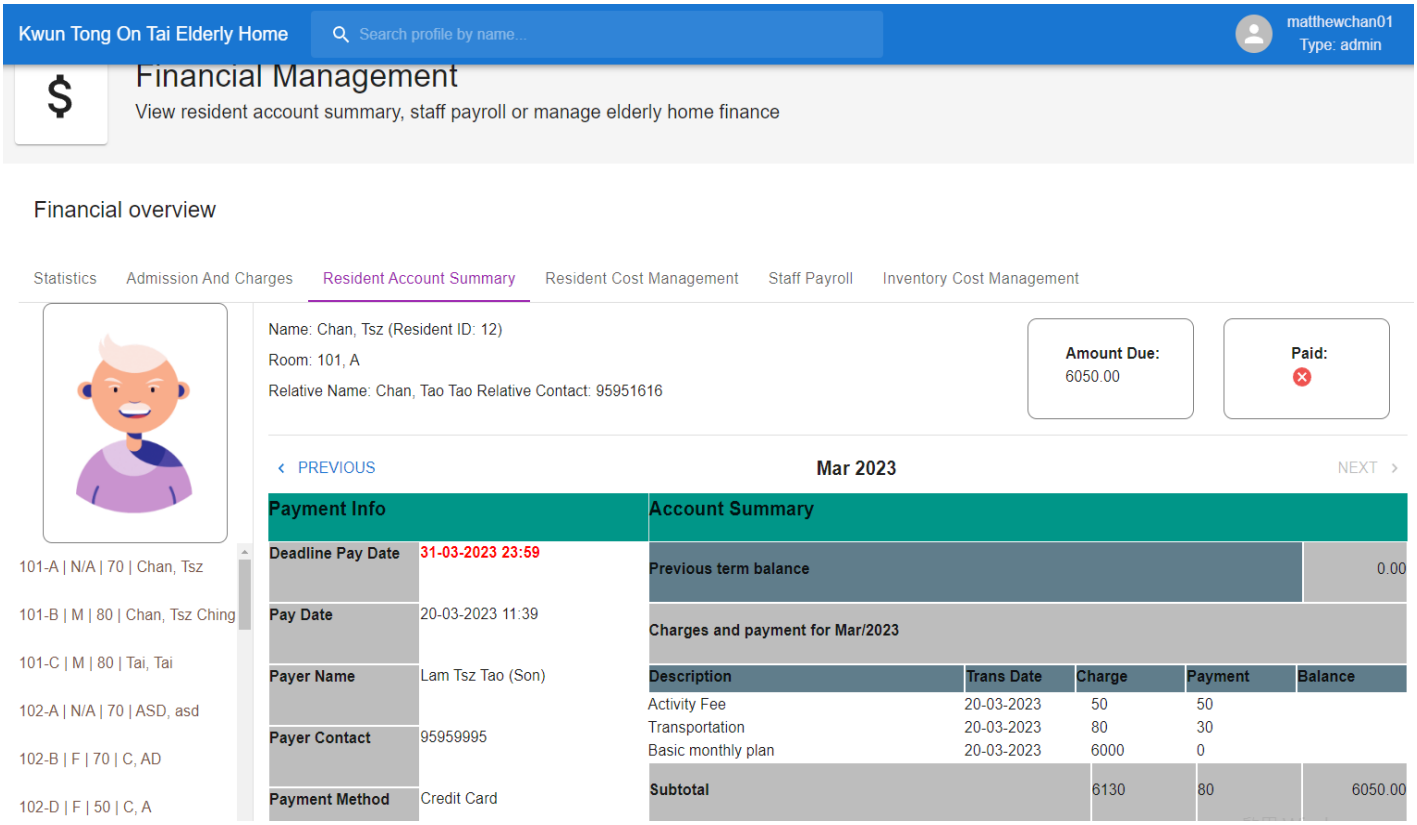


Figure 26 Resident Account Summary

4.5.2 Facility Management

Figure 27 shows the facility management in elderly homes. In this page, users can click on the points dotted on the floor plan to check the status of the rooms or beds. They can also change the floor to change the floor plan and the room points. It is useful in case there are any special circumstances that have to disable the rooms or beds. Users can also know the vacancy of the selected points.

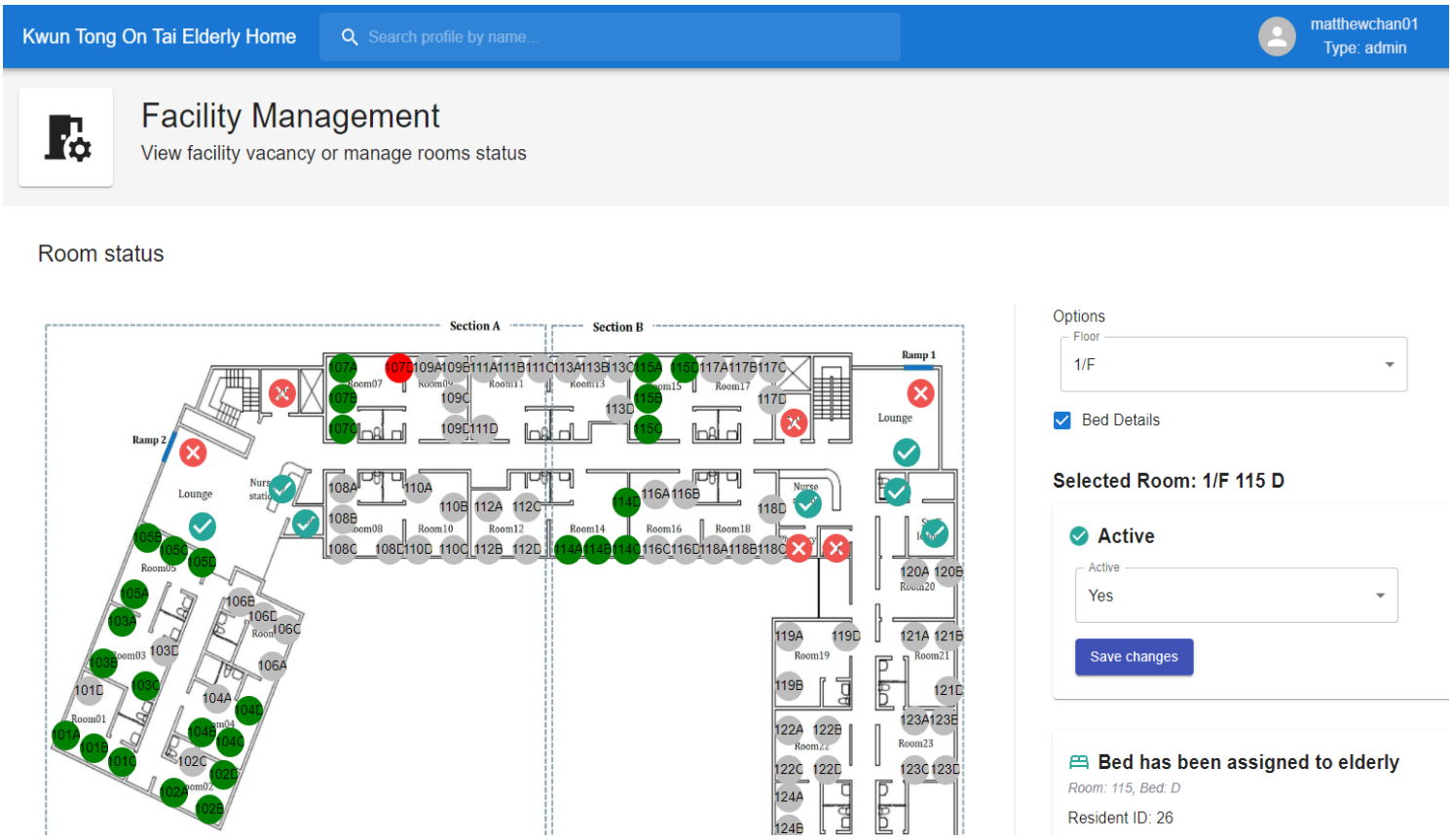


Figure 27 Facility Management

4.5.3 Work Management

Figure 28.1 & Figure 28.2 are the components of work management. Figure 28.1 shows an overview of today's work records to check if there are any missing routines. Figure 28.2 displays the detailed version of the work management overview. Each of the items in the overview is spread into one single table row. If the routine items have been performed to multiple residents, the item is further spread into multiple single table row so that each residents routine is one table record. The detailed management overview is simply for data records and references.

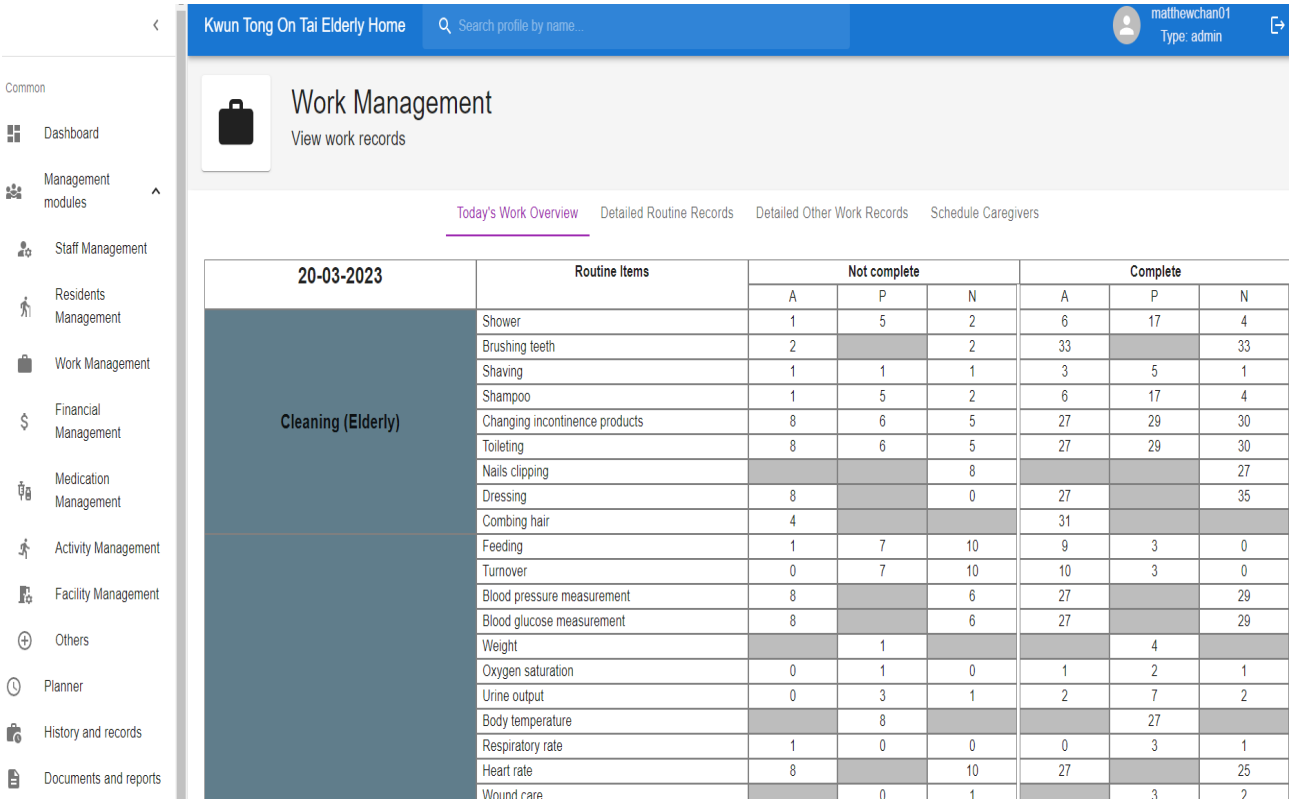


Figure 28.1 Work management overview

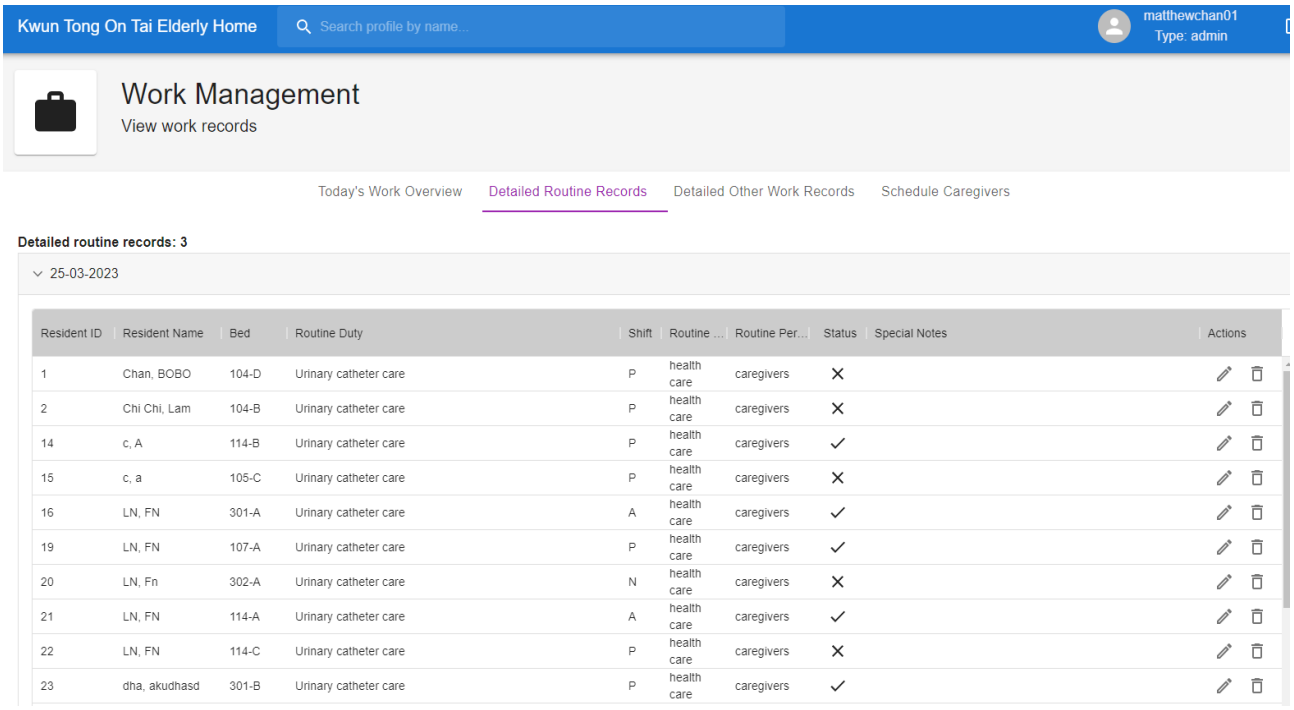


Figure 28.2 Work management detailed information

4.6 Optimization

4.6.1 Authentication

The purpose of authentication design is to ensure application security, which is one of the important elements in a management system. Users are required to have authorization tokens to perform different functions in the application. JSON web tokens (JWT) will be used to achieve such a design.

4.6.1.1 JSON web tokens (JWT)

JWT is a standard industry method that defines secure data transmission between clients and servers using a secret. It consists of 3 components, header, payload and signature. Figure 17.1 shows the structure of JWT. The header stores the information of signing algorithm, while the payload and signature are the encoded

JSON payload data and the encoded concatenation of header and payload respectively. Figure 17.2 displays a detailed methodology for JWT creation.

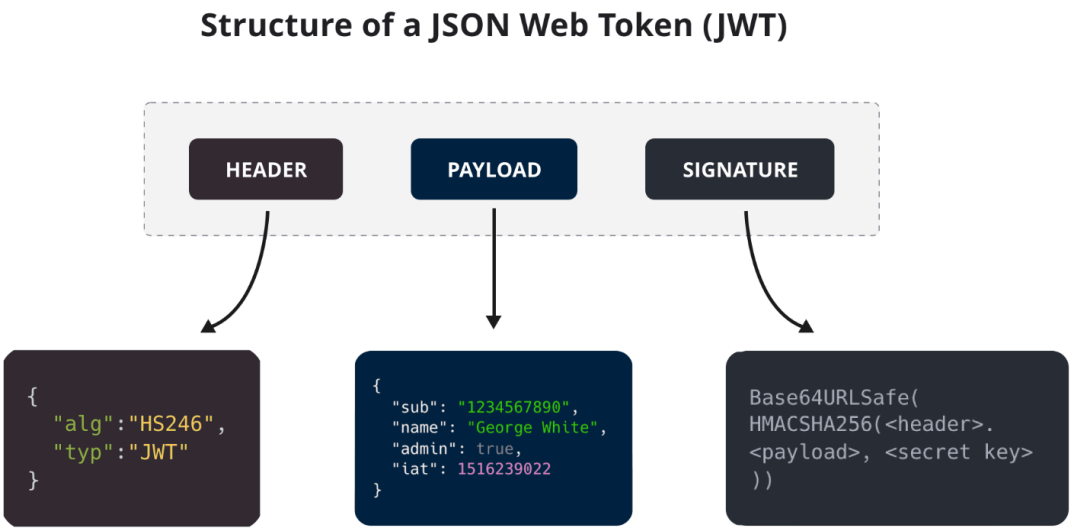


Figure 26.1 JSON Web Token structure

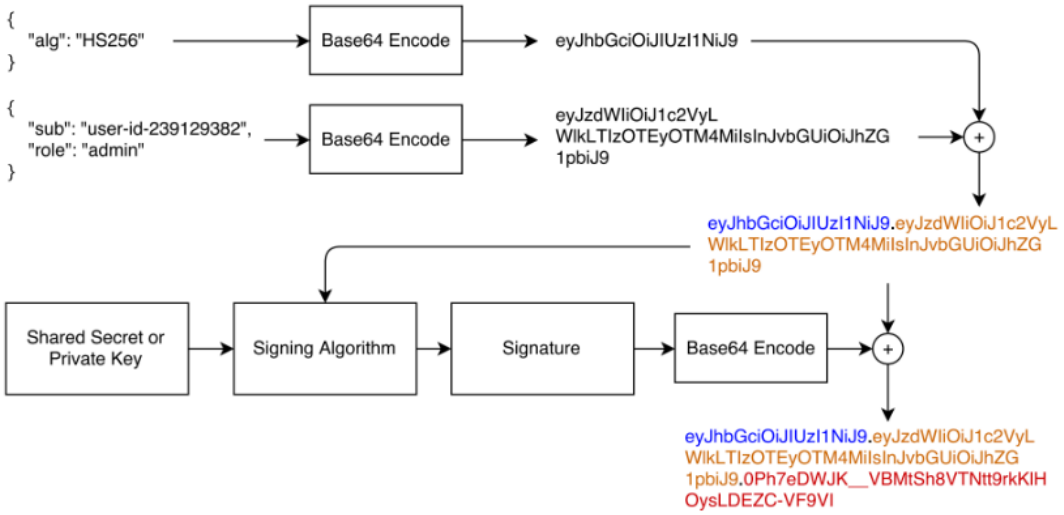


Figure 26.2 JSON Web Token design

4.6.1.2 JWT implementation

JWT can be created in npm in the format of <header>.<payload>.<signature> using the jsonwebtoken package. Figure 17.3 shows the function of creating the token.

```
// create jwt
const createToken = (_id) => {
  return jwt.sign({ _id }, process.env.SECRET, { expiresIn: "1h" });
};
```

Figure 26.3 JSON Web Token creation

To achieve the purpose of protecting the react routes by authorization, it requires the verification of JWT when performing actions. Figure displays the logic of authentication in the system. The requireAuth function requires the verification of JWT so that further actions can then proceed. It also attaches the user object to any requests so that requests can be easily determined according to the user id.

```
const jwt = require("jsonwebtoken")
const User = require('../models/userModel')

const requireAuth = async(req, res, next) => {

  // verify authentication
  const { authorization } = req.headers

  if(!authorization){
    return res.status(401).json({error: 'Authorization token required'})
  }

  const token = authorization.split(' ')[1]
  //console.log(token)
  try {
    // when login, it will generate the token -> then find the user with this authorization token
    const {_id} = jwt.verify(token, process.env.SECRET)
    // attach user object in requests
    req.user = await User.findOne({ _id }).select('_id')
    // allow move on to next function
    next()

  } catch (error) {
    console.log(error)
    res.status(401).json({error: 'Request is not authorized'})
  }

};

module.exports = requireAuth
```

Figure 26.4 JSON Web Token implementation

4.6.1.3 Testing - Postman

The testing of the authentication algorithm is performed by Postman. If the user does not log in or is not authorized, performing actions inside the system is not

allowed with the response error message of ‘Authorization token required’ (Figure 18.1) and ‘Request is not authorized’ (Figure 18.2) respectively.

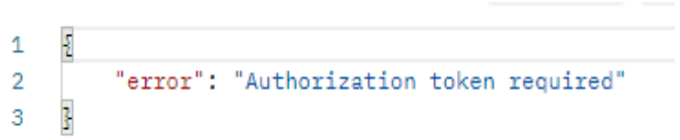


Figure 27.1 Authentication error

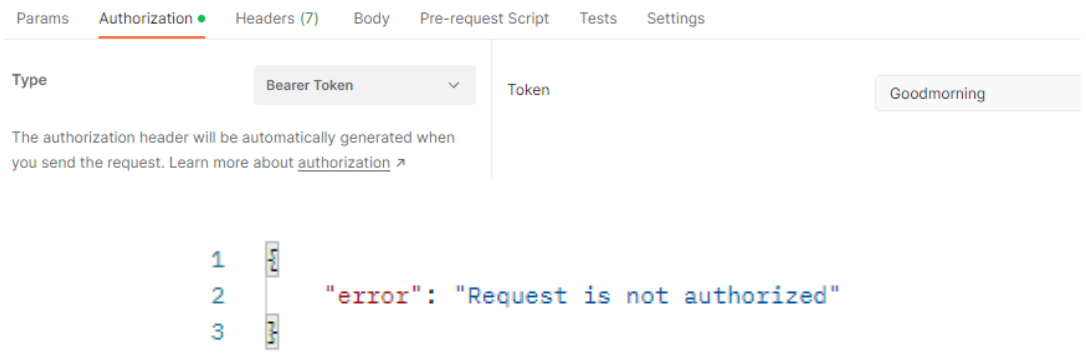


Figure 27.2 Authentication error

If the user has logged in to the system, performing actions are allowed as the token is generated whenever the user log in. The token verification will take place when performing further actions in the system. Figure 18.3 & Figure 18.4 shows the JSON response of fetching staff information with the attachment of the authorization token after login to the system.

Figure 27.3 Success login

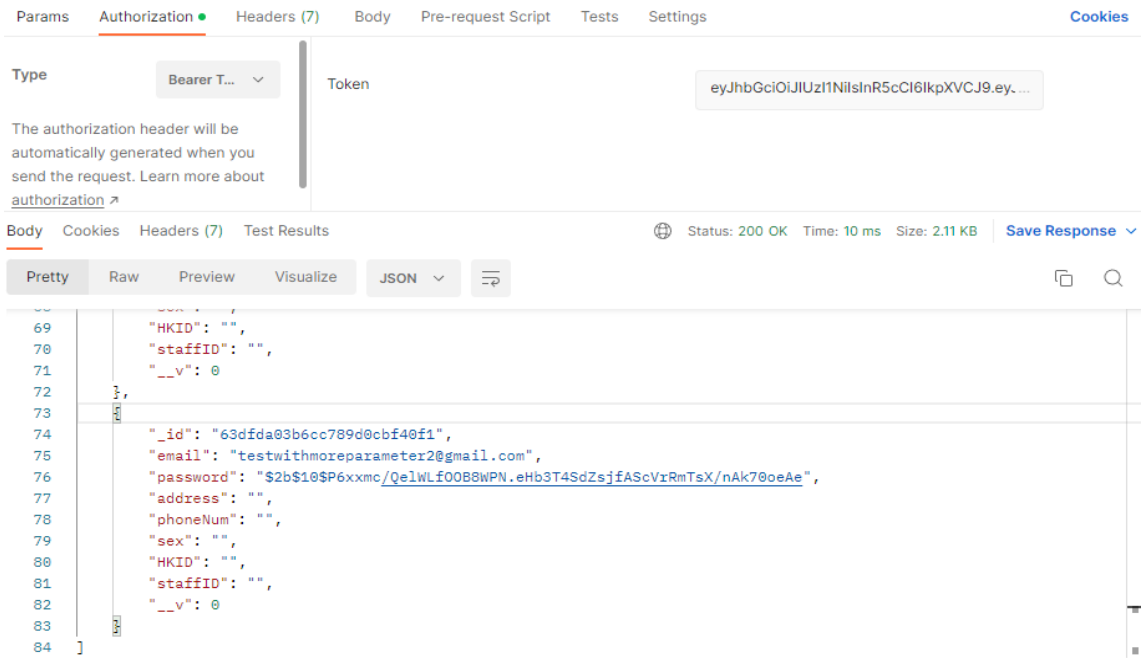


Figure 27.4 Success authentication

4.6.2 Searching optimization algorithm

The searching algorithm is designed for the purpose of minimizing requests and achieving a better query result. In general, if a input textbox is needed to handle the input by users, whenever the search query inside the textbox changes, it will send requests to the endpoint and responses is made to update the state of the results. However, this may not be satisfactory as endpoints may block the requests if

multiple requests are continuously sent. Besides, the intermediate results of the query are not important. Figure 19.1 shows the searching of names that contains the string ‘lam’, when typing ‘lam’, three requests are made with the respective strings of ‘l’, ‘la’, ‘lam’. However, the results of name that includes ‘l’ and ‘la’ such as ‘HELLO WORLD’ and ‘LOREM’ are not our focus.

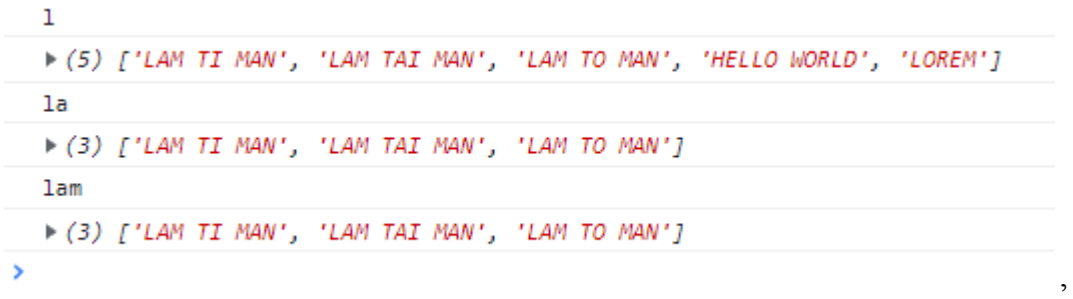


Figure 28.1 Problems in general searching

The effects of the above example may seem insignificant. However, when the dataset is larger and the application is more complex, the searching optimization can help React application minimize the rendering of components.

To tackle the problem, the debouncing algorithm is used to handle the searching of queries.

4.6.2.1 Debouncing

```
24
25  const useDebounceValue = (value, time = 250) => {
26    const [debounceValue, setDebounceValue] = useState(value);
27    useEffect(() => {
28      const timeout = setTimeout(() => {
29        setDebounceValue(value);
30      }, time);
31
32      return () => {
33        clearTimeout(timeout);
34      };
35    }, [value, time]);
36
37    return debounceValue;
38  };
39
40
41
42  const Searchbar = () => {
43    const [query, setQuery] = useState("");
44    const [suggestion, setSuggestion] = useState([]);    'suggestion' is assigned
45    const debounceQuery = useDebounceValue(query);
46
47    useEffect(() => {
48      setSuggestion([])
49      if(debounceQuery.length> 0){
50          console.log(debounceQuery)
51          const data = getAutoComplete(debounceQuery);
52          console.log(data)
53          setSuggestion(data);
54
55      }
56      //anytime query changes, useEffect will run
57    }, [debounceQuery]);
58
```

Figure 28.2 Debouncing

Debouncing is used to determine the finalized query that users input by setting the timeout to each string the user input. It records the time of input whenever the query has changed. If the user has finished typing and the time has passed 250ms, it will return a finalized value for further function.

4.6.2.2 Performance

Figure shows the results of the improved query searching algorithm. The function no longer takes ‘l’ or ‘la’ but ‘lam’ as input, and the response data also matches the expectations.

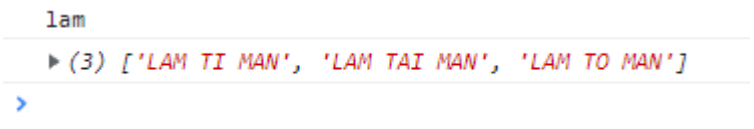
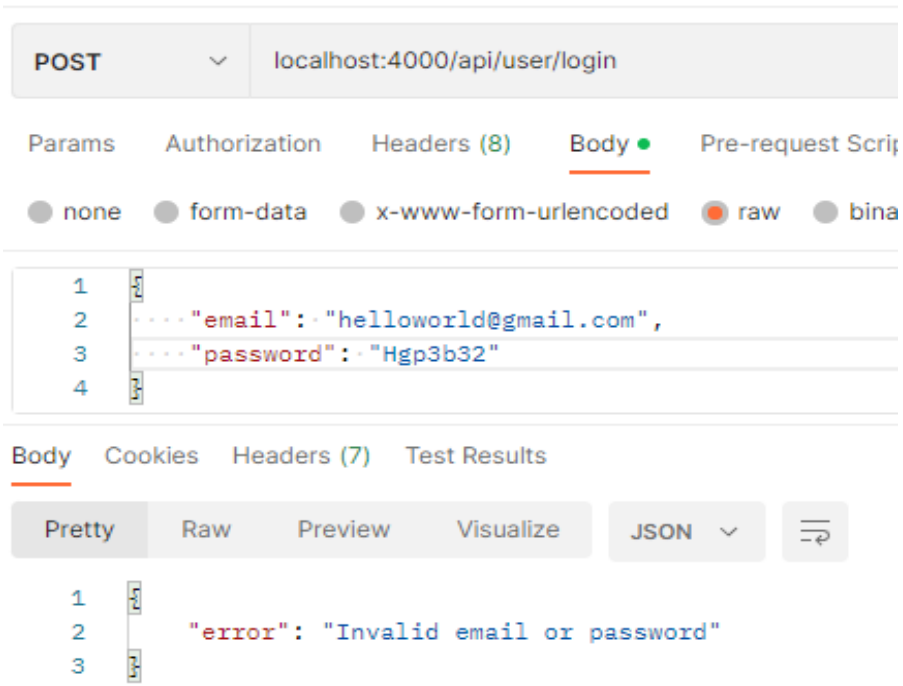


Figure 28.3 Improved searching query

5. Result and evaluation

5.1 Test cases

Test ID	Test Name	Step	Expected Result
1	User login with incorrect data	Step1: input following data account: "helloworld@gmail.com" password: "Hgp3b32	It will receive an "Invalid email or password" error



Test ID	Test Name	Step	Expected Result

2	User signup	Step1: input following data
		{
		"account": "",
		"email": "matthew.ctn56@gmail.com",
		"password": "Hgps3b32",
		"userType": "Admin"
		}
			It will receive an "All fields must be filled" error

```
{
    "error": "All fields must be filled"
}
```

5.2 Test plan

Administrative colleagues

Test ID	Test Name	Status	Test Date
1	User login with incorrect data	Pass	14/2/2022
2	User signup	Pass	14/2/2022
3	Add new staff	Pass	20/2/2022
4	Export staff table to csv	Pass	20/2/2022
5	Add new residents	Pass	20/2/2022

6	Residents changes in resident overview	Pass	20/2/2022
7	Set default routine items to elderly	Pass	20/2/2022
8	Generate detailed routine work record	Pass	3/3/2023
9	Update services cost	Pass	3/3/2023
10	Change 2/F Activity Room 1 to inactive	Pass	3/3/2023

Caregivers

Test ID	Test Name	Status	Test Date
1	User login with incorrect data	Pass	14/2/2022
2	User signup	Pass	14/2/2022
3	Add routine items	Pass	20/2/2022
4	Add medication items	Pass	22/2/2022
5	100% complete in today’s routine	Pass	26/2/2022

Relatives

Test ID	Test Name	Status	Test Date
1	User login with incorrect data	Pass	14/2/2022

2	User signup	Pass	14/2/2022
3	Click on gallerys	Pass	26/2/2022
4	View notes	Pass	26/2/2022

6. Conclusion

To conclude, it can be seen that the system can be implemented in a web-based platform as an elderly home management system. Thus, it is projected that the project has been completed with promising results in the test phases.

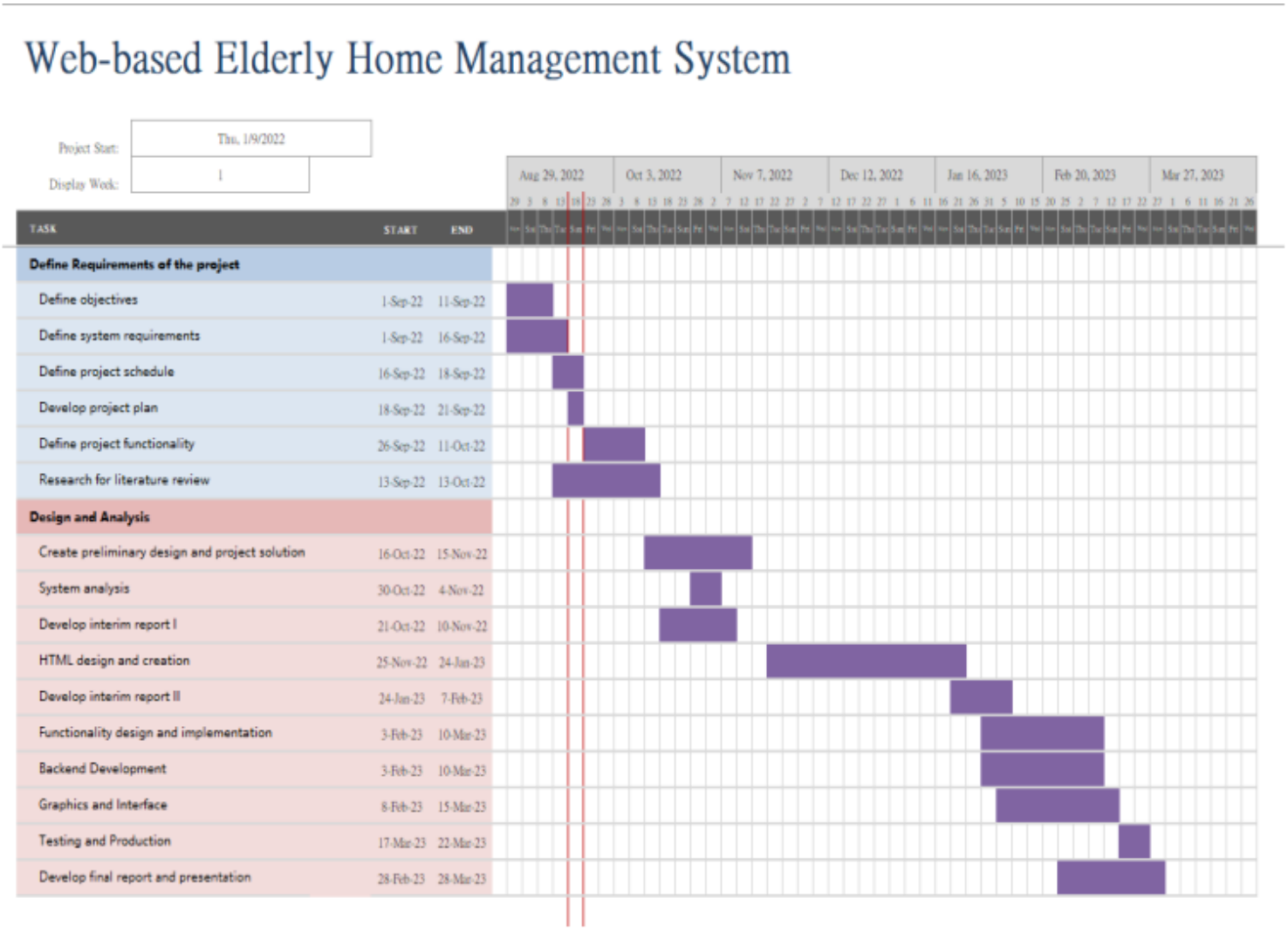
Firstly, the system requirement and factors for the project were collected in Chapter 1, and users' requirements and project items are concluded after the review in existing research studies.

Then, the system is designed with the MERN stack with a detailed description of the technology with its advantages in Chapter 3.

With the detailed implementation in Chapter 4, the system is beyond complete after the adoption of different proposed technologies along with its algorithms.

Finally, the promising result in test phases indicates the success and the practical use of this project.

7. Project Schedule



References

Abejirinde, I. O., English, M., Muinga, N., Paton, C., & Zweekhorst, M. (2020). Designing paper‐based records to improve the quality of nursing documentation in hospitals: A scoping review. *Journal of Clinical Nursing*, 30, 1–2, 56–71.

Acharya, P., Bail, K., Blackburn, J., Gibson, D., Kaak, V., Kozlovskaia, M., Redley, B., & Turner, M. (2022). Using health information technology in residential aged care homes: An integrative review to identify service and quality outcomes. *International Journal of Medical Informatics*, 165, 104824.

Adam, J. (2022). *The React Framework (which is actually a JS library) – A Brief Overview From Use Cases To Features, Components, Strengths & Limitations*. Retrieved November 15, 2022 from https://kruschecompany.com/react-framework-library/

Agency for Healthcare Research and Quality. (2013). *Health Information Technology Integration*. Retrieved October 15, 2022, from https://www.ahrq.gov/ncepcr/tools/health-it/index.html

Aghayi, E., Deen, M., Majumder, S., Memarzadeh-Tehran, H., Mondal, T., Noferesti, M., & Pang, Z. (2017). Smart Homes for Elderly Healthcare—Recent Advances and Research Challenges. *Sensors*, 17, 11, 2496.

Ahmadi, M., Davaridolatabadi, N., Sadoughi, F., & Shahi, M. (2016). Health Information Management System for Elderly Health Sector: A Qualitative Study in Iran. *Iranian Red Crescent Medical Journal*, 18, 2, 1.

Alwan, M., Manard, B. B., Resnick, H. E., & Stone, R. I. (2009). Use of Electronic Information Systems in Nursing Homes: United States, 2004. *Journal of the American Medical Informatics Association*, 16, 2, 179–186.

American Psychological Association. (2011). *The Minimum Data Set*. Retrieved October 20, 2022, from https://www.apa.org/pi/about/publications/caregivers/practice-settings/assessment/tools/minimum-data

Beachley, M., Gugerty, B., Hawk, W., Karp, J., Koszalka, M., Maranda, M. J., Morrison, S., Navarro, M. B., Newbold, S., Poe, S. S., & Wilhelm, D. (2007). *Challenges and Opportunities in Documentation of the Nursing Care of Patients*. Retrieved October 10, 2022, from https://mbon.maryland.gov/Documents/documentation_challenges.pdf

Bourgeois, D., & Bourgeois, D. T. (2014). *Information Systems for Business and Beyond*. Retrieved October 17, 2022, from https://pressbooks.pub/bus206/chapter/chapter-1/#footnote-25-1

Cahyadi, A. T , & Pratama, M. A. T. (2020). Effect of User Interface and User Experience on Application Sales. *IOP Conference Series: Materials Science and Engineering*, 879, 1. https://doi.org/10.1088/1757-899x/879/1/012133

Castle, N. G., & Liu, D. (2008). Health Information Technology in Nursing Homes. *Journal of Applied Gerontology*, 28, 1, 38–58.

Census and Statistics Department. (2017). *Hong Kong Population Projections 2017-2066*. Retrieved October 5, 2022, from

https://www.statistics.gov.hk/pub/B1120015072017XXXXB0100.pdf

Centers for Medicine & Medicaid Services. (2021). *Minimum Data Set 3.0 Public Reports.* Retrieved October 20, 2022, from https://www.cms.gov/Research-Statistics-Data-and-Systems/Computer-Data-and-Systems/Minimum-Data-Set-3-0-Public-Reports

Chen, Z., Qi, H., & Wang, L. (2021). Study on the Types of Elderly Intelligent Health Management Technology and the Influencing Factors of Its Adoption. *Healthcare*, 9, 11, 1494.

Chew, B. H., Rokhani, F. Z., Sazlina, S. G., Su, J., & Zhao, Y. (2021). The expectations and acceptability of a smart nursing home model among Chinese elderly people: A mixed methods study protocol. *PLOS ONE*, 16, 8. https://doi.org/10.1371/journal.pone.0255865

Couchbase. (2022). *NoSQL Databases – What They Are and Why You Need One*. Retrieved November 10, 2022 from https://www.couchbase.com/resources/why-nosql

Damyanov, I., & Tsankov, N. (2019). On the Possibilities of Applying Dashboards in the Educational System. *TEM Journal*, 8, 2, 424-429. https://doi.org/10.18421/TEM82-15

Deed, I. (2023). *Pros and Cons of Web Development Frameworks*. Retrieved February 10, 2023, from

https://www.pangea.ai/dev-web-development-resources/best-practices/

Devi, S. A., Nitish, N., Rachapudi, V., Samaikya, S., & Sathvik, U. P. (2020). Performance Comparison of applications with and without Web Frameworks. *International Journal of Advanced Trends in Computer Science and Engineering*, 9, 2, 1020–1028. https://doi.org/10.30534/ijatcse/2020/19922020

Goh, M. H. (2019). *Understanding Your Organisation: Nursing Homes.* Retrieved November 11, 2022, from https://blog.bcm-institute.org/blog/understanding-your-organisation-nursing-home

Griffiths, R., Jefferies, D., & Johnson, M. (2010). A Meta-study of Essentials of Quality Nursing Documentation. *International Journal of Nursing Practice*, 16, 2, 112-124.

Hector, D. S. (2010). *A Retrospective Analysis of Nursing Documentation in the Intensive Care Units of an Academic Hospital in the Western Cape*. Retrieved October 9, 2022, from https://core.ac.uk/download/pdf/37323671.pdf

Jen, M. Y., Kerndt, C. C., & Korvek, S. J. (2022). *Health Information Technology*. Retrieved October 15, 2022, from https://www.ncbi.nlm.nih.gov/books/NBK470186/

Kinnunen, U., & Saranto, K. (2009). Evaluating nursing documentation - research designs and methods: systematic review. *Journal of Advanced Nursing*, 65, 3, 464-476.

Mandal, A., Pal, S. C., & Saha, D. (2015). *User Interface Design Issues for Easy*

and Efficient Human Computer Interaction: An Explanatory Approach. Retrieved November 14, 2022, from https://www.ijcseonline.org/pdf_spl_paper_view.php?paper_id=18

MongoDB. (n.d.). *What Is The MERN Stack?*. Retrieved November 12, 2022, from https://www.mongodb.com/mern-stack

MongoDB. (n.d.). *NoSQL Vs SQL Databases*. Retrieved February 12, 2022, from https://www.mongodb.com/nosql-explained/nosql-vs-sql

MongoDB. (n.d.). *Key-Value Databases*. Retrieved February 12, 2022, from https://www.mongodb.com/databases/key-value-database

Pan American Health Organization (2001). Problems of Clinical and Administrative Records. In H. F. Marin, R. J. Rodrigues, C. Delaney, G. H. Nlelsen & J. Yan (Eds.), *Building Standard-based Nursing Information Systems* (pp. 12). Washington, D.C: Pan American Health Organization

Sharma, A. (2022). *What Is Express JS In Node JS?*. Retrieved November 10, 2022 from https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js

Shrestha, P. (2016). *Challenges and Impact of Transforming Paper-Based Nursing Documentation into Electronic Form: A Study in Nepal*. Retrieved October 9, 2022, from https://munin.uit.no/bitstream/handle/10037/12787/thesis.pdf?sequence=1

Smiley, R. (2014). *Adoption of Health Information Technology in Nursing Homes*.

Retrieved October 6, 2022, from https://dc.uthsc.edu/cgi/viewcontent.cgi?article=1041&context=hiimappliedresearch

TechTarget. (2021). *3-tier application architecture*. Retrieved November 14, 2022 from https://www.techtarget.com/searchsoftwarequality/definition/3-tier-application

The Office of the National Coordinator for Health Information Technology. (n.d.). *Health IT: Advancing America's Health Care*. Retrieved October 17, 2022, from https://www.healthit.gov/sites/default/files/pdf/health-information-technology-fact-sheet.pdf

United Nations, Department of Economic and Social Affairs, Population Division. (2002). *World population ageing 1950-2050*. New York: United Nations

U.S. Department of Health & Human Services. (2020). *Health Information Technology*. Retrieved October 15, 2022, from https://www.hhs.gov/hipaa/for-professionals/special-topics/health-information-technology/index.html

Yu, P. (2006). ELECTRONIC VERSUS PAPER-BASED NURSING DOCUMENTATION SYSTEMS: THE CAREGIVERS WEIGH IN. *Journal of the American Geriatrics Society*, 54, 10, 1625–1626.

Figure 1: Goh, M. H. (2019). *Organisation Chart of a Nursing Home* [Figure]. Understanding Your Organisation: Nursing Homes. Retrieved November 11, 2022, from https://blog.bcm-institute.org/blog/understanding-your-organisation-nursing-home

Figure 2: Castle, N. G., & Liu, D. (2008). *Conceptual Framework for Health Information Technology in Nursing Homes* [Figure]. Health Information Technology in Nursing Homes. *Journal of Applied Gerontology*, 28, 1, 38–58. http://doi.org/10.1177/0733464808321887

Figure 3.1: Caretech System Limited. (2022). *vDoCare Management System* introduction [Video]. YouTube. https://www.youtube.com/watch?v=kQX76ThtUL0

Figure 3.2: Caretech System Limited. (2022). *vDoCare Management System* [Figure]. vDoCare Management System. Retrieved October 30, 2022 from https://www.caretech.com.hk/

Figure 3.3: Glazier. (n.d.). *HMS Management System Interface* [Figure]. HMS Management System. Retrieved October 24, 2022 from http://sscms.glacierbiz.com/glacier/channels/37.html

Figure 3.4: Glazier. (n.d.). *HMS Management System Mobile Application* [Figure]. HMS Management System. Retrieved October 24, 2022 from http://sscms.glacierbiz.com/glacier/channels/38.html

Figure 4: Deremuk, I. (2022). *Web Application Architecture* [Figure]. Modern Web

Application Architecture Explained: Components, Best Practices and More.

Retrieved November 11, 2022 from

https://litslink.com/blog/web-application-architecture

Figure 5: The Net Ninja. (2022). *MERN Stack Tutorial #1 - What is the MERN Stack?* [Video]. YouTube. https://www.youtube.com/watch?v=98BzS5Oz5E4

Figure 14.1 & Figure 14.2

https://medium.com/bb-tutorials-and-thoughts/why-state-management-is-important-for-react-apps-e72a36d81edd

Figure 17.1 JSON Web Token structure
https://supertokens.com/blog/what-is-jwt

Figure 17.2 JSON Web Token design
https://5xruby.tw/posts/what-is-jwt

Appendix

Monthly Log

Month	October 2022
Content	According to the project schedule, the tasks (Sep-Oct) are as follow:

	● define objectives ● define system requirements ● define project schedule ● develop project plan ● define project functionality ● research for literature review, are completed. For tasks in Nov: ● create preliminary design and project solutions ● system analysis ● develop interim report I These tasks are still on progress, and will be submitted with details in Interim report I with the above tasks. Extra on progress tasks: -> React.js study -> MERN stack application study through Youtube

Month	November 2022
Content	In this month, not much progress has been made due to different coursework and the coming exams. ● define project functionality ● define system requirements ● HTML design and creation

Month	December 2022
Content	According to the project schedule, the proposed tasks in

	December are: ● HTML design and creation (the task is still in progress, the system login authentication is designed) ● creation of class diagram (some preliminary design) In January, I will keep focusing on the design of the structure of the web-based application (starting from the interface of administrative colleagues) and setup of the database (database design, connection of the application and the database using Express and MongoDB)

Month	January 2022
Content	According to the Gantt chart schedule, the proposed task in January includes: ● develop interim report II (the task is still in progress, at the moment, I have modified some content in the literature review, detailed methodology and preliminary system implementation are described in the report. Besides, different UML diagrams such as class diagrams, sequence diagrams and use case diagrams (including some use case specifications) are completed.) ● Tasks scheduled in February such as functionality design and implementation, and backend development are also in progress. System functionalities are properly designed with complete UI design, but not yet the implementation. Some progress on backend development has also been made, such as building some database schema and environment setup for database and

	middleware.

Month	February 2022
Content	According to the Gantt chart schedule, the proposed task in February includes: Functionality design and implementation: Most of the functionality of the administrator interfaces are well designed and implemented, the other interfaces are still in progress and it is expected to meet the deadline Backend development: Backend development are completed Graphics and interface: The development is still in progress along with the functionality design Testing and production: Preliminary testing (e.g. HTTP requests) are tested with Postman with positive results. unit testing is still on progress Develop final report and presentation: The development of final report and presentation will be postponed to mid-March when finishing all interfaces implementation.

Month	March 2022
Content	• Complete final report • Complete the development and test phases of the project In progress: • demo clip preparation • presentation preparation

Gantt chart for project schedule