Queuetie!

Our IoT-Enhanced Campus Dining Solutions

50.046 Cloud Computing and Internet of Things

Project Report

Guo Ziniu (1004890) Meng Fanyi (1004889)

Wang Shiyu (1004874) Wang Zhuoran (1004867)

Table of Contents

1. Problem Framing

1.1. Background

Navigating the bustling landscape of campus life, particularly within the confines of a compact canteen space, highlights the critical importance of efficient crowd monitoring and management. In the absence of a streamlined crowd management method, the current state of affairs in canteens often translates into congestion, difficulties in finding available seating, and the emergence of prolonged queues, especially during peak hours.

The existing challenges stem from the reliance on outdated and error-prone traditional methods for monitoring traffic within canteens. The absence of an efficient system exacerbates issues such as overcrowding and extended waiting times, directly impacting the overall dining experience for students and staff.

1.2. Problem Statement

Traditional methods of monitoring traffic in canteen are outdated, error-prone and inefficient. The lack of historical data makes predicting peak times difficult, leading to overcrowding and extended waiting time. As a solution, we propose a real-time camera-based system that accurately monitors the number of people in a canteen to make crowd management more cost-effective and accurate.

1.3. Potential Use Case

Beyond its immediate application in canteens, the proposed system holds the versatility to extend its utility to various locations. It stands as a versatile solution capable of monitoring queues in diverse settings. The technology's adaptability allows it to be seamlessly integrated wherever the flow of people demands tracking and efficient management. This flexibility positions our real-time camera-based system as a comprehensive and scalable solution for optimising crowd management across a spectrum of environments.

2. Application Design and Development

2.1. Architecture Design

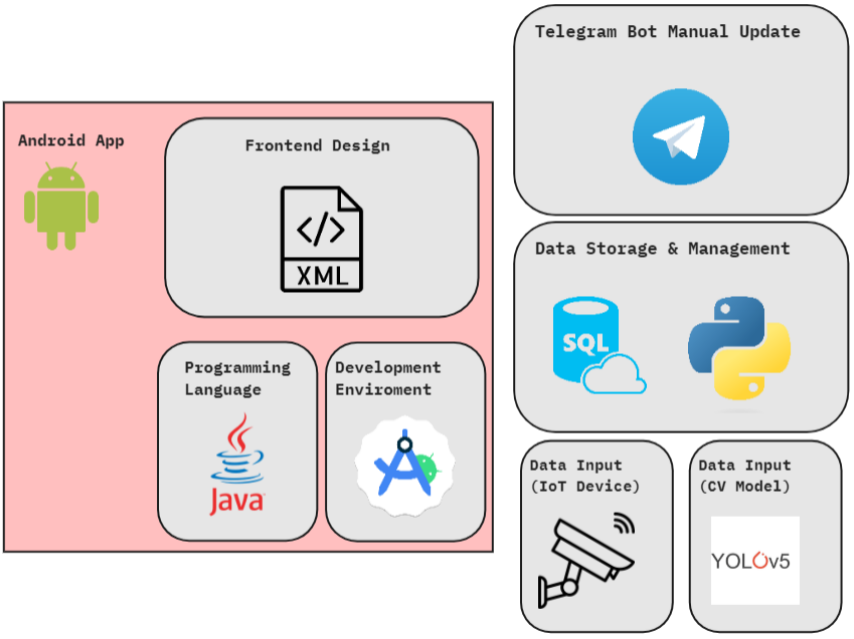

Figure 1: Project Architecture Design

Our system comprises an Android app and a Telegram bot as frontends, both connected to a Python-based backend server. The server offers APIs for seamless access to the MySQL database from the Android app. The IoT device, equipped with a camera for image capture, serves as the primary data source.

The IoT device communicates with the Python server, where image data is processed by a Computer Vision (CV) model. Processed data is periodically populated into the MySQL database. This structured approach enables efficient data retrieval through the Android app, forming the foundation of our IoT-Enhanced Campus Dining Solutions. This architecture supports real-time monitoring, analysis, and historical data availability for predictive analytics, contributing to effective crowd management.

2.1.1. Hardware Setup and Operation of the IoT Camera

The system operates by ingesting streaming videos as input, a function facilitated by webcams due to their cost-effectiveness and ease of installation. Our design involves

situating one webcam at the canteen entrance to monitor the overall foot traffic and another in front of each vendor to capture queue dynamics.

While initial considerations included using Raspberry Pi as the control unit for each camera, the decision was made to employ a computer as the control unit during the demo. This adjustment optimises performance, especially given that the server runs locally rather than in the cloud. This hardware setup ensures the efficient functioning of our IoT camera system, contributing to real-time video capture and analysis for enhanced crowd monitoring and management in the campus dining environment.

2.1.2. Computer Vision (CV) Model

The purpose of integrating CV models is to accurately identify both the number of people in a designated area and assess queue dynamics by processing video feeds from the cameras. Throughout our exploration, we delved into both traditional and contemporary human detection models, including HOGs and Faster RCNNs. Following a comprehensive performance evaluation, considering the real-time nature of our live stream data, we opted for YOLO (You Only Look Once), a cutting-edge object detection model renowned for its exceptional speed and accuracy (Yawar, 2023).

Given budgetary constraints, we made a strategic decision to run the CV model locally, foregoing the use of cloud services. This approach ensures both cost-effectiveness and efficient real-time processing, aligning with the practical considerations of our project implementation.

2.1.3. Data Storage

In our quest for an optimal data storage solution, we initially explored Azure Data Lake due to its prowess in handling vast datasets, especially pertinent for day-to-day video streaming generating large datasets. Acknowledging its robust security features and seamless integration with big data tools like Apache Hadoop and Apache Spark, it appeared to be a promising choice. However, the prohibitive cost and potential latency associated with storing raw or semi-structured data prompted us to seek alternative options.

Enter Azure SQL Database, a relational database finely tuned for structured data. By transforming data from the CV module into tables with predefined schemas, we significantly

mitigate costs. Querying the data through SQL proves more straightforward compared to navigating the intricacies of Hadoop or Spark. To facilitate historical data analysis for estimating and predicting waiting times and the number of people, we design several tables:

- **Table Feature1**: Stores the direct output of the CV model, detecting the number of people in a canteen area, with the 'time_only' attribute for simplified queries.
- **Table Feature3_Info**: Captures the current operational status of a stall, regularly updated by vendors.
- **Table Stall1_Receipt**: Records the timestamp of each receipt, serving as the service timestamp for each customer, retrievable from vendors' bank statements.
- **Table Stall1_Cam**: Archives the direct output of the CV model, detecting the number of people in front of a stall.
- **Table Stall1_Sale**: Documents the total daily orders for a stall, serving as a predictor for future sales. This structured approach to data storage ensures efficiency in both data retrieval and analysis, integral to the success of our IoT-Enhanced Campus Dining Solutions.

Attribute	Format	Description
timestamp	datetime	Primary key
weekday	int	From 1 to 7, representing Monday to Sunday
time_only	time(7)	Time when video frame is taken, 10:00-19:00
number_of_people_detected	int	Number of people captured by the cameras

Table 1: Design of Table Feature1

Attribute	Format	Description
vendor	varchar(20)	Vendor name

menu	varchar(200)	List of dishes provided by the vendor
availability	bit	0: closed, 1: open

Table 2: Design of Table Feature3_Info

Attribute	Format	Description
receipt_timestamp	datetime	Receipt timestamp
weekday	int	From 1 to 7, representing Monday to Sunday
time_only	time(7)	Time of receipt, 11:00-19:00

Table 3: Design of Table Stall1_Receipt

Attribute	Format	Description
timestamp	datetime	timestamp
weekday	int	From 1 to 7, representing Monday to Sunday
time_only	time(7)	Time of receipt, 10:00-19:00
number_of_people	int	Number of people captured by the cameras

Table 4: Design of Table Stall1_Cam

Attribute	Format	Description

record_date	date	
weekday	int	From 1 to 7, representing Monday to Sunday
tot_sale	int	Number of receipts on record_date

Table 5: Design of Table Stall1_Sale

2.2. App Feature 1: Real-Time Monitoring

2.2.1. Highlights

Feature 1 provides real-time monitoring data of the number of people in the canteen to users. The data recorded by the IoT device will be populated into our SQL database, which will eventually be reflected on the traffic history graph.

Figure 2: Demonstration of Feature 1 - Real Time Monitoring

2.2.2. Estimation of Total Number of People

It's unrealistic to install cameras to cover all areas of the canteen, which will also brings multi-camera crossing problems. Instead, we capture the people in an area of the canteen. By assuming that the number of people in a region is in proportion to the area of the region, we can estimate the total number of people in the canteen by counting the number of people in this region via CV model.

2.3. App Feature 2: Crowd Prediction

2.3.1. Highlights

Feature 2 is designed to enhance the dining experience by providing predictive insights into the crowd dynamics within the canteen. Through an intuitive user interface (UI), individuals can select their preferred date and time for a visit to the canteen, and the system generates an estimate of the expected number of people at that specific moment.

The objective of this feature is to empower users with valuable information for better crowd management. By proactively predicting busy periods, individuals can strategically plan their meals to avoid inconveniences such as a lack of available seats and lengthy queues. This predictive functionality not only optimises the overall dining experience but also contributes to the efficiency of canteen operations, ensuring a more seamless and enjoyable dining environment for all users.

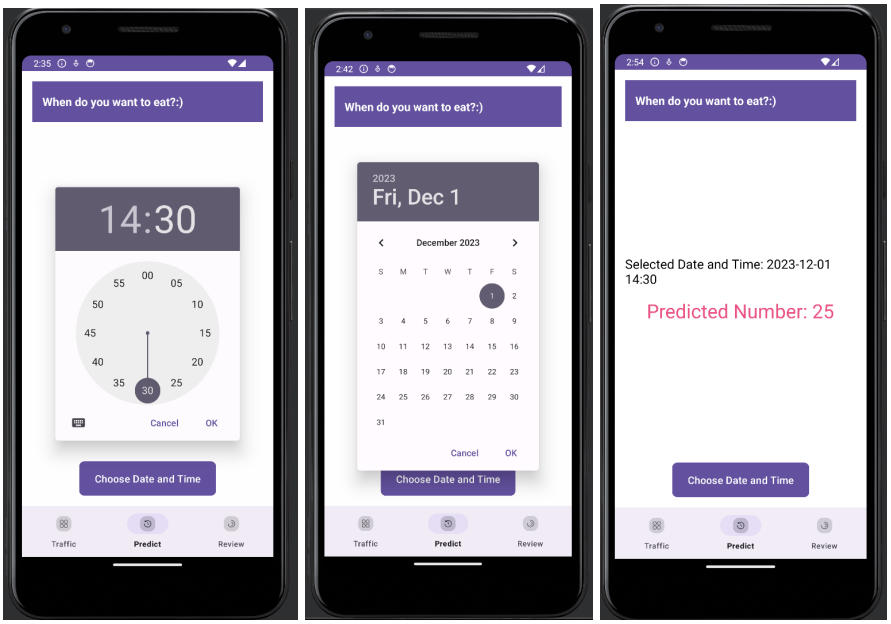

Figure 3: Demonstration of Feature 2 - Crowd Prediction

2.3.2. Prediction of Total Number of People

The calculation leverages historical data stored in the database, considering the day of the week specified by the user. For instance, if a user selects a Monday, the system calculates the average number of people present in the canteen during Mondays at the chosen time. This calculation is based on the assumption that the distribution of crowd numbers during the operating hours of each day follows a Poisson Distribution.

2.4. App Feature 3: Dining Insights

2.4.1. Highlights

Feature 3 provides real-time information on the availability of different food stalls and estimated waiting times, enhancing the dining experience by reducing unnecessary waiting.

Vendors are able to update their menu and availability any time via a Telegram bot. It will also show the predicted number of total customers, minimising waste and ensuring availability during peak hours.

The Telegram bot handler is on the same server where the Python backend is configured so that it can notify the server to return different results when a modification is made.

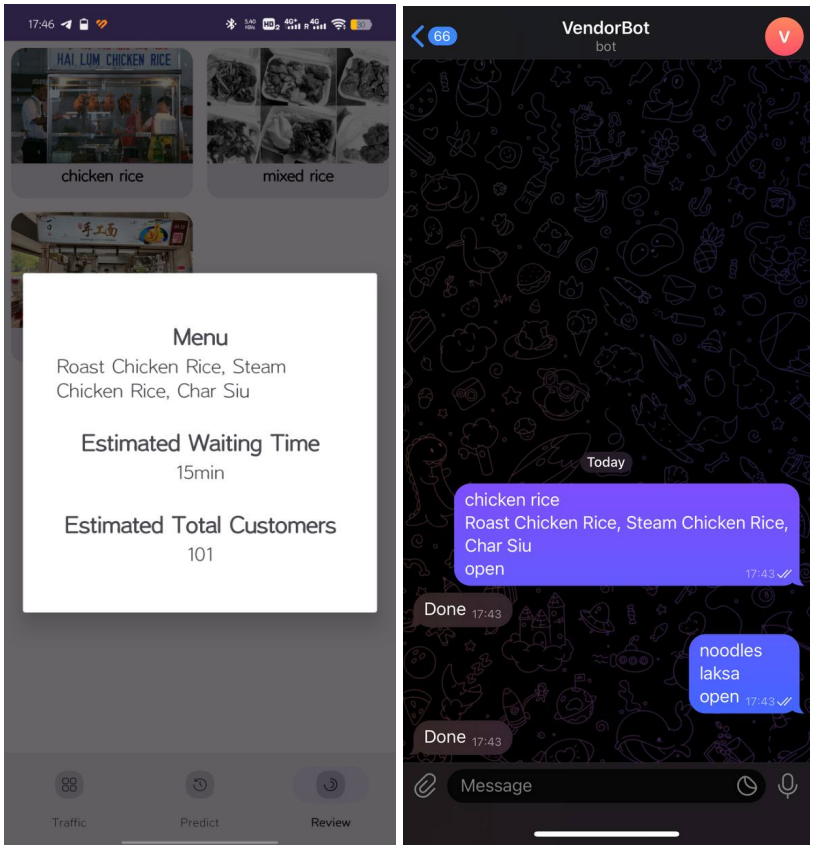

Figure 4: Demonstration of Feature 3 - Dining Insights

2.4.2. Queuing Model: M/M/1

In developing the mathematical simulation model for queuing dynamics, we have made key assumptions to streamline the representation of the system. These assumptions provide the foundation for an M/M/1 queuing model:

1. Poisson Arrival Process: We assume that the arrival of individuals to the queue follows a Poisson process. Also, we assume traffic on the same weekday follows the similar distribution. This means arrivals are random and occur independently of each other, with a constant average rate over time.
2. Exponential Service Times: The service times within the queue are assumed to be exponentially distributed. This aligns with the Markovian nature of the M/M/1 model, where service times are memoryless and the probability of service completion is independent of the duration already spent in service.
3. First-Come, First-Served Discipline: The queuing system adheres to a First-Come, First-Served (FCFS) discipline. Individuals are served in the exact order of their arrival, without any prioritisation or skipping in the queue.
4. Equal Probability for Queue Selection: Although in practical scenarios, individuals tend to join the shortest queue, for our model, we assume that every arrival has an equal probability of joining any queue. This assumption is based on the premise that the queue lengths would be relatively equal if managed efficiently.
5. People who finish services will leave the area: This reduces the influence of irrelevant factors and improves the accuracy of estimation.
6. Full coverage of the camera: The camera in front of the vendor can capture the whole queue. People not in a queue will not be captured. This improves the accuracy of the estimation of the arrival rate.

Under these assumptions, an M/M/1 queuing model for each queue has been developed in Python, allowing for a simplified and standardised approach to analysing queue dynamics. To calculate the arrival and service rate at a particular time, we retrieve historical data on the same time and same weekday in the last month. We retrieve the receipt timestamp and estimate service time based on the time difference. We assume that intervals larger than 3 minutes are considered idle. By taking the average of past service time per customer, we get the estimated future service rate. To calculate the arrival rate, we retrieve the number of people in front of the stall from CV detecting results. Combining with service information retrieved before, we can know how many people finish the services and leave the queue, and how many people join the queue. Again, by taking the average of the current arrival rate, we get the estimated current arrival rate. With these two data, an M/M/1 queuing model is built. The estimated waiting time for each stall is estimated by this model and displayed to users. Only when users go to feature 3 page does the app send requests to the server. The server

then calculates the estimated waiting time and responds. The delay is increased, while the workload and overhead are reduced.

3. Testing and Evaluation

3.1. Usability

To evaluate usability, we conduct testing to measure latency. Several user devices simultaneously send requests to the server. As the number of requests increases, the processing time of the server increases. When there are more than eight devices, the waiting time to receive responses from the server is around one minute. This is due to the limitations of the devices used. When only one device sends requests, the waiting time is less than three seconds.

3.2. Accuracy

All the testing is conducted under the assumption of high accuracy of the YOLO CV model.

Due to various reasons, we are unable to collect enough real historical traffic data to test our system. Hence, we generate historical data for 3 months by assuming that the traffic per day follows Poisson Distribution. Also, the traffic on the same weekday follows the same distribution. Random errors are added to each data entry.

We separate our generated data into training and testing datasets. The first 2 months are the training dataset and the last month is the testing dataset. Initially we want to build a model to predict the traffic. The linear regression model has a higher accuracy. However, we find that the accuracy of taking the average is close to that of linear regression. Thus, we choose to use averaging to predict traffic. The MSE is 12.674.

To test the performance of the M/M/1 model, the same methodology is applied. Because there's no real data on waiting time, the accuracy of arrival rate and service time is used to evaluate the model. The MSE of arrival rate is 4.873 (persons per minute). The MSE of service time is 1.453 (minutes). Based on these two errors, the waiting time calculated by the M/M/1 model is 4.978 (minutes).

3.3. Limitations and Future Improvement

Due to limited funds, we only use one camera to implement our features. To ensure full functionality of the system, more cameras are required and further integration and testing need to be performed.

As the number of requests increases, the processing time of the server increases. This is due to the limitations of the devices used. Using cloud services instead will be one of the solutions to this.

Azure SQL database provides security and privacy for all data collected. However, more approaches to ensuring security may be further needed to protect safe connections among all devices, for example, encryption of messages.

We make the necessary assumptions to generate the data to test our system. However, these assumptions may not adhere to real situations. To improve the accuracy of our system, real data is required to build and train models to predict traffic.

4. Reference

Yawar, M. (2023, November 9). *Human Detection in Computer Vision.* Coding ninjas studio.

https://www.codingninjas.com/studio/library/human-detection-in-computer-vision

5. Appendices

Code: GitHub link

Demo Video: Youtube Link

Medium: Medium link

Slides: Final Presentation Slides