ELECTRONICS VENDING MACHINE
(CHAREON VENDING)

BY

CHAWAPON THAMRONGVEERAHCART

NATCHANON PATRASETTACHAI

SANCHAI SUN

A PROJECT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN ROBOTICS AND AI

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

FACULTY OF ENINEERING

KING MONGKUT'S INTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Electronics Vending Machine (Chareon Vending)
Student Name	Chawapon Thamrongveerachart Student ID. 62011108 Natchanon Patrasettachai Student ID. 62011159 Sanchai Sun Student ID. 62011238
Degree	Bachelor of Engineering in Robotics and AI

Project Advisor signed: _____

(Dr. Poom Konghuayrob)

Project Co-Advisor signed: _____

(Dr. Sarucha Yanyong)

Project Title	Electronics Vending Machine (Chareon Vending)
Student Name	Chawapon Thamrongveerachart Natchanon Patrasettachai Sanchai Sun
Degree	Bachelor of Engineering in Robotics and AI
Project Advisor	Dr. Poom Konghuayrob
Academic Year	2022

ABSTRACT

In this project, a vending machine that dispenses electronic components to university students and faculty members is designed and created. The goal of the vending machine is to make getting electronic components like transistors, resistors, and capacitors which are frequently required for a variety of projects and experiments easy and accessible. The vending machine will have an easy-to-use interface that enables customers to browse and choose from a wide selection of electronic parts as well as pay using a variety of payment methods, including mobile payment. The device will be built with an inventory management system that tracks the availability of the electronic components and automatically places new orders as needed to keep the device in working order.

The main goals of this project are to create a reliable, user-friendly vending machine that can provide a variety of electronic components to satisfy the needs of the university community. The vending machine will be designed and prototyped, evaluated, and improved to ensure that it satisfies the required performance standards. The project will involve investigating and analyzing the needs and preferences of the target users.

Overall, this project will support the creation of creative and useful solutions that improve the university's capacity for teaching and research while also benefiting the campus community.

ACKONWLEDGMENTS

We would like to take this opportunity to express my sincere gratitude to all the individuals and organizations who have helped me in completing this university paper. Primarily, we would like to extend my heartfelt thanks to my advisor, Dr. Poom Konghuayrob, for his invaluable guidance and support throughout my research. His valuable insights and feedback have been instrumental in shaping my work, and we are deeply grateful for his encouragement and financial support. We also want to thank Mr. Thitipong Thepsit for his technical assistance and help with the Flutter code. His advice and input were extremely helpful when creating the software for my project. We are also grateful to the Duck group for their generous contribution to the vending machine, which was instrumental in the success of my project. Their support has been instrumental in helping me achieve my research goals. We would also like to thank Mr. Chousak Chousangsuntorn from Seagate Technology for providing us with invaluable contacts and information. His support was crucial in enabling me to make valuable connections within the industry and gain a deeper understanding of my research topic. Finally, we would like to acknowledge the Robotics and AI Engineering department for providing me with a place to work and laboratory facilities to conduct my research. The resources and support provided by the department have been invaluable in helping me achieve my research goals. Once again, we would like to express my heartfelt thanks to everyone who has contributed to this paper. Without your support, this project would not have been possible.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

The purpose of this university project is to develop a vending machine that provides students and faculty members with convenient access to electronic components for use in their projects and experiments. The vending machine aims to address the challenges associated with obtaining electronic components, such as limited availability, soaring prices, and long wait times. By providing a vending machine that is stocked with a wide range of components and available 24/7, students and faculty members can save time and effort in acquiring the components they need. Additionally, this project aims to enhance the educational and research capabilities of the university by providing a practical solution that supports the learning and experimentation needs of the university community. The vending machine will contribute to the development of innovative and direct learning experiences that prepare students for real-world applications in their fields of study. Furthermore, this project will provide an opportunity for students to gain valuable skills and knowledge in the areas of product design, prototyping, and testing, as well as in the use of advanced technologies such as inventory management systems, payment processing systems, and user interfaces. Overall, the purpose of this university project is to create a practical solution that supports the educational and research goals of the university, while also providing a valuable service to the university community.

1.1 Problem

Engineering students need to purchase electronics components for their projects, which are often not readily available on campus. Typically, students must travel to off-campus stores or order online, which can be time-consuming and costly. Even when components are available in store, the prices can be inflated, making it difficult for students to complete their projects within budget.

This situation presents a significant challenge to engineering students who require these components to successfully complete their projects. The delays in procuring components can result in project delays, reduced efficiency, and increased expenses for students. Furthermore, the lack of accessibility to electronics components can discourage some students from pursuing engineering as a career.

To address this problem, a vending machine can be developed to provide a convenient and affordable way for students to purchase the necessary electronics components directly on campus. The vending machine will be designed to offer a variety of components commonly used in engineering projects, such as Microprocessor, LEDs, and more. The machine will be placed in a convenient location on campus, allowing students to purchase the components quickly and easily they need without having to travel off-campus or pay inflated prices.

The development of such a vending machine will have several benefits for engineering students. Firstly, it will reduce the time and cost involved in procuring electronics components, allowing students to complete their projects more efficiently. Secondly, it will improve accessibility to components, making it easier for students to pursue their interest in engineering. Thirdly, it will provide a valuable resource for the faculty, as the vending machine can be used to supplement in-class learning and provide students with firsthand experience in selecting and using electronics components.

In conclusion, the development of a vending machine for selling electronics components for engineering students in

1.2 Business model

In this business, you invest in vending machines that are strategically placed in high-traffic areas such as office buildings, schools, shopping malls, hospitals, and recreational areas. These machines are stocked with a range of products that cater to the needs and preferences of your target market.

1. Passive Income: Once you have installed and stocked your vending machines, they can generate revenue around the clock, even when you're not present. Customers can make purchases at any time, providing you with a passive income stream.
2. Low Overhead Costs: Vending machine businesses typically have low overhead costs compared to other types of businesses. You don't need a brick-and-mortar store, and you can operate with minimal staff. This means reduced expenses on rent, utilities, and employee salaries.
3. Flexibility and Scalability: Vending machines offer flexibility in terms of location and product selection. You can test different locations and products to find the most profitable combinations. Additionally, as your business grows, you can easily scale by adding more machines or expanding into new markets.

1.3 Timeline

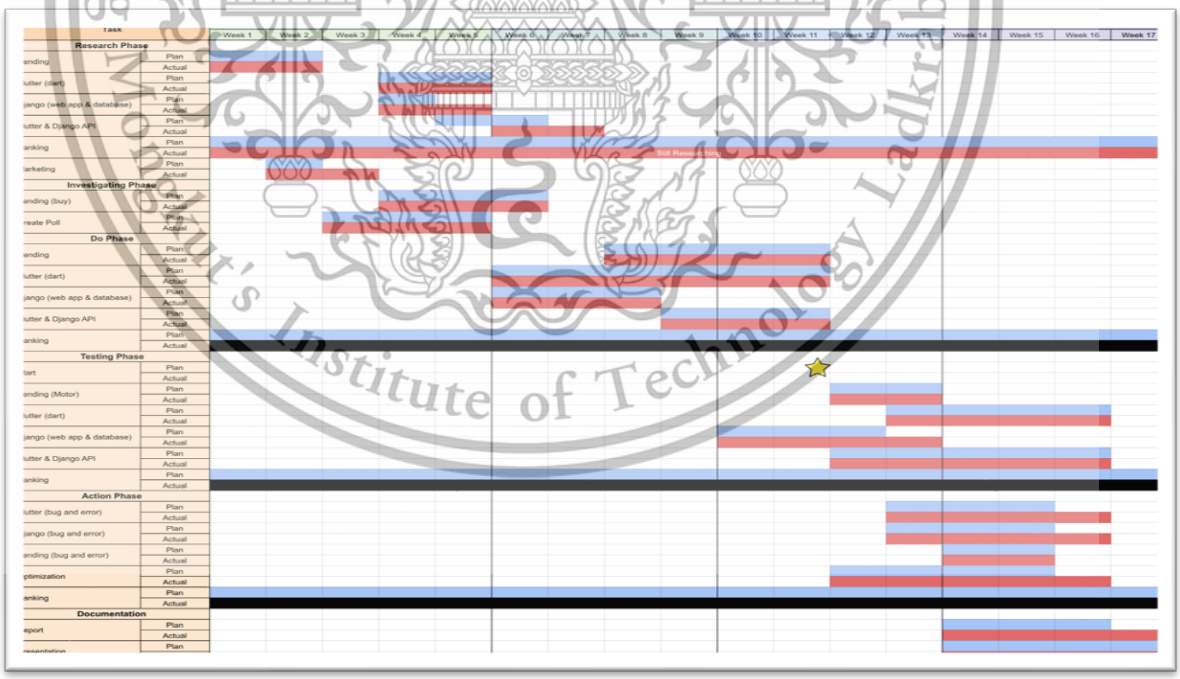


Figure 0 Project Timeline

CHAPTER 2

SUPPORTED THEORY AND BUSINESS MODEL

2.1 FRONTEND

The "front-end" of a website or web application is the area of a website or web application that a user interacts with directly. It includes all the components that a user sees and uses, including the text, images, layout, design, and interactive elements. In front-end development, these elements are created and manipulated using programming languages like HTML, CSS, and JavaScript to make them responsive, aesthetically pleasing, and user-friendly. Additionally, front-end developers make sure that the website or web application loads quickly and effectively and is compatible with a variety of devices and browsers. Creating interactive features, optimizing the website's performance, designing the user interface, and evaluating the website to make sure it works as intended are some frequent tasks in front-end development.

2.1.1 UX/UI

Two key terms in the field of design are UX and UI, particularly in relation to web design and software development. User Experience, or UX, is the term used to describe the general interaction that a user has with a website, app, or other digital product. This covers everything from the product's usability and ease of use to the feelings it evokes in the user. UI, or user interface, refers to the particular user interface that a user interacts with when using a digital product. This includes all the graphical components—buttons, icons, menus, and other controls—that the user uses to interact with the product and navigate through it. In actuality, UX and UI are intertwined and frequently overlap. Understanding user needs and behavior is a key component of UX design, which focuses on creating products that satisfy those needs. The goal of UI design is to create simple, aesthetically pleasing interfaces that are easy for users to use and navigate. Successful digital

products that satisfy user needs and deliver a pleasant experience depend on effective UX/UI design.

2.1.2 Flutter

Figure 1 Flutter logo

Google developed the open-source Flutter UI (user interface) toolkit for creating natively compiled applications for desktop, web, and mobile platforms. It enables programmers to create high-performance, aesthetically pleasing applications for numerous platforms, including Android, iOS, Windows, macOS, and Linux, using a single codebase. Dart, a programming language used by Flutter, was created by Google as well. Dart is an object-oriented, garbage-collected language that can be translated into JavaScript for web applications and machine code for performance. Widgets serve as the foundation for Flutter's architecture and are used to build complex and dynamic user interfaces. Additionally, it provides a vast array of programmable widgets and tools for creating user interfaces that are both attractive and responsive. Additionally, Flutter has a "hot reload" feature that enables developers to instantly see changes they make to their code, speeding up and streamlining the development process. Overall, Flutter is a strong and adaptable toolkit that provides a cutting-edge and streamlined method for developing cross-platform apps.

2.1.3 Android Studio

Figure 2 Android Studio logo

An integrated development environment (IDE) for creating Android applications is called Android Studio. It was created by Google and serves as the official IDE for creating Android applications. With the help of Android Studio, developers can efficiently build, test, and deploy Android apps. Tools for creating user interfaces, managing dependencies, testing, and debugging code are available in Android Studio. Additionally, it provides an emulator that can be used to run and test apps without a physical device. The creation and packaging of apps are automated by Android Studio using the Gradle build system, making it simpler for developers to manage and update their projects. Overall, the Android development community uses Android Studio extensively because it is a strong tool for developers to produce high-quality Android applications.

2.1.4 Visual Studio Code

Microsoft created the free and open-source code editor known as VS Code (short for Visual Studio Code). Developers frequently use it to code, debug, and create diverse types of software applications. Syntax highlighting, code completion, debugging, version control integration, and support for extensions are just a few of the features that VS Code provides. It is a popular

option among developers working with various platforms and technologies because it is highly customizable and supports a variety of programming languages. VS Code can be downloaded from the Microsoft website and is compatible with Windows, macOS, and Linux operating systems.

Figure 3 Vscode logo

2.1.5 Dart

Google created the open-source programming language Dart. It was first introduced in 2011 and is employed in the development of desktop, mobile, and web applications. Dart is a class-based, object-oriented language that can support interfaces and mixins. Dart is made to be quick, scalable, and easy to maintain. It has both an ahead-of-time (AOT) compiler that creates optimized native code for production deployment and a just-in-time (JIT) compiler that enables developers to quickly iterate on their code during development. Dart is simple for developers to learn and use because its syntax is similar to other well-known programming languages like C++, Java, and JavaScript. Dart also has a robust type of system, which helps programmers find errors earlier in the development cycle and produce more dependable code. Dart can be used to create applications that run on the client and the server. It includes a collection of core libraries that offer basic features like concurrency, I/O, and collections. Dart also has a robust ecosystem of third-party packages and frameworks, including the well-known Flutter framework for creating mobile and web applications.

Figure 4 Example of Dart

2.1.6 Android Operating System

Figure 5 Android logo

An operating system (OS) for mobile devices like smartphones, tablets, and smartwatches is called Android. Based on the Linux kernel and other open-source software, Google created it. For mobile devices, Android offers

a user interface, programs, and middleware. Additionally, it supports a variety of hardware elements, including GPS receivers, cameras, and sensors. With so many devices using it, Android has grown to be among the most well-liked operating systems in the world. Its open-source status, which makes it simple for developers to make and modify applications for the platform, can be credited with its popularity.

2.1.7 Dart Package

A package in the Dart programming language is a group of resources and lines of code created to address a particular issue or offer a particular functionality. Libraries, executables, and assets like fonts and images can all be found in a Dart package. Packages may be shared and found by other users by publishing them to the Dart Package Manager. The pub command-line tool allows developers to manage packages, such as installing, upgrading, and publishing them. Dart packages are frequently used to reuse previously written code or to share code among several projects. Code within a project can also benefit from package organization and modularization, which makes it simpler to manage and maintain. Developers who use packages have more control over dependencies and compatibility issues because they can specify the version of the package they want to use. Packages can also come with examples and documentation to help developers learn how to use them.

2.2 BACKEND

A web application's backend, also spelled "back-end" or "back end," refers to the server-side of a software program. The server, database, APIs, and other elements required for the operation of the application are all included in the code and technologies that power it from the background. The backend is in charge of handling user requests, processing data, and controlling the business logic of the application. This indicates that the backend is in charge of getting data out of a database, storing it there, running calculations on it, and giving the necessary data to the front-end or client-side of the application. Ruby on Rails, Node.js with Express, Python with Django

or Flask, and PHP with Laravel or Symfony are a few popular backend programming languages and frameworks. The code that drives these technologies is written by backend developers, and they also make sure the application is secure, scalable, and effective.

2.2.1 Django

A high-level Python web framework known as Django uses the Model-View-Controller (MVC) architectural design pattern. It was created to make it easier for developers to create intricate, database-driven websites quickly and effectively. The Django Software Foundation is in charge of maintaining Django, which is open-source software. Many features are already included in Django, such as an Object-Relational Mapping (ORM) layer that enables programmers to communicate with databases using Python code rather than SQL queries. Additionally, Django offers a robust templating engine that makes it simple to create dynamic HTML pages. Automatic URL routing, user authentication, and an admin interface that is customizable are additional features. Django is a well-liked choice among programmers with experience in Python because it is built on the platform. A wide variety of websites, including content management systems, social networks, e-commerce sites, and more, have been built using Django.

2.2.2 Python

First released in 1991, Python is a well-known high-level, interpreted programming language. With a syntax that prioritizes readability and simplicity, it was created to be simple to read and write. Python is a flexible language that can be used for a variety of tasks, such as developing websites, analyzing data, performing scientific computations, developing artificial intelligence, and more. Python's ease of use and elevated level of productivity are two of its key benefits. Python code tends to be more concise and

condensed than code written in other programming languages, which can result in quicker development times and fewer bugs. Additionally, Python has a sizable and vibrant developer community that contributes to a broad range of libraries and tools, making it simple to locate solutions to typical programming issues. The adaptability and versatility of Python also contribute to its popularity. It works on many different platforms, including Windows, Mac OS, Linux, and mobile devices, and supports a variety of programming paradigms, including object-oriented, functional, and procedural programming. In addition, Python has a sizable and expanding collection of libraries and frameworks, including Django, Flask, NumPy, Pandas, TensorFlow, and many others, which make a variety of programming tasks simple to complete.

Figure 6 Python and Django logo

2.2.3 SQLite

A well-liked software library called SQLite offers a relational database management system (RDBMS). It is a transactional SQL database engine that is open source, serverless, self-contained, and used in many different types of software and platforms. Because it is integrated directly into the application that uses it, SQLite differs from more established client-server database management systems like MySQL and PostgreSQL. This indicates that there is no need for a separate server process or installation because the SQLite

database is built into the application itself. In addition to offering many of the features found in other RDBMSs, such as tables, indexes, triggers, and views, SQLite supports a sizable subset of the SQL92 and SQL99 standards. Additional programming languages supported by it include C, C++, Java, Python, Ruby, and others. SQLite is frequently used in a variety of applications, including mobile apps, web browsers, operating systems, embedded systems, and more because of its small footprint, portability, reliability, and ease of use.

Figure 7 SQLite logo

2.2.4 Java

Java is a popular high-level, object-oriented programming language for creating a range of applications, including desktop, web, and mobile ones. James Gosling created it at Sun Microsystems, which is now owned by Oracle Corporation, and it was first made available in 1995. Platform independence, which allows Java code to be written once and run on multiple platforms without needing to be recompiled, is one of the language's key features. Because Java programs are converted into an intermediate bytecode that can be executed on any platform that has a Java Virtual Machine (JVM) installed,

this is made possible. Because the bytecode is executed by the JVM, Java is a portable language. Java is renowned for its security, scalability, and robustness. It has a sizable and vibrant developer community that has helped shape the creation of numerous well-known Java frameworks and libraries, including Spring, Hibernate, and Apache Struts. Enterprise software development, Android mobile app development, web development, game development, and many other fields and applications use Java. It is frequently employed in applications involving artificial intelligence, machine learning, and big data.

Figure 8 Java logo

2.3 HARDWARE

2.3.1 Vending Machine

Figure 9 Vending Machine

A vending machine is an automated device that, in exchange for payment, dispenses a variety of goods, including snacks, drinks, cigarettes, lottery tickets, and even electronic devices. Coins, bills, credit cards, and mobile payments are frequently accepted at vending machines as payment types. A variety of products are typically displayed on the machine's front panel or by using a touchscreen interface. Customers choose the item they want to buy before paying for it. The product is then released from the machine, typically into a compartment or chute at the base of the device. Vending machines are frequently found in public places like malls, airports, schools, and offices where they offer people convenient and accessible ways to make quick and simple purchases of goods.

2.3.2 Motor Driver

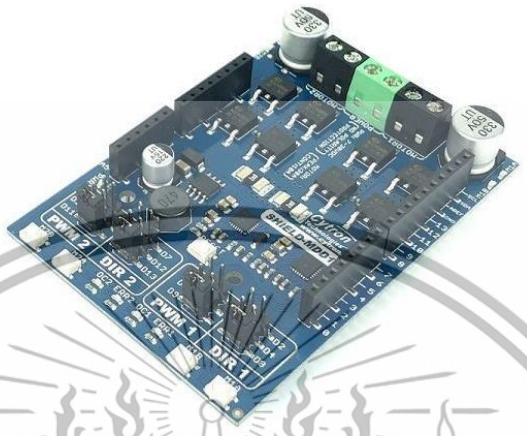


Figure 10 Example of Motor Driver

An electric motor's rotation, speed, and direction are managed by a motor driver, an electronic device. It is an essential part of many different electronic machines and gadgets that use electric motors, including robots, drones, electric cars, machinery for the industrial sector, and home appliances. A motor driver is typically made up of a microcontroller or specialized circuit that receives signals from a controller, like a computer or joystick, and converts them into the proper signals to control the motion of the motor. To control the motor, the driver may employ a number of techniques, including pulse width modulation (PWM), an h-bridge configuration, or stepper motor control. Motor drivers play a crucial role in ensuring that a motor operates correctly and efficiently by supplying it with the necessary current and voltage. Additionally, they can shield the motor from damage brought on by excessive voltage, current, or heat. A motor driver may have various features and specifications, such as input/output interfaces, control algorithms, voltage, and current ratings, depending on the application.

2.3.3 Signal Codec

A technology called a signal codec, which stands for "signal compression/decompression," is used to transmit and store digital signals like audio, video, and data more efficiently. In order to reduce the amount of data required to transmit or store the signal while maintaining an acceptable level of quality, the codec (coder-decoder) compresses the signal into a smaller size. In order for the codec to function, the original signal must be examined, and any redundant or unnecessary information must be removed, such as sounds that are too faint for the human ear to detect or pixels in an image that are too similar to one another. The signal can then be compressed to use less bandwidth or storage space for transmission or storage. The codec decompresses the signal to return it to its original state when it is retrieved or received. Although the decompressed signal might not be an exact replica of the original, it should be close enough for human senses to recognize it as high-quality.

2.3.3.1 Encoder

A signal encoder uses technology to turn analog signals like sound, video, or data into digital signals that can be used by digital devices for processing, storing, and transmitting. By measuring the analog signal at regular intervals and converting those measurements into a stream of binary code, the encoder analyzes the analog signal and transforms it into a digital form. For instance, in the case of audio, the analog sound waves are transformed into a number of digital samples that represent the wave's amplitude at particular points in time. Then, digital devices like computers and digital audio players can store, transmit, or process this digital representation. The effective transmission and storage of digital signals depend on the process of signal encoding. Signal codecs can be used to compress a signal after it has been encoded in order to reduce its size and save bandwidth and storage space. In conclusion, signal encoding involves transforming analog signals into digital form, whereas signal codecs are employed to reduce the size of digital signals for transmission and storage.

2.3.3.2 Decoder

The compressed or encoded digital signal is examined by a signal decoder, which then transforms it into an analog signal that can be played back on speakers, displayed on a screen, or otherwise processed by analog devices. For instance, a digital audio player can decode a compressed and encoded digital audio signal to produce an analog audio signal that can be heard through speakers or headphones. By using algorithms and methods particular to the encoding format, the encoding process is reversed during the decoding process. Although the decoded signal might not be an exact replica of the analog signal's original, it should be close enough for human senses to recognize it as high-quality. In conclusion, a signal decoder is a piece of equipment that transforms digital signals that have been encoded into their analog counterparts.

2.3.4 Android Tablet

Figure 11 Example of Android Tablet

A tablet computer running the Google-created Android operating system is known as an Android tablet. Android tablets perform similarly to other tablets like the Apple iPad and the Microsoft Surface, but they run the Android OS as opposed to the iOS or Windows operating systems. The screen sizes of Android tablets range from 7 inches to 18 inches or more. Users can access apps, browse the web, read e-books, watch videos, and carry out other

functions akin to those of a traditional computer on these typically touch-screen devices. Google Play Store, which enables users to download and install a variety of apps, games, and other digital content, is available to Android tablet users. Additionally, they frequently include Wi-Fi or cellular connectivity that enables users to use online services and access the internet wherever there is a wireless network or cellular data coverage. Some Android tablets also include extra features like keyboard covers, stylus support, and other accessories that can improve their functionality and suitability for particular tasks. A flexible and adaptable platform for mobile computing, entertainment, and productivity is provided by Android tablets overall.

2.4 BANKING API

A banking API (Application Programming Interface) is a set of protocols and tools that let outside programmers' access and communicate with a bank's software or application, frequently in order to create new financial services or applications. Without forcing the user to use a different platform or bank's interface, developers can incorporate banking features like account information, payment processing, money transfers, and other financial services into their own applications, platforms, or software by using banking APIs. For instance, a developer might use a banking API to create a program that eliminates the need for users to visit the bank's website or app in order to view account balances, transfer money, or make payments. Banks and other financial institutions frequently provide banking APIs as a way to enhance the user experience and offer customers value-added services. Additionally, they enable banks to work with outside developers to produce cutting-edge financial goods and services. To protect the security and privacy of customers' financial information, banking API usage is subject to security and regulatory compliance requirements.

2.5 COMMUNICATIONS

2.5.1 Rest API

A set of guidelines for use when developing web services are defined by the REST (Representational State Transfer) API. HTTP requests are used by RESTful APIs to GET, POST, PUT, and DELETE data from the internet. Because a RESTful API is stateless by design, the server does not keep track of the client's previous interactions with it. Instead, each request made by the client includes all the data required for the server to process it. A RESTful API uses HTTP methods to define operations on resources, which are identified by URIs (Uniform Resource Identifiers). For instance, a POST request may result in the creation of a new resource while a GET request to a particular URI might return a representation of a resource. JSON (JavaScript Object Notation) or XML (Extensible Markup Language) are two common formats used by RESTful APIs to represent data. The body of an HTTP request or response can contain this data for transmission between the client and the server. Utilizing a RESTful API has benefits such as simplicity, scalability, and flexibility. It enables simple system integration and can be used to create platform-independent web applications. Additionally, RESTful APIs are simple for developers to understand and use because they employ industry-standard HTTP protocols and data formats.

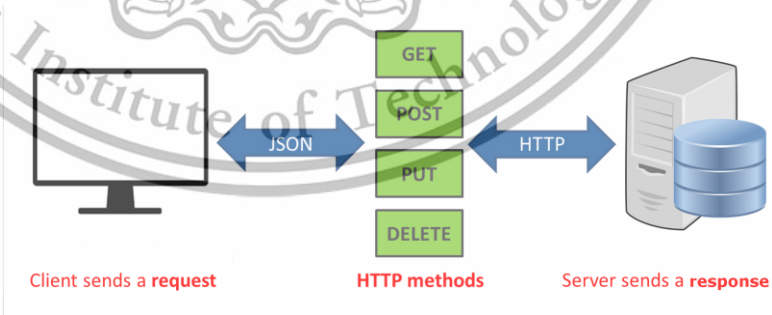


Figure 12 Rest API

2.5.2 TTL Communications

Transistor-Transistor Logic, or TTL, is a type of digital communication that uses voltage levels to represent binary signals. TTL stands for transistor-transistor logic. A common type of logic family used in digital circuits, including microcontrollers and other electronic devices, is called TTL. The binary signal is represented in TTL communication by two voltage levels: a high voltage level, typically 5V, that denotes a logic "1," and a low voltage level, typically 0V, that denotes a logic "0." TTL communication transmits data over a single wire using pulses of voltage, one for each bit. Short-range communications, such as those between components on a printed circuit board (PCB) or between a microcontroller and other peripheral devices, frequently use TTL communication. It is fast and reliable, only requiring a few components, and is relatively easy to implement. TTL communication has drawbacks, though, such as a constrained range because of its sensitivity to noise and interference. Additionally, since the voltage signal tends to deteriorate with distance, it is not appropriate for long-distance communication. TTL communication, which is a widely adopted digital communication standard, offers a straightforward and dependable way to send binary data over short distances.

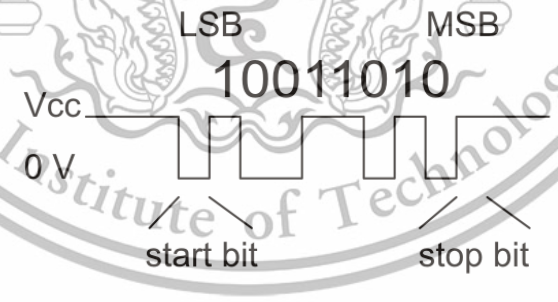


Figure 13 TTL Communications

2.5.3 RS232 Communications

The serial communication protocol known as RS232 (Recommended Standard 232) is used to send data between two devices using a point-to-point connection. The Electronic Industries Association (EIA) first introduced it in 1962, and since then, it has been widely used in many different applications, including computers, modems, printers, and other electronic devices. The "TX" (transmit) and "RX" (receive) lines, which are used in RS232 communication, are used for both sending and receiving data. The sender and receiver must beforehand agree on the data transfer rate and other settings, such as parity and stop bits, as the communication is asynchronous, and data is transmitted without the use of a clock signal. Due to signal degradation, RS232 has a restricted range of about 15 meters and is typically not appropriate for long-distance communication. In industrial and laboratory settings, where short-range communication is adequate and dependability is crucial, it is still frequently used. Newer serial communication standards like USB and Ethernet, which provide faster data transfer rates and greater distance capabilities, have largely replaced RS232 in recent years.

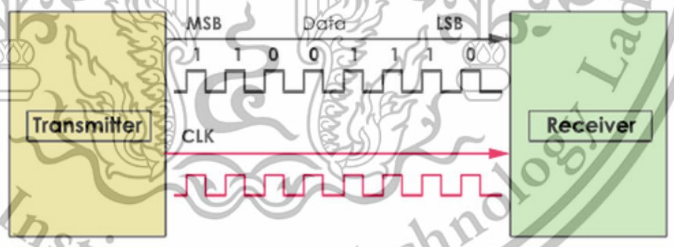


Figure 14 RS232 Communications

2.5.4 RS485 Communications

Applications for industrial and building automation use the serial communication standard RS485. It uses differential signaling to transmit data over great distances with high immunity to noise and interference. As an asynchronous communication protocol, it does not transmit data with a clock signal. Data is transmitted using a two-wire bus in RS485 communication, with one wire serving as the signal wire (A) and the other as the complementary signal wire (B). The transmission of data is represented by the difference in voltage between these two wires, with positive voltage standing for logic 1, and negative voltage for logic 0. As a result, the RS485 protocol is able to achieve high noise immunity and long-distance communication because noise and interference have no effect on the voltage difference between the two wires. With the help of RS485, multiple devices can be connected to the same bus and communicate with one another using specific addresses thanks to the technology's support for multi-point communication. This qualifies RS485 for use in systems like industrial control systems and building automation systems, which require multiple devices to communicate with one another.

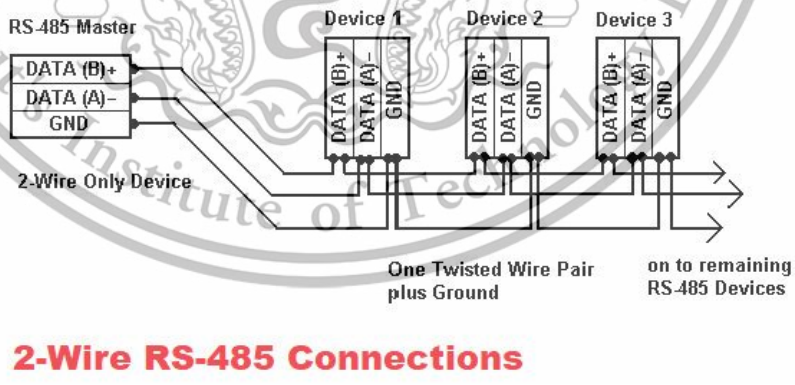


Figure 15 RS485 Communications

2.6 BUSINESS MODEL

An organization's value creation, delivery, and capture processes are outlined in its business model. It outlines the essential components of a business's strategy, such as its target market segments, value proposition, revenue sources, cost structure, and distribution channels. A business model typically outlines a company's costs and revenue sources in order to show how the company expects to make money. It might also include information on how the business intends to set itself apart from rivals and forge a lasting competitive advantage. There are numerous varieties of business models, such as:

1. Direct sales model: The business uses its own sales channels, like a website or physical store, to sell goods or services to customers directly.
2. Subscription-based business model: Customers pay a recurring fee to access a company's goods or services.
3. The business provides a free, basic version of its product or service, but charges for additional features or usage.
4. Model of the marketplace: The business offers a platform that connects buyers and sellers and charges a transaction fee.
5. Franchise: The business licenses its name and way of doing things to franchisees who run their own stores.
6. Advertising model: The business makes money by renting out space for advertisements on its website or platform.

In general, a company's business model is an important component of its strategy because it directs decision-making and ensures long-term success.

CHATER 3

METHODOLOGY AND IMPLEMENTATION

3.1 FRONTEND

First, we designed the UI in Figma website for planning our application direction and request some feedback from sample users for the implementation. We did an application on the React.js platform then we changed to Windows application, after discussing it with our advisor. We have decided to use an Android tablet, so Flutter is the way.

3.1.1 Android Studio

In Android Studio, we used it for debugging and testing our developed applications for the Vending machine. It's a tool that will simulate an android device running on our computer. In it, we can choose the version of android and the version of android device. In this project, I chose an Android tablet with 10.5-inch screen, and android version 12 Tiramisu.

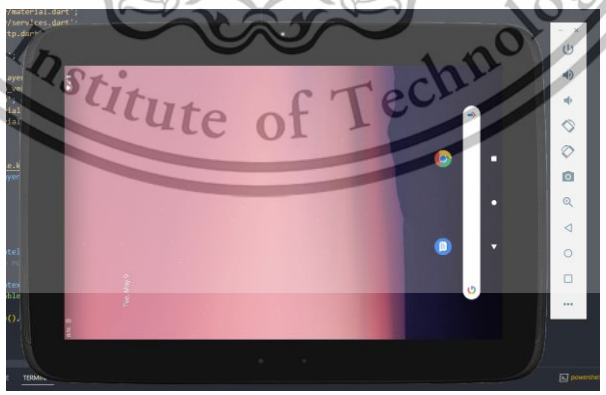


Figure 16 Android Studio Emulation

3.1.2 UI

Designed from Figma first.

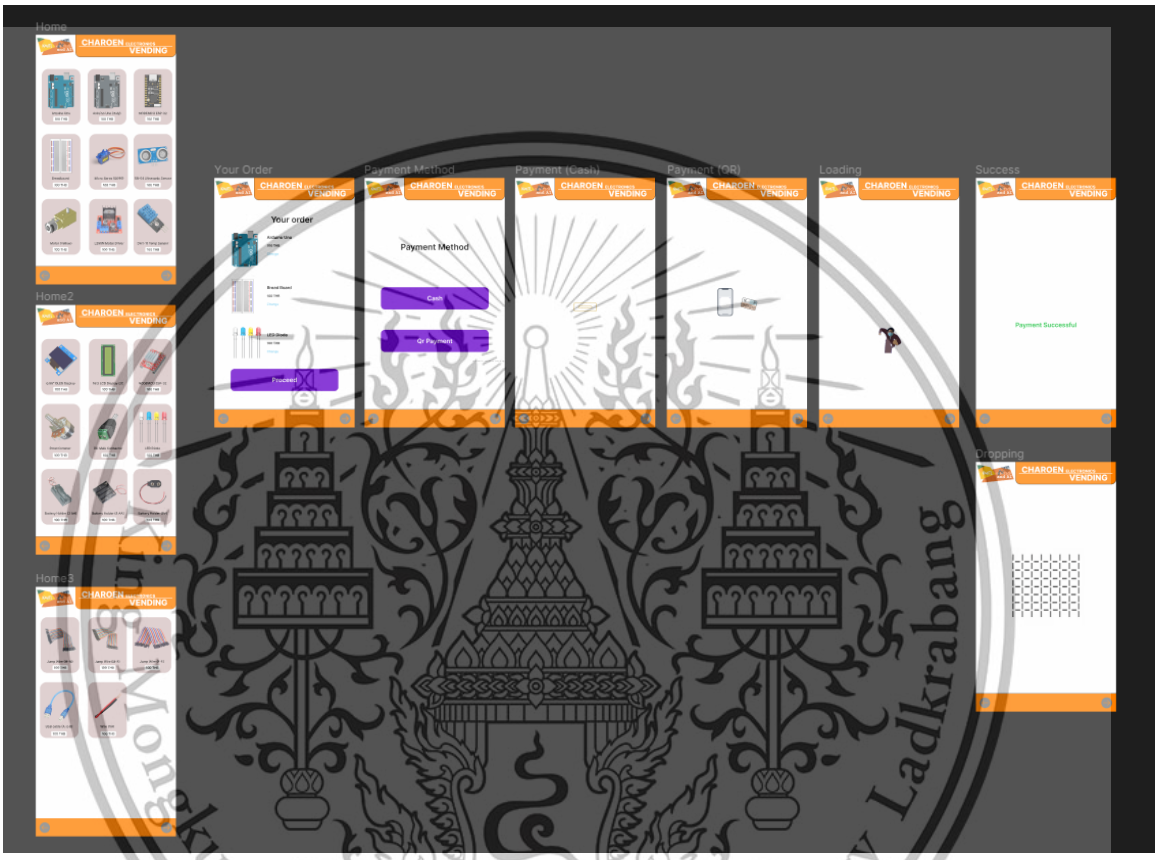


Figure 17 UI Design

3.1.3 Flutter

In this part of the project. We started to develop an application from scratch. First, we have to learn Dart language and learn about the features of Flutter to implement in our application. Thanks to YouTube Channel “KongRuksiam” for the lesson of everything about how to develop an application in Flutter. In the application of our project, we are separated into 5 main parts.

3.1.3.1 main.dart

In main.dart, we included everything from other parts of the application. Start with the feature for playing the background music, all the parameters that we use in our code, home screen and other screen of an application. Features such as hiding the top and bottom bar of an android device.

Figure 18 part of main.dart

As you can see in Figure 18, the main function of the application is to launch the first screen of an application and disable system UI, which means to disable the top and bottom screen navigation bar.

3.1.3.2 jsonreceive.dart

This part of the application is to decode the data from the backend parts, which is transferring in format of json format. This part will receive the data from backend and transform the json format into string, integers and so on.

Figure 19 part of jsonreceive.dart

From the code above, it will read the id, name, amount and qr from the product that we put inside the Vending Machine.

3.1.3.3 receive.dart

This part of the application has only one function, which is to fetch the data from the backend. All the data will be fetched every time that tablet has been clicked on. After we fetch the data, we keep the value in each variable for further uses in our application such as, checking the stock amount, checking the id of the product.

3.1.3.4 menuwidget.dart

This part of the application has many functions. First, it includes all the popups such as out of stock, paid, and qr payment. This code includes the part that is used to communicate with the vending machine also, making the decision about out-of-stock product.

```
Future<bool> fetchStockAmountqr(int id) async {
  bool state = false;
  try {
    var url_ = Uri.parse('https://jamie.krai.io/api/ind-stock/${id}/');
    var response_ = await http.get(url_);
    _jsondata = jsondataFromJson(response_.body);
    if (_jsondata?.qr == 1) {
      state = true;
      print("PAID");
    } ls {
      state = fals ;
      print("FAIL");
    }
  } catch (e) {
    print(e);
  }
  return state;
}
```

Figure 20 part of decision making.

```dart
void _getPorts() async {
  _ports = [];
  List<UsbDevice> devices = await UsbSerial.listDevices();
  if (!devices.contains(_device)) {
    _connectTo();
    print("Conected");
  }
}
```

Figure 21 getting the connection.

3.1.3.5 senddata.dart

This code only has a function to reduce the stock amount in database by sending command through rest api to patch to update the stock in database.

```dart
Future<bool> sendData(int a) async {
  http.Response apiResponse = await http.patch(Uri.parse(url));
  var apiData = jsonDecode(apiResponse.body);
  bool state = await fetchStockAmount(a);
  print(state);
```

Figure 22 patch data

3.1.4 Flutter Package

For the Flutter package we used in this application included

3.1.4.1 cupertino_icons

This package is used for adding icons into our application, this package provides easy changes to icon, auto re-size an icon image.

3.1.4.2 http

This package is used for communicating with rest api by adding PATCH, GET, OUT function into our application.

3.1.4.3 js

This package is used for encoding and decoding the data for communicating with database through rest api, by sending and receiving data in json format.

3.1.4.4 audioplayers

This package is used for playing any sound or music in mp3 format inn our application. This function works when we are on the menu page to play loop background music.

3.1.4.5 usb_serial

This package is used for communicating with our motor driver board to control the motor in the vending machine.

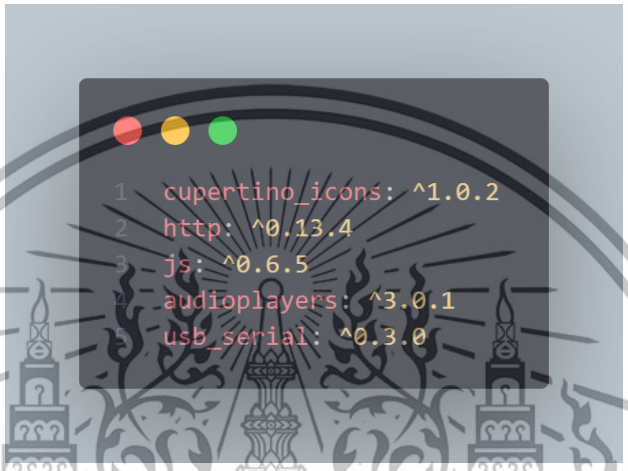


Figure 23 packages

3.2 BACKEND

For the backend, we will mostly utilize the Django Framework for creating a website, making database for the vending machine, and communication via REST Framework.

3.2.1 Django

First, we start by creating the Django project for making the base website which will also function as a website where customers can come to check the stock of the product within the vending machine.

There will be two main parts for creating the Django website which are the Frontend and the Backend. The frontend will be managed by JavaScript code and HTML while the backend will be managed mostly by Python.

3.2.1.1 Django's Backend

The backend part for Django will manage most of the script and command that will be running in the frontend. But before we are going to start coding the backend, we need to create the database by migrating the models.py within the Django project folder.

```
1   from django.db import models
2
3   # Create your models here.
4   class Stock(models.Model):
5       name = models.CharField(max_length=50)
6       amount = models.IntegerField()
7       status = models.IntegerField()
8       time = models.DateTimeField(auto_now=True)
9
10      def __str__(self):
11          return self.name + " : " +str(self.name)
```

Figure 24 models.py of Django

We will name the class of our model as Stock which will include 4 fields. Those fields are [name] for the item name, [amount] for the amount of each item in the stock, [status] which will be used during the payment mockup and [time] which was originally intended to be use with the real-time chart, but it isn't necessary anymore. Another field that we need is [id] but it was auto generated by Django. After we include all the fields that we need, we will migrate those fields into the Django database.

After we migrate the data, we will start coding the backend. The 4 main pages that needed to be included for the website are Home, Stock, Chart and Real-Time Chart. The backend of the code will be written in the views.py

```
from django.shortcuts import render,redirect
from django.http import HttpResponse, JsonResponse
from myapp.models import Stock

from django.contrib import messages

from rest_framework.response import Response
from rest_framework.renderers import JSONRenderer
from rest_framework.parsers import JSONParser
from rest_framework.decorators import api_view
from rest_framework import status, generics
from .serializers import StockSerializer

import json

import matplotlib.pyplot as plt
import io
from django.shortcuts import render
import time
import random
```

Figure 25 Libraries for views.py

First, we will start by writing the backend for the index. The index will act as a template for all the pages present on this website which will include the querry set for the database and include function for using time.

```python
def index(request):
    # all_stock = Stock.objects.all()
    labels = []
    data = []
    t = time.localtime()
    current = time.strftime(" %H : %M : %S ", t)

    queryset = Stock.objects.order_by('-amount')[:25]
    for stock in queryset:
        labels.append(stock.name)
        data.append(stock.amount)

    return render(request,"index.html",{
        'labels': labels,
        'data': data,
        'current': current,

    })
```

Figure 26 Index for views.py

Next is going to be the backend for the stock pages which will mainly be defining the variable of the data in JavaScript form for the frontend to use.

```python
def stock(request):
    all_stock = Stock.objects.all()
    return render(request,"stock.html",{"all_stock":all_stock})
```

Figure 27 Stock for views.py

Then we start writing the backend of the chart which will involve writing the amount of stock to be corresponding to the name and id of the item from the database. The type of chart will be Bar-Chart, so we haven't made it a real-time chart yet.

```python
def chart(request):
    labels = []
    data = []
    t = time.localtime()
    current = time.strftime(" %H : %M : %S ", t)

    queryset = Stock.objects.order_by('id')[:25]
    for stock in queryset:
        labels.append(stock.name)
        data.append(stock.amount)

    return render(request,"chart.html",{
        'labels': labels,
        'data': data,
        'current': current,
    })
```

Figure 28 Chart for views.py

And for Real-Time chart, we will need to utilize REST Framework for consistently getting the data of the amount of stock so the data of the chart can run in real-time. The type of chart will be a Bar-Graph instead of Bar-Chart.

```python
def rtchart(request):
    queryset = Stock.objects.order_by('id')[:25]

    # Get the latest amounts for all stocks
    latest_data = {stock.id: stock.amount for stock in queryset}

    # Render the template with the initial data
    datasets = []
    for id, stock in enumerate(queryset, start=1):
        r = random.randint(0, 255)
        g = random.randint(0, 255)
        b = random.randint(0, 255)
        color = f'rgba({r},{g},{b},3)'

        datasets.append({
            'label': f'{id}: {stock.name}',
            'data': [{'x': stock.name, 'y': latest_data[stock.id]}],
            'borderColor': color,
            'borderWidth': 3,
        })

    current = time.strftime("%H:%M:%S", time.localtime())

    return render(request, 'rtchart.html', {
        'datasets': datasets,
        'current': current,
    })
```

Figure 29 Real-Time Chart for views.py

3.2.1.2 Django's Frontend

The frontend of Django will be the appearance of the website. The code that we will be using in this part is JavaScript with HTML format. Each the frontend of the website will be corresponding to the backend, which means that the number of pages will be corresponding as previous section.

We start by creating the template for the overall appearance of Django which we will be using bootstraps template to make the website looks more appealing and user friendly. Then we use the template pages to create other pages for the website.

```
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="s
    <link href="http://cdn.pydata.org/bokeh/release/bokeh-2.3.2.min.css" rel="stylesheet" type=
    <link href="http://cdn.pydata.org/bokeh/release/bokeh-widgets-2.3.2.min.css" rel="styleshee


    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>
    <script src="https://cdn.amcharts.com/lib/5/index.js"></script>
    <script src="https://cdn.amcharts.com/lib/5/xy.js"></script>
    <script src="https://cdn.amcharts.com/lib/5/themes/Animated.js"></script>

    {% block title %}
    {% endblock %}

</head>

<body style="background-color: rgb(63, 60, 53);">

    <h1 class="text-center" style="color:rgb(255, 119, 29); font-family:Impact, Haettenschwei
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
          <a class="navbar-brand" href="/">KMITL</a>
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
              <li class="nav-item">
                <a class="nav-link" aria-current="page" href="/">Home</a>
              </li>
              <li class="nav-item">
```

Figure 30 base.html

Figure 31 Website Template

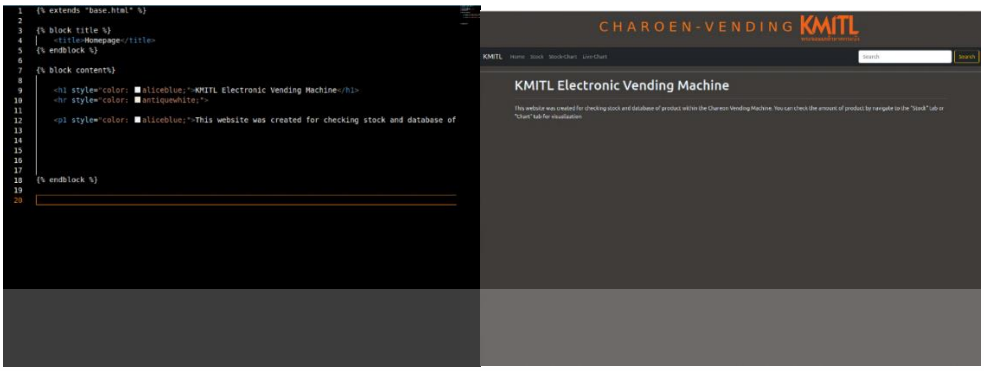


Figure 32 Homepage

Figure 33 Stock Page

Figure 34 Chart Page

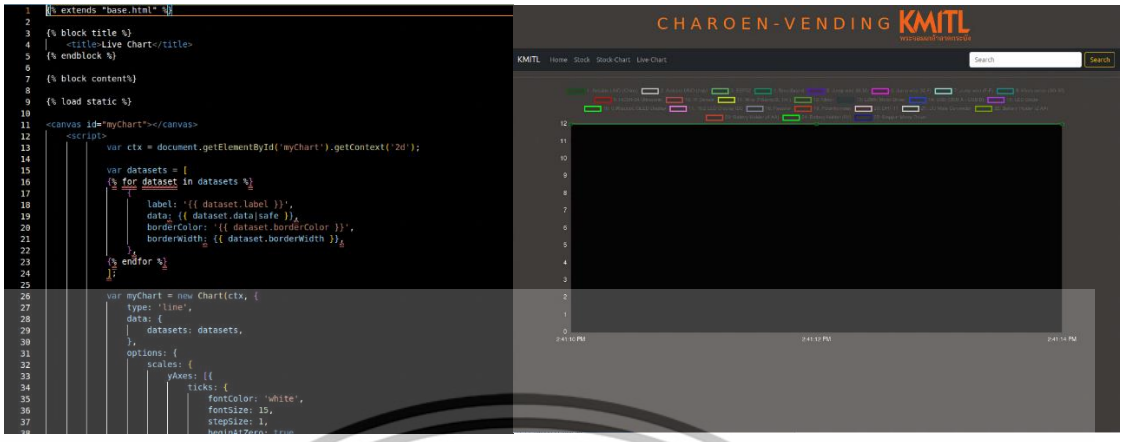


Figure 35 Real-Time Chart Page

3.2.2 Database

When we first create the database through Django's model.py, the data will be set to a default value. We can change those values into the desired value by using a program called DB Browser for SQLite.

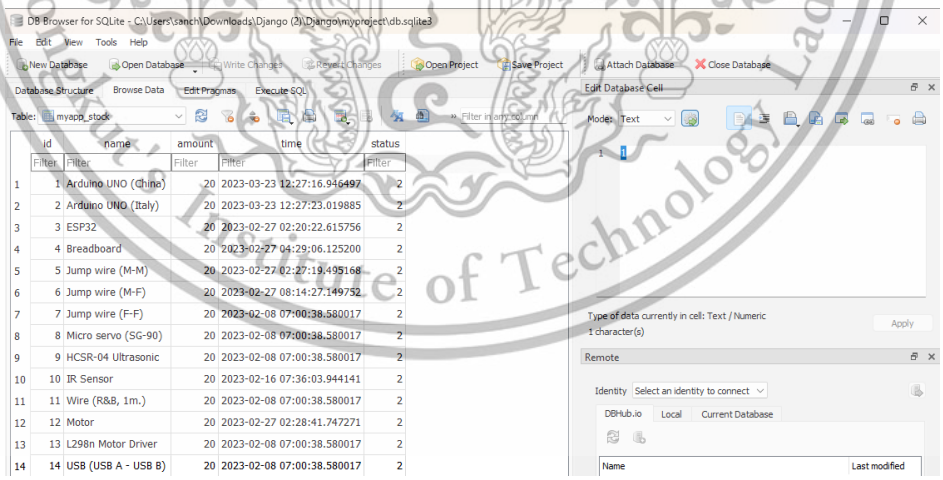


Figure 36 SQLite

We can also use the program to edit the data, remove the data or increase the data depending on the task. This is one of the ways to change the database manually during some emergency situations.

We also need to set up a serializer.py for implementing the database with REST API framework. Serializers are a core component of Django's built-in framework for creating Web APIs. They are typically used in conjunction with Django's generic views to convert the data returned by a database query or other data source into a format that can be easily consumed by clients.

```
from rest_framework import serializers
from .models import Stock

class StockSerializer(serializers.ModelSerializer):
    class Meta:
        model = Stock
        fields = ('id','name','amount','status') # '__all__'

        def update(self, instance, validated_data):
            instance.id = validated_data.get('id', instance.id)
            instance.name = validated_data.get('name', instance.name)
            instance.amount = validated_data.get('amount', instance.amount)
            instance.status = validated_data.get('status', instance.status)
            instance.save()
            return instance
```

Figure 37 serializer.py

3.2.3 API (REST Framework)

We start this process by installing the REST API framework as a python library and import the library into the views.py file (which is already done if you follow the library list during the backend section).

```
from rest_framework.response import Response
from rest_framework.renderers import JSONRenderer
from rest_framework.parsers import JSONParser
from rest_framework.decorators import api_view
from rest_framework import status, generics
from .serializers import StockSerializer
```

Figure 38 REST library list

After we import the libraries, we must write down the code in views.py to establish the usage of REST API. We also write down the command and sequences after receiving those commands.

```
@api_view(['GET','POST','PUT','PATCH',])
def all_stock(request):
    if (request.method == 'GET'):
        all_stock = Stock.objects.all()
        serializer = StockSerializer(all_stock, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
    if (request.method == 'POST'):
        json_data = request.body
        stream = io.BytesIO(json_data)
        python_data = JSONParser().parse(stream)
        serializer = StockSerializer(data=python_data)
        if serializer.is_valid():
            serializer.save()
            res = {'msg': 'Data Created'}
            json_data = JSONRenderer().render(res)
            return HttpResponse(json_data, content_type='application/json')
        return HttpResponse(JSONRenderer().render(serializer.errors), content_type=
    if (request.method == 'PUT'):
        json_data = request.body
        stream = io.BytesIO(json_data)
        python_data = JSONParser().parse(stream)
        id = python_data.get('id', None)
        if id is not None:
            stock = Stock.objects.get(id=id)
            serializer = StockSerializer(stock, data=python_data)
            if (serializer.is_valid()):
                serializer.save()
                res = {'msg': 'Data Updated'}
                json_data = JSONRenderer().render(res)
                return HttpResponse(json_data, content_type='application/json')
            res = {'msg': 'data not valid'}
            json_data = JSONRenderer().render(res)
            return HttpResponse(json_data, content_type='application/json')
        res = {'msg': 'data not valid'}
        json_data = JSONRenderer().render(res)
        return HttpResponse(json_data, content_type="application/json")
    if (request.method == 'PATCH'):
        json_data = request.body
        stream = io.BytesIO(json_data)
        python_data = JSONParser().parse(stream)
        id = python_data.get('id', None)
        if id is not None:
            stock = Stock.objects.get(id=id)
            if stock.amount == 0:
                res = {"msg": "Out of Stock"}
                json_data = JSONRenderer().render(res)
                return HttpResponse(json_data, content_type='application/json; charset=utf-8')
            elif stock.amount > 0:
                stock.amount -= 1
                stock.save()
                res = {'msg': 'Data Updated'}
                json_data = JSONRenderer().render(res)
                return HttpResponse(json_data, content_type='application/json')
            else:
                res = {'msg': 'Invalid data'}
                json_data = JSONRenderer().render(res)
                return HttpResponse(json_data, content_type='application/json')
        res = {'msg': 'data not valid'}
        json_data = JSONRenderer().render(res)
        return HttpResponse(json_data, content_type="application/json")
```

Figure 39 serializer.py

After establishing the REST API, we also have to create the URL link for the API page and also create the URL link for other pages too by writing the link that you want in the urls.py file.

```python
from django.urls import path
from myapp import views
from .views import stock_api, edit_stock, change_stock_amount, edit_status, change_status_0, change_status_1, change_status_2, update_stock
urlpatterns = [
    path('',views.index, name='stock'),
    path('update-stock/', update_stock),
    path('stock',views.stock),
    path('adst',views.adst),
    path('edit_stock/', edit_stock, name='edit_stock'),
    path('change_stock_amount/', change_stock_amount, name='change_stock_amount'),
    path('edit_status/', edit_status, name='edit_status'),
    path('change_status_0/', change_status_0, name='change_status_0'),
    path('change_status_1/', change_status_1, name='change_status_1'),
    path('change_status_2/', change_status_2, name='change_status_2'),
    path('additem',views.additem),
    path('api/all-stock/',views.all_stock),
    path('api/ind-stock/<id>/',views.ind_stock),
    path('chart',views.chart),
    path('rtchart',views.rtchart),
    path('api/stocks/', stock_api, name='stock_api'),
    path('conf/', views.conf, name='conf'),
    # path('my_endpoint/', views.my_endpoint, name='my_endpoint'),
]

```

Figure 40 urls.py

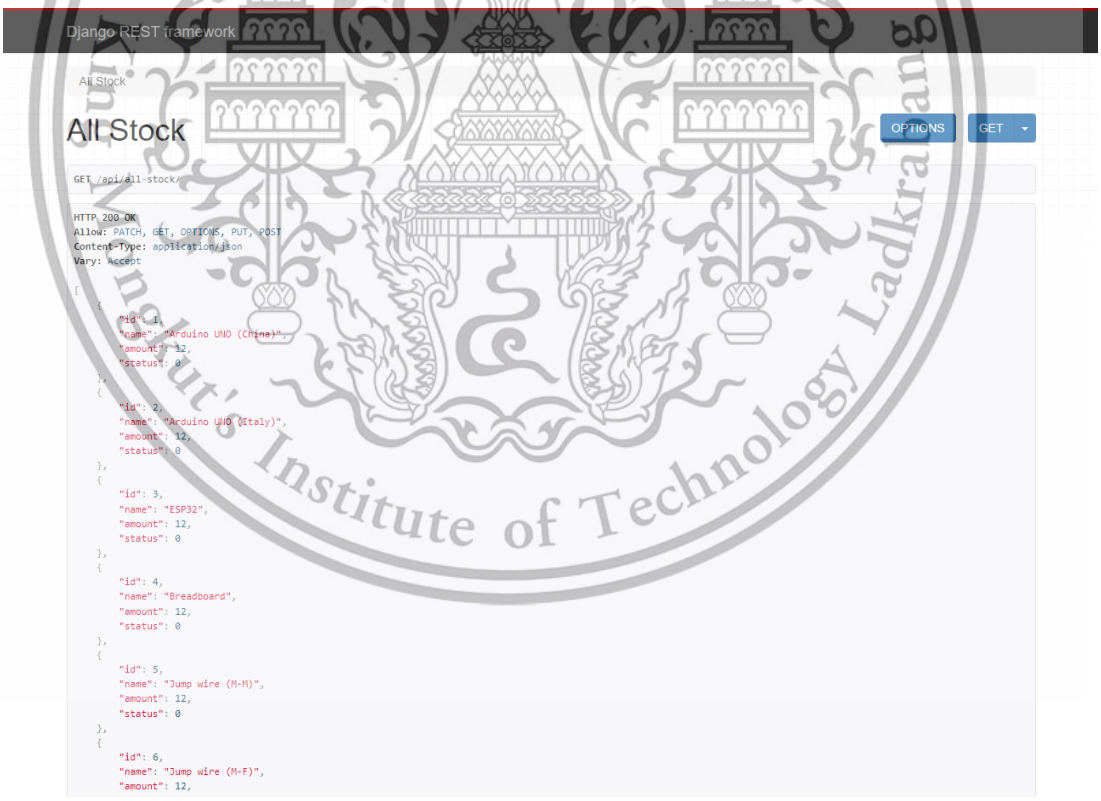


Figure 41 API page

3.3 Hardware

For the hardware parts, we need to do some drilling, wiring, and designing. For the drilling we designed to drill the 2 big fans holes to attach the fan for cooling system. For the wiring, we need to arrange all the wires to be in place neat and clean. For designing, we designed a side unit for the machine, inside including tablet, fans, router, and dongles.

Figure 42 Side unit

3.4 Business

For this part, we started with our business model then created a poll to find out the most used product in university, find out the top 25 product. Then we made a business model canvas.

3.4.1 Business model

Included 7 steps.

- Research the market: Determine the demand for Arduino boards and related items in the area where the vending machine will be located. Look for potential competitors and their pricing strategies.

- Obtain the necessary equipment: Purchase or lease a vending machine that can hold and dispense Arduino boards and related items.
- Stock the vending machine: Obtain a supply of Arduino boards and related items from a distributor or supplier at a wholesale price.
- Set the prices: Determine the retail prices for the Arduino boards and related items based on your research and the cost of goods.
- Promote the vending machine: Advertise the availability of the Arduino boards and related items through social media, flyers, and other marketing methods.
- Monitor and maintain the vending machine: Regularly check the vending machine for any malfunctions and restock it as needed. Keep track of sales and adjust prices or inventory as necessary.
- Expand the business: Once the vending machine is successful, you can consider expanding to multiple locations or offering a wider range of products.

Overall, the vending machine business model is a good way to sell Arduino boards and related items if there's a demand for it in your area. It allows for a low cost of entry and can be run with minimal staff and expenses.

3.4.2 Choosing Vending Machine

- Size and capacity: The vending machine should be large enough to hold the Arduino boards and related items and have enough capacity to hold a sufficient quantity to meet customer demand.
- Security: Make sure the vending machine has adequate security features to prevent theft and vandalism.
- Payment options: Consider what payment options you want to offer customers, such as cash, credit/debit cards, or mobile payments.
- Display: Look for vending machines with clear and attractive displays that will make it easy for customers to see the Arduino boards and related items.
- Additional features: Some vending machines come with additional features such as automatic restocking, remote monitoring, and inventory management. These can make it easier to run your vending machine business.

- Brand and model: Research on different brands and models of vending machine that are suitable for your business. You can find some vending machines that are specifically designed for small items such as electronics, and you can find some that have a more flexible design that can adapt to different items.

Ultimately, the best vending machine for your business will depend on your specific needs and the characteristics of your target market. It's a good idea to visit vending machine supplier and test the vending machine in person before deciding.

3.4.3 Survey

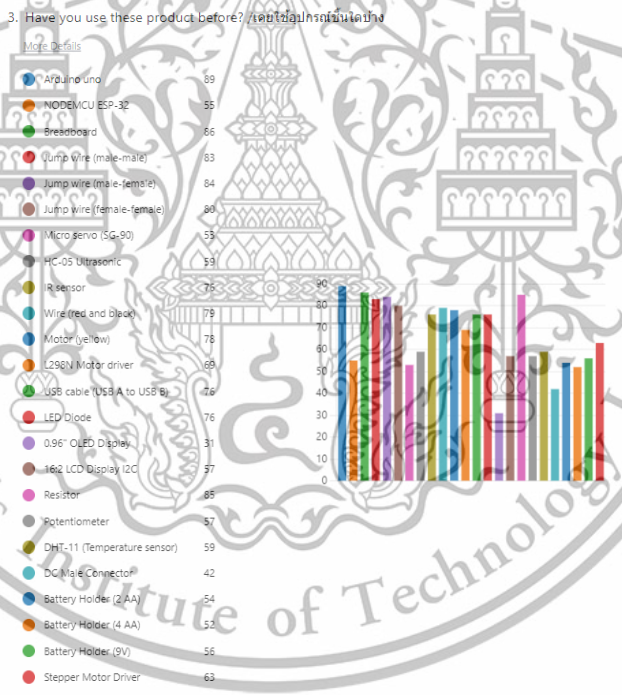


Figure 43 Response : top 25

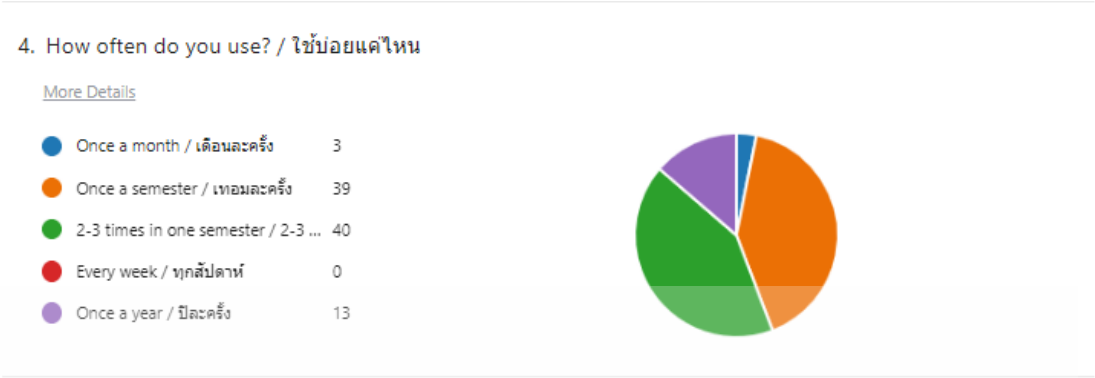


Figure 44 Response : Often used.

3.4.4 Banking API

For the banking API, every bank requires a limited partnership with at least 6-month registration, for the implementation of banking service for provide an api is 50,000 baht. So, we cut this off and use another way for mocking up the payment first.

CHAPTER 4

DEMONSTRATION AND RESULTS

4.1 FRONTEND RESULT

4.1.1 Home screen

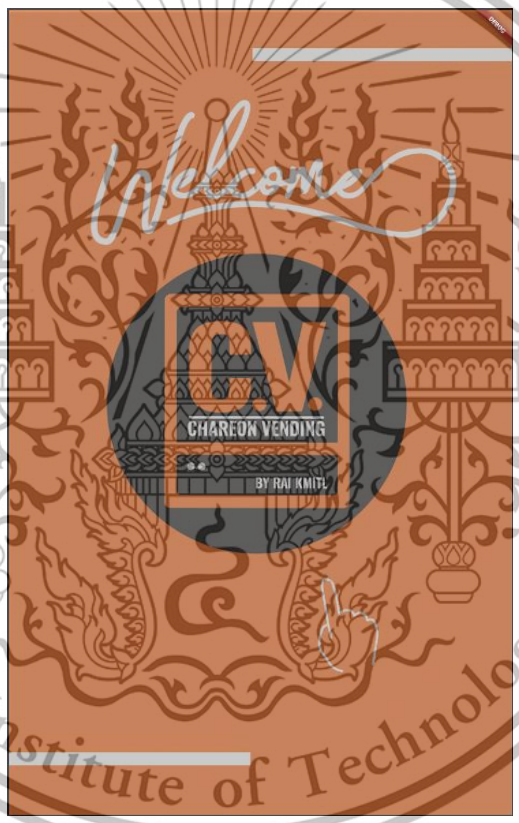


Figure 45 Home screen

In this home screen, the logo of our application will be rolling from left to right, waiting for the user to touch the screen to proceed to the next page.

4.1.2 Menu screen

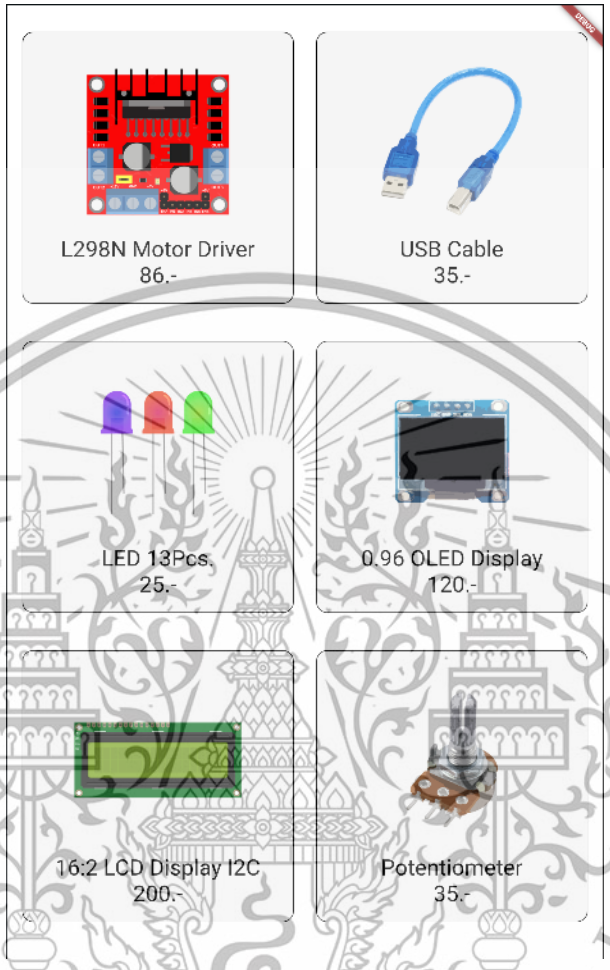


Figure 46 Menu screen

This screen will show all the items in the vending machine, displayed by 2 times 13 grid, sum up to 25 items. These items can be added or removed later. After clicking on the item that you want. The screen will show the pop-up to confirm the order. This screen can be moved by scrolling up and down to select the item. Also, the top and bottom navigation bar of the android will be hidden all the time when using the application, preventing all of the notification and someone trying to exit the application.

4.1.3 Confirmation screen

Figure 47 Confirmation screen

This screen will pop up after a customer chooses the items. It has two choices, the first is to confirm, and the application will bring the customer to the payment screen, cancel button uses to close he pop up and make the customer choose another item by navigating to the menu screen. If the item was out of stock the application will bring into another page called out of stock. The customer cannot process the to the payment.

4.1.4 Out of stock screen

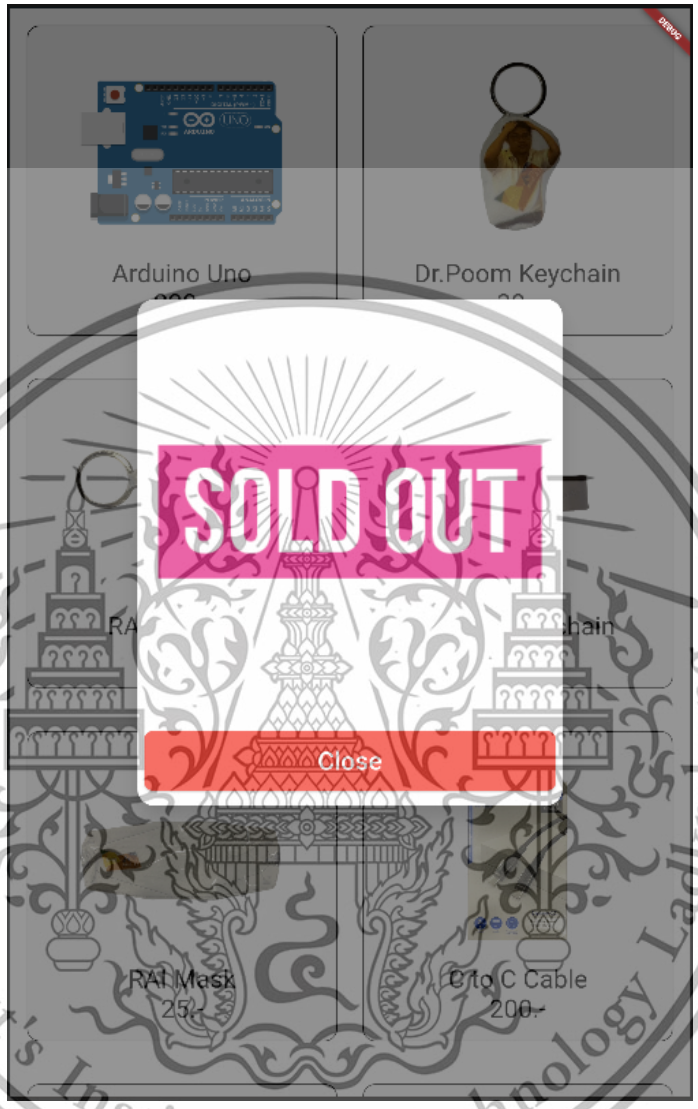


Figure 48 Out of stock screen

This pop up will show when the item in the vending machine is out of stock, by checking with the database. The application will not let the customer process the payment screen. The customer can just close the pop up and select other items that they want. This one is for preventing the customer to pay and did not getting the items.

4.1.5 Payment screen

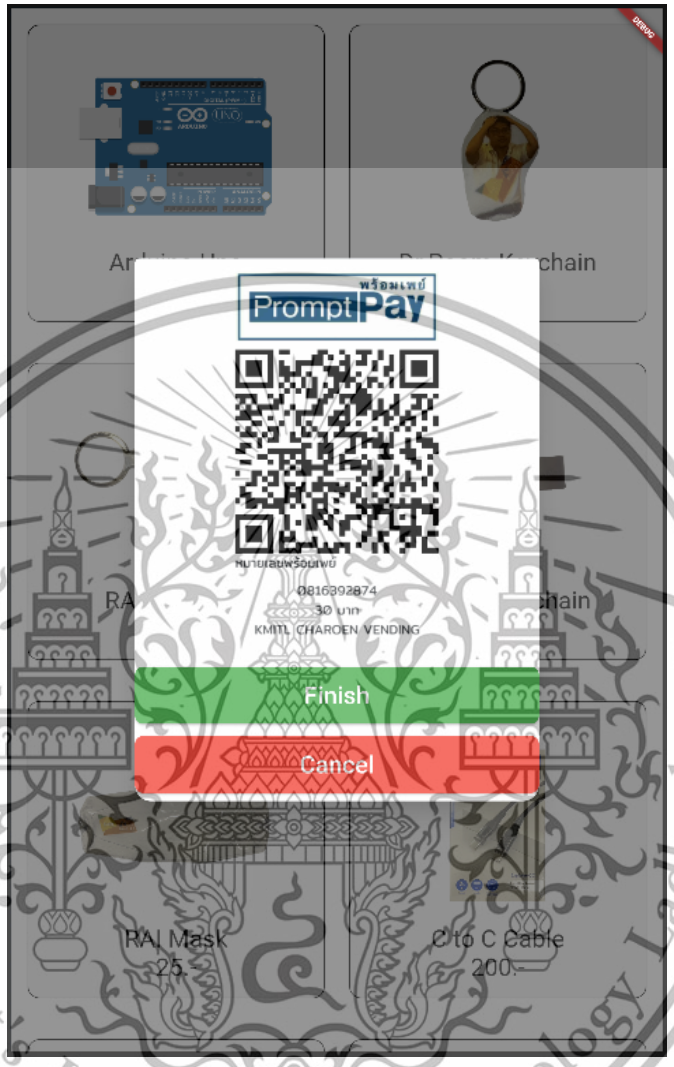


Figure 49 Payment screen

This screen is to make the customer pay for the items. The QR code for payment that we use in our application is just the mockup QR code. Which cannot be used daily because it needs to be confirmed manually. After the customer paid, they had to click on finish to finish the payment. The system will check if the payment is successful, the vending machine will drop the items, if not, it will show the payment fail screen.

4.1.6 Payment fail screen

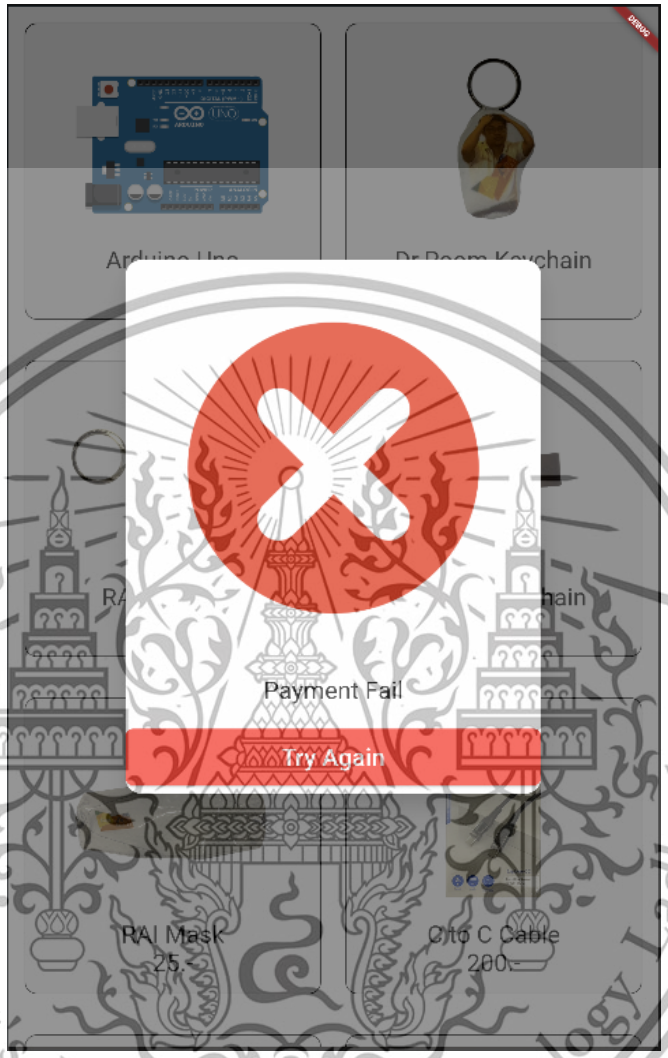


Figure 50 Payment fail screen

This screen will show when the payment is failed. The customer can click on try again and it will bring the customer back to the payment screen and try to pay for the item again.

4.1.7 Successful screen

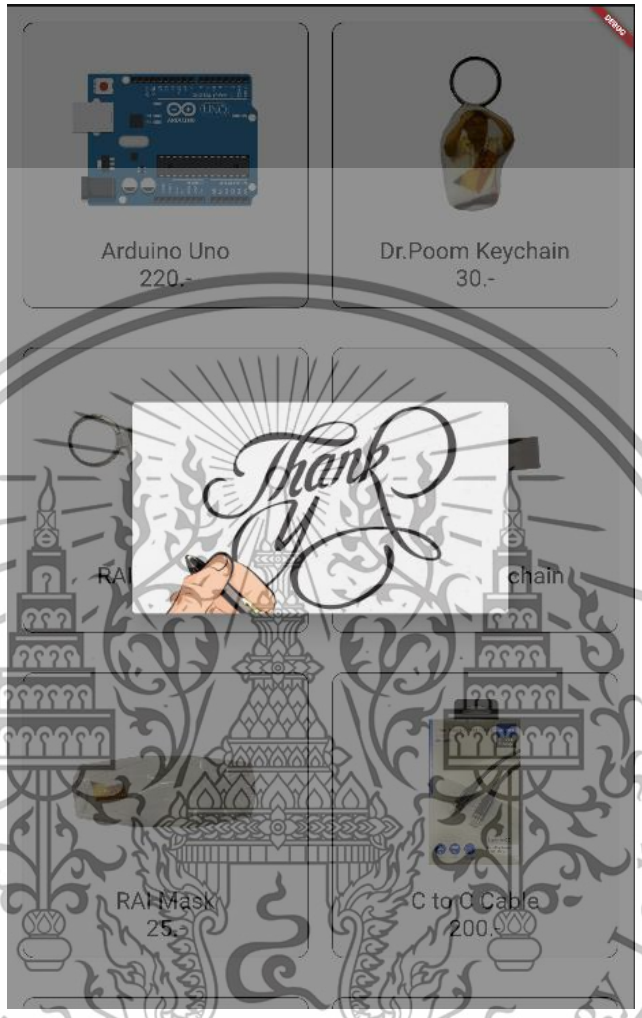


Figure 51 Successful screen

If this screen showed up, which means the payment was successful. The screen will be showing the “Thank you” gif and will close automatically after 10 seconds and bring the application back to home screen. If the customer wants to buy another item, they can press anywhere on the screen, and it will navigate back to the menu screen.

4.2 BACKEND RESULT

4.2.1 Django

After we finish setting up the Django, we are going to publish it as a public website via public domain. In this step, we must consult with the staff within the Robotics and AI Faculty to publish it in the faculty's server. The link to the website is https://jamie.krai.io/ .

Figure 52 Published website

We also included a private page which is only accessible to the admin of the website to change the stock's details. Adding this page will be convenient to the admin of the server during the restock of the machine.

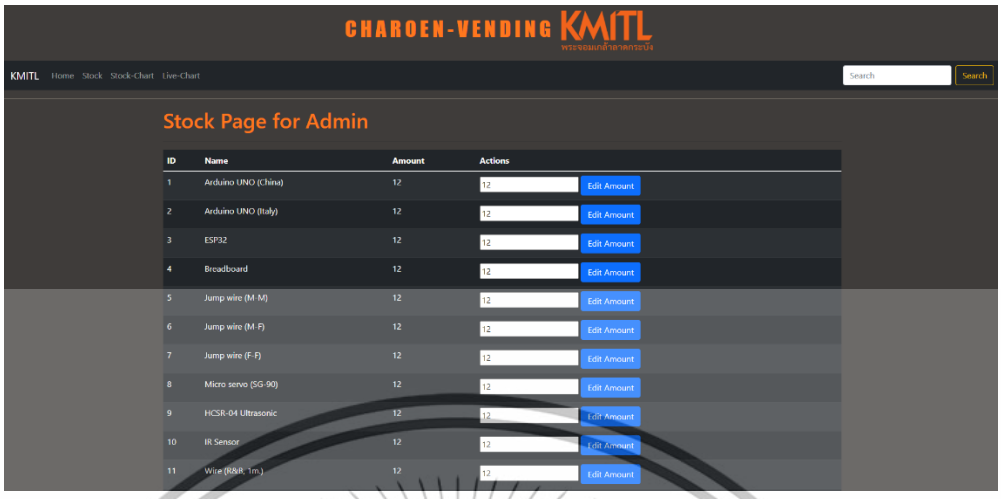


Figure 53 Stock Page for Admin

Figure 54 Django Administration

4.2.2 Database & API Result

After establishing the Database and API, we can test the communication result by using Postman. We will be focusing on the GET and PATCH method. The GET method is necessary for the Flutter part to check the stock when the customer presses the product. When we choose GET, we

will receive the stock data from the Django database which we can write the querry to get the amount of stock.

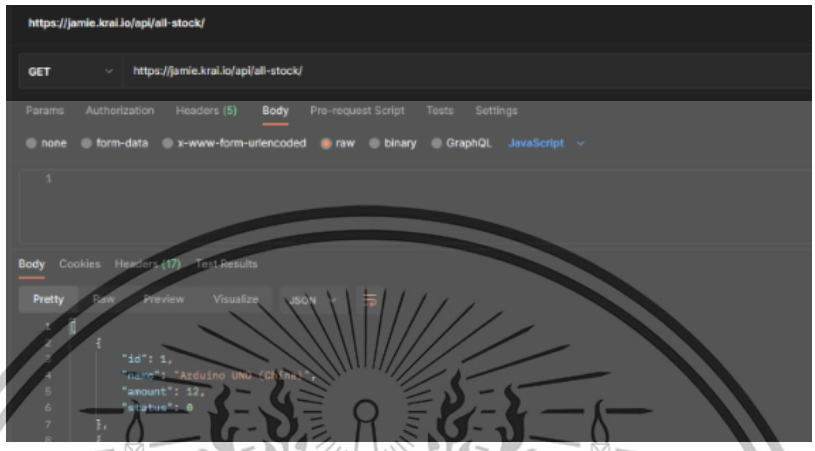


Figure 55 GET result

Next is the PATCH method which is used for decreasing the amount of stock. We will need to specify the querry of queries for the product that we want to decrease in this section. Each time that we send the PATCH command, it will decrease the amount of stock by one.

Figure 56 PATCH result

We also apply the GET method with the Real-Time Chart which will continuously get the data of stock in each timecode and display the decreasing data on the chart.

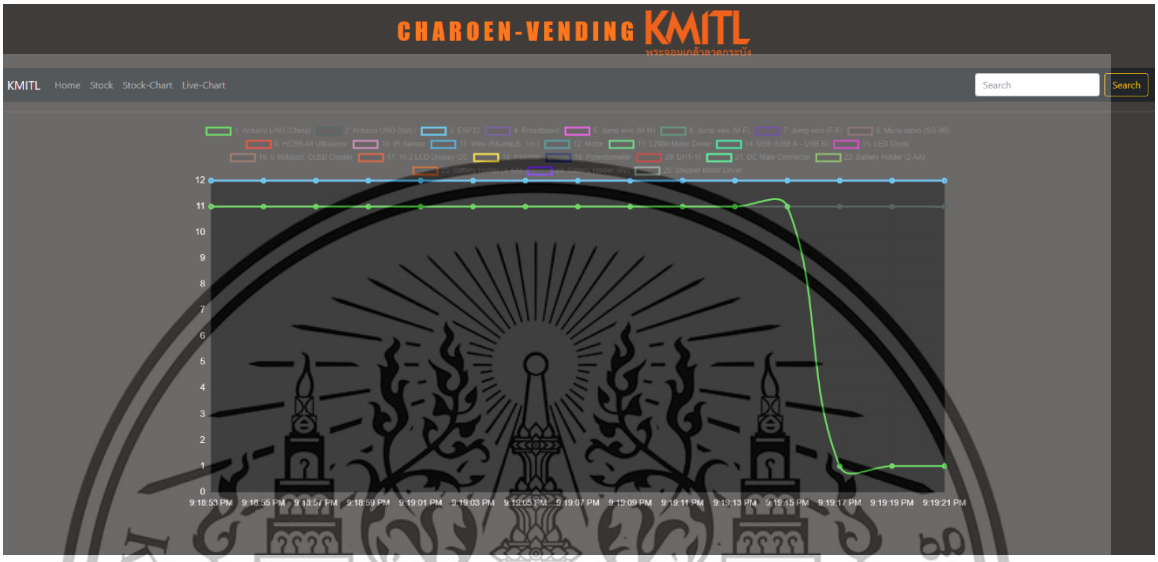


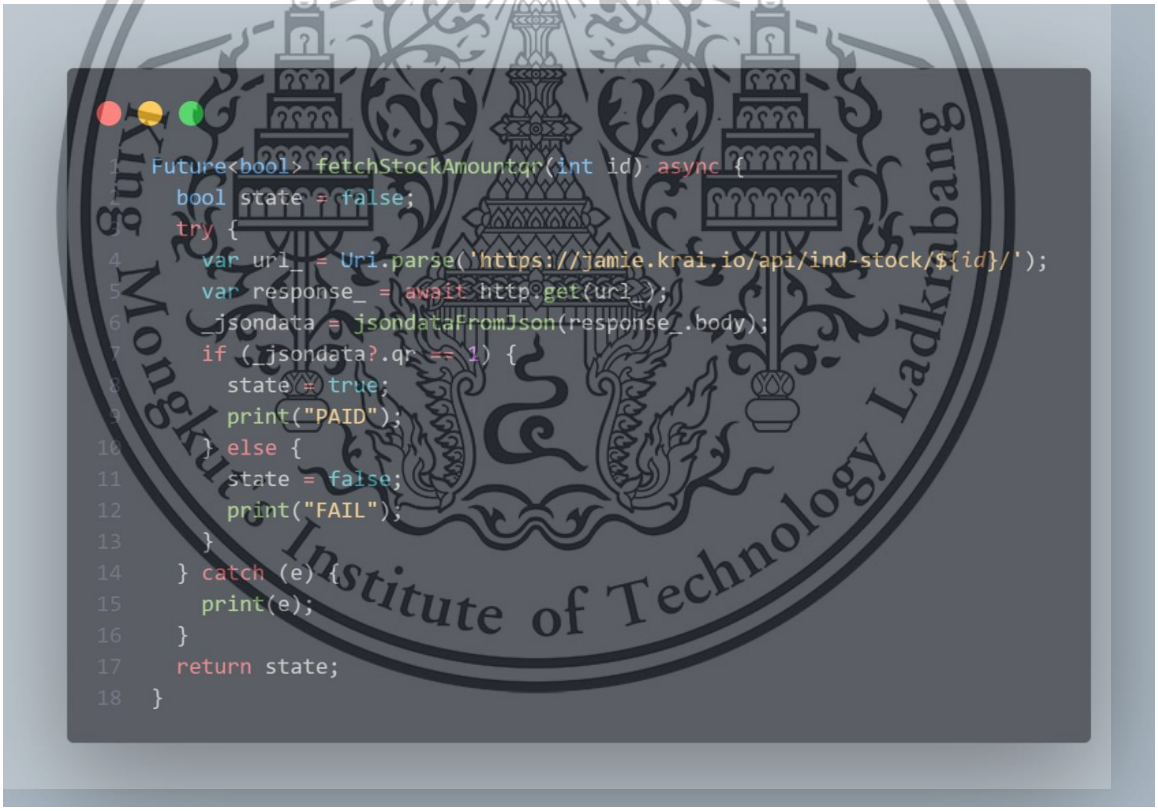
Figure 57 GET method with Chart

After testing out the API, we will apply them into the Flutter code which will establish the communication between the Vending Machine and the Django Database by changing the code language into dart and make it send the signal towards the URL of the API.

When the Vending Machine GET the stock's amount that is equal to 0, the Vending machine will show the Out of Stock screen to prevent the customer to buy the product. But if the product is not out of stock, the Flutter will proceed to the PATCH command which will decrease the stock amount after customer bought the product.

```dart
Future<bool> sendData(int a) async {
  http.Response apiResponse = await http.patch(Uri.parse(url));
  var apiData = jsonDecode(apiResponse.body);
  bool state = await fetchStockAmount(a);
  print(state);
```

Figure 58 JSON decode for dart



```dart
Future<bool> fetchStockAmountqr(int id) async {
  bool state = false;
  try {
    var uri_ = Uri.parse('https://jamie.knai.io/api/ind-stock/${id}/');
    var response_ = await http.get(uri_);
    _jsondata = jsondataFromJson(response_.body);
    if (_jsondata?.qr == 1) {
      state = true;
      print("PAID");
    } else {
      state = false;
      print("FAIL");
    }
  } catch (e) {
    print(e);
  }
  return state;
}
```

Figure 59 GET and PATCH for dart

4.3 Hardware

4.3.1 Communication

For communication with hardware. The motor driver board will connect to the signal codec, then connect to the android tablet.

4.3.1.1 Motor driver board

This motor driver board requires an HEX function to control the motors, which means the data or value that is sent to the motor driver needs to be in HEX format function. Include start bits, stop bits and data bits.

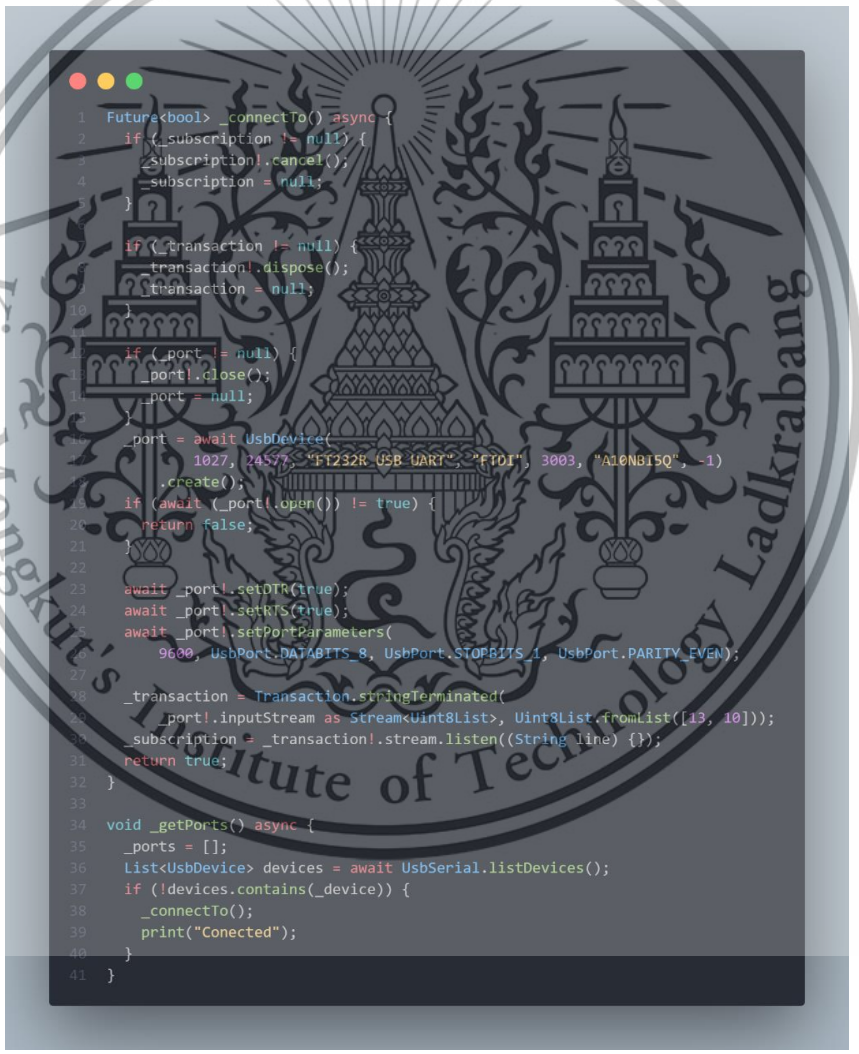
4.3.1.2 Signal Codec

For this project, we used the one called FT232RL USB to RS232 RS485 TTL Interface Converter Industrial Isolation. Which will help to isolate the signal from android tablet to motor driver board. The connection is just using a usb A to connect from tablet to the codec and 3 wires, ground, A and B to connect to motor driver

Figure 60 Signal Codec

4.3.1.3 Android Tablet code

For the android tablet code which runs by flutter, we have a tough time finding a package that works, finally any of the package on pub.dev and on the internet was not working at all. So, we have to find one of the closet possibilities that might work and read its library, changing some of the library code and creating a new function for sending data and connecting to a motor driver board.



```
Future<bool> _connectTo() async {
  if (_subscription != null) {
    _subscription!.cancel();
    _subscription = null;
  }

  if (_transaction != null) {
    _transaction!.dispose();
    _transaction = null;
  }

  if (_port != null) {
    _port!.close();
    _port = null;
  }
  _port = await UsbDevice(
          1027, 24577, "FT232R USB UART", "FTDI", 3003, "A10NBI5Q", -1)
      .create();
  if (await (_port!.open()) != true) {
    return false;
  }

  await _port!.setDTR(true);
  await _port!.setRTS(true);
  await _port!.setPortParameters(
      9600, UsbPort.DATABITS_8, UsbPort.STOPBITS_1, UsbPort.PARITY_EVEN);

  _transaction = Transaction.stringTerminated(
      _port!.inputStream as Stream<Uint8List>, Uint8List.fromList([13, 10]));
  _subscription = _transaction!.stream.listen((String line) {});
  return true;
}

void _getPorts() async {
  _ports = [];
  List<UsbDevice> devices = await UsbSerial.listDevices();
  if (!devices.contains(_device)) {
    _connectTo();
    print("Conected");
  }
}
```

Figure 61 getting ports

```
void portwrite(int id) async {
  id = id;
  await _connectTo();
  if (id == 1) {
    await _port!.write(Uint8List.fromList([
    print('CLICKED');
    print(id);
  }
```

Figure 62 Sending the HEX function

We need to blur out the HEX function of the vending machine, because it is the Duck group confidential details.

4.3.2 Vending Machine

For the side unit that we design, we use 3D printing to print it out, and use some putty and primer to clear the surface and make the surface smooth. After that we painted it black. Inside of the side unit, we have tablet, WIFI router, cooling fans and dongle

Figure 63 Side Unit

After we got the side unit. We drill the holes and attach it to the vending machine. Then do the decoration by adding stickers wrapping around the vending machine.

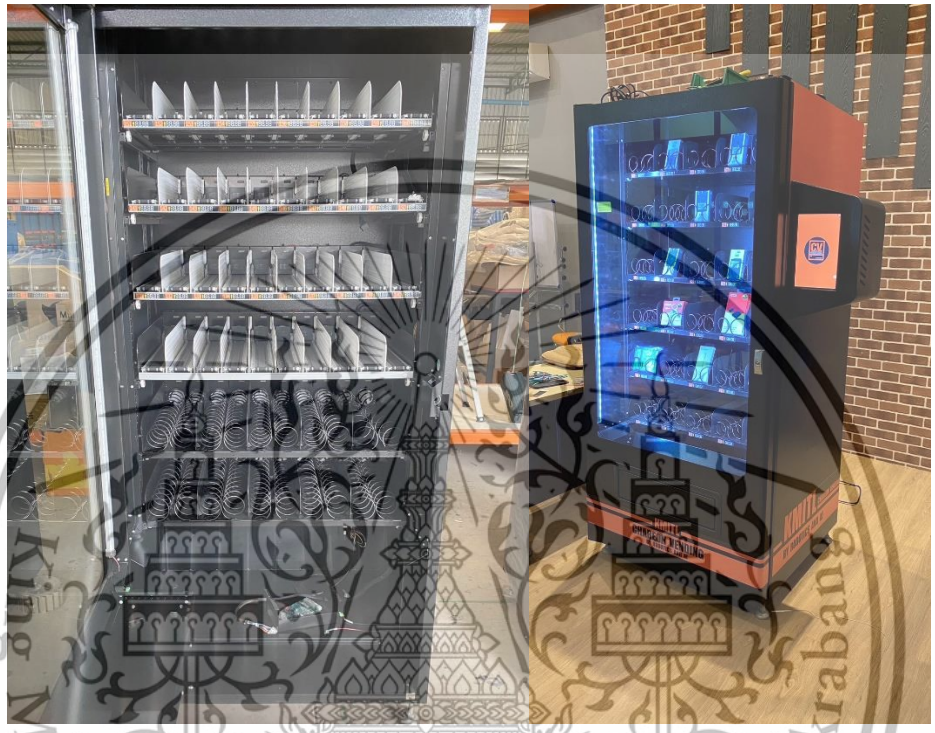


Figure 64 Before and After

4.4 PAYMENT MOCKUP

Since requesting a Banking API takes a long time to establish, we will create a mockup to be a place holder during this process. The mockup method will include Flutter getting data from the database which has a field called status. The admin of the website will be the one controlling the status to allow or not allow the signal to go through.

4.4.1 Django Payment Status

In the field of data, we have included a field called status. We set it to 1 and 0. If status is 1, it means that the payment is successful, and it will allow the purchase to go through. But if the status is 0, it will not allow the purchase to go through.

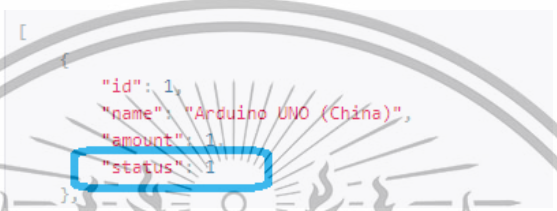


Figure 65 Status field

We also created a status change page for the website admin to change the status of the product after receiving confirmation for purchasing.

Figure 66 Status page

4.4.2 Flutter Payment Code

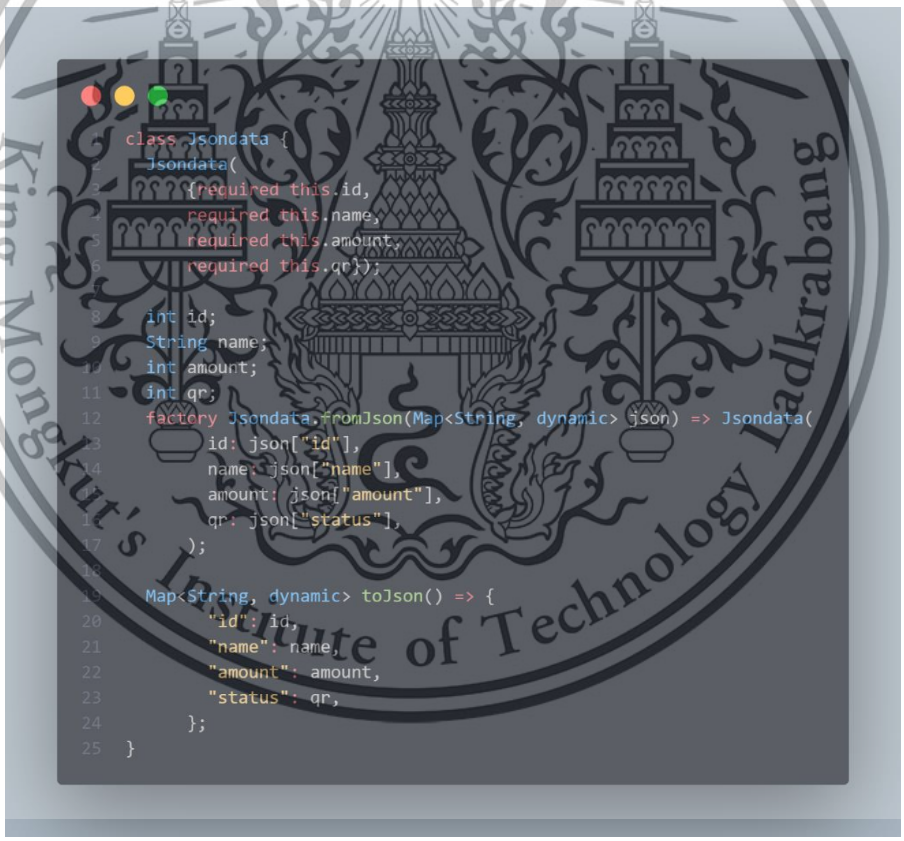
For the flutter code, we will include an image of QR code for each product in the Vending Machine. The format of QR code will be the prompt pay payment which is the commonly used payment in Thailand.

Figure 67 QR Code

Then we will write the Flutter Code to receive the status through GET method which will occur after getting the amount of stock. The customer will have to press a confirmation button after scanning the QR code for Flutter to GET the status.

```
if (id == 1)
            ClipRRect(
              borderRadius: BorderRadius.circular(5),
              child: Transform.scale(
                scale: scale,
                child: Image.asset('assets/image/220.png'),
              ),
            )
```

Figure 68 Pop up screen for QR Code



```
class Jsondata {
  Jsondata(
      {required this.id,
      required this.name,
      required this.amount,
      required this.qr});

  int id;
  String name;
  int amount;
  int qr;
  factory Jsondata.fromJson(Map<String, dynamic> json) => Jsondata(
        id: json["id"],
        name: json["name"],
        amount: json["amount"],
        qr: json["status"],
      );

  Map<String, dynamic> toJson() => {
        "id": id,
        "name": name,
        "amount": amount,
        "status": qr,
      };
}
```

Figure 69 Flutter Code for GET data

CHAPTER 5

CONCLUSION, FUTURE PLAN AND SUGGESTION

5.1 Conclusion

As a result of our project, a comprehensive vending machine system designed especially to satisfy the needs of college students looking for electronics parts has been successfully created. We have developed a highly functional and user-friendly platform that enables students to conveniently access and buy the required electronic components by designing and implementing our own frontend and backend systems. We became aware of the difficulties college students encounter when trying to buy electronics components as this project progressed. Students frequently have trouble locating particular parts, or they may only have limited access to stores that carry them. Our vending machine system offers a centralized and convenient campus-wide solution to these problems.

Our system's frontend has an easy-to-use interface with a pleasing appearance. We concentrated on making the user experience seamless so that students could easily browse the options, view comprehensive product information, and decide with confidence. We aimed to improve the overall convenience and effectiveness of the purchasing process by incorporating features like search functionality, filtering choices, and a user-friendly shopping cart. On the backend, we created a strong and expandable system that effectively handles inventory management, sales tracking, and transaction processing. To make the vending machine's operations as efficient as possible, our backend architecture combines secure payment gateways, algorithms for inventory management, and real-time analytics. We aimed to ensure a dependable and responsive system that can handle high volumes of transactions while maintaining data integrity and security by utilizing cloud-based technologies and best practices in software development. Additionally, our vending machine system offers more than just the ability to buy electronics components conveniently. It encourages an innovative and self-sufficient culture among university students. We enable students to explore

and experiment with electronics projects by giving them access to readily available components, allowing them to bring their ideas to life. This system helps the university achieve its goal of creating a learning environment that fosters imagination, critical thinking, and practical experience. Additionally, by serving as a focal point for electronics enthusiasts, our project has the potential to encourage student collaboration. Like-minded people can come together at the vending machine, which can lead to discussions, knowledge exchange, and potential project collaborations. The educational experience is further enhanced by this sense of belonging and camaraderie, which also aids students in honing important communication and teamwork skills.

In conclusion, the successful design and implementation of our electronics-parts vending machine system specifically targets college students. We have developed a user-friendly and effective platform with our frontend and backend systems that not only addresses the difficulties students face in accessing these components but also fosters a culture of independence, innovation, and cooperation within the university community. Moving forward, we think that our project will significantly improve students' educational experiences and help create a setting that is more supportive of both academic and personal development.

5.2 Future Plan

Future plans for this project include integrating a banking API for payment processing and putting machine learning into practice for facial recognition and membership management. These are the two main areas we can concentrate on to further improve the vending machine system.

First off, incorporating a banking API would give students more payment options and improve the vending machine system's overall convenience. Students could use their debit or credit cards, mobile payment apps, or other digital payment methods by connecting to a trustworthy and secure banking API. The payment process would be simplified by this integration, cash transactions would no longer be necessary, and the highest level of security for financial transactions would be guaranteed. Additionally, it would make it possible for us to track and manage payments more effectively, producing accurate reports and raising financial transparency.

Second, adding machine learning for facial recognition and membership management would give the vending machine system an extra layer of customization and security. We could create a facial recognition system that recognizes registered users, offering them a personalized experience and enabling quicker transactions by utilizing machine learning algorithms. For students, the process would be more seamless if this feature eliminated the need for physical membership cards or authentication codes. Facial recognition could also help us track usage patterns, collect insightful information on student preferences, and give customers personalized electronic component recommendations based on their past purchases. Additionally, the application of machine learning could aid in the improvement of inventory control. We can determine which components are likely to be in higher demand and adjust inventory levels accordingly by examining purchasing patterns, demand projections, and other pertinent data. The instances of out-of-stock items would be reduced thanks to this proactive approach, which would also guarantee a more effective supply chain management system.

In conclusion, our long-term goals for this project include incorporating a banking API for better payment options and security as well as putting machine learning to use for facial recognition and membership management. These improvements would make the vending machine system even more efficient, offer students practical payment options, offer a customized experience, and improve inventory control. We can continue to enhance the system's overall functionality, effectiveness, and user experience by embracing these developments, which will help the entire university community.

5.3 Suggestion

We advise concentrating on application updates that prioritize convenience and offer more thorough information about the products in order to further enhance the vending machine system and the user experience.

1. Improve the application's interface to make it even more intuitive and user-friendly. Optimize the overall layout, make sure the icons and

buttons are clear and simple to understand, and streamline the navigation flow to provide a seamless browsing experience.

2. Quick Search and Filtering: Integrate sophisticated search and filtering features into the application. Users will be able to instantly search for particular electronic components using keywords, specifications, or categories thanks to this. Additionally, give users the ability to filter their search results and find the desired products quickly by offering options like price range, brand, or compatibility.
3. Expand the product information section to include thorough details and specifications for each item. Product Details and Specifications. Include key details like dimensions, technical requirements, compatibility information, and any other pertinent information. This will help users choose the best components for their projects and make informed decisions.
4. Make sure the application presents high-quality product images from a variety of perspectives. Users will be better able to comprehend the physical characteristics and features of the components they are interested in thanks to clear and detailed images. Their purchasing decisions may be significantly influenced by this visual representation.
5. Reviews and Ratings from Users: Include a system for reviews and ratings from users within the application. Users will be able to do this to share their opinions and experiences with the goods they have purchased. Real customer feedback and ratings can help foster trust and support other users' decision-making.
6. Implement a "Related Products" or "Customers also bought" section within the application. Related Products and Recommendations. This function can recommend extra parts that are frequently bought together or suggest different options based on user preferences and previous purchases. Giving users these recommendations can increase cross-selling opportunities and help users find pertinent products they might have missed.
7. Wishlist and Notifications: Allow users to add products they want to a Wishlist they can access from the application. Users will be able to save items for later reference or to receive notifications when they are back in stock or on sale thanks to this feature. Users may receive push notifications alerting them to new product introductions, sales, or price reductions for items on their wish lists.

8. User Feedback and Support: Include a feedback mechanism that enables users to make suggestions or report any problems they run into in the application. The user experience will be improved overall thanks to this feedback and areas that need to be improved. Additionally, make it simple for users to contact customer support via email or a live chat feature to get any questions or issues resolved right away.

We can increase convenience and give users more thorough product information by putting these recommendations for application updates into practice. These upgrades will result in a smoother and more informative shopping experience, boosting customer satisfaction and creating a closer bond between the vending machine system and its patrons.

REFERENCES

DjangoGirls. (2022, Jan 1). *Django Tutorial*. Retrieved from https://tutorial.djangogirls.org/.

DjangoRESTframework. (2022, Jan 1). *Django REST framework documentation.* Retrieved from https://www.django-rest-framework.org/.

DjangoSoftwareFoundation. (2022, Jan 1). *Django database API*. Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/sql/.

DjangoSoftwareFoundation. (2022, Jan 1). *Django Documentation*. Retrieved from https://docs.djangoproject.com/.

DjangoSoftwareFoundation. (2022, Jan 1). *Django models*. Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/models/.

DjangoSoftwareFoundation. (2022, Jan 1). *Django ORM: Working with databases*. Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/.

Flutter. (2023, Jan 18). *Debugging Flutter apps*. Retrieved from https://docs.flutter.dev/testing/debugging.

Flutter. (2023, Jan 18). *Flutter-Set Up an editor*. Retrieved from https://docs.flutter.dev/get-started/editor.

Flutter. (2023, Jan 18). *Write your first Flutter app*. Retrieved from https://docs.flutter.dev/get-started/codelab.

KongRuksiamOfficial. (2019, Feb 3). *Django Tutorial for Beginners | Full Course [Video]. Youtube*. Retrieved from https://www.youtube.com/watch?v=XLMLveR2BYo.

Official, K. (2021, Jan 3). พัฒนาแอพด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE]. Bangkok, Thailand.

Pub. (2023, Feb 22). *Pub Dev*. Retrieved from https://pub.dev.

StackPython. (n.d.). *Django REST Framework API Python Tutorial*. Retrieved from https://stackpython.co/tutorial/django-rest-framework-api-python.

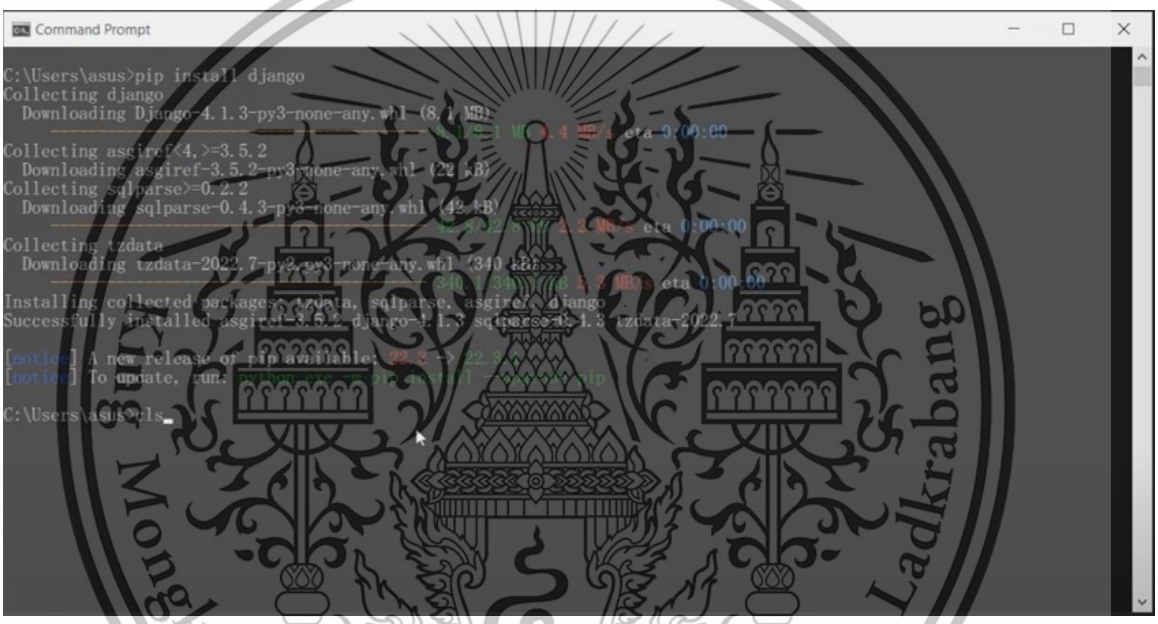
Vincent, W. S. (2019). Django for beginners: Build websites with Python and Django. *Independently published*.

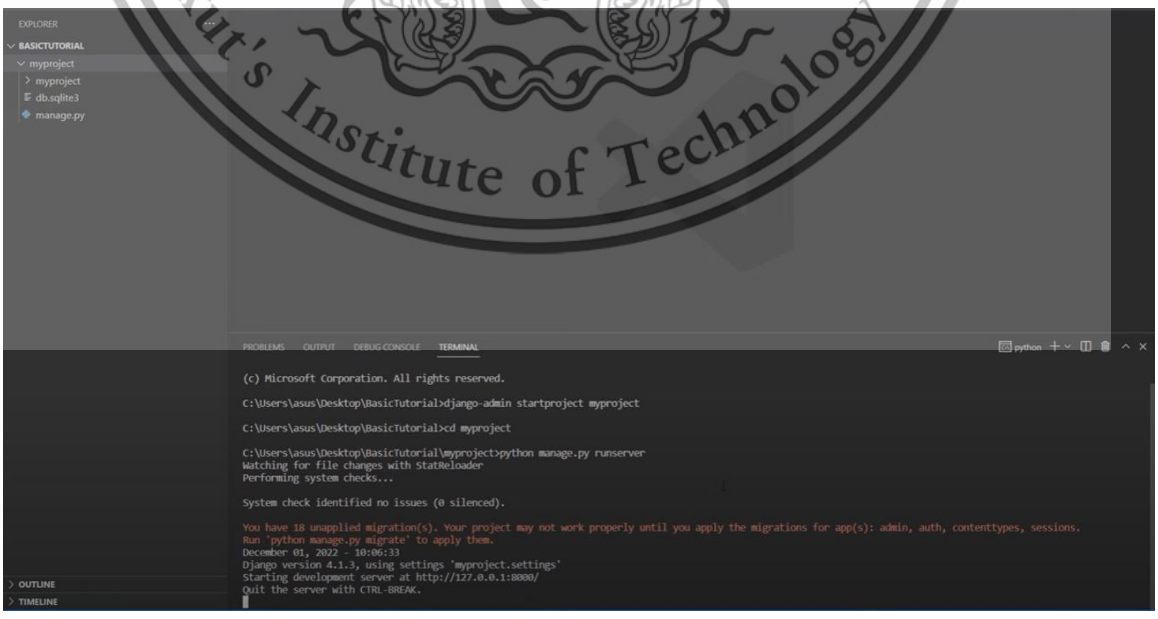
APPENDICES

APPENDIX A

DJANGO INSTALLATION

Installation Tutorial:
https://www.youtube.com/watch?v=XLMLveR2BYo&ab_channel=KongRuksiamOfficial





REST Framework & Django Database Installation Tutorial:

https://stackpython.co/tutorial/django-rest-framework-api-python

Django project & app

เมื่อทำการติดตั้ง Django และ Django REST Framework เสร็จแล้ว ก็จะเป็นการเริ่มต้นสร้างโปรเจคท์และแอพ

```
(env) django-admin startproject mysite .
(env) cd mysite
(env) python manage.py startapp api
```

requirements.txt

```
(env) pip freeze > requirements.txt
```

จะมีแพ็คเกจและไลบรารีที่ติดตั้งดังนี้

```
asgiref==3.3.4
Django==3.2.2
djangorestframework==3.12.4
pytz==2021.1
sqlparse==0.4.1
typing-extensions==3.10.0.0
```

ทำการรีจิสเตอร์แอพใน `settings.py`

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'api',  # Oup app
    'rest_framework'  # DRF
]
```

Database Model - SQLite

เมื่อสร้างโปรเจคท์และแอพเสร็จเรียบร้อย ถัดมาก็จะเป็นการสร้าง model ซึ่งในที่นี้จะหมายถึงการสร้าง class/object เพื่อติดต่อกับกับฐานข้อมูล เป็นอันว่ามักจะเข้าใจกันดีถ้าพูดถึง model นั้นแน่นอนว่าจะมีความเกี่ยวข้องกับ database นั่นเอง โดย database ที่ใช้ในโปรเจคท์นี้ก็จะใช้ SQLite ซึ่งเก็บข้อมูลในรูปแบบของไฟล์ ไม่ต้องทำการคอนฟิกเซิร์ฟเวอร์หรือพอร์ตอะไรต่าง ๆ ให้ยุ่งยาก เรียกได้ว่ามีมาให้กับ Django และพร้อมใช้กันได้เลย

Note: SQLite จะนิยมใช้กับโปรเจคท์เล็ก ๆ หรือไม่ว่าจะเป็นในช่วงของการ development ใน localhost เท่านั้น แต่ถ้าโปรเจคท์ใหญ่ ๆ มีผู้ใช้เยอะ ๆ มี concurrency สูง ๆ จะไม่เหมาะ แต่สำหรับคำแนะนำของผู้เขียน ถ้าจะ build โปรเจคท์จริง ๆ ควรใช้ PostgreSQL ทั้งในส่วนของ development และ production นั่นก็คือ keep the same environment as possible

ทำการออกแบบและสร้างตารางใน `models.py`

```
from django.db import models


class Task(models.Model):
    title = models.CharField(max_length=80)
    description = models.TextField()
    date_created = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField()

    class Meta:
        ordering = ['-date_created']
        db_table = 'task'

    def __str__(self):
        return self.title
```

จากนั้นทำการรันคำสั่งเพื่อ migrate database โดยต้อง makemigrations ก่อน

```
(env) python manage.py makemigrations
```

จะได้

```
Migrations for 'task':
  api\migrations\0001_initial.py
    - Create model Task
```

จากนั้นทำการ migrate

```
(env) python manage.py migrate
```

Serializers

serializer คือกระบวนการแปลง model object ไปเป็น JSON ฟอร์แมตเพื่อส่งไปที่ client หรือจะมีส่วนกลับอีกตัวที่มีชื่อว่า deserializer ที่ทำหน้าที่ตรงกันข้ามคือแปลง JSON ไปเป็น Django object เพื่อให้ Django สามารถอ่านเข้าใจได้

ทำการสร้างไฟล์ใหม่ขึ้นมาชื่อ `serializers.py` และทำการอิมพอร์ตโมดูล `serializers` เข้ามาเพื่อเรียกใช้งานคลาส `ModelSerializer` ซึ่งจะเป็นพระเอกในการ serialize ออปเจคท์นั่นเอง

`serializers.py`

```
from rest_framework import serializers
from .models import Task


class TaskSerializer(serializers.ModelSerializer):

    class Meta:
        model = Task
        fields = ('id', 'title', 'description', 'date_created', 'complete')
```

- `TaskSerializer` : คลาสที่ถูกสร้างขึ้นมา
- `model` : โมเดลที่ต้องการ serialize ซึ่งแน่นอนมีแค่โมเดลเดียวในตอนนี้คือ `Task` ที่ได้สร้างไว้ใน `models.py`
- `fields` : ฟิลด์หรือคอลัมน์ในโมเดลที่ต้องการ serialize ซึ่งในที่นี้คือนำมาใช้ทุกฟิลด์

Views

Views คือส่วนของการเขียน business logic ของ Django เช่นการจัดการกับ request/response ต่าง ๆ ซึ่งปกติแล้วจะมีรูปแบบการเขียนได้ 2 แบบ คือ **Function Based (FBV) View** และ **Class Based View (CBV)** ซึ่งในโปรเจคท์นี้จะเขียนแบบที่สอง เพราะว่าจะสามารถเรียกใช้ฟีเจอร์ต่าง ๆ ของ REST Framework ได้เต็มประสิทธิภาพ ซึ่งพระเอกของเราต่อไปนี้ก็คือ โมดูล `generics` ที่ประกอบไปด้วย APIView ต่าง ๆ ให้เราสามารถเลือกใช้งานได้อย่างสะดวกและรวดเร็วมาก ๆ

ไปที่ไฟล์ `views.py`

```python
from django.shortcuts import render
from .serializers import TaskSerializer
from .models import Task
from rest_framework import generics


class TaskList(generics.ListAPIView):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer

    # return Response(queryset.data)
```

- `TaskList` : คลาสที่ถูกสร้างขึ้นมา
- `ListAPIView` : คลาสที่สืบทอดมาจาก `generics` โมดูลเพื่อให้สามารถเข้าถึง list ของ object ซึ่งในที่นี้ก็คือลิสต์ของ task ต่าง ๆ ที่จะแสดงผลนั่นเอง
- `queryset` : ตัวแปรที่เก็บข้อมูลที่ query มาจากฐานข้อมูลในตาราง `Task` นั่นเอง
- `serializer_class` : กำหนดคลาสที่ได้ serialize ไว้ก่อนหน้านี้ ใน `serializers.py` ซึ่งต้องอิมพอร์ตเข้ามาใช้งานด้วยเช่นกัน นั่นก็คือ `TaskSerializer`

Note: การเขียนแบบ Class Based View จะขึ้นต้นด้วยคีย์เวิร์ด `class` และตัวที่เป็น Function Based View จะขึ้นต้นด้วย `def`

บทความแนะนำ/อ่านเพิ่มเติม CBV vs FBV

URLs / Route / End-points

ต่อมาจะเป็นการกำหนด URLs ให้กับโปรเจคท์ โดยทำอิมพอร์ตโมดูล `views` จากแอพ `api` เข้ามาเพื่อเรียกใช้งานคลาสและออปเจคท์ต่าง ๆ ในโมดูล

ไปที่ไฟล์ `mysite/urls.py`

```
from django.contrib import admin
from django.urls import path, include
from api import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/tasks/', views.TaskList.as_view()),
]
```

- `'api/tasks/'` URL end-point ของ API ที่กำลังสร้าง
- ทำการเพิ่ม `TaskList` ที่เข้าถึงได้ผ่าน `views.` เข้าไปใน path เพื่อแม็ปเข้าไปเป็นอาคิวเมนต์ลำดับถัดไปต่อจาก URL end-point เพื่อที่เวลา request มาจาก client จะถ้าเข้ามาเจอฟังก์ชั่นหรือแอคชั่นใน end-point นี้

RESTful Structure & URL endpoints

Endpoint --> Method --> Action

- `/api/tasks/` --> `GET` --> ดึงข้อมูล tasks มาแสดงผลทุก task
- `/api/tasks/<id>/` --> `GET` --> ดึงข้อมูลมาแสดงเฉพาะ task นั้น ๆ
- `/api/tasks/` --> `POST` --> สร้าง task ขึ้นมาใหม่
- `/api/tasks/<id>/` --> `PUT` --> อัปเดตข้อมูลใน task นั้น ๆ
- `/api/tasks/<id>/` --> `DELETE` --> ลบ task นั้น ๆ

APPENDIX B

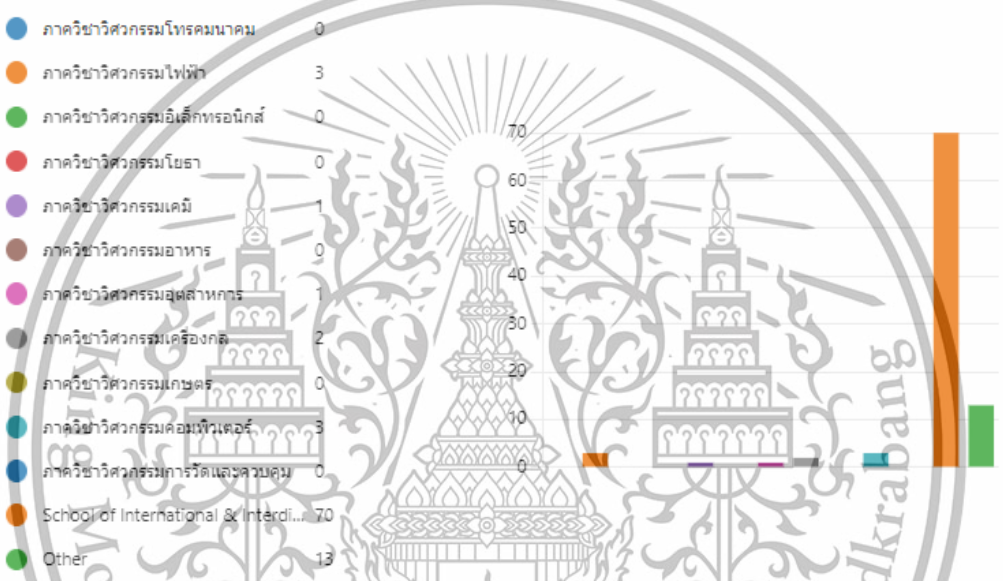
SURVEY OF ELECTRONIC PRODUCTS

ID	Start time	Completion time	Email	Name	Major / ภาควิชา	Year / ชั้นปี	Have you use these	How often do you u
1	1/17/23 10:09:08	1/17/23 10:09:44	anonymous		School of Internationa	4	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
2	1/17/23 10:11:29	1/17/23 10:12:24	anonymous		School of Internationa	2	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
3	1/17/23 10:11:41	1/17/23 10:13:14	anonymous			3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
4	1/17/23 10:11:51	1/17/23 10:13:30	anonymous		School of Internationa	2	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
5	1/17/23 10:13:32	1/17/23 10:14:25	anonymous		School of Internationa	4	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
6	1/17/23 10:13:47	1/17/23 10:14:41	anonymous		School of Internationa	3	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
7	1/17/23 10:14:43	1/17/23 10:16:37	anonymous		School of Internationa	2	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
8	1/17/23 10:16:38	1/17/23 10:19:22	anonymous		Rai	3	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
9	1/17/23 10:23:36	1/17/23 10:24:10	anonymous		ภาควิชาวิศวกรรมคุลสา	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
10	1/17/23 10:23:52	1/17/23 10:25:50	anonymous		School of Internationa	1	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
11	1/17/23 10:29:13	1/17/23 10:30:28	anonymous		ภาควิชาวิศวกรรมเครื่อง	4	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
12	1/17/23 10:33:46	1/17/23 10:34:25	anonymous		School of Internationa	4	Arduino uno;NODEMC	Once a year / ปีละครั้ง
13	1/17/23 10:37:57	1/17/23 10:38:47	anonymous		School of Internationa	1	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
14	1/17/23 10:36:16	1/17/23 10:39:32	anonymous		Robotics and Ai	3	Arduino uno;Breadboa	Once a year / ปีละครั้ง
15	1/17/23 10:39:40	1/17/23 10:40:40	anonymous		ภาควิชาวิศวกรรมไฟฟ้า	4	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
16	1/17/23 10:40:05	1/17/23 10:40:57	anonymous		School of Internationa	4	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
17	1/17/23 10:39:53	1/17/23 10:41:07	anonymous		School of Internationa	4	Arduino uno;Jump wir	Once a semester / เทอมละครั้ง
18	1/17/23 10:42:30	1/17/23 10:43:54	anonymous		School of Internationa	1	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
19	1/17/23 10:59:18	1/17/23 10:59:54	anonymous		School of Internationa	2	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
20	1/17/23 11:01:26	1/17/23 11:02:36	anonymous		School of Internationa	1	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
21	1/17/23 11:08:06	1/17/23 11:09:07	anonymous		School of Internationa	4	NODEMCU ESP-32;Ardi	Once a semester / เทอมละครั้ง
22	1/17/23 11:34:53	1/17/23 11:35:47	anonymous		TNI BJ	4	Micro servo (SG-90);US	Once a year / ปีละครั้ง
23	1/17/23 12:07:02	1/17/23 12:07:59	anonymous		School of Internationa	4	Stepper Motor Driver;I	2-3 times in one semester / 2-3 ครั้งต่อเทอม
24	1/17/23 12:07:56	1/17/23 12:08:32	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
25	1/17/23 12:20:39	1/17/23 12:21:22	anonymous		School of Internationa	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
26	1/17/23 12:36:57	1/17/23 12:38:18	anonymous		RAI	2	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
27	1/17/23 12:38:58	1/17/23 12:40:07	anonymous		ภาควิชาวิศวกรรมเคมี	4	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
28	1/17/23 12:51:11	1/17/23 12:52:30	anonymous			3	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
29	1/17/23 13:11:02	1/17/23 13:11:42	anonymous		School of Internationa	4	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
30	1/17/23 15:21:37	1/17/23 15:22:51	anonymous		ภาควิชาวิศวกรรมไฟฟ้า	4	Arduino uno;Jump wir	Once a month / เดือนละครั้ง
31	1/17/23 15:41:47	1/17/23 15:43:23	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
32	1/17/23 16:57:10	1/17/23 16:58:07	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
33	1/17/23 17:41:49	1/17/23 17:42:59	anonymous		School of Internationa	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
34	1/17/23 18:51:57	1/17/23 18:52:32	anonymous		School of Internationa	2	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
35	1/17/23 20:09:17	1/17/23 20:10:55	anonymous		การผลิตและส่งสนเทศ	4	IR sensor;Motor (yello	2-3 times in one semester / 2-3 ครั้งต่อเทอม
36	1/17/23 20:10:55	1/17/23 20:11:58	anonymous		วิศวกรรมสิ่งทอ	4	Breadboard;Jump wire	2-3 times in one semester / 2-3 ครั้งต่อเทอม
37	1/17/23 20:11:51	1/17/23 20:12:26	anonymous		Bme	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
38	1/17/23 22:06:42	1/17/23 22:07:15	anonymous		Imse	4	Jump wire (male-male	Once a year / ปีละครั้ง
39	1/17/23 22:41:49	1/17/23 22:42:59	anonymous		ภาควิชาวิศวกรรมคอมพิ	4	Wire (red and black);U	Once a year / ปีละครั้ง
40	1/17/23 23:29:10	1/17/23 23:30:28	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
41	1/18/23 2:01:40	1/18/23 2:02:34	anonymous		School of Internationa	2	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
42	1/18/23 2:28:53	1/18/23 2:29:37	anonymous		School of Internationa	4	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
43	1/18/23 3:22:18	1/18/23 3:23:14	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
44	1/18/23 13:07:06	1/18/23 13:07:43	anonymous		ภาควิชาวิศวกรรมคอมพิ	4	Arduino uno;Breadboa	Once a year / ปีละครั้ง
45	1/18/23 13:43:01	1/18/23 13:43:22	anonymous		ภาควิชาวิศวกรรมคอมพิ	4	Arduino uno;LED Diod	Once a year / ปีละครั้ง
46	1/18/23 19:52:40	1/18/23 19:53:36	anonymous		ภาควิชาวิศวกรรมเครื่อง	4	Battery Holder (2 AA);	2-3 times in one semester / 2-3 ครั้งต่อเทอม
47	1/18/23 20:36:20	1/18/23 20:36:57	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
48	1/18/23 20:36:25	1/18/23 20:37:14	anonymous		School of Internationa	2	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
49	1/18/23 20:36:50	1/18/23 20:37:30	anonymous		School of Internationa	1	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
50	1/18/23 20:36:58	1/18/23 20:37:40	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a year / ปีละครั้ง
51	1/18/23 20:36:23	1/18/23 20:37:43	anonymous		School of Internationa	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
52	1/18/23 20:36:26	1/18/23 20:37:53	anonymous		School of Internationa	1	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
53	1/18/23 20:36:14	1/18/23 20:38:05	anonymous		ภาควิชาวิศวกรรมไฟฟ้า	3	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
54	1/18/23 20:37:26	1/18/23 20:38:27	anonymous		School of Internationa	1	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
55	1/18/23 20:38:39	1/18/23 20:39:10	anonymous		School of Internationa	3	Arduino uno;IR sensor	Once a semester / เทอมละครั้ง
56	1/18/23 20:39:22	1/18/23 20:40:28	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
57	1/18/23 20:40:42	1/18/23 20:42:15	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a year / ปีละครั้ง
58	1/18/23 20:40:33	1/18/23 20:42:38	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
59	1/18/23 20:41:45	1/18/23 20:42:46	anonymous		School of Internationa	3	NODEMCU ESP-32;Ardi	2-3 times in one semester / 2-3 ครั้งต่อเทอม
60	1/18/23 20:41:51	1/18/23 20:42:59	anonymous		School of Internationa	2	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
61	1/18/23 20:42:35	1/18/23 20:43:22	anonymous		School of Internationa	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
62	1/18/23 20:46:23	1/18/23 20:47:26	anonymous		School of Internationa	3	Arduino uno;IR sensor	Once a year / ปีละครั้ง
63	1/18/23 20:46:24	1/18/23 20:47:30	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
64	1/18/23 20:54:02	1/18/23 20:54:54	anonymous		School of Internationa	4	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
65	1/18/23 20:55:05	1/18/23 20:56:01	anonymous		School of Internationa	2	Arduino uno;Jump wir	2-3 times in one semester / 2-3 ครั้งต่อเทอม
66	1/18/23 20:57:07	1/18/23 20:58:01	anonymous		School of Internationa	4	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
67	1/18/23 20:58:39	1/18/23 20:59:43	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
68	1/18/23 21:02:52	1/18/23 21:03:46	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
69	1/18/23 21:08:21	1/18/23 21:09:11	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
70	1/18/23 21:19:09	1/18/23 21:19:45	anonymous		School of Internationa	2	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
71	1/18/23 21:23:55	1/18/23 21:28:45	anonymous		RAI	2	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
72	1/18/23 21:29:57	1/18/23 21:33:50	anonymous		School of Internationa	2	Arduino uno;Breadboa	Once a year / ปีละครั้ง
73	1/18/23 22:06:37	1/18/23 22:08:13	anonymous		Rai	2	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
74	1/18/23 22:27:05	1/18/23 22:28:20	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
75	1/18/23 22:47:32	1/18/23 22:48:09	anonymous		School of Internationa	1	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
76	1/18/23 23:54:52	1/18/23 23:55:49	anonymous		School of Internationa	4	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
77	1/19/23 0:06:51	1/19/23 0:07:26	anonymous		RAI	1	Arduino uno;Breadboa	Once a month / เดือนละครั้ง
78	1/19/23 3:23:16	1/19/23 3:23:45	anonymous		School of Internationa	4	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
79	1/19/23 10:24:31	1/19/23 10:25:48	anonymous		School of Internationa	2	Breadboard;Arduino u	Once a semester / เทอมละครั้ง

80	1/19/23 11:53:28	1/19/23 11:55:12	anonymous		School of Internationa	2	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
81	1/20/23 14:27:16	1/20/23 14:27:50	anonymous		School of Internationa	1	Arduino uno;Jump wir	Once a semester / เทอมละครั้ง
82	1/20/23 14:27:06	1/20/23 14:28:16	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
83	1/20/23 14:27:28	1/20/23 14:28:57	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
84	1/20/23 14:30:28	1/20/23 14:31:04	anonymous		School of Internationa	4	Arduino uno;Breadboa	2-3 times in one semester / 2-3 ครั้งต่อเทอม
85	1/20/23 14:30:59	1/20/23 14:31:39	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
86	1/20/23 14:34:35	1/20/23 14:35:54	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
87	1/20/23 15:03:49	1/20/23 15:04:35	anonymous		School of Internationa	3	Arduino uno;NODEMC	2-3 times in one semester / 2-3 ครั้งต่อเทอม
88	1/20/23 15:07:14	1/20/23 15:08:24	anonymous		Rai	3	Motor (yellow);Resist	Once a semester / เทอมละครั้ง
89	1/20/23 15:12:47	1/20/23 15:13:28	anonymous		School of Internationa	3	Arduino uno;Micro ser	Once a year / ปีละครั้ง
90	1/20/23 15:45:02	1/20/23 15:47:59	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
91	1/20/23 15:48:00	1/20/23 15:48:45	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a semester / เทอมละครั้ง
92	1/20/23 17:42:51	1/20/23 17:43:53	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
93	1/21/23 22:23:18	1/21/23 22:24:07	anonymous		School of Internationa	3	Arduino uno;NODEMC	Once a semester / เทอมละครั้ง
94	1/21/23 22:22:46	1/21/23 22:24:12	anonymous		Robotics and AI	3	Arduino uno;Breadboa	Once a month / เดือนละครั้ง
95	1/22/23 0:37:46	1/22/23 0:38:39	anonymous		School of Internationa	3	Arduino uno;Breadboa	Once a year / ปีละครั้ง

1. Major / ภาควิชา

More Details



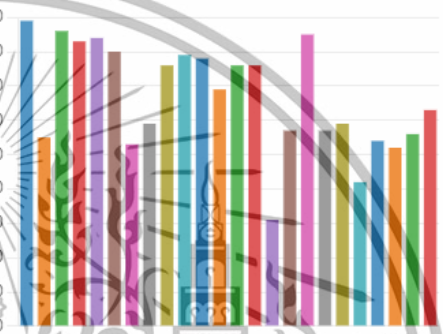
2. Year / ชั้นปี

More Details

3. Have you use these product before? /เคยใช้อุปกรณ์ชิ้นใดบ้าง

More Details

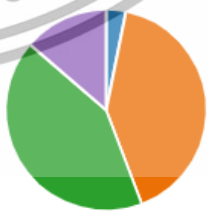
Arduino uno	89
NODEMCU ESP-32	55
Breadboard	86
Jump wire (male-male)	83
Jump wire (male-female)	84
Jump wire (female-female)	80
Micro servo (SG-90)	53
HC-05 Ultrasonic	59
IR sensor	76
Wire (red and black)	79
Motor (yellow)	78
L298N Motor driver	69
USB cable (USB A to USB B)	76
LED Diode	76
0.96" OLED Display	31
16:2 LCD Display I2C	57
Resistor	85
Potentiometer	57
DHT-11 (Temperature sensor)	59
DC Male Connector	42
Battery Holder (2 AA)	54
Battery Holder (4 AA)	52
Battery Holder (9V)	56
Stepper Motor Driver	63



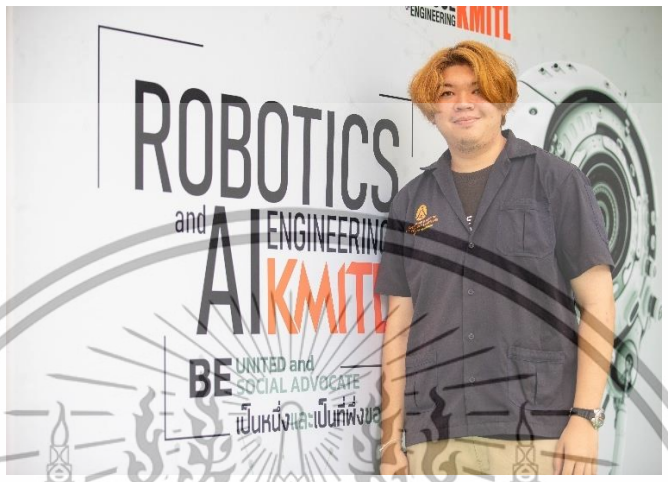
4. How often do you use? / ใช้บ่อยแค่ไหน

More Details

Once a month / เดือนละครั้ง	3
Once a semester / เทอมละครั้ง	39
2-3 times in one semester / 2-3 ...	40
Every week / ทุกสัปดาห์	0
Once a year / ปีละครั้ง	13



BIOGRAPHY



Chawapon Thamrongveerachart is a highly skilled student from KMITL (King Mongkut's Institute of Technology Ladkrabang) utilizing an impressive range of technical skills, including 3D CAD (Computer-Aided Design), automation, Flutter (a popular mobile app development framework), and microcontroller programming. Born and currently residing in Bangkok, Thailand, Chawapon has established themselves as a knowledgeable professional in their field. Chawapon holds a Bachelor of Engineering degree in Robotics and AI Engineering from King Mongkut's Institute of Technology Ladkrabang, a prestigious educational institution known for its focus on technology and engineering. Throughout their academic journey, Chawapon has encountered and successfully overcome various challenges in their pursuit of knowledge and expertise. Chawapon's personal traits and qualities are marked by their openness to new experiences and ideas. This mindset allows them to embrace unfamiliar challenges and continuously expand their horizons. Beyond their professional endeavors, CHAWAPON finds solace and inspiration in activities such as listening to music and playing the guitar, highlighting their artistic and creative side. While the information provided does not include a specific quote that Chawapon lives by, it seems unlikely that their motto would be "Always give up," as this contradicts their resilient nature and their ability to overcome difficult learning experiences. It's possible that there may be a miscommunication or error in the information provided.

Sanchai Sun is a Robotics and AI Engineer from KMITL specializing in Python, with a focus on machine learning and Django. With a strong passion for technology, Sanchai has honed his skills in Python programming and has a solid understanding of machine learning concepts.

During his internship at Seagate Technology Thailand, Sanchai worked as a Machine Learning Engineer in the Advanced Manufacturing Engineer (AME) team. In this role, he contributed to the development and implementation of machine learning algorithms for optimizing manufacturing processes. His work involved leveraging Python and machine learning techniques to enhance efficiency and productivity.

One of Sanchai's notable contributions includes his involvement in the Chareon Vending project. As part of this project, he played a key role in developing the backend infrastructure, designing the database, and facilitating API communication between the vending machines and the associated website. Sanchai's expertise in Django allowed him to create robust and efficient solutions, ensuring seamless functionality and effective communication between different components of the project.

ELECTRONICS VENDING MACHINE
(Chareon Vending)

Chawapon Thamrongveerachart, Natchanon Patrasettachai, Sanchai Sun
Robotics and AI Department of Engineering
King Mongkut's Institute of Technology LadKrabang
Bangkok, Thailand
th.chawapon@gmail.com, sanchaisaeng@gmail.com

Abstract— **A vending machine project aims to provide university students and faculty members with easy access to electronic components like transistors, resistors, and capacitors. The machine will feature a user-friendly interface, a wide selection of parts, and various payment options, including mobile payment. It will also have an inventory management system to track availability and automatically restock components. The project focuses on creating a reliable, user-friendly vending machine that meets performance standards, considering the needs and preferences of the users. This initiative enhances teaching, research, and benefits the campus community.**

Keywords— Vending Machines, Interface, API, Database, Communication

I. Introduction

This university project aims to develop a vending machine that provides convenient access to electronic components for students and faculty. It addresses challenges such as limited availability and high prices. By offering a wide range of components 24/7, it saves time and effort. The project enhances educational and research capabilities, fostering hands-on learning and preparing students for real-world applications.

II. Company Introduction

A. Company History

RAI Solution is an exceptional and visionary company that emerged from a dynamic environment of engineering innovation. It was founded during the groundbreaking development of the revolutionary Chareon vending machine, spearheaded by the brilliant minds of KMITL's Robotics and AI engineers. RAI Solution stands as a testament to their unwavering commitment to pushing the boundaries of technological advancement. With a team of ingenious individuals at its core, RAI Solution thrives on transforming cutting-edge concepts into practical realities.

B. Business Process

In this business, you invest in vending machines that are strategically placed in high-traffic areas such as office buildings, schools, shopping malls, hospitals, and recreational areas. These machines are stocked with a range of products that cater to the needs and preferences of your target market.

1. Passive Income: Customers can make purchases at any time, providing you with a passive income stream.
2. Low Overhead Costs: Vending machine businesses typically have low overhead costs compared to other types of businesses.
3. Flexibility and Scalability: Vending machines offer flexibility in terms of location and product selection.
4. Diverse Product Range: Vending machines can dispense a wide range of products, including snacks, beverages, fresh food, coffee, personal care items, and even electronics.

III. Problem

Engineering students often face challenges when purchasing electronics components for their projects. These components are not readily available on campus, requiring students to travel off-campus or order online, which can be time-consuming and costly. Additionally, inflated prices and limited accessibility can further hinder students' progress and discourage their pursuit of engineering as a career. To address these issues, the development of a vending machine on campus can provide a convenient and affordable solution for students to purchase the necessary components.

A. Limited Availability and Accessibility

- Electronics components are not readily available on campus.
- Students have to travel to off-campus stores or order online.

B. Time-consuming and Costly Procurement Process

- Traveling off-campus or waiting for online orders can be time-consuming.
- Components purchased outside campus stores can be inflated in price.

C. Negative Impact on Project Efficiency and Budget

- Delays in procuring components lead to project delays.
- Increased expenses due to high prices impact on students' budgets.

In this context, developing a vending machine for selling electronics components on campus will alleviate these challenges, providing a convenient and affordable solution for engineering students. By addressing these problem statements, the vending machine aims to improve accessibility, reduce procurement time and costs.

IV. SOLUTION

To address the challenges faced by engineering students in procuring electronics components for their projects, the development of a vending machine on campus presents an innovative and practical solution. This vending machine will offer several advantages:

A. Enhanced Availability and Accessibility

The vending machine will be strategically placed on campus, providing easy access to electronics components for students.

B. Streamlined Procurement Process

The vending machine will offer a simple and efficient purchasing process, saving students time and effort.

C. Cost-effectiveness and Budget-friendly

By eliminating inflated prices commonly found at off-campus stores, the vending machine will provide affordable pricing for components.

V. METHODOLOGY

A. FRONTEND

First, we designed the UI in Figma website for planning our application direction and request some feedback from sample users for the implementation. We did an application on the React.js platform then we changed to Windows application, after discussing it with our advisor. We have decided to use an Android tablet, so Flutter is the way.

1. Android Studio

In Android Studio, we used it for debugging and testing our developed applications for the Vending machine. It's a tool that will simulate an android device running on our computer. In it, we can choose the version of android and the version of android device. In this project, I chose an Android tablet with 10.5-inch screen, and android version 12 Tiramisu.

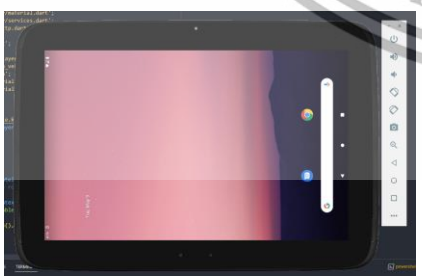


Figure 5.1 Android Studio Emulation

2. Flutter

In this part of the project. We started to develop an application from scratch. First, we have to learn Dart language and learn about the features of Flutter to implement in our application. Thanks to YouTube Channel "KongRuksiam" for the lesson of everything about how to develop an application in Flutter. In the application of our project, we are separated into 5 main parts.

- main.dart,- we include everything function from other parts of the application into this file.

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    SystemChrome.setEnabledSystemUIOverlays([]);
    return MaterialApp(
      home: FirstScreen(),
    );
  }
}
```

Figure 5.2 main.dart

- jsonreceive.dart - This part of the application is to decode the data from the backend parts.
- receive.dart - This part of the application fetches the data from the backend.
- menuwidget.dart - This part of the application includes all the popups such as out of stock, paid, and qr payment.
- senddata.dart - This code only has a function to reduce the stock amount in database by sending command.

B. BACKEND

1. Django

First, we start by creating the Django project for making the base website which will also act as a website where customers can come to check the stock of the product within the vending machine.

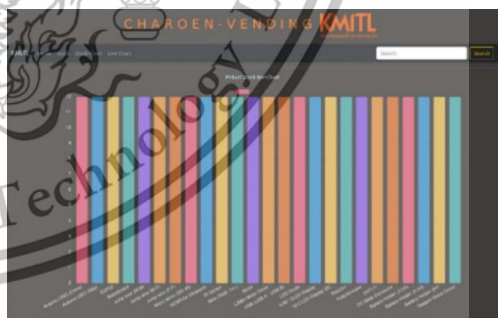


Figure 5.3 Chart Page

1. Database

We create the database to connect with the Website and Vending Machine to be able to track the item and its stock.

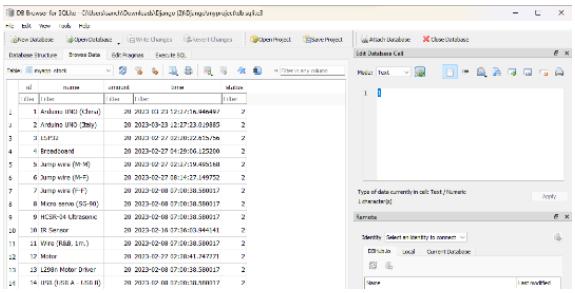


Figure 5.4 SQLite

C. API & COMMUNICATION

By using REST Framework, we can make the website and flutter communicate which makes the customer able to interact with the backends through frontend of the vending machine.

Figure 5.5 Flutter API code

D. HARDWARE

For designing, we designed a side unit for the machine, inside including tablet, fans, router, and dongles.

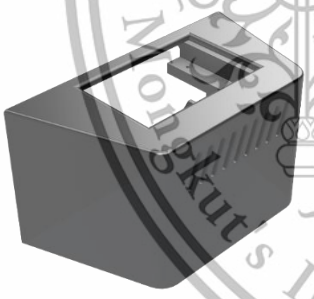


Figure 5.6 Side Unit

VI. THEORY

A. FRONTEND

The "front-end" of a website or web application is the area of a website or web application that a user interacts with directly. It includes all the components that a user sees and uses, including the text, images, layout, design, and interactive elements. In front-end development, these elements are created and manipulated using programming languages like HTML, CSS, and JavaScript to make them responsive, aesthetically pleasing, and user-friendly.

B. BACKEND

A web application's backend, also spelled "back-end" or "back end," refers to the server-side of a software program. The server, database, APIs, and other elements required for the operation of the application are all included in the code and technologies that power it from the background. The backend is in charge of handling user requests, processing data, and controlling the business logic of the application. This indicates that the backend is in charge of getting data out of a database, storing it there, running calculations on it, and giving the necessary data to the front-end or client-side of the application.

VII. DESIGN PROCESS

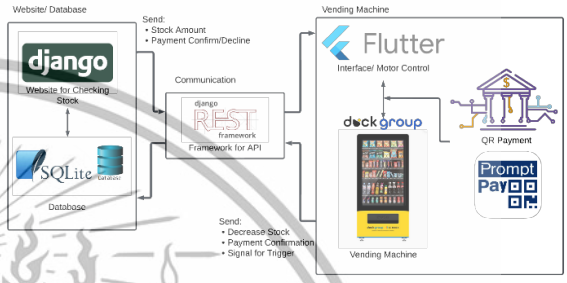


Figure 7.1 Relation Chart

VIII. EXPERIMENTAL RESULT

A. FRONTEND

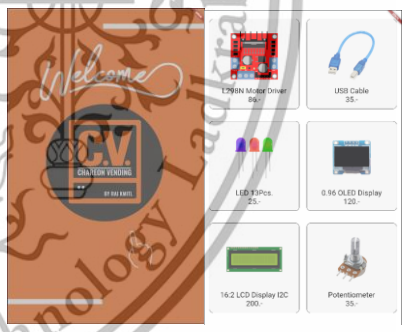


Figure 8.1 Frontend Result

Using Flutter to create an application that connects to the vending machine for user interface and motor control has proven to be highly effective and efficient. The integration of Flutter's cross-platform capabilities and intuitive user interface design has resulted in a seamless and user-friendly experience for both students and faculty members.

B. BACKEND

Figure 8.2 Backend Result

After we finish setting up the Django, we are going to publish it as a public website via public domain. In this step, we must consult with the staff within the Robotics and AI Faculty to publish it in the faculty's server. The link to the website is https://jamie.krai.io/ .

C. API & COMMUNICATION

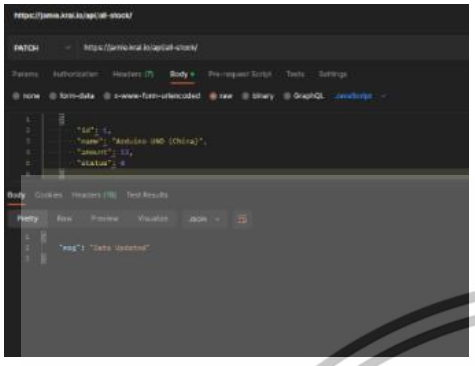


Figure 8.3 API Test Result

After establishing the Database and API, we can test the communication result by using Postman. We will be focusing on the GET and PATCH method. The GET method is necessary for the Flutter part to check the stock when the customer presses the product. When we choose GET, we will receive the stock data from the Django database which we can write the querry to get the amount of stock.

D. HARDWARE

Figure 8.4 Vending Machine

ACKNOWLEDGEMENT

We would like to take this opportunity to express my sincere gratitude to all the individuals and organizations who have helped me in completing this university paper. Primarily, we would like to extend my heartfelt thanks to my advisor, Dr. Poom Konghuayrob, for his invaluable guidance and support throughout my research. His valuable insights and feedback have been instrumental in shaping my work, and we are deeply grateful for his encouragement and financial support. We also want to thank Mr. Thitipong Thepsit for his technical assistance and help with the Flutter code. His advice and input were extremely helpful when creating the software for my project. We are also grateful to the Duck group for their generous contribution to the vending machine, which was instrumental in the success of my project. Their support has been instrumental in helping me achieve my research goals. We would also like to thank Mr. Chousak Chousangsuntorn from Seagate Technology for providing us with invaluable contacts and information. His support was crucial in enabling me to make valuable connections within the industry and gain a deeper understanding of my research topic. Finally, we would like to acknowledge the Robotics and AI Engineering department for providing me with a place to work and laboratory facilities to conduct my research. The resources and support provided by the department have been invaluable in helping me achieve my research goals. Once again, we would like to express my heartfelt thanks to everyone who has contributed to this paper. Without your support, this project would not have been possible.

REFERENCES

[1] DjangoGirls. (2022, Jan 1). *Django Tutorial*. Retrieved from https://tutorial.djangogirls.org/.

[2] DjangoRESTframework. (2022, Jan 1). *Django REST framework documentation.* Retrieved from https://www.django-rest-framework.org/.

[3] DjangoSoftwareFoundation. (2022, Jan 1). *Django database API.* Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/sql/.

[4] DjangoSoftwareFoundation. (2022, Jan 1). *Django Documentation.* Retrieved from https://docs.djangoproject.com/.

[5] DjangoSoftwareFoundation. (2022, Jan 1). *Django models*. Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/models/.

[6] DjangoSoftwareFoundation. (2022, Jan 1). *Django ORM: Working with databases*. Retrieved from https://docs.djangoproject.com/en/3.2/topics/db/.

[7] Flutter. (2023, Jan 18). *Debugging Flutter apps*. Retrieved from https://docs.flutter.dev/testing/debugging.

[8] Flutter. (2023, Jan 18). *Flutter-Set Up an editor*. Retrieved from https://docs.flutter.dev/get-started/editor.

[9] Flutter. (2023, Jan 18). *Write your first Flutter app*. Retrieved from https://docs.flutter.dev/get-started/codelab.

[10] KongRuksiamOfficial. (2019, Feb 3). *Django Tutorial for Beginners | Full Course [Video]. Youtube*. Retrieved from https://www.youtube.com/watch?v=XLMLveR2BYo.

[11] Official, K. (2021, Jan 3). พัฒนาแอพด้วย Flutter สำหรับผู้เริ่มต้น 7 ชั่วโมงเต็ม [FULL COURSE]. Bangkok, Thailand.

[12] Pub. (2023, Feb 22). *Pub Dev*. Retrieved from https://pub.dev.

[13] StackPython. (n.d.). *Django REST Framework API Python Tutorial*. Retrieved from https://stackpython.co/tutorial/django-rest-framework-api-python.

[14] Vincent, W. S. (2019). Django for beginners: Build websites with Python and Django. *Independently published*.

CONTRIBUTION EVALUATION

- Chawapon Thamrongveerachart (45%)
  - Fronted
  - Research
  - Flutter (UI)
  - TTL Communication (Vending Machine)
  - API Communication (Flutter side)
  - Payment Mockup (Flutter side)
  - Hardware
  - Report
  - Presentation

- Sanchai Sun (45%)
  - Backend
  - Research
  - Django (Website)
  - Database (Product ID, Stock)
  - API Communication (Django side)
  - Payment Mockup (Django side)
  - Hardware
  - Report
  - Presentation

- Natchanon Patrasettachai (10%)
  - Research
  - Payment (uncomplete)