5-1-2021

ESTATE MANAGEMENT

Rhea Pottathuparambil

ESTATE MANAGEMENT

by

RHEA POTTATHUPARAMBIL

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2021

ACKNOWLEDGMENTS

I would like to thank my Professor, Dr. Christopher McMurrough and Mr. Kent Pawlak for guiding through the entire project. I would also like to thank my group members for their teamwork and support throughout the entire project.

May 4, 2021

ABSTRACT

ESTATE MANAGEMENT

Rhea Pottathuparambil, B.S. Computer Science

The University of Texas at Arlington, 2021

Faculty Mentor: Christopher McMurrough

The Estate Management system is based on IoT (Internet of Things). The Estate Management system tracks and updates the status of various devices of an institution and provides a graphical user interface to display their status. Many design choices were carefully evaluated for each subsystem. These evaluations were done by comparing the constraints, ramifications, and advantages of each model. The design and implementation of front-end layer of the project is also discussed. Regular meetings with Transact Inc. were held to better understand the project and gather the system requirements. Tools such as React, HTML, and CSS were used to design and develop the dashboard for the project.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Product Concept

Our Estate Management system is based on IoT (Internet of Things). The Estate Management system tracks and updates the status of various devices of an institution and provides a graphical user interface to display their status. The main purpose of this project was to develop a dashboard or estate manager to have all hardware devices report into a central system. This central system monitors the connected IoT devices and stores the history of status of each device. Each institution has their own central app to view their connected devices however they are restricted to view the details of other institutions.

1.2 Purpose and Use

The Administrator can view the status of the devices of all the associated institutions categorized by their institution ID and perform the administrative actions. The secondary component of this project is the client application that communicates between the central host and the IoT devices. Any event occurring in the IoT devices are transferred through this app to the central host.

1.3 Intended Audience

This system will be used mostly by institutions like universities to see the status of devices for campus access, such as: card readers, door locks, and contactless cards. The system alerts the owner (institution) of any device failure and monitors the general behavior of the devices. The system uses Azure IoT to contact back and forth with the connected IoT devices.

CHAPTER 2

SYSTEM OVERVIEW

2.1 Introduction

Our system consists of the three major layers. The first layer is the front-end layer which is also called web app. The second layer is the back-end layer which is the major layer of our system. The third layer is the IoT Edge consisting of IoT devices. The descriptions of each layer are given in the following sections.

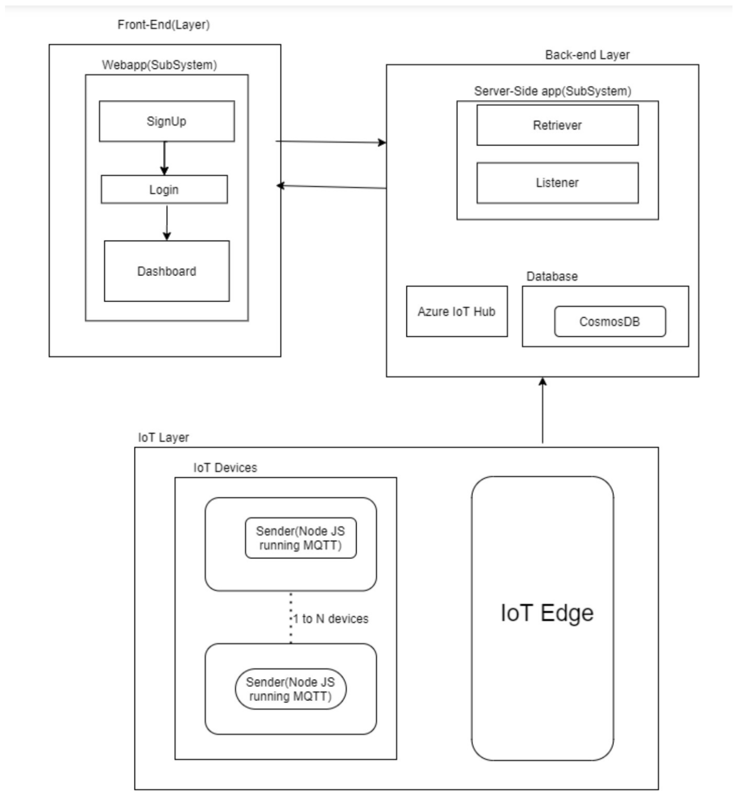

Figure 2.1: System Overview of Estate Managing Service

2.2 Front-End Layer

The Graphical User Interface (GUI) is the layer designed for the user or admin. Its main function is to register the authorized user and give access to the user and administrator (admin). This layer consists of a registration and login page. The user is redirected to the home page after the user can log in successfully. We have a navigation bar (navbar) at the top which includes the links for Home, Devices, Maps, and Sign-Out. This toggling navbar is highly responsive which reduces size to fit in any screen. The home page consists of the Dashboard showing the list of all the available devices based on the privilege level of the user. Admin has control over all the devices whereas a user or organization has control over the devices owned by them.

This dashboard shows the list of devices as a grid view. When the user clicks the grid view, the status of the device with their information is visible. The user can also view the history of data of each device. The history of the data is seen in the form of a graph. There is a map section in the navigation bar where the user can see where each device is located. Animations were also made to make smooth transitions.

2.3 Back-End (Server) Layer

There are two servers in this layer, making it the backbone for the GUI of the web application. The first server is responsible for listening to the data from the Azure IoT Hub. Azure IoT hub acts like a public endpoint for sharing the information to the databases and front-end layer. This layer also consists of databases for storing the data received from the devices. The data in the database is grouped according to the organization. The second server is our NodeJS server. This server has public API end points which allows the data to be exchanged between the front-end and back-end layers. The API routes include user, authentication, and devices. These are enclosed under the same folder named "routes". The front-end side is responsible for dispatching actions related to these routes. The user info and devices information are sent as a payload from the routes.

2.4 IoT Layer

This layer consists of IoT devices and IoT Edge. The devices are registered in the Azure IoT hub. The Message Queuing Telemetry Transport (MQTT) protocol is used for exchanging information as a JSON string from the device to the back-end server or vice-versa. All the connected devices communicate with the IoT Edge which communicates with the back-end layer.

CHAPTER 3

FRONT-END LAYER

3.1 Introduction

The front-end layer is the user facing layer of the project. I was majorly responsible for the implementation and delivery of our front-end system. OAuth authentication was also used to allow users to sign in and log in safely and securely. This authentication also increased the overall security of the system and restricts the users to only view the information they have access to. This prevents the data from being accessed by the wrong people.

3.2 Sign-Up

The first subsystem under the front-end includes the user sign-up system. This subsystem provides a user interface to allow users to sign-up to our estate manager. The web app homepage navbar contains a link for sign-up. Each user is provided a sign-up form with the following contents: First name, Last name, email, and password. HTML5 was used to validate all the attributes of the form elements. All user details are stored in the database in plain text except for the password. Hashing is implemented for storing passwords to make our user system more secure.

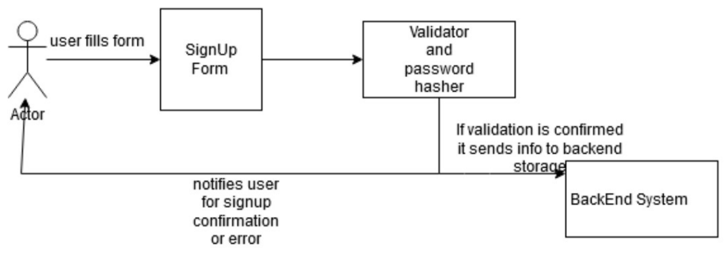
user fills form
SignUp Form
Validator and password hasher
Actor
If validation is confirmed it sends info to backend storage
BackEnd System
notifies user for signup confirmation or error

Figure 3.2: Architectural Dataflow of Sign-up Subsystems

Table 3.2: Sign-up Subsystem Interfaces

ID	Description	Inputs	Outputs
#01	Successful validation	valid user information	Message dialog for successful sign-up
#02	Validation fail	one or more invalid info	Inform user about errors

3.3 Login

The second subsystem under the front-end is user login. The web app homepage navbar will also contain a link for login. Users will be prompted to login with their email address and password before accessing the web app. HTML5 controls are used to validate all the attributes of the form.

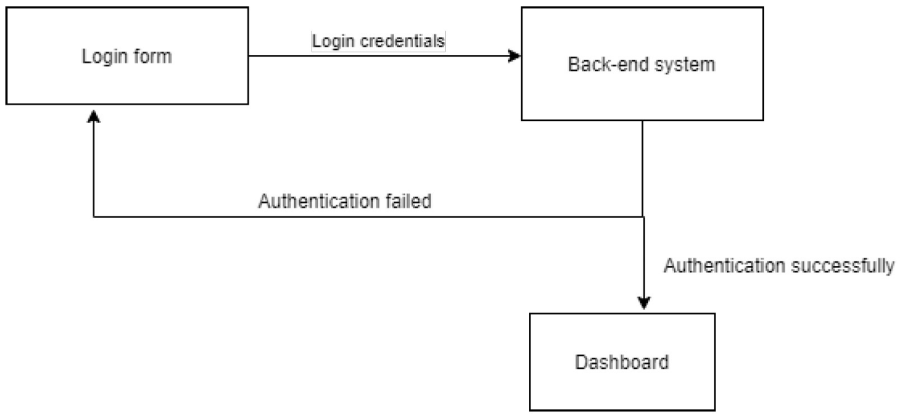

Figure 3.3: Architectural Dataflow of Login Subsystems

Table 3.3: Login Subsystem Interfaces

ID	Description	Inputs	Outputs
#1	Successful login	Valid credentials	Dashboard
#2	Unsuccessful login	Invalid credentials	Login form with error message

3.4 Dashboard

The third subsystem of the web app is the Dashboard. This subsystem is the most important part of the front-end layer as it displays all the important information to the users. After a successful login, the user has the access to the dashboard of the web app which is constantly monitoring the status of the connected devices. The web app used by each institution displays the information about the device ID, device type, device status, and history of the connected devices. Transact support team has the privilege to monitor the devices connected around all institutions categorized by their institution ID.

Many design decisions were made while building the dashboard of the system. These decisions were made against constraints of our project and using results of various surveys conducted with our peers. The UI and UX was designed taking the intended audience and their convenience into consideration. Data was also visualized using react chart library to allow easy understanding of complex data. The webpage was also made responsive to allow users the flexibility to view the web app from various devices.

The dashboard uses the internal API to request data from the server. Most of the data processing was done on the back-end layer to increase the efficiency and make the transitions of the web app fast and smooth. The information related to the devices is shown to the user in a table to make the web app user-friendly. The dashboard also allows the users to search queries and sort the columns of the table to allow easier access to the information. A complex search can also be performed using a combination of searching and sorting. Pagination was also implemented to prevent long tables and to improve the appearance of the web app. Pagination helps display a large amount of data in pages which ultimately helps the user in viewing the data in small chunks.

Finally, the most important reason for this project was to reduce maintenance cost and efforts to maintain many devices. Therefore, the dashboard also draws the devices on a map using various custom markers to allow the users to easily locate the devices. This location information is directly received from the IoT devices in the form of coordinates and stored in the database in the back-end layer. This information is then accessed by the dashboard through the internal API and displayed on the map using the google maps API.

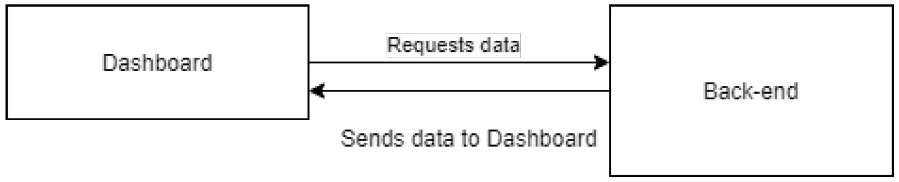

Figure 3.4: Architectural Dataflow of Dashboard Subsystems

Table 3.4: Dashboard Subsystem Interfaces

ID	Description	Inputs	Outputs
#01	User Menu Choices	user clicks link	render respective dash UI
#02	User Logout Link	user clicks logout	Redirect to home-page

CHAPTER 4

BACK-END LAYER

The back-end subsystem is the part of computer application which is not available directly to the users. This sub system is responsible for listening to the data from Azure IoT hub. When an event is triggered, it takes the information from that event and stores it in the database (Cosmos DB). In addition to this, the back end also handles the client-side requests and responds with the data accordingly. The back-end subsystem is further divided into server-side application, database, and Azure IoT.

4.1 Azure IoT

An Azure IoT Hub acts as a central message hub. It facilitates bi-directional communication between the IoT application and the devices.

Table 4.1: Azure IoT Subsystem Interfaces

ID	Description	Inputs	Outputs
#1	Azure IOT HUB	JSON from registered device	Data to the database

4.2 Server-Side

This subsystem is composed of two parts: Listener and Retriever. The Listener listens to events triggered by Azure IoT hub and stores the event information to the Cosmos DB database. The Retriever retrieves required information from the database asked by the client-side.

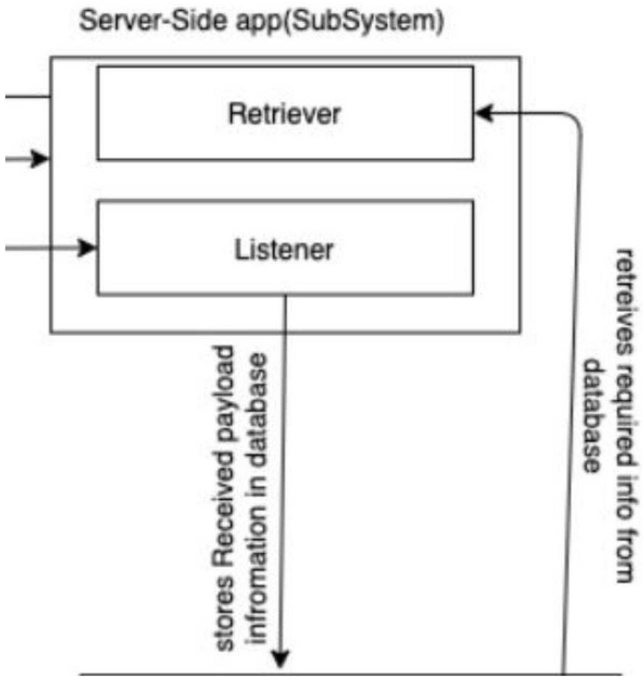
Server-Side app(SubSystem)
Retriever
Listener
stores Received payload infromation in database
retreives required info from database

Figure 4.2: Architectural Dataflow of Server-Side Subsystems

Table 4.2: Server-Side Subsystem Interfaces

ID	Description	Inputs	Outputs
#1	Log in and Authentication	username and password from client	The server returns a Boolean value indicating whether the login was failed or successful.
#2	Device Information	Requests from client for device information.	Response to Client in from of JSON data.
#3	Event Listener	The server will be listening to any event listener triggered by azure IOT. When an event occurs it recieves JSON from the IOT Hub.	Stores the JSON data received in the database.
#4	Retriever	The retriever gets API calls from the client side	Retrieves the required data from the database and sends it to the client side.

4.3 Database

The database is an integral part of the project. The database contains critical information for the efficient function of the system. The information stored in the database is obtained from the server.

Table 4.3: Database Subsystem Interfaces

ID	Description	Inputs	Outputs
#1	Saving Data	Query from the server	Status of the query
#2	Data Retrievability (The data should be extractable using queries)	Data from server	Data in JSON.
#3	Resetting Data (The database resets the data every month.)	No input	No output

CHAPTER 5

IOT LAYER

5.1 Introduction

This layer simply consists of the IoT devices. These devices are registered in the Azure IoT hub. MQTT protocol is used for exchanging information as a JSON string from the device to the back-end server or vice-versa. An error is sent to the back-end service if the device is unable to connect with the Azure IoT Hub.

5.2 IoT Edge

This subsystem will be responsible for sending data from the devices to the Azure IoT hub synchronously.

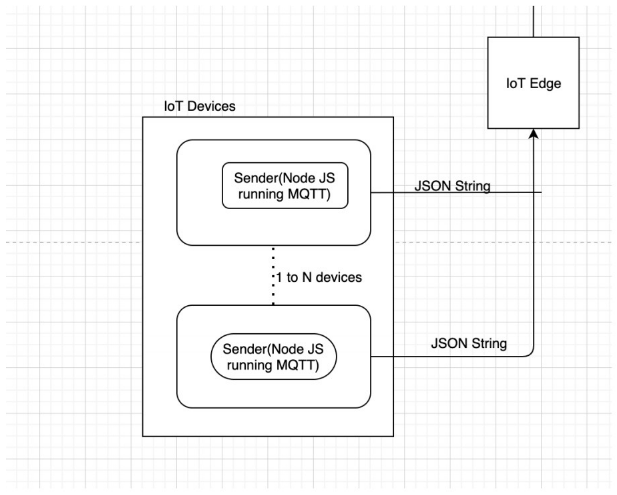

Figure 5.2: Architectural Dataflow of IoT Edge Subsystems

5.3 IoT Devices

Each IoT device has a MQTT protocol for exchanging data with the back-end layer. This protocol is responsible for requesting data from the device and sending the data to the Azure IoT Hub. In addition, a client application was developed to run as a service and report to the estate manager. This client application talks to the back-end layer and is responsible for dispatching actions like device information, status, transactions, firmware update, device failures and errors, and sending alerts to the user. The client application can run on the following operating systems: Windows, Android, Micro OS, and Mac iOS.

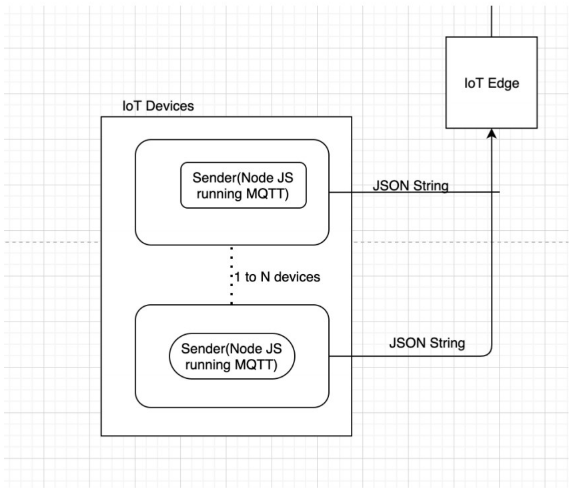

Figure 5.3: Architectural Dataflow of IoT Devices Subsystems

Table 5.3: IoT Devices Subsystem Interfaces

ID	Description	Inputs	Outputs
#1	Azure IoT Hub Registry	Device ID IoT Hub connection String	Device connected
#2	MQTT protocol	Device Information institution ID device ID device serial number firmware version status	JSON string of device in-formation

CHAPTER 6

CONCLUSION

The implementation of the Estate Management system was a success. After several testing, validation, and verification rounds, we were able to build a functioning product as requested by our sponsors (Transact Inc.). All the requirements and deliverables set forth by our sponsors were also met. Working on this project has been very enriching and has been an amazing learning experience. This project allowed me to network and work with professionals in my field. It also helped me gain firsthand experience and will help me in my transition into the industry. I am very thankful to the Honors College, our sponsors, and my professor for this experience.

REFERENCES

"Descriptive Models for Internet of Things." IEEE Xplore, ieeexplore.ieee.org/abstract/document/5564232?casa_token=2r7FTo6rWPIAAAA A%3A jNzE4dyF3G_mi_eSO-IqVUiMkPB60tzO1vOPN-a5_roY_jB5oQHD3HNEiAAwQ6TaSkDX0G1SFA.

"Research on the Architecture of Internet of Things." IEEE Xplore, ieeexplore.ieee.org/abstract/document/5579493?casa_token=gpfhGcXgm7kAAA AA%3 AtJ3gK-bEv-vCWjdxTrNrEJft3qSCe_u5GS8RmYuW_vD_8fjQA-DJXh4ywGggFpHGEOggTsB7rk.

BIOGRAPHICAL INFORMATION

Rhea Pottathuparambil is currently pursuing an Honors Bachelor of Science in Computer Science with a minor in Business Administration. She aspires to become a data scientist.