COMP4801 Final Year Project
A Better System for HKU Dorms

Student: Ho Joshua (3035689831)

Supervisor: Dr Pan Jia
Abstract

The laundry process for students residing in dormitories at the University of Hong Kong is currently a manual and time-consuming process that can result in long wait times and frustration for students. The lack of a streamlined and efficient system for reserving washers and monitoring their availability creates unnecessary inconvenience and stress for students. This project aims to solve this issue by creating a web application that allows students to view real-time status of washers and make reservations to give them a more convenient and hassle-free experience. The front-end and back-end consist of popular frameworks fastAPI and Next.js respectively, with MySQL used as the database. The SSOCR, an open-source OCR software is used to retrieve real-time status of washers. Moreover, Raspberry Pi cameras and smart plug technology alongside the Tuya IoT platform are utilized to automate and improve the efficiency of the laundry process. The web application incorporates all functionalities with a friendly UI and the integration of other technologies is successful.
Acknowledgements

First, I would like to express my gratitude to Dr Pan Jia for his supervision and the Department of Computer Science for the hardware support.

Second, I would like to express my appreciation for my teammates who has given me support throughout the project.
Figures

Figure 1 Pi Camera V2.1 .......................................................... 9
Figure 2 Raspberry Pi Model 3b+ ........................................... 9
Figure 3 WHD09 Smart Plug .................................................. 10
Figure 4 Assets of the Tuya Cloud Project ............................. 11
Figure 5 Seven Parts of a Seven Segment Digit ...................... 13
Figure 6 Visualization of the Algorithm .................................. 14
Figure 7 System Architecture .................................................. 17
Figure 8 RGB Visualization .................................................... 18
Figure 9 HSV Visualization .................................................... 19
Figure 10 Noise Removal Code .............................................. 20
Figure 11 Raw Images ........................................................... 20
Figure 12 Monochrome Images .............................................. 21
Figure 13 Vertical Result of an Image After Processing ............. 22
Figure 14 Unskewing Pre-Processing Steps ............................. 23
Figure 15 Comparison of skewness ........................................ 23
Figure 16 Noise Removal Steps .............................................. 24
Figure 17 Images with noise ................................................... 25
Figure 18 Fully Processed Images .......................................... 25
Figure 19 Bound Boxes of an Image ....................................... 25
Figure 20 Before and After Dilation ........................................ 26
Figure 21 Code of startocr.py ............................................... 27
Figure 22 Login Screen .......................................................... 31
Figure 23 Authentication using nextauth ................................ 32
Figure 24 Home Page ........................................................... 33
Figure 25 Invalid Input ......................................................... 34
Figure 26 Valid Input ............................................................ 34
Figure 27 Reservations List Page .......................................... 35
Figure 28 QR Code Page ....................................................... 36
Figure 29 Time Remaining Page ........................................... 36
Figure 30 Payment Page ....................................................... 37
Figure 31 Stripe Checkout Page ............................................. 38
Figure 32 Energy Consumption Page .................................... 39
Figure 33 Email Notification .................................................. 39
# Table of Contents

*Abstract* ................................................................................................................................. 2

*Acknowledgements* .................................................................................................................. 3

1. *Introduction* .......................................................................................................................... 6
   1.1 Background ......................................................................................................................... 6
   1.2 Current Solutions .............................................................................................................. 6
   1.3 Related Works .................................................................................................................. 7
   1.4 Objectives ....................................................................................................................... 7
   1.5 Outline ............................................................................................................................ 8

2. *Methodology* ...................................................................................................................... 8
   2.1 Hardware ......................................................................................................................... 8
   2.2 Tuya IoT platform .......................................................................................................... 10
   2.3 SSOCR .......................................................................................................................... 12
   2.4 Frontend ........................................................................................................................ 14
   2.5 Backend ........................................................................................................................ 14
   2.6 Stripe ............................................................................................................................. 15
   2.7 QR Code ......................................................................................................................... 15
   2.8 Authentication ................................................................................................................. 16
   2.9 Related Libraries ............................................................................................................ 16
   2.9 Overall System Architecture ......................................................................................... 16

3. *Results* .................................................................................................................................. 17
   3.1 Image pre-processing ....................................................................................................... 17
      3.1.1 Monochromatic Representation .............................................................................. 18
      3.1.2 Skew Correction .................................................................................................... 21
      3.1.3 Noise Removal ....................................................................................................... 24
      3.1.4 Supplementary Pre-Processing Steps .................................................................... 26
   3.2 Raspberry Pi Python Scripts ............................................................................................ 26
   3.3 Database schema .............................................................................................................. 28
   3.4 API endpoints .................................................................................................................. 29
   3.5 Web App ......................................................................................................................... 30

5. *Conclusion* .......................................................................................................................... 40

6. *Future Plans* ....................................................................................................................... 40

7. *References* ......................................................................................................................... 41
1. Introduction

Chapter 1 is divided into 5 parts. Firstly, a background of the project, followed by current solutions, related works and objectives. An outline for the rest of the paper is given at the end.

1.1 Background

In today’s world, digitalization has become an integral part of our lives. From food ordering to transportation to shopping, digitalization with the use of apps has made everyday tasks more efficient and convenient. However, the laundry process for students in HKU dormitories has been manual and time-consuming. Students must use cash to tap up their cards and physically enter the laundry room and swipe their cards to reserve one. They may also need to wait a long time before being able to use one. In contrast, other universities have digitalized the laundry process in their dormitories with the help of apps. For example, The Hang Seng University of Hong Kong has their own dedicated app where students can check the time remaining of washers, availability and even use e-payment. Many universities abroad also have these kinds of apps for their students such as University of Michigan’s LaundryView app. As digitalization continues, there is much room for improvement for HKU in this field.

1.2 Current Solutions

Universities that have their own laundry apps do not actually build apps from scratch but rely on third party services. For example, LaundryView is a web-based laundry monitoring system created by CSC Service works, a market leader in commercial laundry solutions, and more than 1000 academic institutions in the US (United States) use their services to provide their students with real-time availability of washers that can be accessed online. Although the exact details of LaundryView is not known, both software and hardware components such as Wi-Fi enabled devices or sensors are installed to individual washers to create a network of machines that can be accessed anywhere. These machines then signal data to a secure cloud-
based backend where their insights monitoring application get data from and route those that are useful to user applications.

1.3 Related Works

Students from HKUST [1] and HKU [2] have attempted to create a smart laundry monitoring system in the past years. The group of HKUST students used single-board computers and cameras to detect the time left on machines to assess the status which will be sent to a server for processing and finally to the users app. On the other hand, the group of HKU students used light sensors to detect the status of the doors which then can be used to detect the status of the machines. Although previous works show comprehensive knowledge of the topic, further improvements can be done. For example, one substantial improvement is to incorporate IoT devices and platforms to further enhance and add functionalities. Other useful functions such as reservation, e-payment and notifications were not included in previous works as well.

1.4 Objectives

The project’s objective is to build a web app where students can do different tasks relating to the laundry process and get useful information. The main functionalities of the app include:

- Time checking
- Washer reservation
- Washer availability
- E-payment
- Notifications
- Energy consumption dashboard

Another objective is to successfully incorporate third party services and hardware components such as the Pi camera and the use of the Tuya IoT platform alongside the web app to give the students a complete and seamless laundry experience.
1.5 Outline

The outline of the report is structured as follows. First, Chapter 2 discusses the technologies used in the project. Second, Chapter 3 reveals the results and implementation details of the project. The final Chapters 4 and 5 gives a conclusion and plans that can done to improve the project.

2. Methodology

This chapter discusses the implementation details of the project. Chapter 2.1 discusses the system architecture at a high level. Chapter 2.2 discusses the hardware components and their uses. Chapter 2.3 discusses the SSOCR that is used to retrieve the time in washers and how it works. Chapter 2.4 introduces the Tuya IoT platform and the different APIs that are used in providing the functionalities of the system. Chapter 2.5 and 2.6 discusses the fastAPI, Next.js framework which are used for the app and their benefits. Chapter 2.7 – 2.9 discusses other technologies used in the project. Finally, a system architecture overview is given in Chapter 2.10.

2.1 Hardware

Three hardware components are used in the project: a Raspberry Pi 3 Model B+, a Pi Camera V2.1, and a WHD09 smart plug.

The Pi Camera (see Fig.1) is used to extract the time on the washer’s display which will be shown on the users app. The camera has a still resolution of 8 megapixels and is 25 x 24 x 9 mm, making it a suitable sized camera with sufficient quality to be mounted on the washer for extracting time.
Moreover, the camera is used to scan QR codes provided by users for authentication to allow the students who reserved the washers to use it. Since the camera does not have motorized focus i.e., auto focus, the camera lens is adjusted for close range.

The Raspberry Pi 3 Model B+ (see Fig.2) is a single board computer used to connect the Pi Camera into. The current one has 4GB of memory which is enough for the small tasks it does. Moreover, it has an Ethernet and Wi-Fi port and interface to connect to the internet.

The Raspberry Pi is used to hold the code for image pre-processing before sending the time extracted into a dedicated server. It also sends API calls to the server for various functions such as controlling the smart plugs to switch on/off the washer at appropriate times and QR code validation from the QR code provided by the users.

The smart plug is used for two purposes (see Fig.3). First, to enhance the reservation component’s functionality. Since the project allows only students who reserved the washer to
use it, a mechanism is needed to enforce this. Smart plugs are a cheap and easy to use hardware component that can be used as a proof of concept. Second, to calculate the energy usage for the energy consumption dashboard in the app.

The WHD09 smart plug can be connected to the internet and controlled via the Tuya IoT platform which is discussed later.

2.2 Tuya IoT platform

The Tuya IoT platform is a cloud-based platform that provides many kinds of services and resources for managing and controlling IoT devices. There are several reasons for choosing it. First, it offers integration with many IoT devices from different brands. Second, it offers an easy integration process and the API it offers is well documented thus development is straightforward. Third, it provides a range of security measures such as encryption and authentication. In the project, it is used to control the smart plugs. Integration of the smart plugs to the platform is simple via the Tuya app.
One benefit of the platform is that the smart plugs can be organized in a hierarchy order (see Fig.4). For example, we can create different assets which corresponds to each dorm and group all smart plugs that are contained in the same dorm together. In theory, this can help the administrators of individual dorms maintain the system with ease via the Tuya app. This is even more useful for the web app as it can act as a “second database” where we can fetch all smart plugs corresponding in a dorm and show those only in the web app. For example, if student A lives in XYZ dorm and he logins to the web app, only those smart plugs will be shown. Note that in the project, the smart plugs will correspond to each individual washer. The ideal scenario would be to use IoT washers directly but clearly this is not feasible.
Several of APIs from the platform are used:

<table>
<thead>
<tr>
<th>Route (base URL is same for all endpoints)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>/v1.0/iot-02/assets/{asset_id}/devices</td>
<td>to receive all devices in the corresponding asset e.g., receive all smart plugs corresponding to each dorm since each asset will act as a dorm.</td>
</tr>
<tr>
<td>/v1.0/iot-03/devices/{plug_id}/commands</td>
<td>to control a specific plug e.g., switch on/off</td>
</tr>
<tr>
<td>/v1.0/iot-03/devices/6c3e8b0f824770c4dfvcra/status</td>
<td>to receive data about the plug e.g., power</td>
</tr>
</tbody>
</table>

Tuya provides a Python and Node.js SDK for the APIs with the tuya-connector library. To send calls, an access id and access key is used which can be retrieved from the Tuya cloud project. One feature of the app is to show the energy consumption of the user. Although it might not be necessary to record the energy usage of washers alone, it could be a especially useful feature if other device’s energy consumption is integrated therefore it is a feature worthwhile doing. Retrieving cumulative energy consumption requires subscription thus is not viable. To get the cumulative energy consumption of a user for a particular day, the Raspberry Pi will send periodical calls e.g., every minute to retrieve the current voltage and multiply the current voltage with the time interval until the end time of the reservation. Finally, it will update the user’s energy consumption for that day.

2.3 SSOCR

Seven Segment Optical Character Recognition or SSOCR for short is used to extract the time remaining of washers. This is suitable for the project as it is specifically used to read from seven segment displays which are commonly found in electronic devices like the display panel of washers.

Initially, TesseractOCR, a different OCR software, was considered as an option for the project. However, it was found that TesseractOCR does not have the necessary training to accurately recognize seven segment fonts, which are a key requirement for the project. As a result, it was deemed unsuitable for the project and SSOCR was selected for the task instead.
While TesseractOCR could potentially be trained to recognize seven segment fonts, it is not necessary to use machine learning for this particular task. Recognizing seven segment digits is a simple task that can be accomplished using other techniques.

SSOCR does not use machine learning but rather a technique similar to template matching. The process is divided into two main parts, segmentation and character recognition. Segmentation is the processing of dividing the image into several regions while character recognition is the actual process of recognizing what digit each region is. One caveat of SSOCR is that images fed must be in ideal condition. For example, it must be monochrome, noise-free and unskewed due to the SSOCR’s simple algorithm of detecting seven segment digits. As a result, image pre-processing steps are carried out to make sure the image is ideal and will be discussed in later sections. Before explaining the algorithm, readers should be aware that every seven-segment digit are formed by 7 possible parts (see Fig.5).

![Figure 5 Seven Parts of a Seven Segment Digit](image)

During segmentation (see Fig.6), the algorithm scans the image from the left until the first foreground pixel i.e. pixel of a digit is found. This is considered as the left border of the object. It will continue until it a column with only background pixel is found. This is considered as the right border of the object. This process is repeated to find the specified number of digits, or until no more digits are found. The process is done vertically to find the top and bottom as well. The only difference is that gaps are allowed vertically as certain digits have the middle part unset like the digits 0, 1 and 7.
During character recognition, the algorithm aims to recognize what digit an object is by looking at which parts are included and not included. For example, the number 1 has two parts while the number 8 has all 7 parts. To find the three horizontal parts, a vertical scan is done from the centre top the digit to find the three horizontal parts. Foreground pixels in the upper 1/3, middle 1/3 and bottom 1/3 are counted as part of the top, middle and bottom part respectively. For example, if a digit does not have any pixels in the middle 1/3 then the digit does not have a middle part. A similar process is done to find out whether the digit has the other 4 parts i.e. 2 on the left and 2 on the right. The recognized segments are then used to identify the displayed digit using a table lookup (implemented as a switch statement).

2.4 Frontend

Next.js is used as the frontend framework for the web app. It is a popular open-source framework for building React applications. Next.js has many benefits such as server-side rendering and automatic code splitting for a smoother user experience and provide convenient features such as a file-based routing system. The use of React makes developing and maintaining apps much easier as it uses a component-based architecture where the user interfaces can be broken down into smaller and reusable components.

2.5 Backend

FastAPI, a fast and modern backend framework, is used for the project. FastAPI has several benefits. For example, it is a Python framework meaning it is a more expressive language making it more readable, easier to debug and enable developers to focus more on application logic rather than syntactical issues. Moreover, it has automatic validation of incoming data from requests and automatic data serialization i.e., developers can return Python objects
directly at the endpoints without explicitly converting them into JSON which makes development more convenient.

MySQL is used as the database since data needs to be stored such as reservation and student information etc. The reason for using an SQL database is that the kinds of data stored are structured and transactions are processed in a secure and reliable manner compared to NoSQL databases.

2.6 Stripe

Stripe is a popular payment processing platform that handles online payments securely and smoothly therefore it is a suitable platform to use for the e-payment system of the project. Moreover, it provides various payment methods such as Apply Pay, Google Pay and many more. It opens a wider range of choices for students to pay for their laundry rather than only using cash. For simplicity, the project only uses credit card. Stripe also offers APIs for developers to integrate the platform into their apps and also provides many powerful features.

One feature that is used is the products feature of Stripe. It allows developers to define many kinds of products to sell. Developers can easily adjust the pricing model, price, type of product i.e. recurring or one time etc. All products of the app will be in the form of a top-up payment. For simplicity, the top-up choices provided in the app are HKD 20, HKD 50 and HKD 100. Another feature are web hooks provided by Stripe which are URLs that are called upon successful payment. This feature can help update the database to record the correct balances of the users in a secure manner.

2.7 QR Code

QR codes, short for Quick Response codes, are two-dimensional barcodes that can be read by a QR code scanner or smartphone camera. QR codes are designed to store information in a way that is easily read and processed by machines. The information stored in a QR code can include text, URLs, contact information, and more. They are easy to use and flexible thus are used in many ways such as payment and advertising. They are used to validate washer
reservations made by students for the project. The information encoded will be a 128-bit value to identify each reservation with the help of the uuid library.

2.8 Authentication

NextJS authentication feature is a feature of the NextJS framework and is used in the project. It comes with various authentication providers such as Auth0, Firebase, Ory etc. Auth0 (Google Authentication) is the authentication method used in the project. It is a cloud-based identity management system. Traditional login systems require developers to create a customized front-end to collect the password and pass it to the server for storage. This requires developers to create a suitable front-end and make sure the back-end is secure. However, Auth0 is a universal solution which provides a secure infrastructure to store data and a polished login interface. Moreover, it provides different authentication methods such as MFA or through social providers.

2.9 Related Libraries

The picamera and pyzbar Python libraries are used to control the Pi Camera programmatically and read QR codes respectively. picamera is the standard library for controlling Pi Cameras and provides a simple and intuitive interface. The pyzbar library, on the other hand, is a Python wrapper for the ZBar barcode scanner library. It provides a simple and efficient way to decode QR codes and barcodes from images and video streams.

2.9 Overall System Architecture

A simplified summary of the system architecture is shown below (see Fig.7). All data from the front end will be retrieved through the FastAPI server from the MySQL database and Tuya IoT Cloud. The FastAPI server will interact with the Pi and smart plugs as well as necessary.
3. Results

This chapter discusses the progress and results. Chapter 3.1 focuses on the results of the pre-processing steps on images. Chapter 3.2 goes through the python scripts used in the Raspberry Pi. Chapter 3.3 and 3.4 explains the creation of the database schema and endpoints respectively. Chapter 3.5 walks through the web app and its workflow.

3.1 Image pre-processing

This section discusses the image pre-processing steps before feeding it to SSOCR. As stated earlier. The images have to be monochrome, noise-free and unskewed. A black and white representation is needed by the algorithm to identify the digits i.e. foreground pixels from the background i.e. background pixels. Any foreground pixels will be counted as part of the digit. A noise-free image is needed since the noise may be represented as part of the digit by the algorithm causing it to output the wrong digit. Many washer displays feature light indicators that use the same colour as the time on the display so if the display is converted to monochrome (e.g., turning the time to white and the rest to black), the noise from the light indicators would also be converted to white, which will tamper the results. The skewness may not be a big problem as ideally the camera would be mounted in a fixed position. However it is a good measure to make sure that it is not skewed during the pre-processing steps just in case the camera is tilted.
Some test images are pre-processed to show the effectiveness of the pre-processing steps. These images are used as they possess undesired properties such as noise and they display seven segment digits similar to the washer display.

3.1.1 Monochromatic Representation

The first step is to change the image to black and white. To achieve this, the colour of the time displayed should be changed to white or black and the rest to be the opposite colour. First, we perform masking on the image. It is a process where pixels turn white or black depending on the condition. In this case, all pixels turn white if it is the same colour as the time displayed. However, a representation of the colour must be known beforehand to use it as a parameter. There are many ways to represent a colour, also known as colour space. All colour space represents a colour using a set of numerical values that describe a colour’s property.

One popular colour space is RGB (see Fig. 7) which stands for red, green and blue respectively. It describes a colour based on the intensity of these three colours. Each value can be from the range 0-255 with 255 being the most intense. For example, a purely red colour would have a value of 255,0,0 for its red, green and blue intensity value respectively.
HSV is another colour space which stands for hue, saturation and value respectively (see Fig.8). Hue represents the colour tone, saturation represents the purity and value represents the brightness. Applications use different scales for HSV, for example, GIMP uses H=0-360, S=0-100, V=0-100 while OpenCV uses H=0-179, S=0-255, V=0-255.

HSV is used to extract the displayed time instead of the RGB colour space because HSV is more resistant to lighting changes meaning that shades of the same colour will have similar hue values but a much different RGB value. Moreover, since the colour information is separated from the luminance, it makes it easier to define the colour range that is needed.

Using a few lines of code, we can successfully turn an image to black and white (see Fig.9). Masking is done to images to turn all pixels of a specific colour white and the rest black. For example, the colour of the time used in one of the raw images is an orange colour (see Fig.10) and the HSV range used is from [0,10,190] to [40,200,255]. The range covers all the possible orange shades of the time. Although HSV is easier to use than RGB, in some cases unnecessary spots/shapes become foreground pixels thus noise removal is done to fix this issue.
Figure 10 Noise Removal Code

```python
hsv=cv.cvtColor(image,cv.COLOR_BGR2HSV)
# Define lower and upper limits of the number colour
number_color_lo=np.array([0,10,190])
number_color_hi=np.array([40,200,255])
# Mask image to only select colors in the range
mask=cv.inRange(hsv,number_color_lo,number_color_hi)
# Change those points to white
image[mask>0]=(255,255,255)
image[mask<=0]=(0,0,0)
```

Figure 11 Raw Images

Before:

![Before Image](image1)

After:
3.1.2 Skew Correction

The next step is skew correction. A simple way to fix skewness is to find the angle of rotation of one of the digits and rotate the image using the angle. This is suitable as all of the digits will have the same angle of rotation if it exist. However, one important assumption made is that largest object in the black and white image is a digit. With the assumption, we can get all contours in the image using the OpenCV findContours function and use the largest one to get the angle of rotation. This is a reasonable assumption as usually the time on the display of washers is usually the largest. Usually the only other contours besides the digits are the light indicators such as those in image 1. After getting the contour of the digit, the minAreaRect function of OpenCV is used which returns the angle of rotation. The minAreaRect function also gives the minimum bounding box of the contour. The angle is then used to rotate the image to make it straight. The angle is the angle between the x axis of the image and the longer axis of the rectangle.
However, one issue is that the resultant image after rotating may cause the time to be shown in a vertical format (see Fig.12) since the unskewing step considers a vertical or horizontal orientation both to be correct as long as it’s not skewed. To solve this issue, if the width of the minimum area rectangle is more than the width then it is an indication that the time is shown in a vertical format. Therefore the image is rotated by 90 or -90 degrees once again depending on the slope of the minimum area rectangle. After getting the correct angle, we apply the rotation matrix to the image. The code for this step is shown below (see Fig.13).
The result of this pre-processing step is shown below. As shown in the image, the image is much less skewed compared to the raw image shown at the beginning.
### 3.1.3 Noise Removal

The final step is to remove all the noise i.e. all objects in the image that is not a part of the time. As stated earlier, these noise will be treated by the SSOCR algorithm as digits and may tamper with the results. There are many possible noise such as the light indicators in image 1. Moreover, the colon in between numbers i.e. ‘:’ should be removed as this will affect the results as well. Therefore the output of image 3 after pre-processing should be 0457. The time will need to be reformatted into 04:57 i.e. MM:SS later on.

```python
# Define the area and height of the digit's bounding box
digit_area = None
digit_height = None
# Find the sorted contours because the position of the digits are changed after fixing skewness
contours, hierarchy = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
area = map(cv2.contourArea, contours)
area_index = sorted(area_index, key = lambda x: x[0], reverse = True)
# New black image to paste only digits into to feed to SSOCR
new_canvas = np.zeros(image.shape, dtype=np.uint8)
for i in area_index:
    # Get bounding box of contour
    rec = cv2.boundingRect(contours[i[1]])
    # Assume the biggest contour is a digit and get its box height and area
    if digit_height == None:
        digit_height = rec[3]
        digit_area = rec[2]*rec[3]
    # Check if its a digit based on the bounding box
    if rec[1] < digit_area * 0.09 <= rec[2]*rec[3] <= digit_area * 1.1:
        # If passed then paste the digit into a new black image to feed to the sssoc
        new_canvas[rec[1]:rec[1]+rec[3], rec[0]:rec[0]+rec[2]] = image[rec[1]:rec[1]+rec[3], rec[0]:rec[0]+rec[2]]
# Feed the pre-processed image to SSOCR to extract time
final_image = image_fromarray(np.array(new_canvas))
final_image.save("final_image.jpg")
cmd = "socr --foreground-white --number-digits=1 final_image.jpg"
output = subprocess.check_output(cmd, shell=True)
```

**Figure 16 Noise Removal Steps**

The main idea to remove any non-digit foreground pixels is by drawing bounding boxes to each possible contour in the image and determine which are the digits we want to extract using the SSOCR and which are just other visual elements that should not be included by comparing their bounding boxes. OpenCV provides the boundingRect function to extract bounding boxes. Clearly, all the digit’s bounding box will have the same area and height except for the number one. However, we can identify the number one if it has the same height and about 1/4 the area of other digit’s bounding box. In this step, the contours must be recalculated as the image might have been rotated to fix skewness from the previous step thus the positions of the digits could have been changed. A new all black image is also created to
paste the digits into. This is the final image we use to feed to the SSOCR i.e. all preprocessing steps are done and can be fed to the SSOCR. Since the bounding boxes of all digits may not be exactly the same, a range is used instead to identify bounding boxes of digits as shown in the code. The –number-digits=–1 flag indicate that the number of digits in the image will be automatically identified and the –foreground=white flag indicates that the foreground is white.

Before:

![Images with noise](image1)

Figure 17 Images with noise

After:

![Fully Processed Images](image2)

Figure 18 Fully Processed Images

For reference, these are the identified bounding boxes for one of them images.

![Bound Boxes of an Image](image3)

Figure 19 Bound Boxes of an Image
3.1.4 Supplementary Pre-Processing Steps

There are some scenarios where more pre-processing must be done. For example, depending on the camera quality, gaps could be found within the digits. Therefore, the bounding boxes will be incorrectly placed. A simple solution is to dilate the image.

Before and After Dilation:

![Before and After Dilation](image)

*Figure 20 Before and After Dilation*

3.2 Raspberry Pi Python Scripts

Two scripts are run in the Pi (search_qr.py and start_ocr.py). search_qr.py continuously waits for a QR code provided by the student and the control is switched to start_ocr.py after the QR code is validated from the server. The start_ocr.py is responsible for continuously extracting the time left from the washer display into the web app for viewing. start_ocr.py ends upon reaching the end date stated by the reservation. The control is handed back to start_ocr.py to wait for the next QR code for the next reservation.
With the help of various libraries, the process for scanning QR code i.e. `search_qr.py` is wrapped in an infinite loop. The image is initially in the form of a byte object then turned into a PIL.Image object. The image is then used as a parameter for the decode function of the `pyzbar` library to extract the decoded value of the QR code. Finally, it is sent to the server for validation and depending on the status code, the time extracting process starts and the plug is switched on for a 204 status code. Otherwise, nothing will happen and the QR code searching process continues until the next student has a valid QR code for a reservation. The end date is also returned by the server to indicate how long `start_ocr.py` should run if the QR code is valid. When the QR code is valid, a API call to Tuya is made to switch on the smart plug.
using the smart plug id retrieve from the previous API call to the FastAPI server. If the call to the smart plug worked then start_ocr.py will start.

The start_ocr.py script has a similar structure as the search_qr.py script. However, it does not run infinitely and only runs for a specific amount of time i.e. until the end date stated by the reservation. The API call is also different as it needs to update the database with the correct time remaining instead of validating QR codes. The image pre-processing step introduced in section 3.1 is also a part of the start_ocr.py script.

3.3 Database schema

The database schema includes four tables: students, reservations, prices, payments and energy_consumption. The students table contains the student information. It has four fields, an email field which is their university email to uniquely identify each of them, a dorm_name field which is the name of their dorm and dorm_id which is the asset id of that dorm stored in the Tuya cloud. The reason for this field is that when a student logs in, the app should show only the washers of their dorm and this can be achieved by sending an API call to the Tuya IoT platform with the dorm_id field which returns the information of relevant washers. The last field is the balance which is updated on every Stripe payment.

The reservation table contains information of each individual reservation made by each student. The field includes an email field which is their university email to uniquely identify each of them, a start_date and end_date field corresponding to the starting datetime a student can use the washer and ending datetime in which the student can no longer use it, both are in the format of YYYY:MM:DD HH:MM:SS, a qr_code field that is used to “enable” the student to use the washer via the switching on of smart plugs, a washer_id field that identifies each washer (technically the smart plug id), a time_remaining field which is the current time displayed on the washer and a status field which reflects the current state of the reservation. The status field has three possible values: incomplete, in_progress and completed and is used to let the students know the status of their reservations in the app.
The prices table simply contains the prices of washer per time. For testing, the prices are arbitrarily chosen. The energy_consumption contains information about the user’s energy usage. It has three fields, date, email and energy. The energy usage for a day is calculated by periodically sending an API call to the server to record the cumulative usage. Lastly, the payments table contains the history of the user’s payments to top up his account. It contains 4 fields, an idea which represents a unique payment, the amount, email and the date of the payment.

3.4 API endpoints

To populate data in the web app, API endpoints are defined as follows:

<table>
<thead>
<tr>
<th>Route</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>/users/{email}/dorm-info (GET)</td>
<td>to get the user’s dorm information</td>
</tr>
<tr>
<td>/users/{email}/balance (GET)</td>
<td>to get the user’s balance</td>
</tr>
<tr>
<td>/users/{email}/transactions (GET)</td>
<td>to get the user’s payment history</td>
</tr>
<tr>
<td>/washers/{washer_id}/price (GET)</td>
<td>to get the price of individual washers</td>
</tr>
<tr>
<td>/reservation/check (GET)</td>
<td>to check if the given dates given by the user overlaps with existing reservations</td>
</tr>
<tr>
<td>/reservation (POST)</td>
<td>to insert a new reservation</td>
</tr>
<tr>
<td>/reservations/{email} (GET)</td>
<td>to retrieve the list of reservations made by the user (to show in the app)</td>
</tr>
<tr>
<td>/reservation/{qr_code} (DELETE)</td>
<td>to delete a reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/time (GET)</td>
<td>to get the current time remaining of the washer for that reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/time (PUT)</td>
<td>to update the time remaining of the washer for that reservation</td>
</tr>
<tr>
<td>/reservation/{qr_code}/status (PUT)</td>
<td>to update the reservation status e.g. reservation changes to in_progress after QR code validation.</td>
</tr>
<tr>
<td>/create-checkout-session (POST)</td>
<td>to create a checkout session for the user to top up money</td>
</tr>
<tr>
<td>Route</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>/payment (POST)</td>
<td>to record the payment by the user after checking out and update the users balance</td>
</tr>
<tr>
<td>/email (POST)</td>
<td>to send an email notification upon completion of laundry</td>
</tr>
<tr>
<td>/energy (PUT)</td>
<td>to update the energy usage of a person for a specific day</td>
</tr>
</tbody>
</table>

Best practices for API design were taken in consideration. For example, consistent and clear names were used and a hierarchical structure for the routes were used. For example, all routes handling reservations starts with the reservation keyword. Moreover, proper methods were used to reflect the actual usage of the routes e.g. PUT for updating and POST for creating. Proper HTTP status codes are also returned from the endpoints. For example, a 204 No Content is returned for endpoints that are successful but have no return values, a 404 Not Found to indicating the item is not found in the database and a 500 Internal Server Error indicating possible network issues on the server side. URL components were encoded and the proper use of path and query parameters were taken into consideration. For example, path params were used for indicating a specific resource while query params were used for filtering.

### 3.5 Web App

In this section, the web app and its workflow is introduced.

The login screen features a sign in button that brings the user to login using their HKU email using next auth introduced earlier (see Fig.21 and Fig.22). Upon logging in, the app stores a session in the server such that he/she is logged in automatically every time he/she visits the app unless the browser is closed.
Figure 22 Login Screen
Upon logging in, users are greeted with a home page (see Fig.23) showing all the washers found in their dorm using the Tuya API as explained in previous sections. There are three components to every page in the app, a header, sidebar and the main content area. The header contains the email of the user, their respective dorm, their current balance and a sign-out page. The balance is newly reflected after every top up payment. The sidebar contains 4 links which are “Washers”, “Your Reservations”, “Add Payment” and “Energy Consumption.” The “Washers” page is the initial page loaded upon login i.e. home page. The “Washers” or home page contains different sections corresponding to each of the washers. The price and current availability are shown for each washer. Note that the availability refers whether the washer is being reserved currently only. Users may click on it to reserve it at other times.
Upon clicking on one of the washers, the reservation form is brought up. In the reservation form, users can choose the date and time of their choice. However, there are some constraints for a valid input and a message will be shown at the bottom showing if its valid or not (see Fig.24 and Fig.25). Valid inputs must have the following properties:

- The date, start time and end time fields must not be empty.

- The start time must precede the end time.

- The start time must be later than the current time.

- The reservation part must not overlap with other reservation period which can be known from the server.
Submitting an invalid reservation will not go through and upon a successful one, users will be redirected to the “Washers” / home page. If the user booked a period with start time == current time, then the washer he/she booked will be marked as unavailable. Moreover, if the user does not have enough balance, then a screen warning the user to add balance will be shown instead.
Users can also visit the “Your Reservations” page where the list of reservations the user has booked are listed. Each reservation will contain the washer reserved, the time period of the reservation and the status of the reservation. There are 3 different possible status, Not Completed, In Progress and Completed. Not Completed refers to when the user has not start using the washer because it is not his/her turn yet or when it is his/her turn already but have not provided the QR code to use the washer. Once the user has provided the QR code then the status will be switched to In Progress. Upon reaching the end time of the reservation, the status will finally become Completed.

Users clicking each of the reservations will show different behavior depending on the status (see Fig.27 and Fig.28). If it is In Complete, then a QR code will be shown to the user since he/she has not started using the washer. If it is In Progress, the time remaining of the washer is shown instead. For Completed reservations, nothing will be shown but the user can delete the reservation and it will be erased from the database.
Figure 28 QR Code Page

Figure 29 Time Remaining Page
The “Add Payment” page is where users can top up their account to use the washers (see Fig.29). As mentioned previously, users can add HKD 20, HKD 50 or HKD 100. A table of their payment history is also recorded. The ID is the last 6 characters of the payment id generated by Stripe. The actual length is much longer but is cut down for presentation purposes. The payment history is all recorded in the database for every user.
Upon clicking one of the top-up options, the front end sends a request to the back end to retrieve a checkout session id and the front end redirects the user to the checkout page using the id (see Fig. 30). Upon filling the payment information, the users are brought back to the home page. Moreover, after successful payment, Stripe calls one of the web hook APIs defined in the project (i.e. /payments) to record the new payment and update the balance of the user in the database.
The “Energy Consumption” page (see Fig. 31) is where users can see their energy usage and is recorded daily with the unit of the Y axis being watts. The total and average energy consumption is also shown and some other information at the bottom.

Figure 32 Energy Consumption Page

The “Energy Consumption” page (see Fig. 31) is where users can see their energy usage and is recorded daily with the unit of the Y axis being watts. The total and average energy consumption is also shown and some other information at the bottom.
An email notification is sent out (see Fig.32) every time the Raspberry Pi detects a 00:00 time during the time extraction using the SSOCR. It is not sent out on the end time because a user may use the washer multiple times depending on the reservation time length. The email is sent out using a Google service via smtp.

5. Conclusion

With digitalization becoming more popular in recent years, the project’s aim is to create a system for HKU dorm students to efficiently and conveniently do laundry tasks. The product is in the form of a web app where students can check the time remaining of a washer, do reservation, use e-payment and many more. Moreover, the web app utilizes modern frameworks such as Next.js and FastAPI. The Raspberry Pi, Pi camera and smart plugs are used to extract washer information and enforce the reservation mechanism. These hardware provides a low cost but robust solution to these problems. Software components were also used such as the Tuya IoT platform to control the smart plugs and retrieve voltage information, Stripe platform to handle e-payment and Google services to send notifications. Testing of both the OCR mechanism and web app functionalities has been tested and show good results thus the concluding the project to be a success.

6. Future Plans

Although the project is successful, many improvements can be made to the projects. For example, the app could use the actual data from HKU such as the actual list of students living in dorms. As devices could be integrated as well such as the AC, printers, dryers etc to leverage app and centralize the different devices and machines into one app. This could greatly improve the use of the app and bring more convenience to the lives of dorm students. However, the methods of could be different for other machines. Other important functionalities that could be introduced is an admin page for staff, a feedback/complaints page and a mobile version of the app.
7. References
