Hotel Management & Automation

Chanadech Jaroenkunathum 61090003
Panpakorn Siripanich 61090020
Paphitchaya Prombutr 61090023
Patchayut Tinnawimutjit 61090049

Bachelor of Engineering in Software Engineering
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2021

Thesis – Academic Year 2021

Bachelor of Engineering in Software Engineering
Department of Computer Engineering, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Title: Hotel Management & Automation

Authors

1. Chanadech Jaroenkunathum Student ID: 61090003
2. Panpakorn Siripanich Student ID: 61090020
3. Paphitchaya Prombutr Student ID: 61090023
4. Patchayut Tinnawimutjit Student ID: 61090049

Approved for submission

.....

(Dr. Pipat Sookavatana)
Advisor

Date ...../...../.....

Acknowledgement

This software engineering project aims to create a system for hotel staff to manage their guests easily and automate their everyday workload somewhat, and the customer's ability to quickly check out or control devices within their room from a mobile app.

In our endeavour to realise the goal of the project, we learned more about technical knowledge such as IoT Protocols, Kotlin, and the MVVM software architecture pattern. The team also had a chance to work in a real working environment with direct feedback from both the staff and customers which proved to be valuable for us all.

We would like to express our gratitude and appreciation to the following for providing assistance and guidance with the project development:

Dr. Pipat Sookavatana, for helping out with the overall design, networking structures of the project, and the other guidelines to create the framework of the software.

To all relatives, friends, and professors who helped support us, either mentally, physically, or financially, we sincerely thank you.

Abstract

This software engineering project aims to create a system for hotel staff to manage their guests easily and automate their everyday workload somewhat, and the customer's ability to quickly check out or control devices within their room from a mobile app.

Our Hotel Management & Automation System consists of two major components; "Intelligent Device Control & Automation" and "Guest Management System". The Intelligent Device Control & Automation is based on the open-source Home Assistant platform, which acts as a controller interface and automation tool for IoT devices that hotel staff and customers use. On the other hand, our project also includes the "Guest Management System" to let the hotel staff manage their guest status or book a new reservation locally and give the guest the ability to check-in/check-out by themselves using the provided mobile application.

The Hotel Management & Automation system is mainly developed using Kotlin. On the server side, Home Assistant is being used as a base for our device control & automation system. The database is currently deployed on Firebase to enable our team to work together across long distances and to quickly experiment with our database tables with a planned migration to Python web framework Django for the 2nd semester to keep the system working without WAN if needed.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Motivation

Many hotels lack the budget to hire as many hotel staff as pre-Pandemic due to the slow recovery of the Thai tourism industry. Coupled with the need to attract new customers and the widespread adoption of smartphones nationwide, we planned to let the customer quickly check-in or check out the hotel and control devices with their smartphones through a single mobile app. That same application should also act as a means for hotel staff to manage their guests and automate tasks to reduce their daily workload all in one system.

1.2 Objectives

1. To develop a system which provides the user with the ability to easily manage and automate all IoT devices for hotels.
2. To develop a guest management system that works according to business requirements.
3. Allow hotel staff the ability to edit the device management & automation by themselves in a secure and convenient way.
4. To provide the user with the ability to use the system even without WAN access.

1.3 Scope of Work

The scope of this project for the second semester can be listed as follows:

- Finish the QR authenticator system for the guest side
- Adapt all of our HA-based repo to HACS standards for easy implementation.
- Make sure the HA functionalities are working properly with the management application side.
- Implements a functioning framework for the software architecture on the application-client side.
- Implements the features specified in both the application and management side.
- Run the client-side application through basic testing procedures.
- Optimize memory usage and power efficiency in the application.

1.4 Thesis Structure

This thesis consists of seven chapters which are arranged as follows:

- Chapter 1 Introduction - refers to the motivation, objectives, scope of work, and thesis structure of this thesis.

- Chapter 2 Related work – proposes the Literature survey, various software that are relevant to this project, and comparison.

- Chapter 3 Background knowledge - explains the knowledge and technology necessary for the reader to understand the thesis.

- Chapter 4 Requirement analysis/ System Architecture and Design – presents the requirement of the system, the use case diagram, and the relevant system architecture diagram.

- Chapter 5 Software Development - explains the concepts, tools, and techniques that are used in developing the project.

- Chapter 6 Results - refers to the results of software demonstration, which consists of the user interfaces of the software and real-world applications.

- Chapter 7 Conclusion and Future work - is the chapter that talks about the conclusion, future work and improvements to the project.

Chapter 2
Related Work

This chapter primarily presents the comparison and evaluation of existing software related to the workflow management systems, which are Tuya Hotel, Cloudbeds,

2.1 Tuya Hotel

Tuya Hotel is a cloud-based, integrated smart hotel management system that includes four core management platforms: brand business operation, hotel operation, service provider operation, guest applet, and core functional modules. Based on the Tuya AIoT platform, this allows for diverse smart device options within the ecosystem.

Tuya Hotel currently works on a monthly subscription basis, with its primary users and partners being from mainland China.[1]

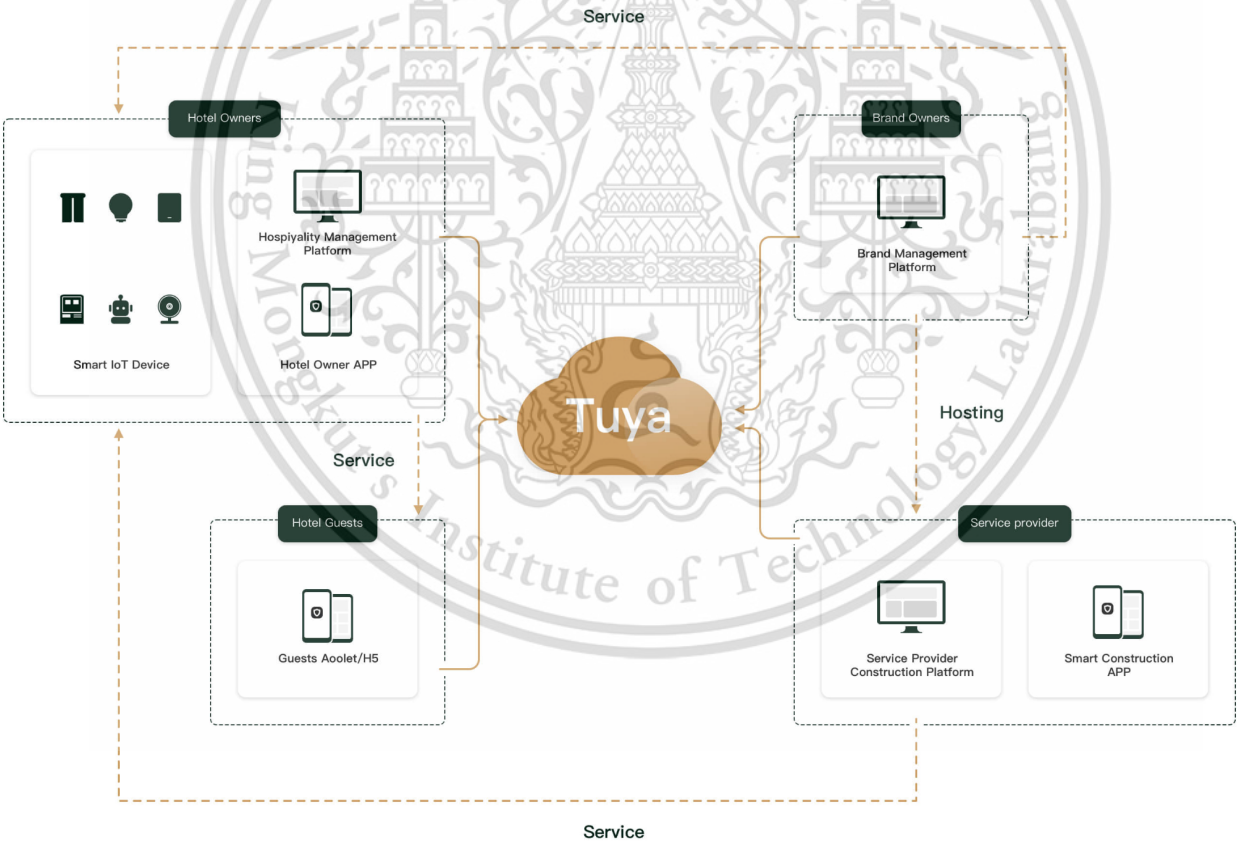

Figure 2.1.1 Tuya Hotel Ecosystem

2.2 Cloudbeds

Cloudbeds is a cloud-based, integrated hotel management software that features channel management, project management, and booking engine solutions in a single, centralised platform. Cloudbeds allow hotel managers to save on both time and expenses and streamline workflows with support for seamless integration of more than 500 third-party applications such as accounting and email marketing tools.

Cloudbeds currently works on a monthly subscription basis geared towards all hotel sizes.[2][3]

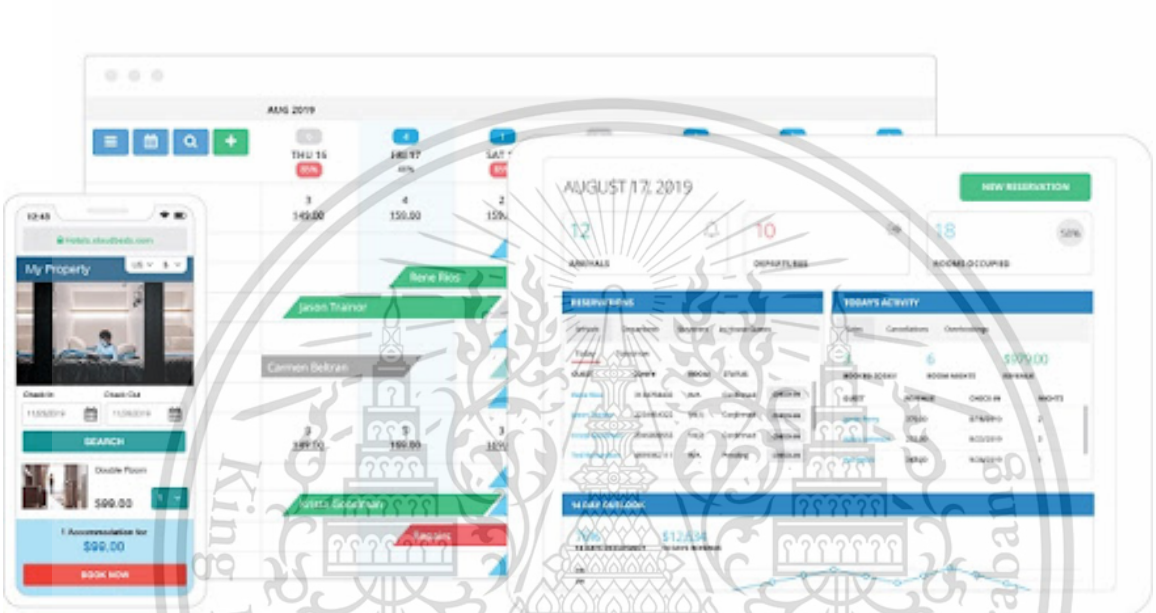

Figure 2.2.1 Cloudbeds

2.3. SabeeApp

SabeeApp is an all-in-one cloud-based hotel management software tailored for independent hotels and other small accommodations like vacation rentals and B&B's. It offers 4 main core modules: Property Management System, Channel Manager, Internet Booking engine and Payment Gateway.

SabeeApp currently works on a monthly subscription basis aimed toward small and medium hotel operators.[4][5]

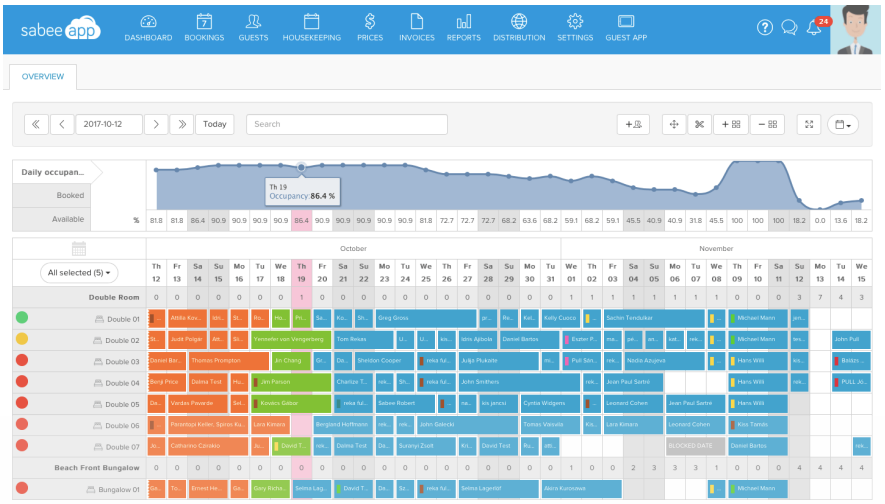

Figure 2.3.1 SabeeApp Hotel Management System

2.4. EasyFo

EasyFo is an offline hotel management system for desktops that supports various operations such as Front Office, Point of Sale, Housekeeping, Engineering, Customer Relation Management, Central Reservation System, Web Booking Engine, Interface System with our software and third party Accounting and Inventory Management system.

EasyFo currently works on a one-time payment basis, with its primary users being from Thailand and neighbouring CLMV countries covering over 800 properties in total.[6]

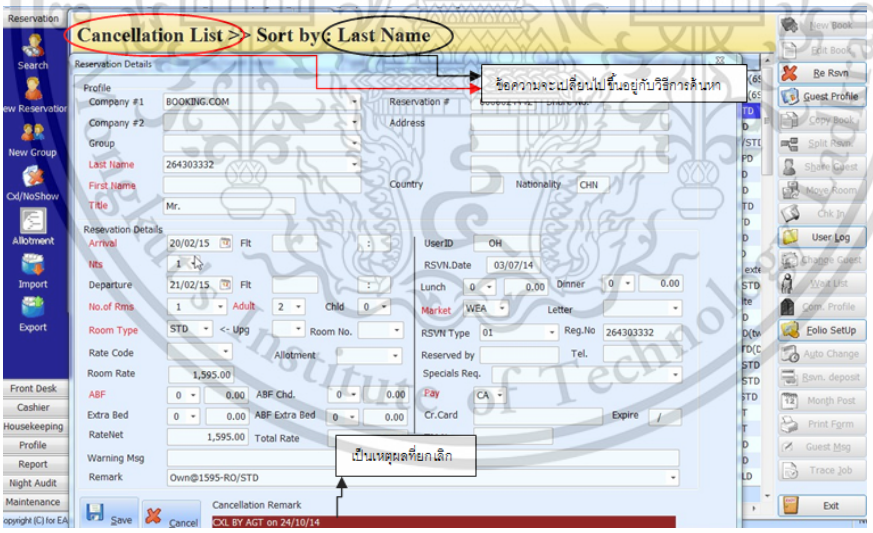

Figure 2.4.1 EasyFo Hotel Management System

Chapter 3
Background Knowledge

3.1 Home Assistant

Home Assistant is open-source software for home automation that is designed to be the central control system for the Internet of Things (IoT) devices, especially from different brands or manufacturers. Home Assistant allows us to control all our devices in one place, and to keep all devices on our and only our local network.[7]

3.2 Firebase Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronised in real time to every connected client. This allows for cross-platforming across Apple platforms, Android, and JavaScript SDKs with all clients sharing one Realtime Database instance and automatically receiving updates with the newest data.[8]

3.3 Django

Django is a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern. The framework emphasises reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models.

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.[9]

3.4 Kotlin

Kotlin is a static type, object-oriented programming (OOP) language that is interoperable with the Java virtual machine, Java libraries and Android.[10]. Kotlin saves time for developers as the less verbose language provides briefer and less redundant code. Kotlin is often regarded as a Java successor. It is interoperable with Java programs and libraries, while not being consistent with syntax. Kotlin also has its own libraries, which were developed with the help of the community in the form of an Android API. Kotlin emphasises clean, functional code and avoids "boilerplate" programming. The language has null safety, which prevents null pointer exceptions. Semicolons at the end of each line are not required, although Kotlin is unconcerned if a developer does so out of habit. Additional features lower the amount of code required to accomplish a goal's complexity and length.

3.5 Android SDK

An official development platform for android applications is provided by Google. The Android SDK enables you to develop Android applications. The Android SDK is a set of software development tools and libraries that are necessary to create Android apps. When Google publishes a new version of Android or an update, it also provides a related SDK that developers must download and install. It's worth mentioning that you may download and utilize the Android SDK without using Android Studio, but for most Android programming, you'll be using Android Studio.
The software package comes with AVD or the Android Virtual Device Manager. Also prompts the user to install the emulation layer acceleration driver for complying processors.[11]

3.6 Android Architecture Component using the MVVM pattern

Android Architecture Components are a part of Android Jetpack.

The Android Jetpack components are a collection of libraries that are individually adoptable and built to work together while taking advantage of Kotlin language features that make us more productive.

These software components have been arranged in 4 categories in which one of the categories is Architecture Components. Other categories are Foundation Components, Behaviour Components and UI Components.

Android architecture components are a collection of libraries that help us in the following, building the robust android application, building the testable android application, and building the maintainable android Apps. Architecture components help in managing our UI component lifecycle and handling data persistence.

Android Architecture Components are as follows:

- Lifecycles, It manages activity and fragment life cycles of our app, survives configuration changes, avoids memory leaks and easily loads data into our UI.
- LiveData, It notifies views of any database changes. Use LiveData to build data objects that notify views when the underlying database changes.
- ViewModel, It manages UI-related data in a lifecycle-conscious way. It stores UI- related data that isn't destroyed on app rotations.
- Room: It is a SQLite object mapping library. Use it to Avoid boilerplate code and easily convert SQLite table data to Java objects. Room provides compile time checks of SQLite statements and can return RxJava, Flowable and LiveData observables.
- Data Binding: It helps in declaratively binding UI elements in our layout to data sources of our app.[14]

3.7 Clean Architecture with MVVM pattern

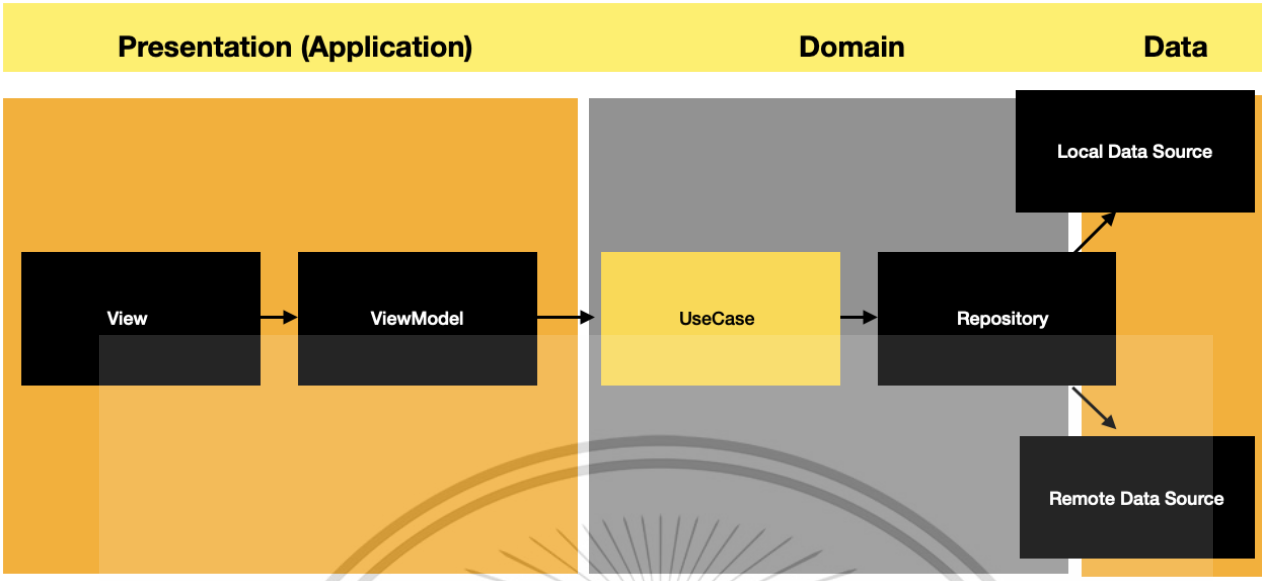

Figure 3.7.1 Clean architecture with MVVM Pattern

Separation of concerns is the primary purpose of clean architecture, which means that each layer is responsible for its own activities. We chose a clean architecture with the MVVM pattern since it allows us to be more readily testable and maintainable, as well as add new features more quickly.

The Layers of MVVM with Clean Architecture can be divided into three separate layers:

- Presentation Layer: This applies to our Activities, Fragments, and ViewModels, among other things. A task should be as simple as possible and never use Activities to store the business logic.
- Domain Layer: All of your application use cases are stored in the domain layer (the use cases have the purpose to act as a link between your ViewModels and Repositories).
- Data Layer: This Data Layer contains all of the repositories available to the domain layer. In general, this layer makes a data source API available to external classes.[12][13][14]

Chapter 4
Requirement Analysis

4.1.1 Functional Requirement

- Home Assistant & Server
    - A single desktop/RPi/NAS shall host both the Home Assistant and the Database System.
    - The required software shall auto-run automatically upon the server device being restarted.
    - Each installation shall be done onsite to guarantee usability should the internet access to outside is unavailable.
    - User passwords shall be generated dynamically per each booking, with their access revoked upon checking out, with the username being their own room number.
    - Admin shall have control for devices in all rooms, though the user themselves only have theirs.
- Internet of Things Hardware
    - Appliances in the guest room should be able to control through WiFi or Room Network and via smart switches.
- Application: Room's Control and Monitoring (Guest)
    - Users can login using the credentials provided by the hotel.
    - Users can toggle lighting on/off and adjust the level of brightness for each bulb.
    - Users can control the power outlet in the domain.
    - Users can control IR devices in the room.
    - Users can read the temperature and humidity of the environment.
    - Users can read the water level of the tank.
    - Users can control the water pump via relay.
    - Users can control the backup generator via relay.
    - Users can set up simple automation conditioning.
    - Users can lock/unlock the door of the room.
- Application: Guest Management System (Hotel's Affiliate Staffs)
    - Users can login to the system using their own staff ID.
    - Users can show the current room listings
    - Users can mark individual rooms with a status (e.g. Needs Cleaning).
    - Users can add or remove a reservation.
    - Users can search and edit an existing reservation.
    - Users can record payment information along with an image capture.
    - Users can check in a guest from a reservation or walk-in.
    - Users can check out guests.

4.1.2 Non-Functional Requirement

- Home Assistant & Server
    - Home Assistant ver. Core-2021.12.3 and above will be used.
    - A server with at least 4GBs of RAM and 128GB SSD is required for such operation.
    - SQLite3 DB shall be used here, deployed locally on a JSON server.
    - Admin users have 2FA enabled by default for security purposes.
- Internet of Things Hardware
    - Appliances in the guest room must support 2.4, 5GHz IEEE802.11b/g/n.
    - Those Appliances in the guest room that communicate through the room network must have security certification at WPA/WPA2.
    - IR Universal Remote Control must be able to send out signals in the range of 10m or more.
- Application: Room's Control and Monitoring (Guest)
    - Data from individual modules is fetched at a variable rate.
    - Data fetched is displayed as a collection on the dashboard.
    - Temperature can be displayed in both Fahrenheit or Celsius format.
    - Clock can be displayed in both 12-hours or 24-hours format.
    - Provide consistent design for user interactable components (e.g. Buttons).
    - Auto-generated password is used for encryption of the door lock.
- Application: Guest Management System (Hotel's Affiliate Staffs)
    - Data from individual entities is fetched at a variable rate.
    - Credentials are encrypted with hash and salt.
    - API calls should not lock out UI threads.
    - Simple UX flow for each process.
- Common Requirements
    - The UTF-8 Standard shall be used for localization.
    - Support Android devices from SDK version 21 onwards.
    - Solution implementation using Kotlin and Java programming language.
    - Utilises Android Jetpack libraries.

4.2 Use Case Diagrams

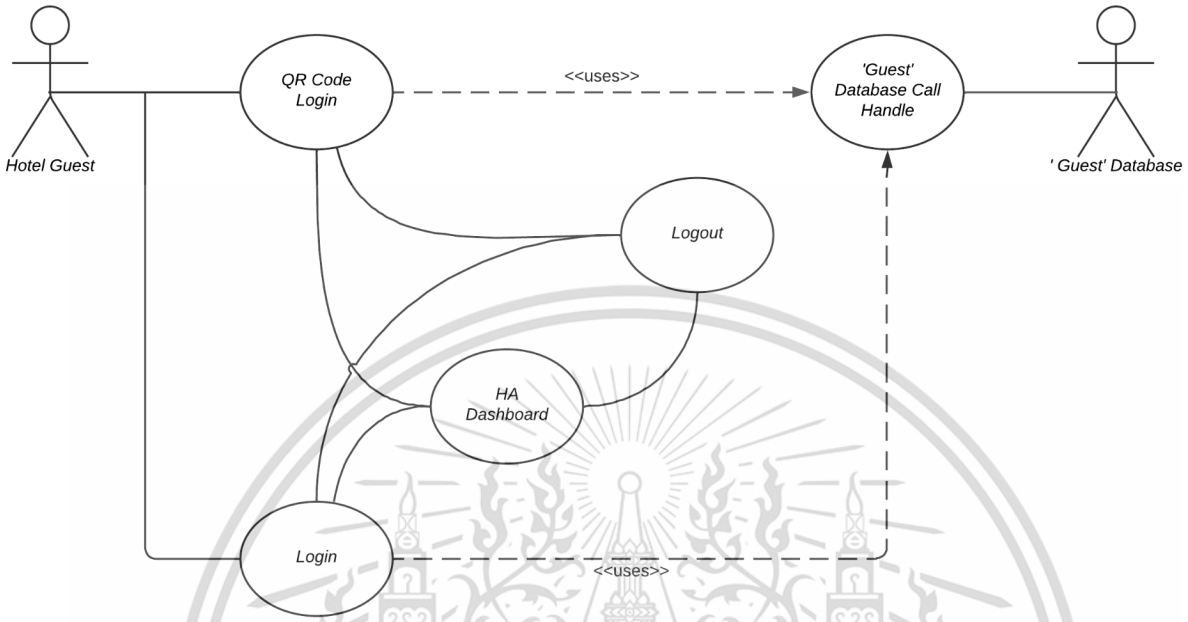

Figure 4.2.1 : Automation Client Use case diagram

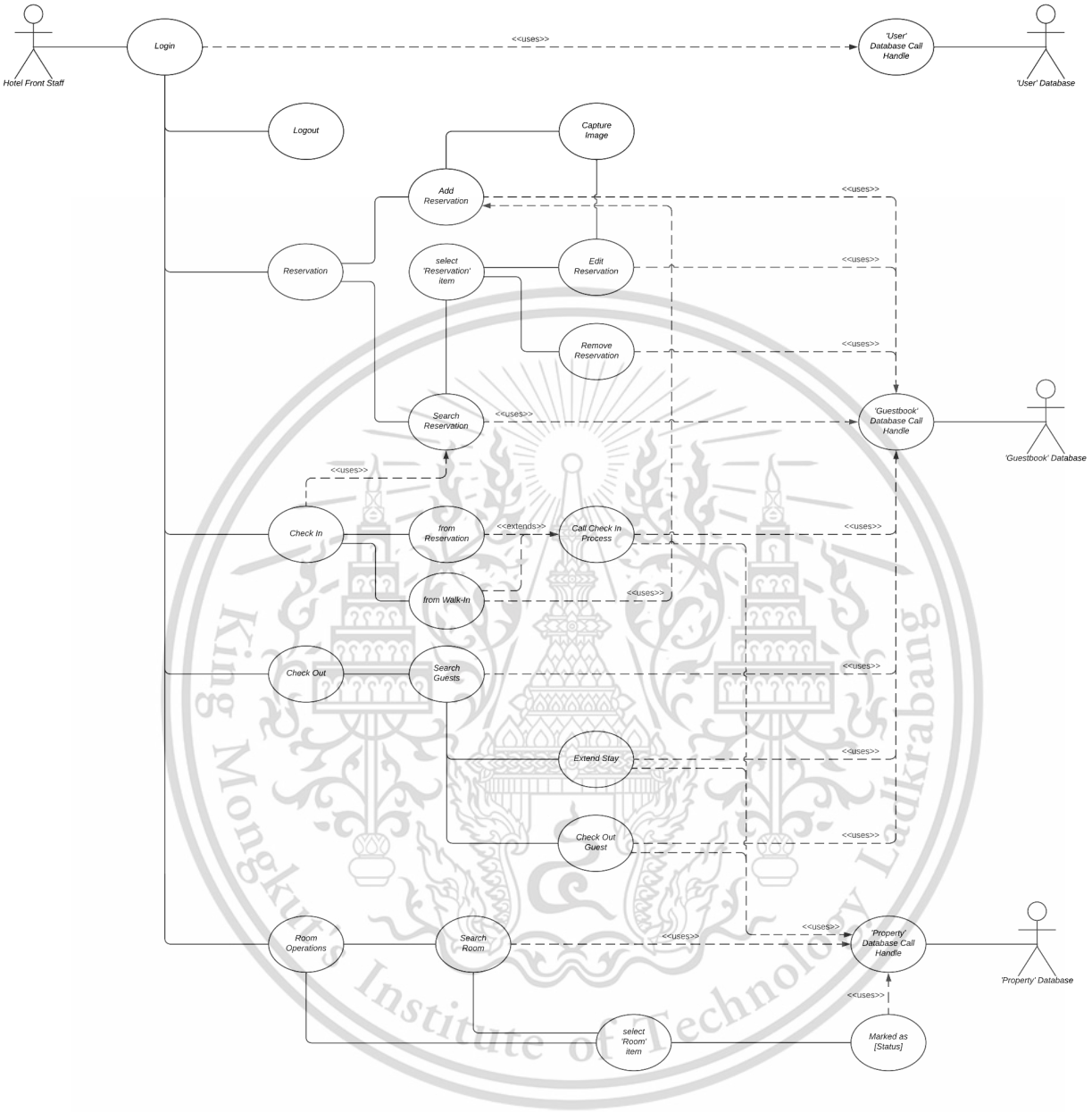

Figure 4.2.2 : Management Client Use case diagram

4.3 Class Diagrams

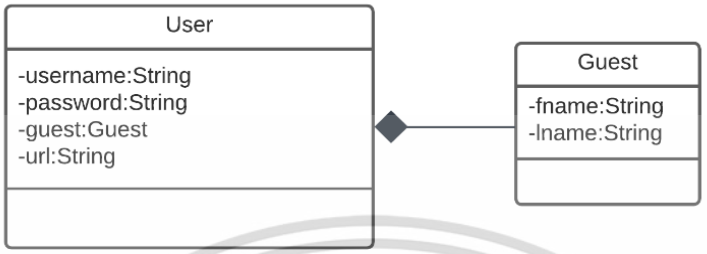

Figure 4.3.1 Automation Application Model Diagram

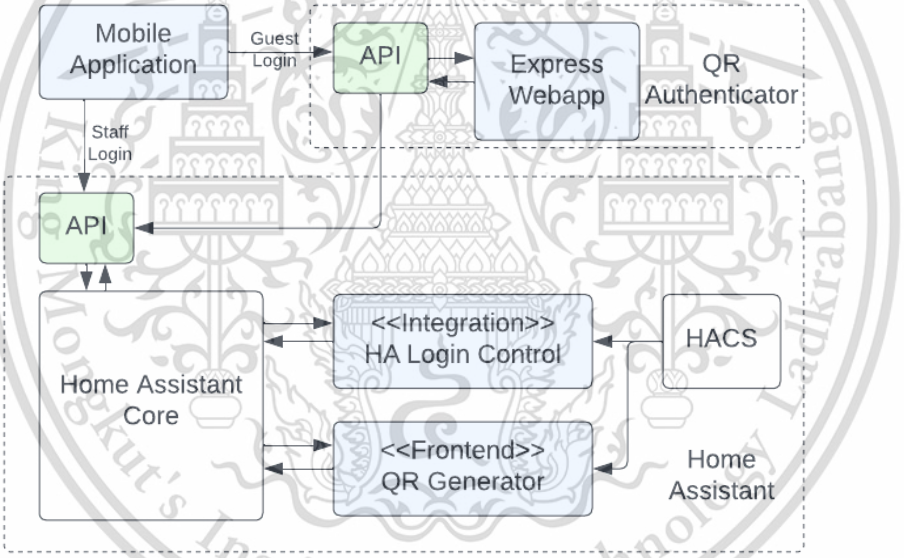

Figure 4.3.2 Automation Application System Architecture

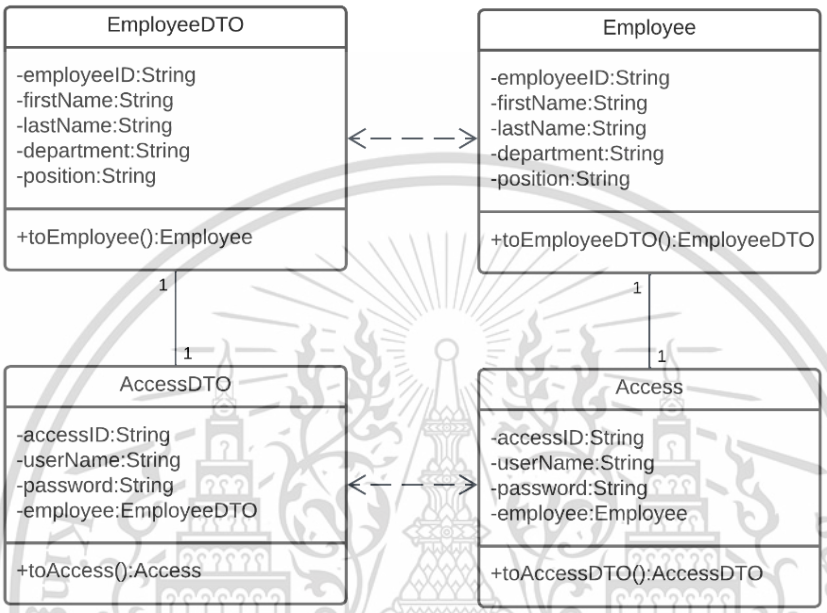

Figure 4.3.3 Management Application Model Diagram (Employee/Access)

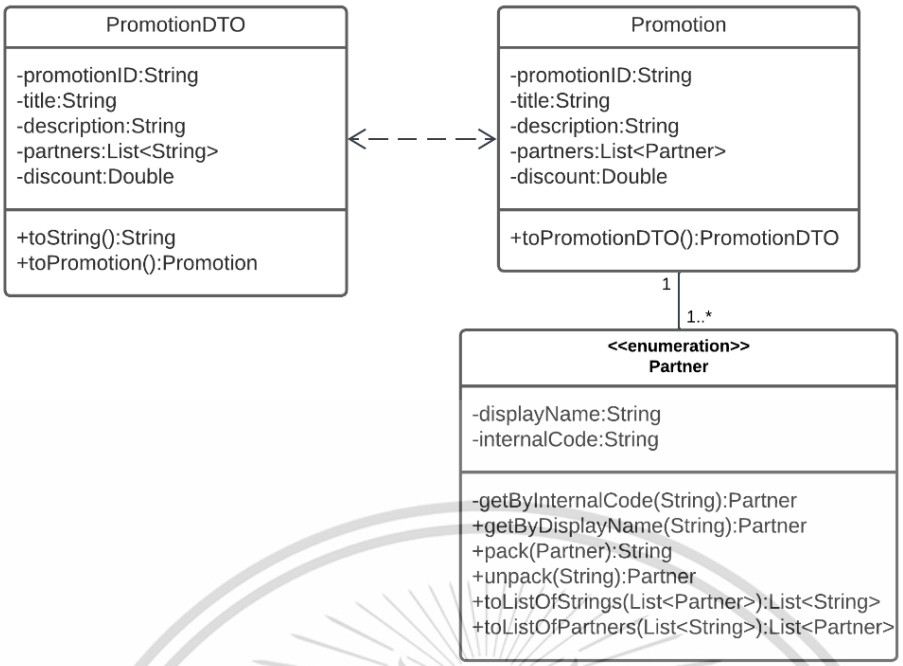

Figure 4.3.4 Management Application Model Diagram (Promotion)

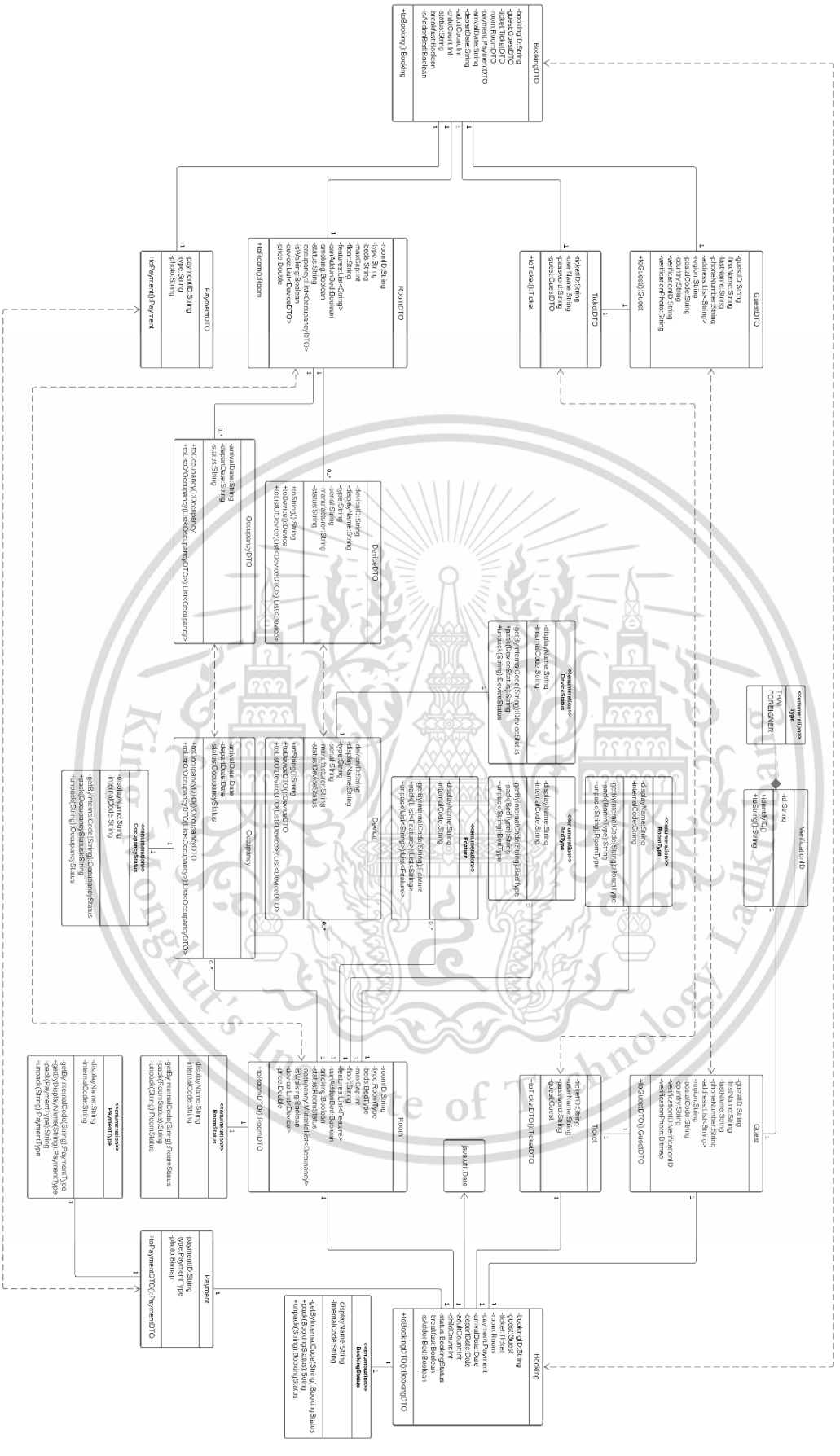

Figure 4.3.5 Management Application Model Diagram (Booking, Overview)

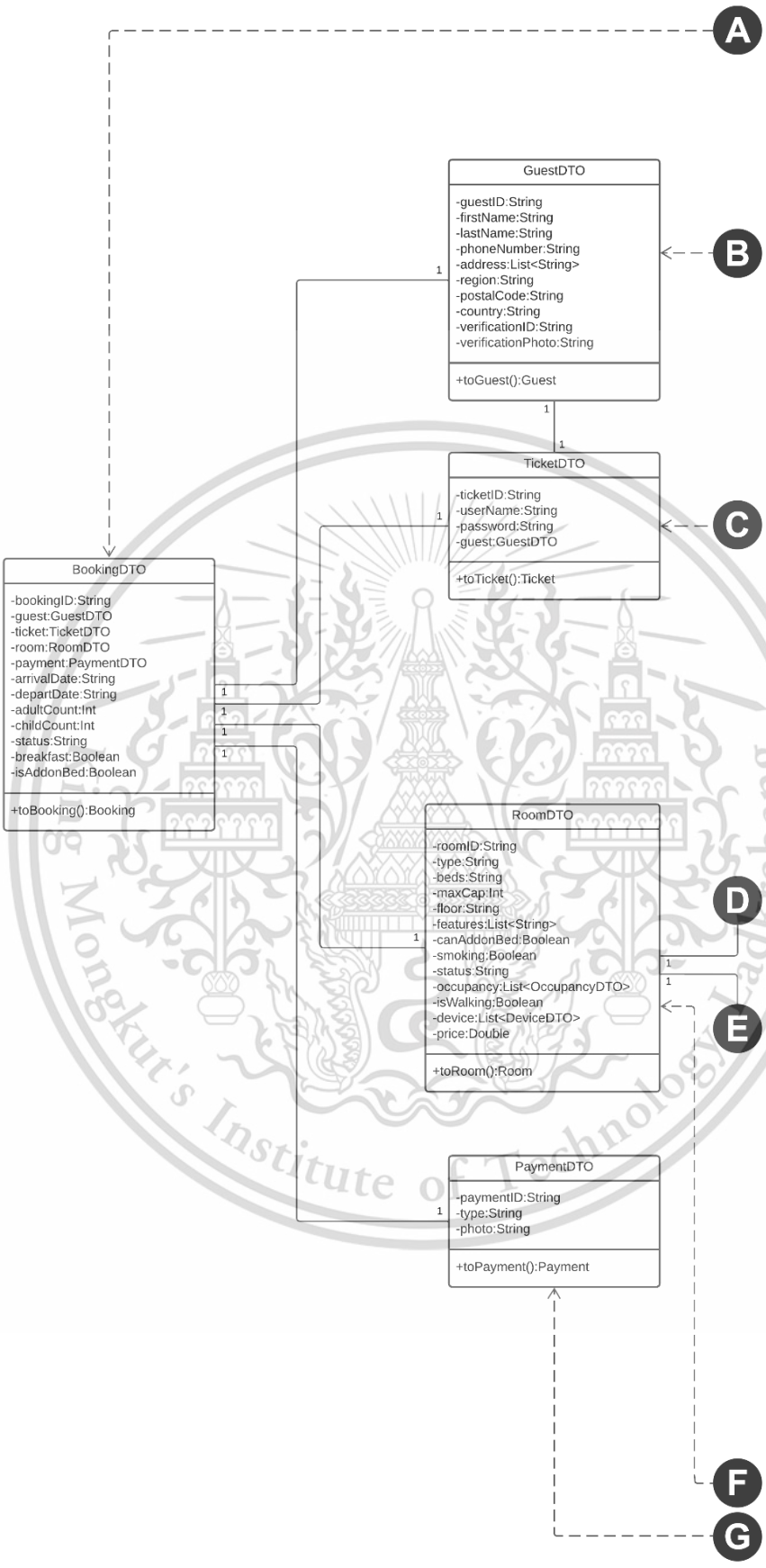

Figure 4.3.6 Management Application Model Diagram (Booking, cont.)

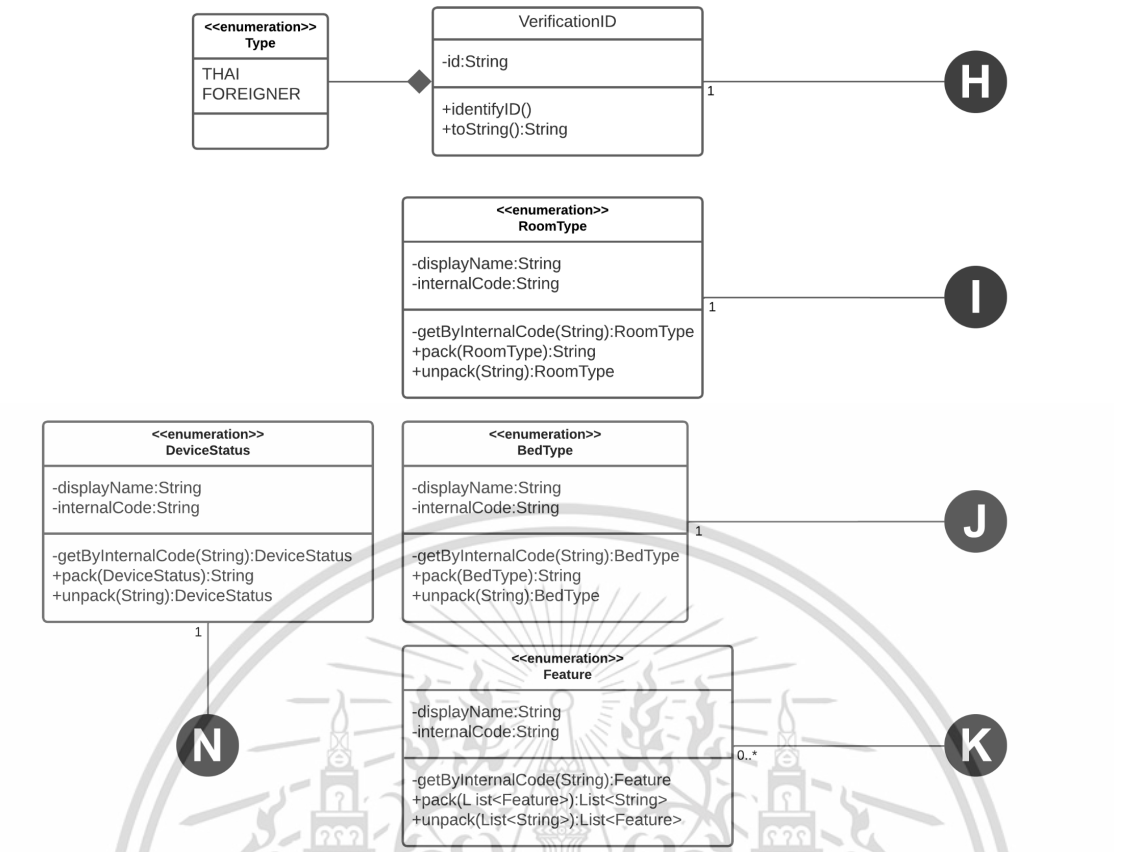

Figure 4.3.7 Management Application Model Diagram (Booking, cont.)

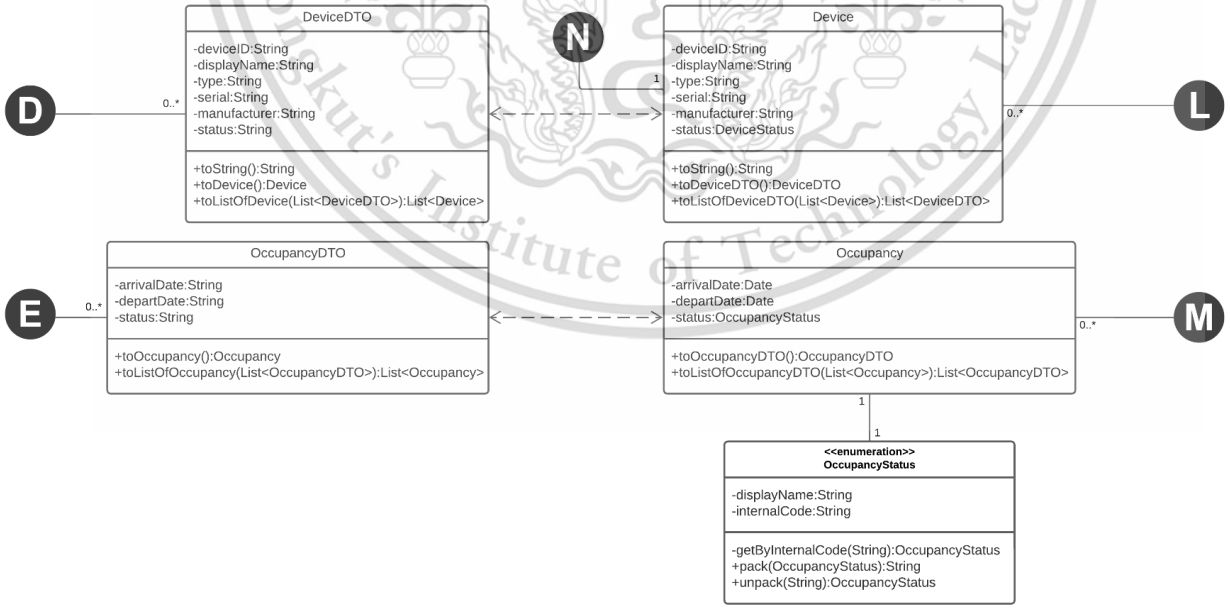

Figure 4.3.8 Management Application Model Diagram (Booking, cont.)

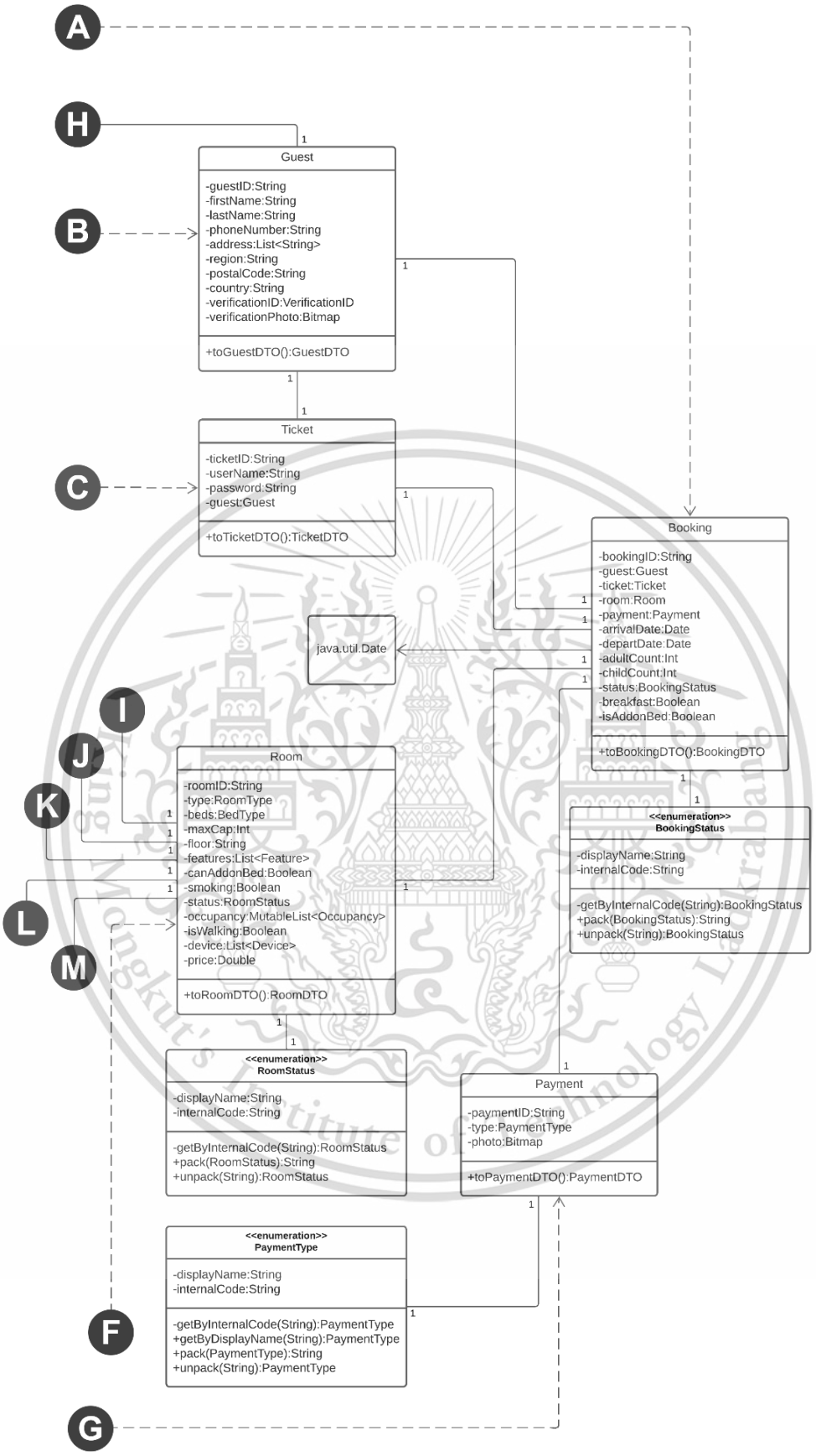

Figure 4.3.9 Management Application Model Diagram (Booking, cont.)

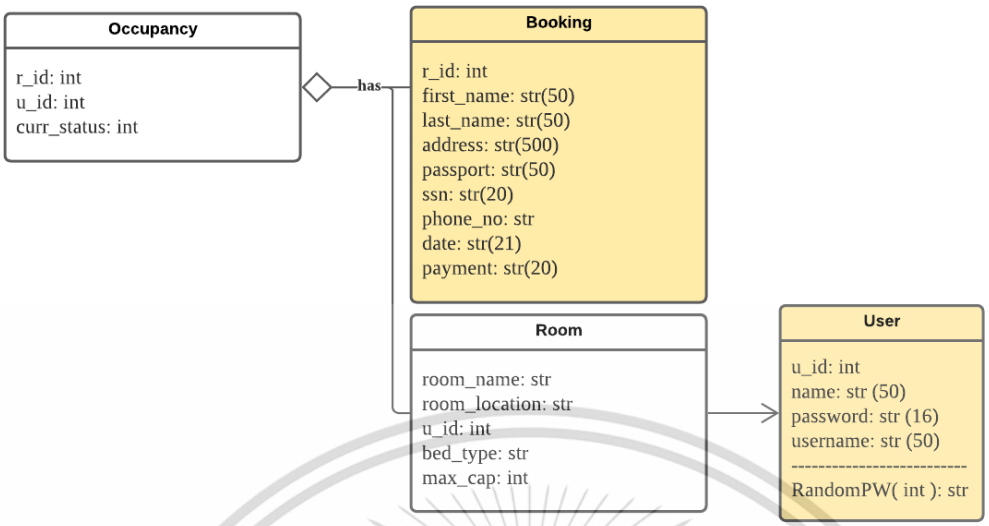

Figure 4.3.10 Server-side Model Diagrams

Chapter 5
Development

5.1 Client-Side Development

5.1.1 Development Tools

For the client side, the tools utilised to achieve the software product are as follows.

a. Android Studio IDE
Main development platform for android application recommended by the company itself. The tool provided the essential and useful features for debugging and deploying the application through the Android Virtual Device via the Android Debug Bridge. This allows compiling to test on multiple devices without the need to access the physical device. The platform is provided by IntelliJ.

b. External Libraries
Multiple libraries are included into the application to provide a more robust code. Kotlin Coroutine libraries are implemented for asynchronous tasks. Hilt Dagger is used to provide dependency injection. Android Jetpack is used for various essential Android components.

c. Android Virtual Device
Android Virtual Devices are virtual machines running on the emulation layer. It is a standalone product which can run independently from the Android Studio IDE. The virtual machine provides the emulation for both x86 and Arm image. This allows the host machine to perform better on multiple applications and provides a better outlook for deployment on the target device.

d. Github
Acts as sub-version control and tracking for development environments. Provide comparison between different aspects of the codes. Allows for parallel development of multiple branches and is easier to set up for nightly builds.

e. Kotlin/Java Programming Language
Kotlin is a programming language that is designed to fully provide interoperability with Java. This allows a code snippet of Java to be called from Kotlin and likewise the other way around. Both provide seamless integration to android development.

5.1.2 Technology Concepts Applied

The application is designed to work closely with a database system. This means the client must be able to perform tasks simultaneously, or asynchronously. The task that is being done should not block another task, especially the user interface.

As the software product is an android application, the range of devices that can be deployed is extensive. The SDK and API level (Android OS Version) is selected from how much change occurs from the previous level. This reduces the workload needed to maintain some part of the project while supporting a few more devices.

Finally, as of the current date, this report is written, Microsoft Windows 11 is planning to support android applications natively on their operating system. Heavy dependencies that are tied to a specific microarchitecture are avoided in implementations, and cross-platform libraries are considered.

5.1.3 Techniques/Algorithm

The main architectural design is built around the concept of MVVM architecture with Clean architecture. Decoupling is emphasised in this architecture. Testing modules could be deployed and testing is done with ease. Structures are divided into the presentation layer, domain layer, and data layer. The presentation layer is further divided into the user interface and the logic that controls it. The domain layer is used to communicate with the data layer through repositories. The repository is then used to fetch the data, which is described by the data model. As such, features can be implemented more easily and provide high maintainability in comparison to other architecture. The disadvantage of this architecture is the amount of objects needed.

Some user ihnterfaces are automatically updated through a technique called live data binding. The data from a collection are refreshed and actuate an action by utilising an event listener object. Data persistence is provided by the view model (presentation layer) of the software architecture.

5.2 Server-Side Development

5.2.1 Development Tools
For Server-side development, the tools used to create our database are as follows:

a. Firebase Realtime Database
   The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time.

b. Django
Django is a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern.

5.3 Internet of Things Development

5.3.1 Development Tools

a. Visual Studio Code
VSCode is an open-source source-code editor, which includes support for debugging, syntax highlighting, IntelliSense, snippet, code refactoring, and embedded Git. It is available for macOS, Linux, and Windows

b. Home Assistant
Home Assistant is open-source software for home automation that is designed to be the central control system for the Internet of Things (IoT) devices.

c. ESPHome
ESPHome is a tool which allows users to create custom firmware for ESP8266 and ESP32 boards. It reads a YAML file and creates a custom firmware binary. With the OTA (Over The Air), it became the main reason we selected to use ESPHome, we can upload our firmware to our ESP boards without having to use a USB cable for uploads.

5.3.2 Technology/Concepts Applied

a. OTA update (over-the-air update)
An over-the-air (OTA) update is the wireless delivery of new software, firmware, or other data to mobile devices such as tablets, smartphones, and Internet-of-Thing (IoT) devices.

b. Home Assistant Automation

Chapter 6

Results

6.1 User Interface (Management)

Figure 6.1.1 Loading page

Loading page: display the picture of the application logo

Figure 6.1.2 Login page

Login page : The user can sign in by entering their username and password on the login screen.

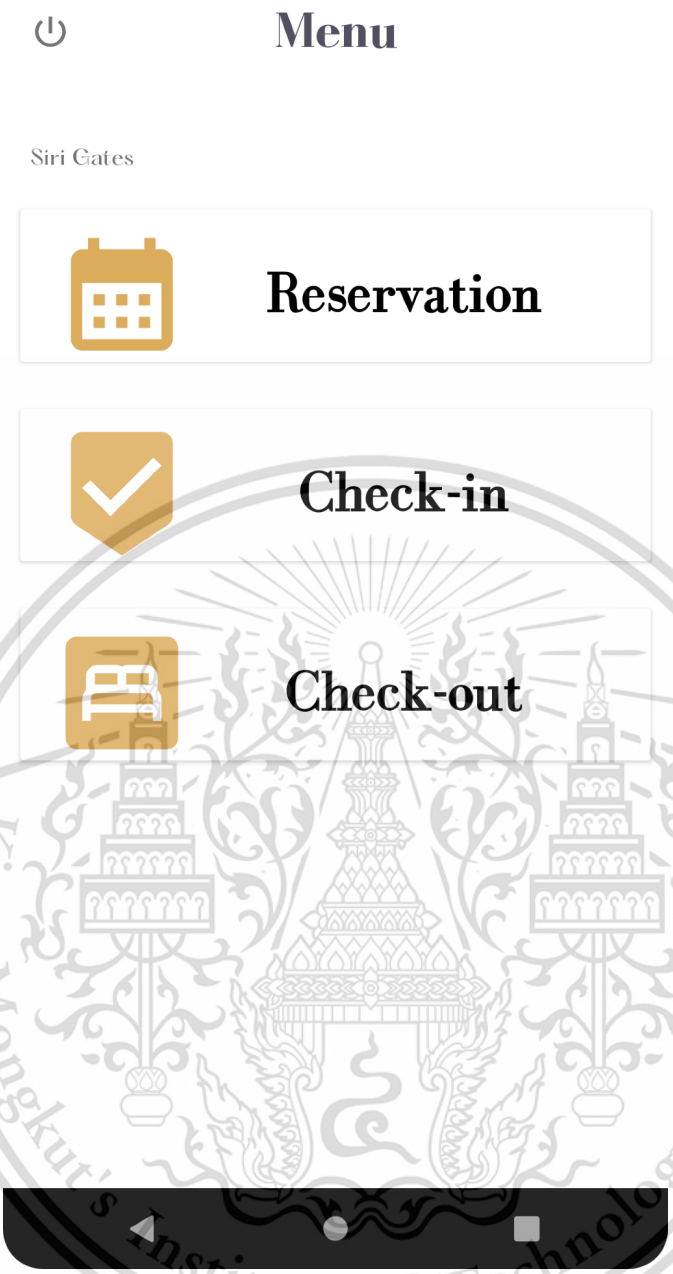

Figure 6.1.3 Main menu page

Main menu page: display the three menus included with Reservation menu, Check-in menu, Check-out menu

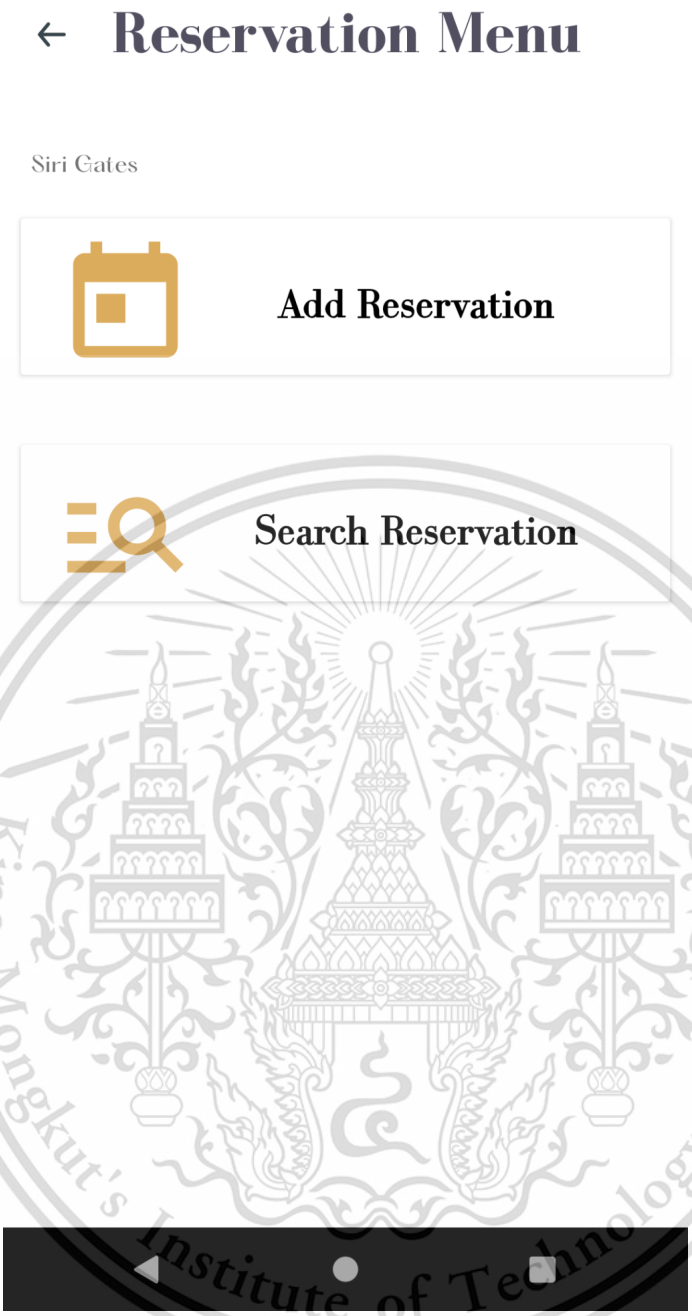

Figure 6.1.4 Reservation Menu page

Reservation menu page: The Add Reservation Menu and Search Reservation Menus are displayed on the Reservation Menu page.

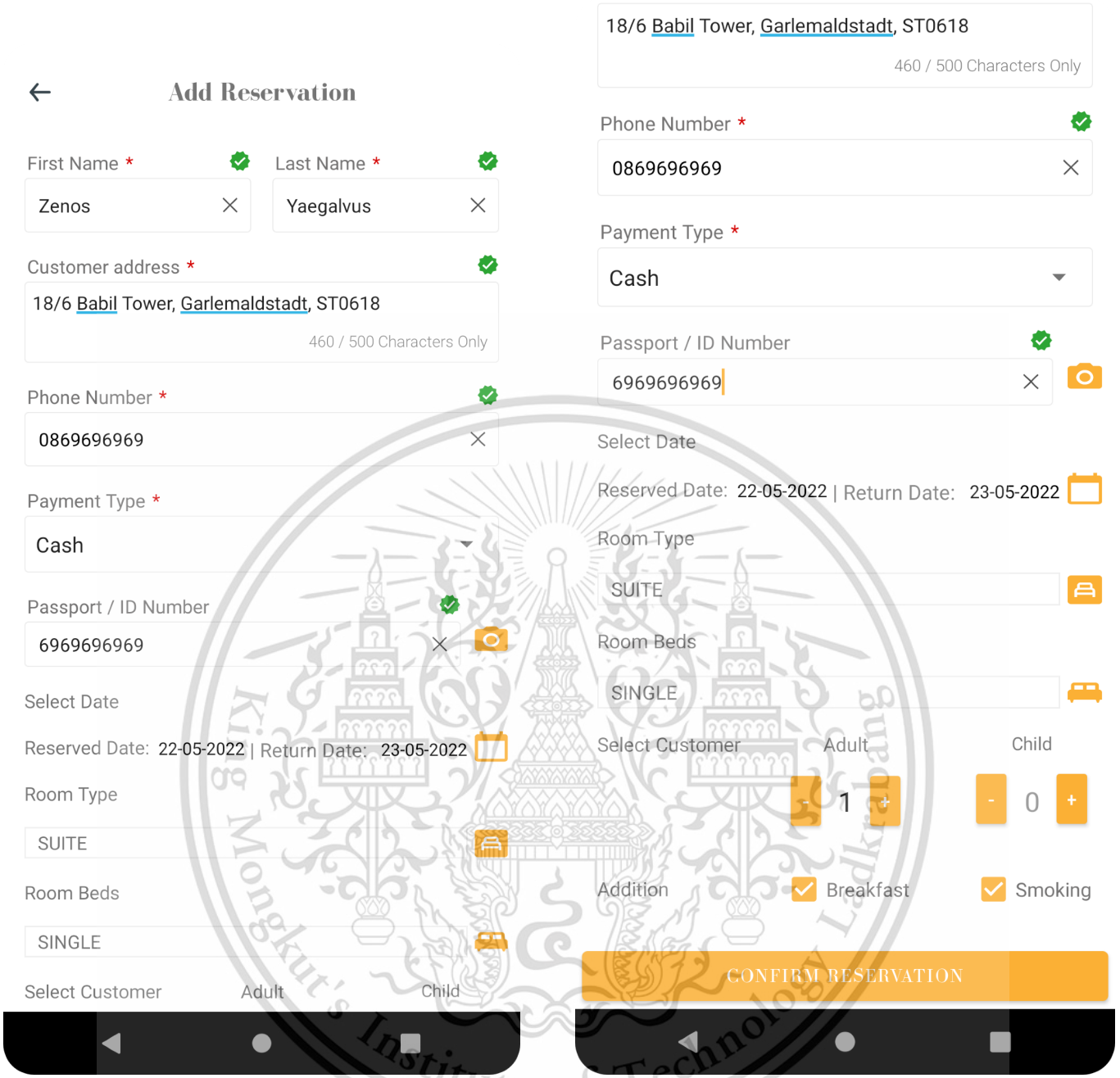

Figure 6.1.5 Add Reservation page

Add Reservation page: allow users to fill out a reservation form that includes their first and last names, customer address, phone number, payment type, reserved date, return date, room type, room beds, numbers of customers, and additional requests.

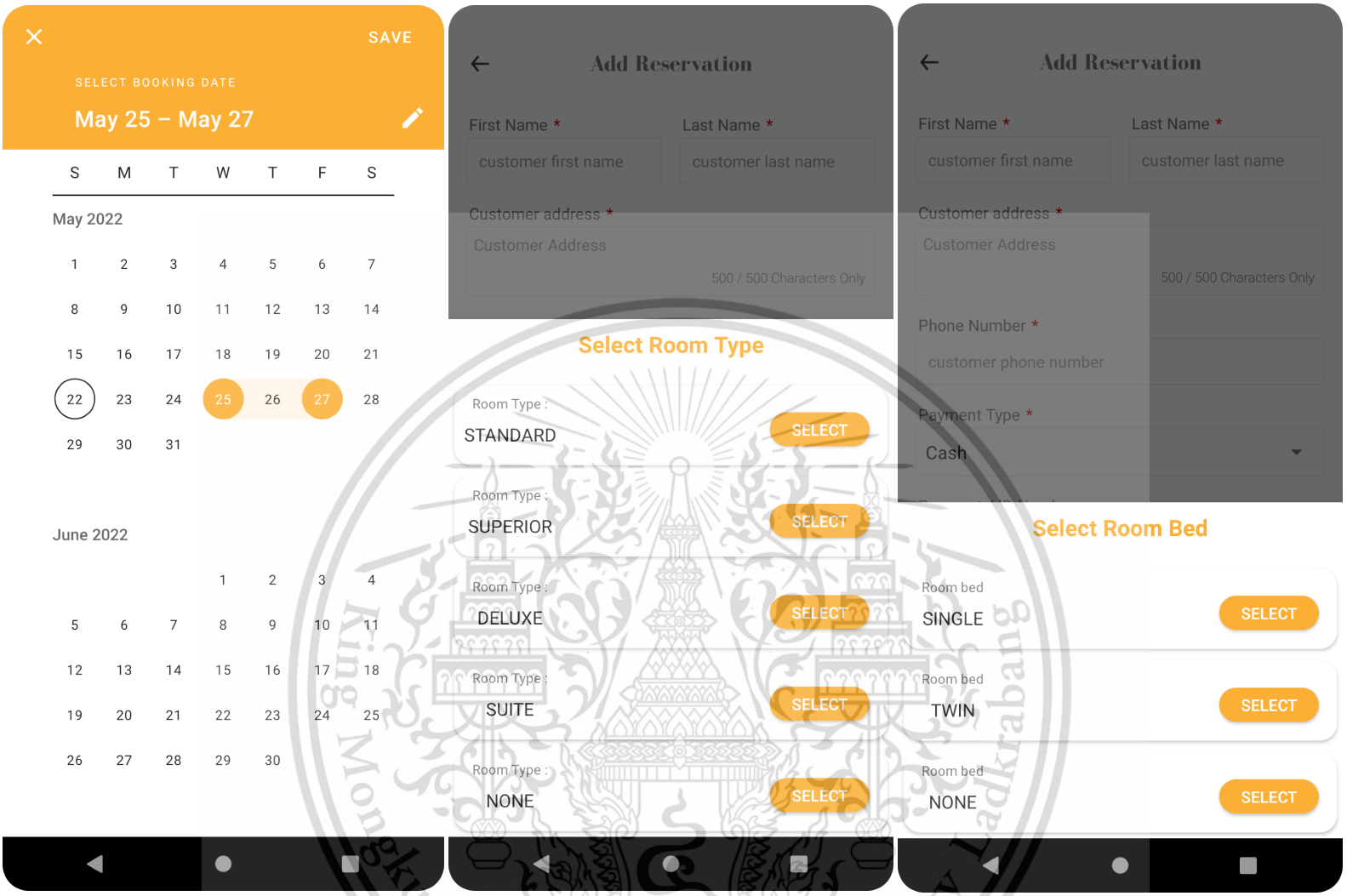

Figure 6.1.6 Add Reservation page (continue)

Add Reservation page: By pressing the calendar icon, the user can choose a reserved date and a return date. By pressing the room icon, the user can choose a room type. By pressing the bed icon, the user can choose a bed type in the room.

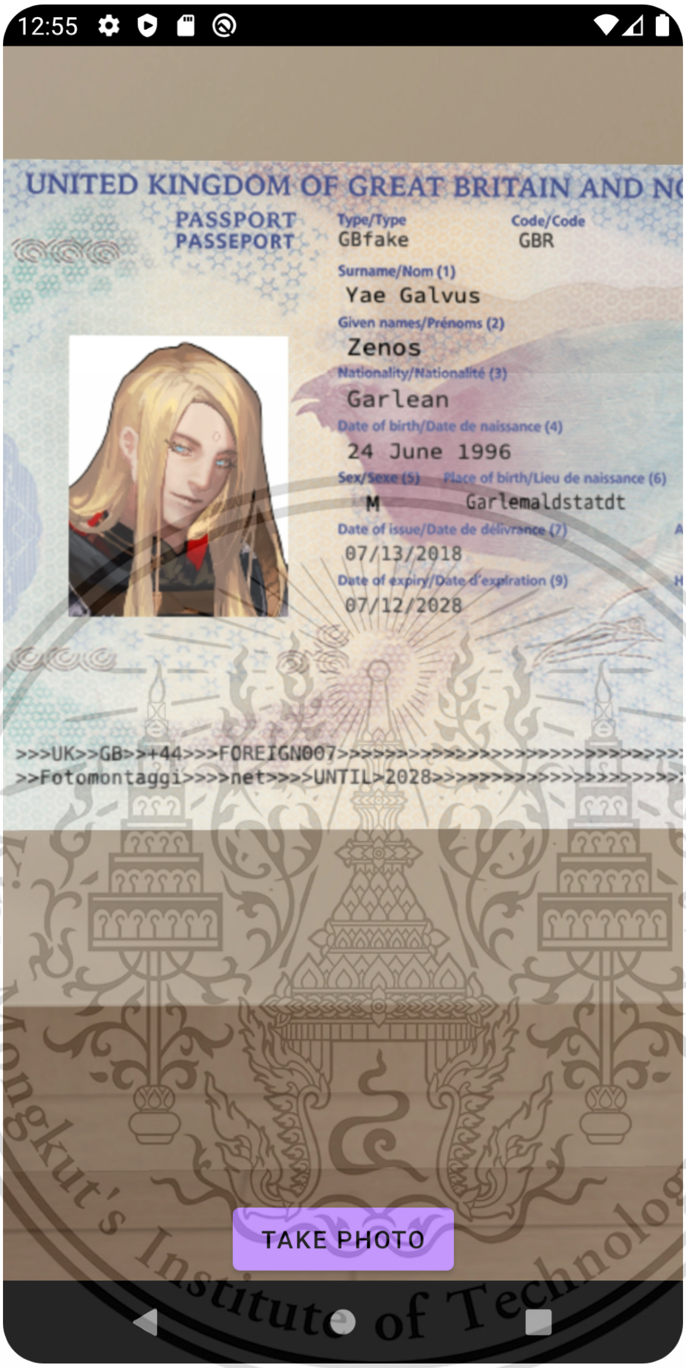

Figure 6.1.6 Add Reservation page (continue)

Add Reservation page: By pressing the camera icon, the user can take a photo and return a picture.

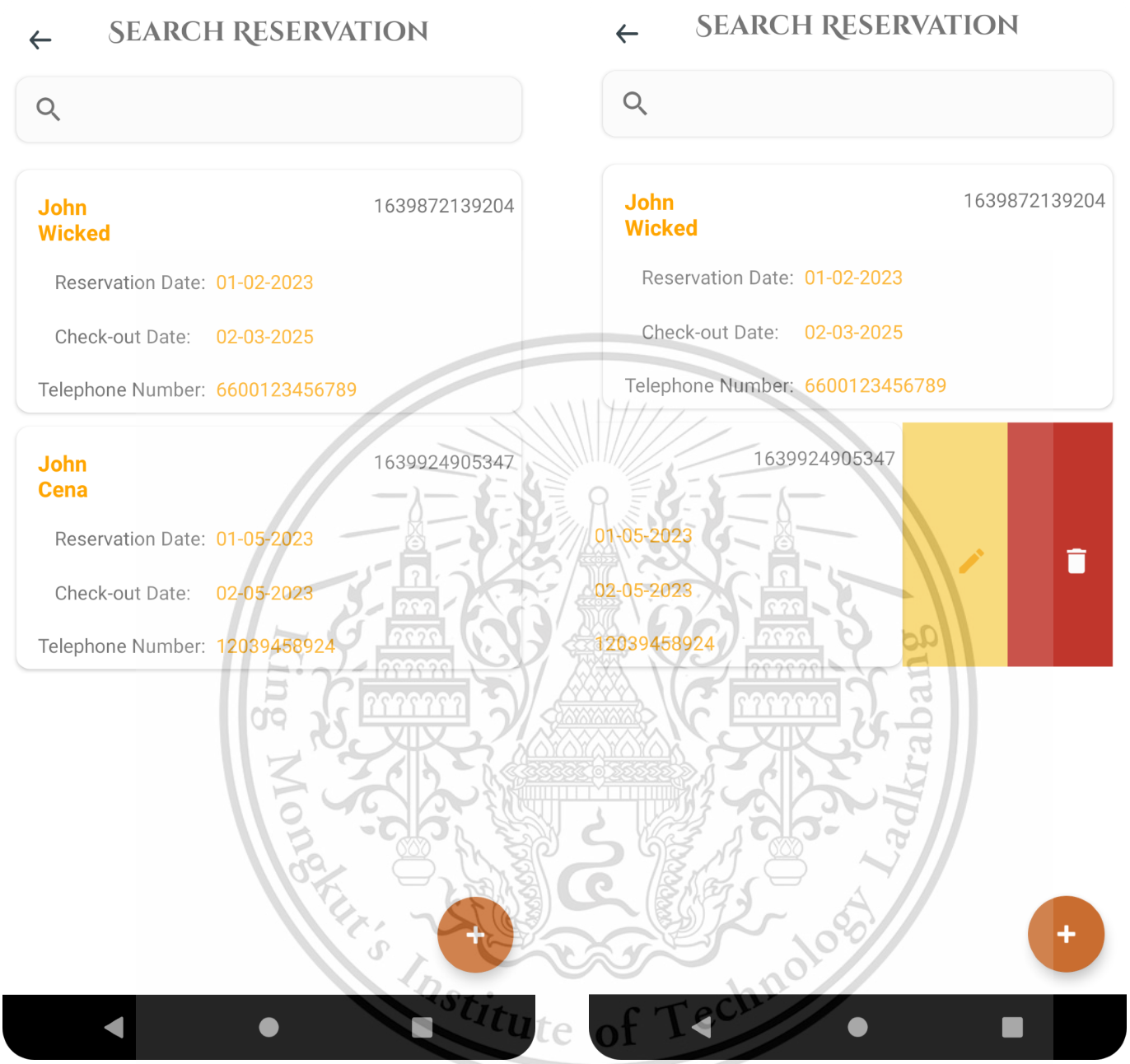

Figure 6.1.7 Search Reservation page

Search Reservation page: The user can search for reservation using the search bar provided. The search result is listed below while the user types. The user can also slide each card to left in order to edit or delete that specific reservation page

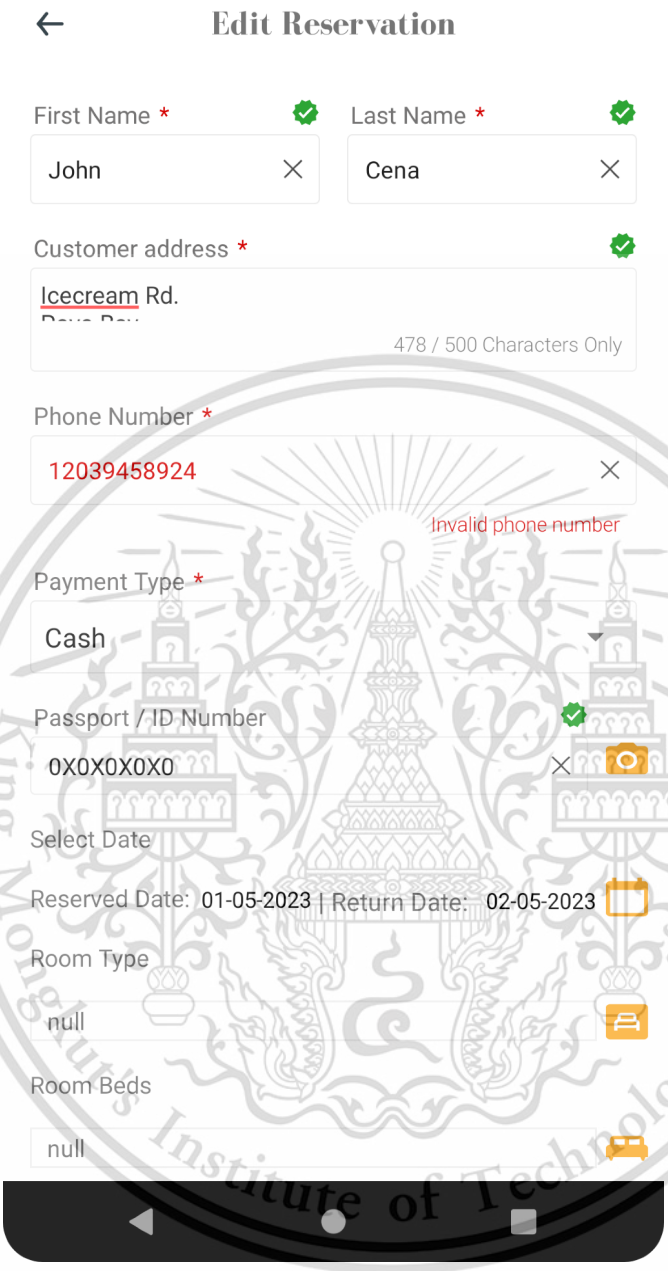

Figure 6.1.8 Edit Reservation Page

Edit Reservation Page: The user can edit the reservation page. The feature is similar to the add reservation page.

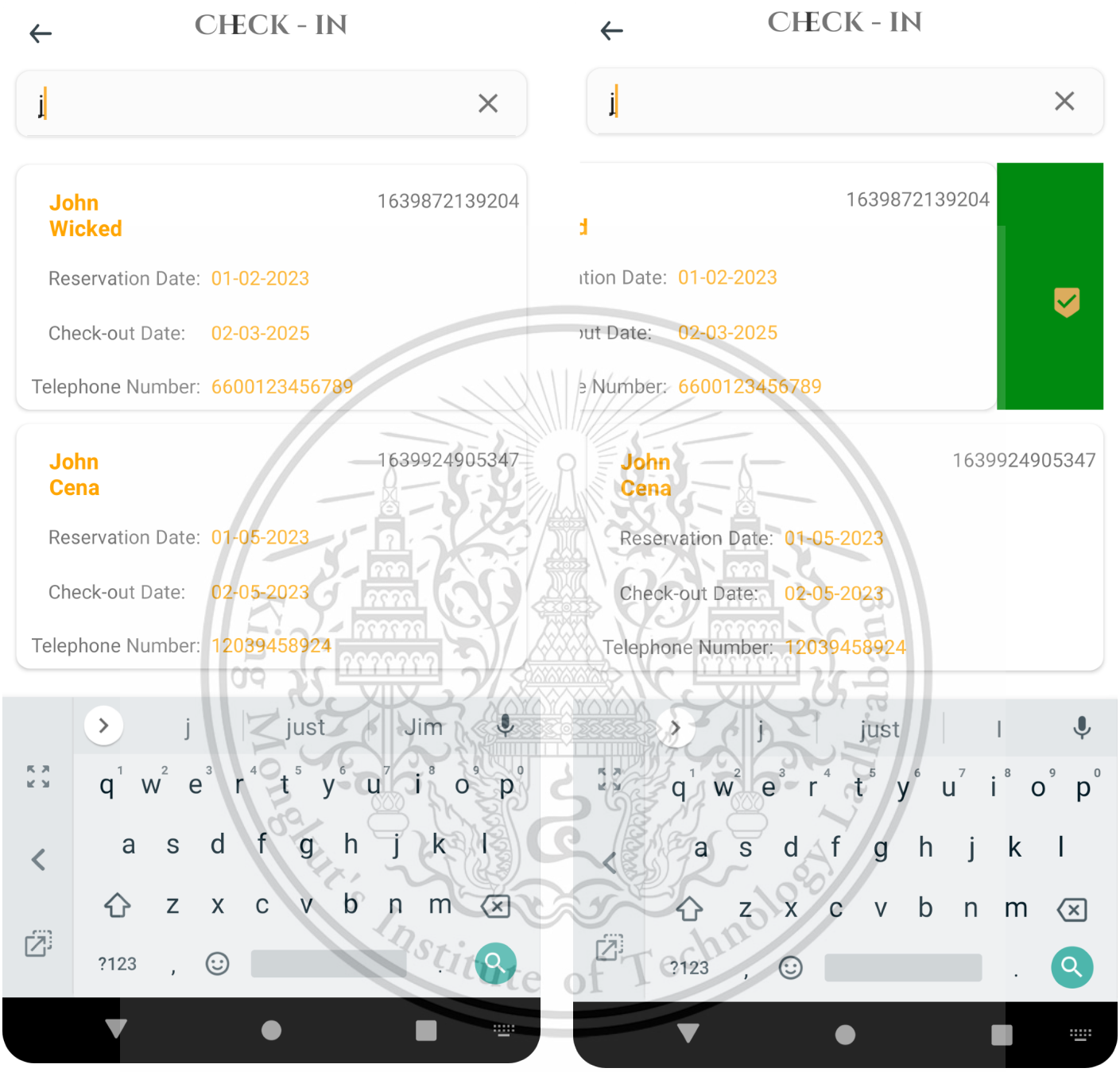

Figure 6.1.9 Check-in Search Page

Check-in Search Page: The user can search using the search bar provided. The search result is listed below while the user types. The user may check in each card by sliding it to the left.

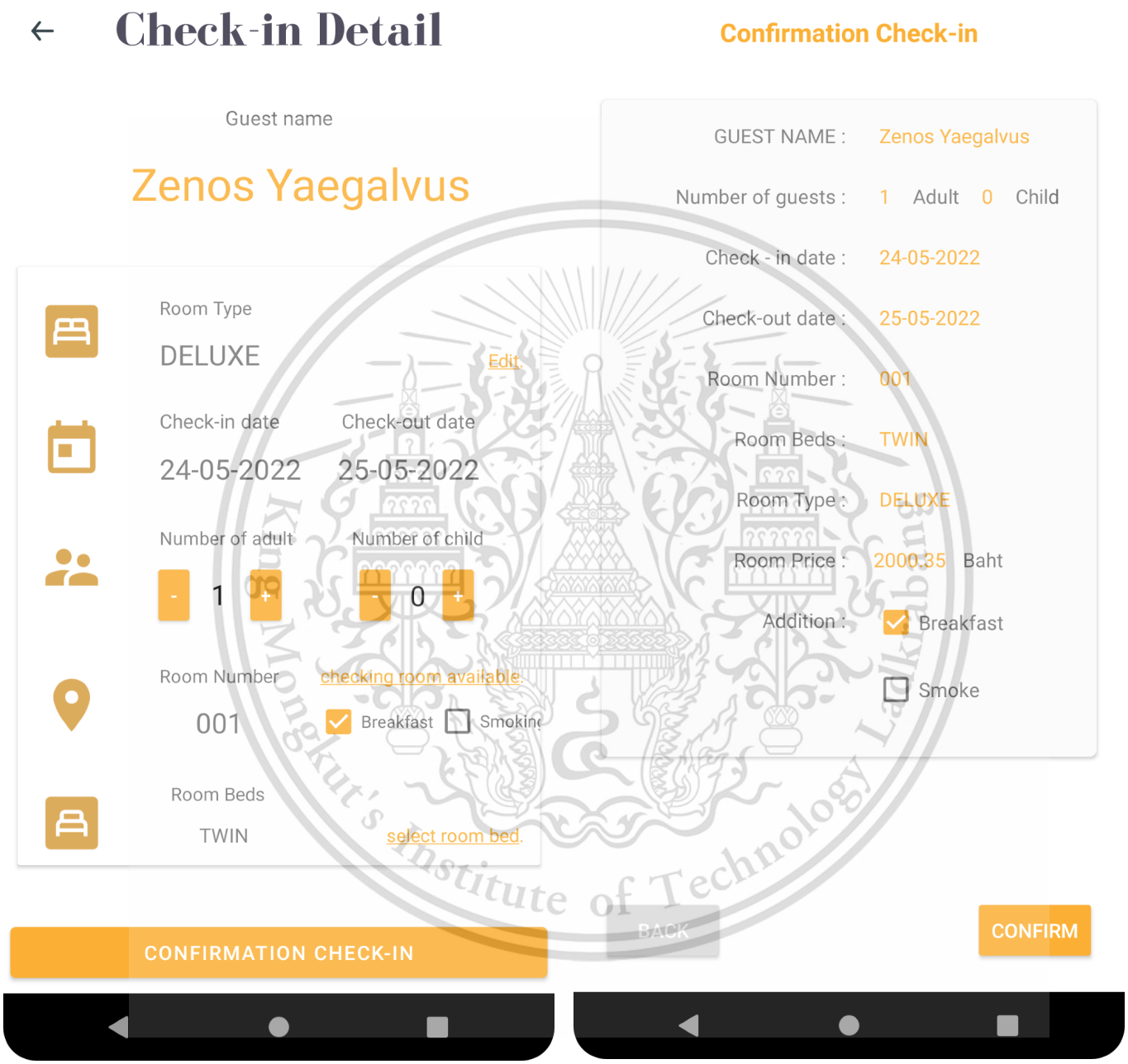

Figure 6.1.10 Check-in Confirmation Detail Page

Check-in Confirmation Detail Page: The user is able to update confirmation details and confirm them after editing.

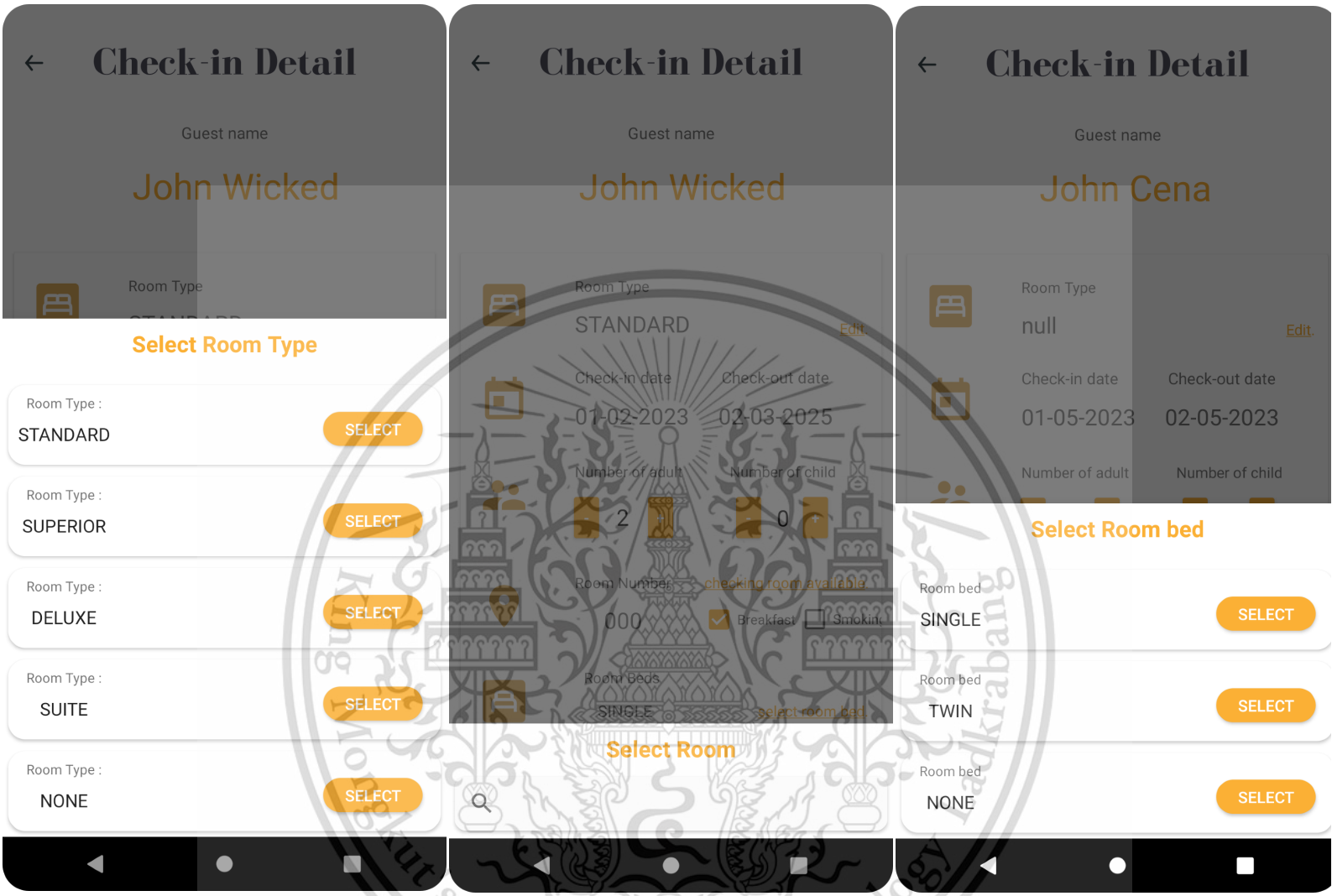

Figure 6.1.11 Check-in Confirmation Detail Page (continue)

Check-in Confirmation Detail Page: The user can reselect room type, room bed. For the checking room available, the user can search using the search bar provided. The search result is listed below while the user types.

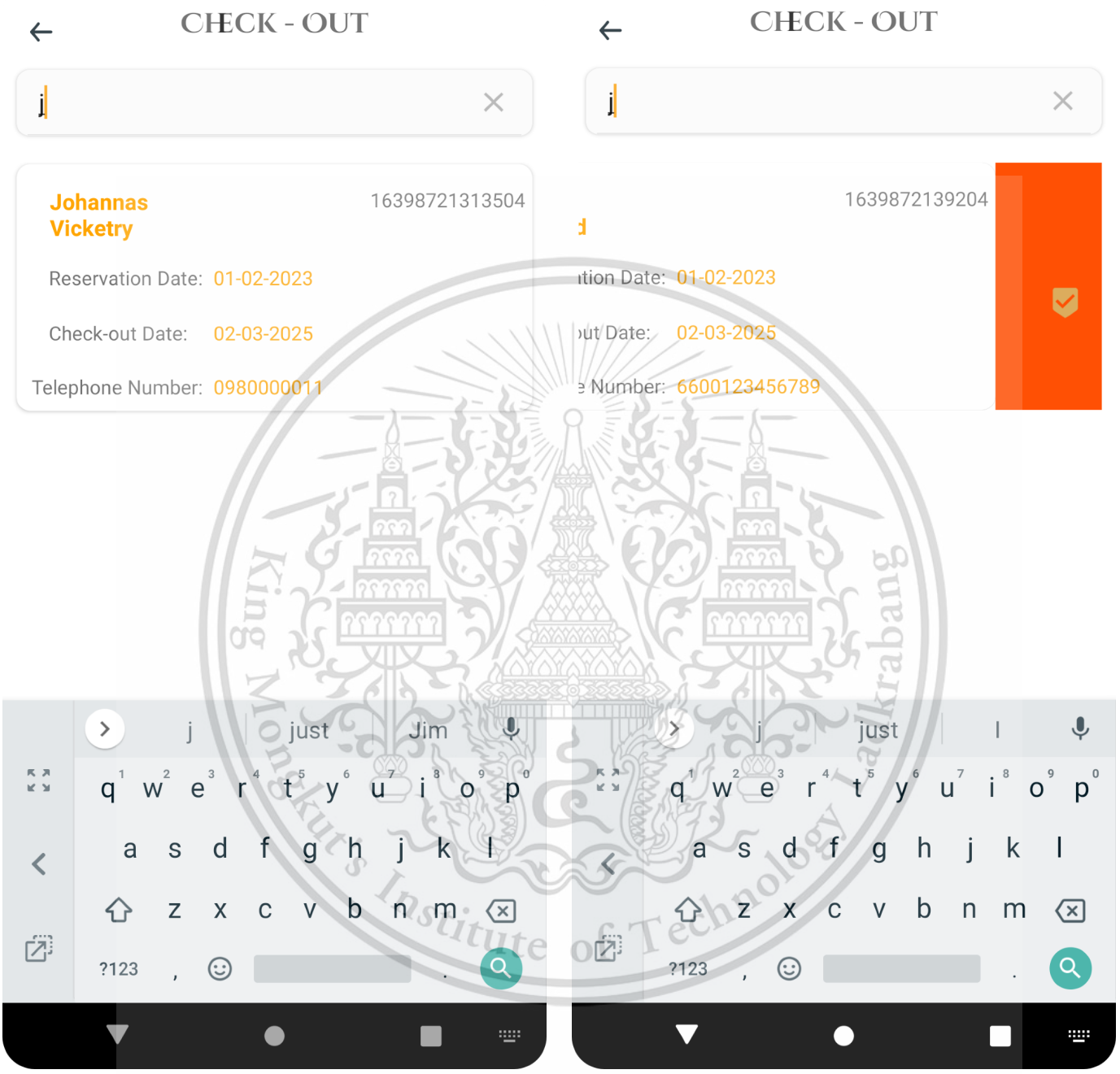

Figure 6.1.12 Check-out Search page

Check-out Search page: The user can search using the search bar provided. The search result is listed below while the user types. The user can also left-slide the card in order to check out a specific card.

6.2 User Interface (Automation)

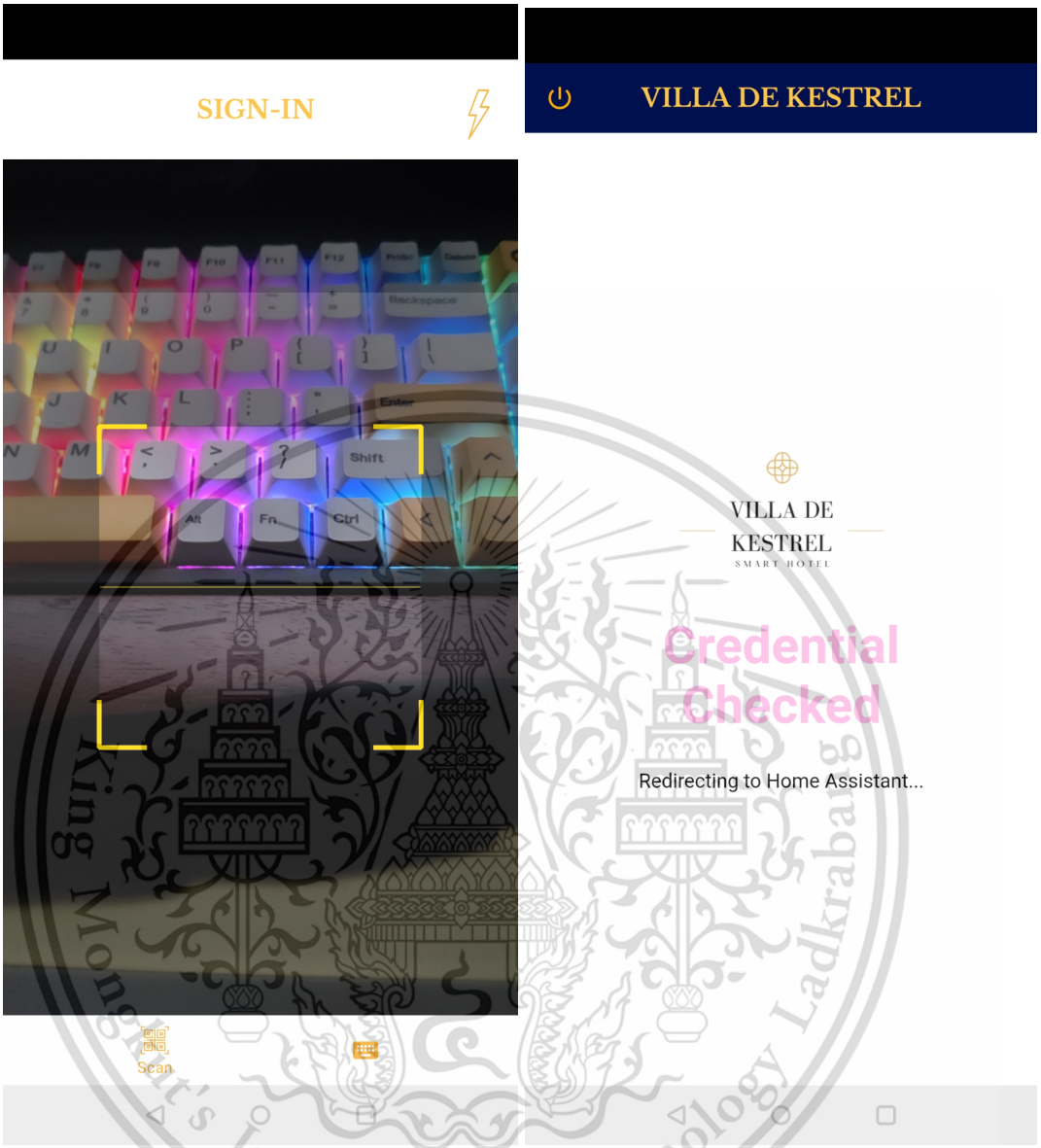

Figure 6.3.1 QR Redirect (Guest)

Upon scanning QR code for the guest side, the server will automatically complete the authentication process for them, resulting in a smooth user experience

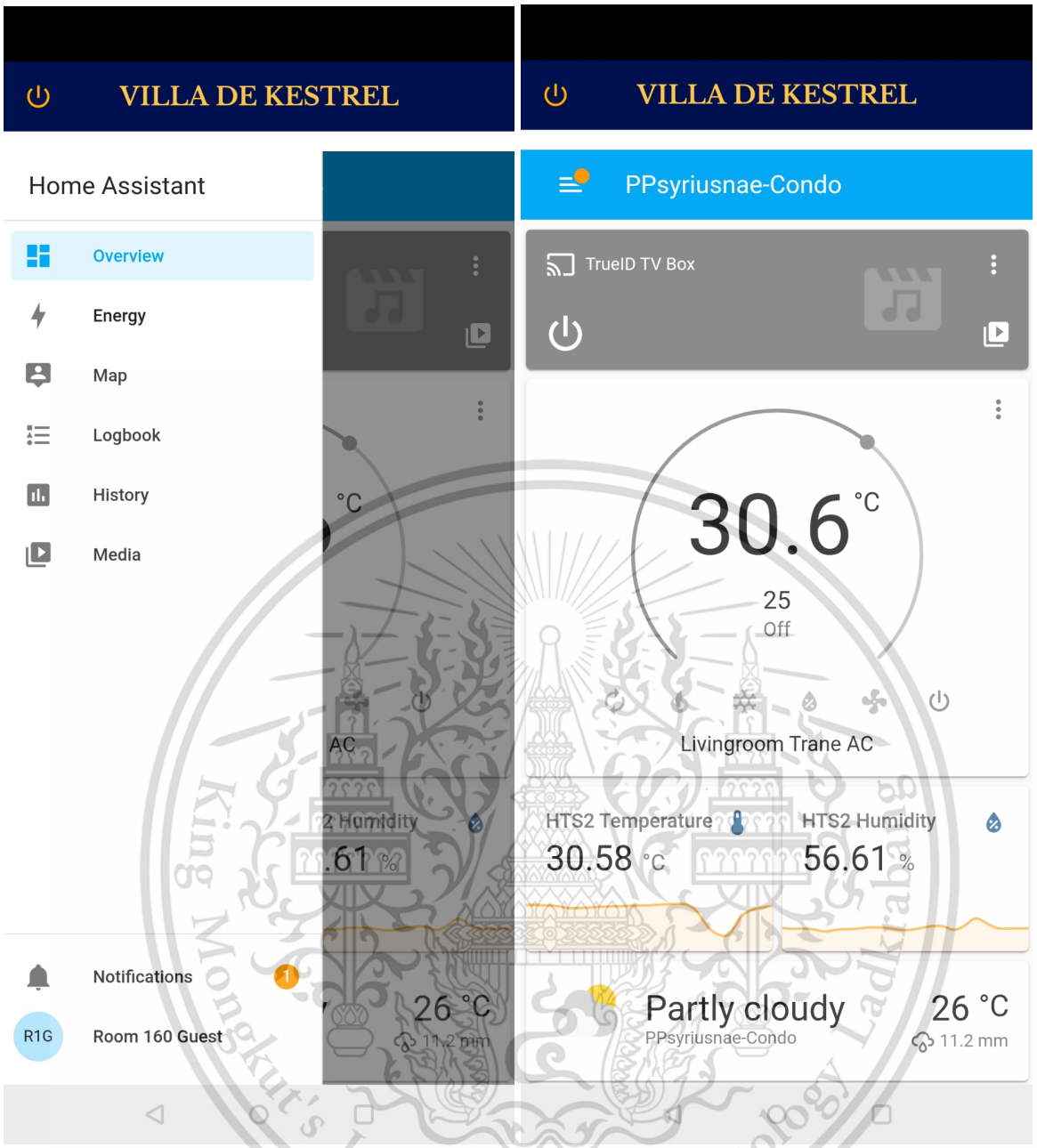

Figure 6.3.2 HA Control Interface (Guest)

Guest can only access control within their own rooms.

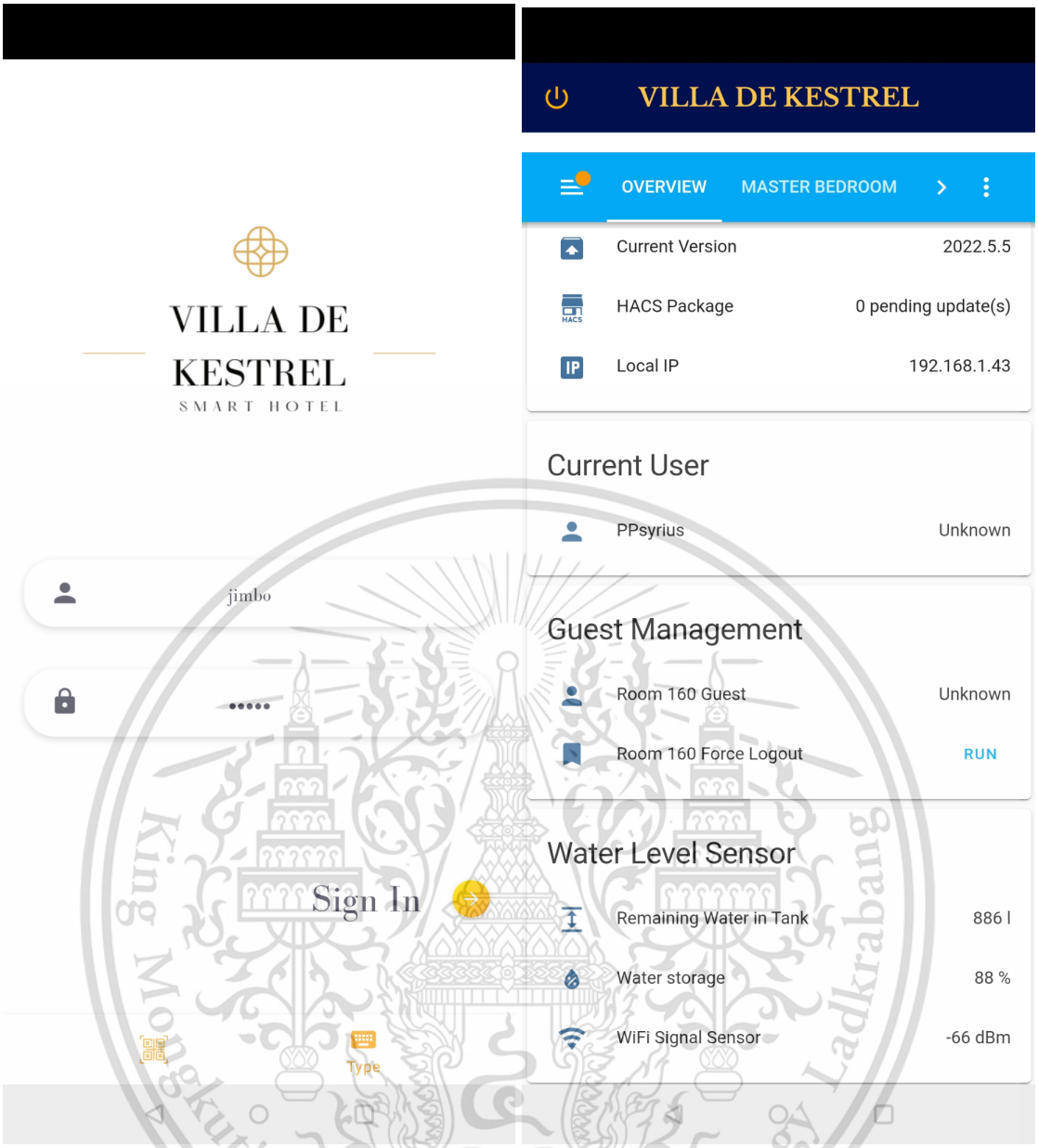

Figure 6.3.3 HA Control Interface (Staff)

Hotel Staff has access to all relevant features for the whole hotel.

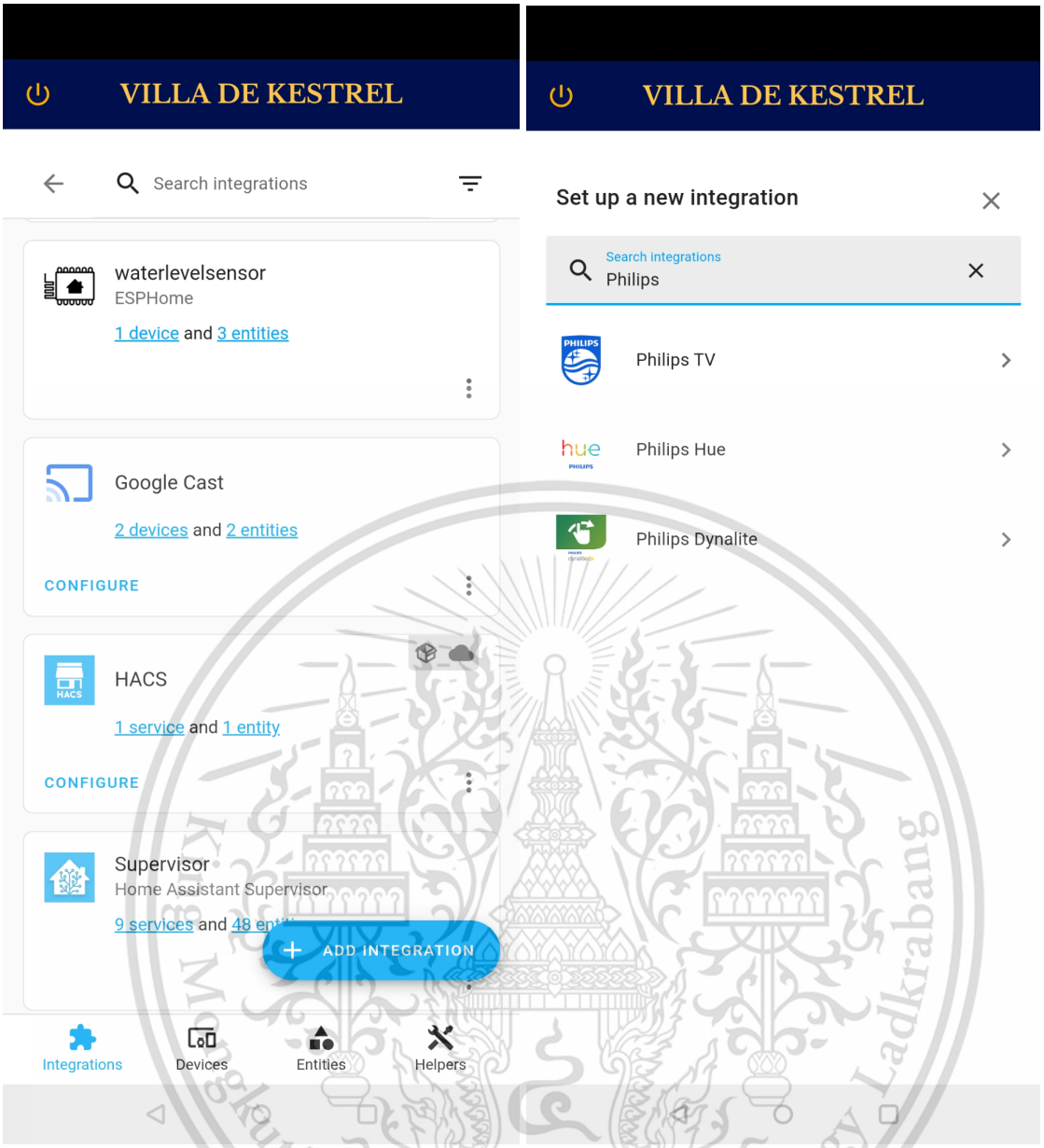

Figure 6.3.4 HA Device Integration Management (Staff)
Staff can easily add in integration for branded IoT devices here should they have one.

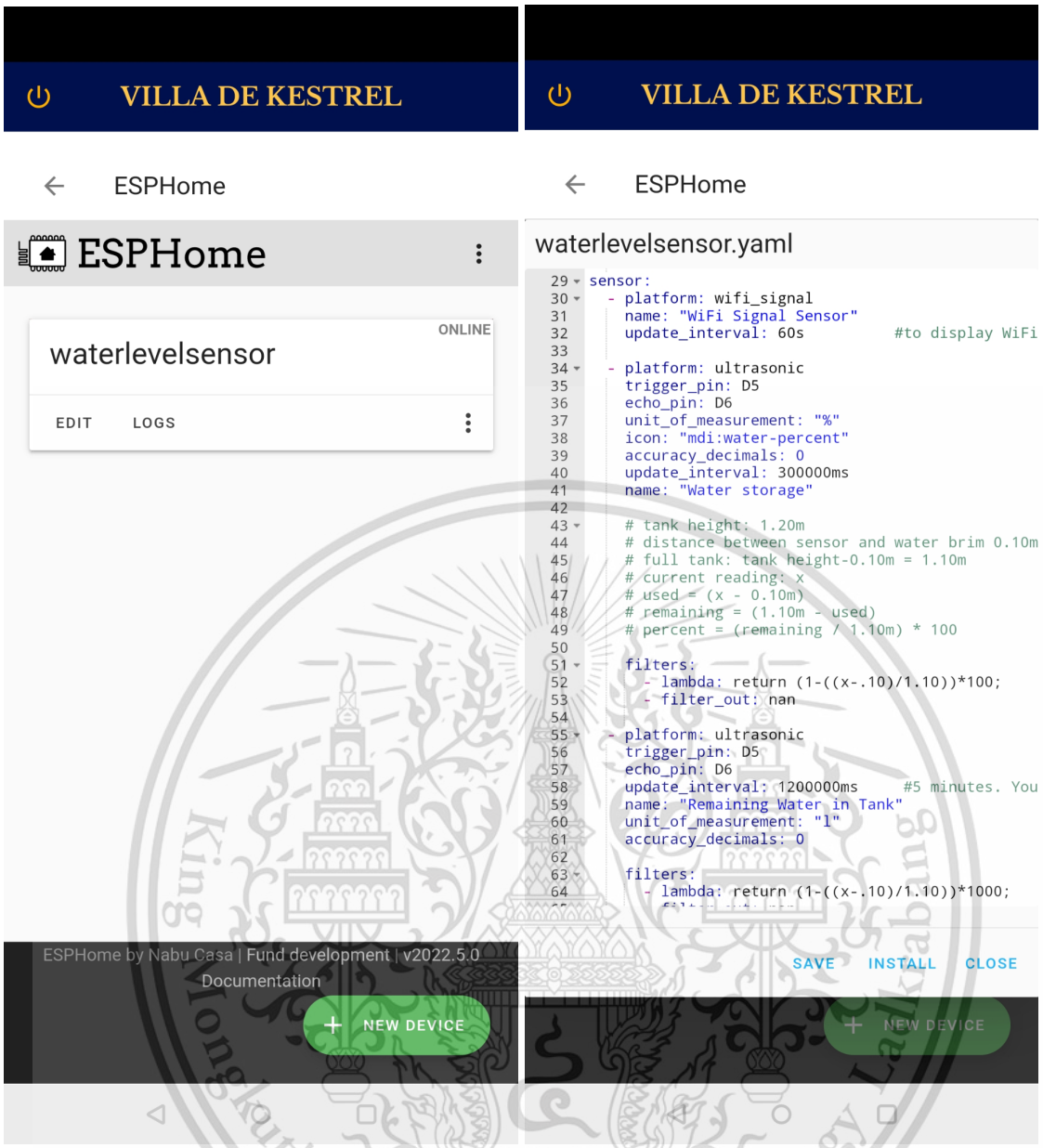

Figure 6.3.5 ESPHome (Staff)

For custom IoT devices based on the ESP platform, OTA updates can be easily issued via this interface.

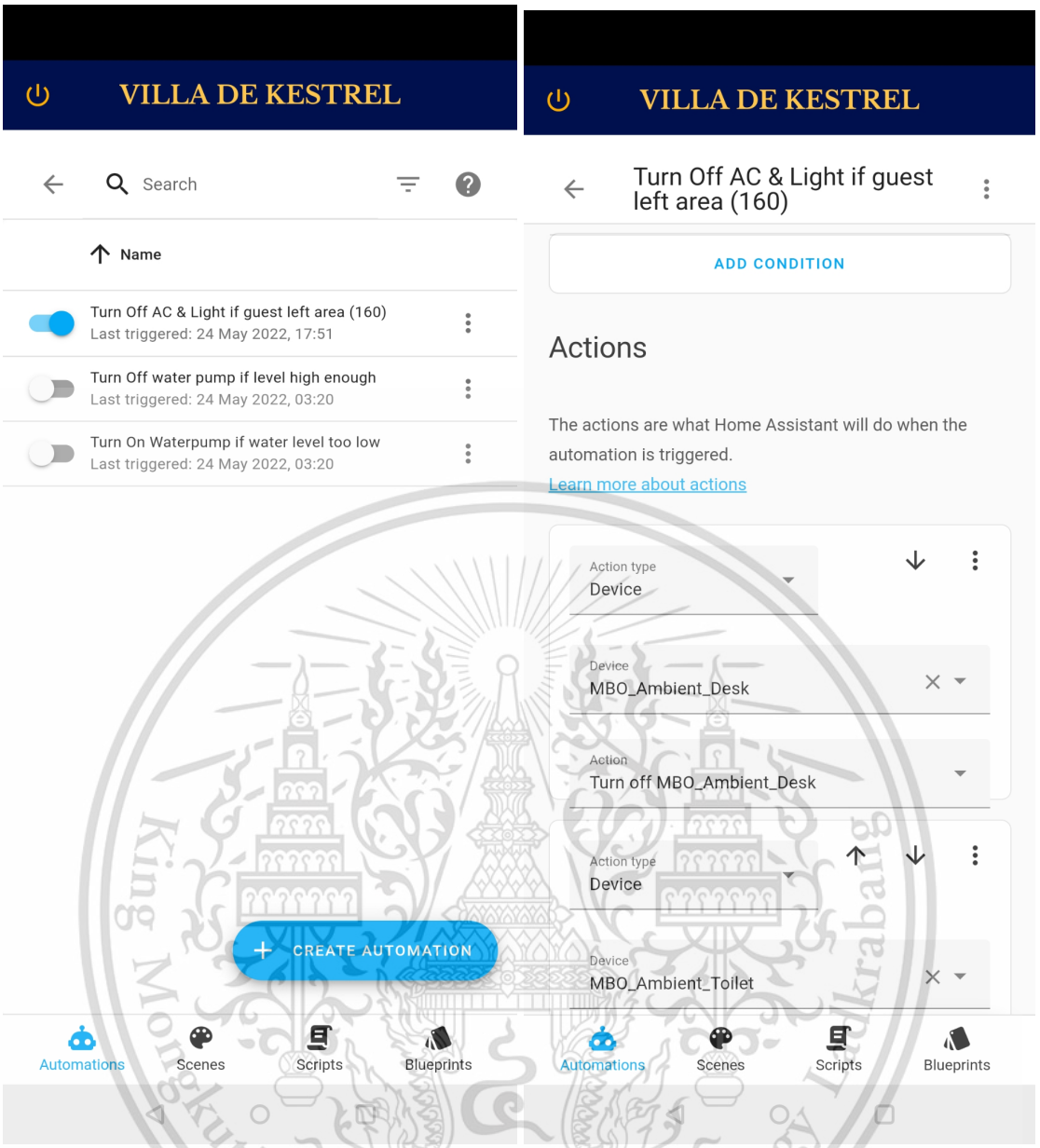

Figure 6.3.6 HA Automation (Staff)

Staffs can enable/disable certain automations here, as well as create new ones as they see fit.

Chapter 7
Conclusion and Future Work

7.1 Summary

In Conclusion, the hotel reservation application provides the user with the ability to log in, add reservations, search for reservations, both normal check-in and walk-in check-in, and check out. On the Add Reservation page, the admin must fill out the input form and photograph the national ID/passport card with a camera. After filling out all of the required fields (first name, last name, customer address, phone number, payment type, passport / ID number, reservation date, and return date), the user can choose room type and room beds, customer count and additional menu and when clicks on confirmation reservation button will be redirected to the reservation menu. The client reservation data from the add reservation menu will be presented on the Search Reservation page. Users may search for customer information by typing the customer's name into the search field on the Search Reservation page, and then swiping the customer item to edit or remove the reservation. The client name, reservation date, check-out date, and telephone number will be shown on the Check-In page. Users may look for a specific client by typing their name into the search field. Users may swipe the customer item which is shown to navigate to the Check-In detail page, and if they have a customer walk-in, the User can click the + button to go to the Walk-in Check-In page. The Walk-in Check-In page uses the same way as an add reservation menu. The customer information from the reservation page will appear on the Check-in Detail page, and when a user clicks the confirmation check-in button, a confirmation check-in page will appear for the user to verify that the customer information is valid. If the user hits the confirm button, the check-in procedure is complete, and the last choice is the Check-Out menu, which displays the data of the customer who has checked in. Users can swipe the customer check-in item to check out, indicating that the check-out is complete.

Similarly, the Hotel Automation side of things streamlines the onboarding process for guests to access the Hotel's Internet of Things capabilities. Unlike the traditional method where a generated password would be given to guests on a piece of paper, the guests can simply scan the QR code to access their room control, with the authentication process handled by a separate server for a smooth experience. In the part of hotel automation, The Customer can use a QR scan to open the Home Automation. Should the guest check out or request a QR renewal, all sessions for that account would be automatically terminated for safety reasons. On the other hand, the hotel staff would benefit more from centralised control of devices within the hotel. For example, staff could quickly check if anything happened in the CCTV, turn on/off the lights in the walkway, or adjust the air conditioner temperature throughout the whole building (or more) from a single application. This is unlike the traditional implementation where different brands of IoT devices would require separate applications just to function, which is quite a hassle to switch around. Furthermore, a unified control allows for better automation integration as

this lessens the amount of delay created by data transmission between all the various brand-specific IoT servers. This also acts as another later of redundancy as both the hotel staff and guests can freely access these features even if the network connection to outside the hotel is down. Another benefit of basing our development on the existing Home Assistant platform is the ESPHome integration, which enables the ability to quickly deploy a custom IoT devices to suit special needs (a water pump controller for example) with an ability to update them all later on via Over-the-air (OTA) update capabilities, thus making the whole system more robust with less maintenance needed throughout the product lifespan.

7.2 Future Work

- Server-side (Home Assistant and Database)
    - Optimization of QR Authenticator server
    - Sensors return signals in an acceptable range
    - Finalise User Interface and User Experience
    - Complete the hotel management system to be fully functional
    - Migrate features to the required architecture design
    - Binds the existing functionally to their respective new design
    - Implements the features on hardware controls
    - Migrate the database from the Firebase to Django based server for better redundancy
- Hotel Management Application
    - Optimize the speed and security of the application
    - Room occupancy mapping for better visualization
    - Implements an internal communication for hotel staff
    - Allows the administrator to edit status enums
    - Compatible with regional and international booking channels
    - One-click reporting features to track hotel property's short- and long-term performance.
    - More beautiful UI design
    - UI for the tablet form factor
    - Dark mode
    - Supporting multiple languages
- Hotel Automation Application
    - Lighter weight
    - Better UI design
    - Dark mode
    - Supporting multiple languages

Reference

[1] Tuya Hotel. *TuyaHotel*. Retrieved from: https://hotel.tuya.com/
[2] Cloudbed. *Cloudbed*. Retrieved from: https://www.cloudbeds.com/
[3] TheKoreaBizwire. *Cloudbed*. Retrieved from: http://koreabizwire.com/cloudbeds-introduces-new-payments-solution-to-further-streamline-hotel-operations/191627
[4] SabeeApp. *SabeeApp*. Retrieved from: https://www.sabeeapp.com/
[5] HotelMinder. *SabeeApp*. Retrieved from: https://www.hotelminder.com/partner=SabeeApp
[6] EasyFo. *EasyFo Hotel Management System*. Retrieved from: https://www.easyfo.com/
[7] Wikipedia. *Home Assistant*. Retrieved from: https://en.wikipedia.org/wiki/Home_Assistant
[8] Firebase Documentation. *Firebase Realtime Database*. Retrieved from: https://firebase.google.com/docs/database
[9] Wikipedia. *Django (Web Framework)*. Retrieved from: https://en.wikipedia.org/wiki/Django_(web_framework)
[10] Android Developers. *Android Kotlin Fundamentals*. Retrieved from: https://developer.android.com/courses/kotlin-android-fundamentals/overview
[11] Android Developers. *Run apps on the Android Emulator*. Retrieved from: https://developer.android.com/studio/run/emulator
[12] Abhishek Tyagi. *Better Android Apps Using MVVM with Clean Architecture*. Retrieved from: https://www.toptal.com/android/android-apps-mvvm-with-clean-architecture
[13] Golnaz Torabi. *Clean Architecture with MVVM*. Retrieved from: https://levelup.gitconnected.com/clean-architecture-with-mvvm-34cc05ab3bc5
[14] Android Developers. *Guide to app architecture.* Retrieved from: https://developer.android.com/jetpack/guide
[15] Android Developers. *Kotlin Bootcamp for Programmers*. Retrieved from: https://developer.android.com/courses/kotlin-bootcamp/overview