

Analytical Parcel Management Information System (APMIS) For Sorting Centers

Hernandez, Jan Cedric

1 SYSTEM INTRODUCTION

1.1 Background

In the past, the emergence of e-commerce giants like Alibaba and JD.com can be attributed to the 2003 SARS outbreak while the significant increase in shareholder value of American Express and Starbucks during the 2008–2009 global financial crisis is related to their strategy shifting towards digital operating model (Candelon et al., 2020). During the pandemic, there was another surge in the e-commerce industry. The COVID-19 pandemic caused travel restrictions and health concerns, prompting a significant shift towards online shopping and digital transactions. E-commerce is still expected to flourish in the market with the growing accessibility to the internet and mobile phones together with several digitalization initiatives reaching rural communities (Fabri & Valverde Márquez, 2020, Dragomirov, 2020).

E-commerce encompasses business transactions done electronically between or among organizations and between organizations and individuals either locally or globally. In the context of the Philippines, Shopee and Lazada are the most locally known e-commerce companies with other competitors such as TikTok Shop, Carousell, and E-bay. These companies provide a platform, usually through mobile apps and websites, where several consumers can buy various products from different sellers in the digital space. This approach also enables digital marketplaces to operate at any time of the day as the systems are designed to work without human support (Fabri & Valverde Márquez, 2020).

The end-to-end processes involved in e-commerce involve order processing, picking, packing, shipping, delivery, and return. As the buyers check out their orders, the seller will be notified regarding the items included in the orders and will proceed to parcel preparation and pick-up requests. The logistics process begins with first-mile logistics, where logistics personnel collect goods from sellers and transport them to a designated hub or warehouse. Following this, the middle-mile process involves transferring the goods from the hub or warehouse to a sorting or distribution center. At these centers, parcels going to the same area are typically sorted together before being transferred to another hub or warehouse. Lastly, the last-mile logistics phase entails delivering parcels directly to buyers to fulfill orders, assuming no return requests exist. On the other hand, the processes involved in sorting centers mostly deal with the management of parcels getting in and out of the facility. When parcels are received, they will be sorted depending on the destination hubs from which the goods will be transported. The parcels can be tagged as received, sorted, stored, or dispatched.

1.2 Problem Statement

Nowadays, the competition of the e-commerce giants is in terms of delivery time as Shopee, Lazada, and Tiktok Shop have several initiatives for the buyer to receive their orders in 1-2 days. Despite the huge growth of e-commerce businesses, there is still a critical need for more efficient parcel management solutions, particularly in sorting centers where parcels are strategically sorted and distributed. There is a need for the operational processes to cope with the increasing demand to avoid processing delays, sorting errors, and parcel tracking inefficiencies. Another problem faced by the operation is also the proper manpower allocation depending on the actual demand wherein there are some seasons or days with excess staff. These inefficiencies not only hinder client satisfaction in terms of delivery time but also generate non-value-adding expenditures. Aside from this, the potential of machine learning is still not maximized in the e-commerce industries with tons of mineable data. To properly handle these difficulties, an analytical parcel

management information system adapted to the specific needs of sorting centers is crucial for operational improvements.

1.3 Objectives

The projects aimed to develop an Analytical Parcel Management Information System (APMIS) for sorting centers operating in the e-commerce logistics industry. Specifically, the case study aimed to:

- digitize parcel tracking and management to lessen manual interventions and human processing errors,
- enable real-time key metrics monitoring through an interactive dashboard, and
- provide day-by-day forecasting on order counts to enable the capability for more efficient resource allocation and immediate process adjustments.

1.4 Scope and Limitations

In the case study, the project focused on the design, development, and implementation of an information system to be used by sorting centers. The system's primary features involve tracking parcel status within the sorting center and analytical features such as supply forecasting and a dashboard with parcel count, processing time, and backlog monitoring capability. The system only deals with processes involved within the sorting center. Information from the e-commerce and third-party logistics (3PL) is only integrated into existing logistics infrastructure through APIs, thus, the system is not involved in the order creation, seller processing, and 3PL operation outside the sorting center. APMIS was designed to have a user-friendly interface to allow efficient navigation of parcel management with a simplified database design for scalability as parcel volume increases over time.

One of the main features of the system is the parcel volume forecasting. It is worth noting that its accuracy depends on the data quality from the upstream and model capability. The initial phase of implementation will be the basis for further model fine-tuning; thus, this capability might not be as accurate as expected at the start. Full usage of the system's features can also be limited by the current hardware infrastructure available within the sorting center such as restrictions on the number of connected devices accessing the system simultaneously. Furthermore, there can be operational disruption due to external factors such as network connectivity and hardware issues that may affect the system performance. It is also important to highlight that the current developer has beginner-level web development skills, which presents ample opportunities for improvement following the beta implementation.

2 SYSTEM REQUIREMENTS ANALYSIS

2.1 Use Cases/Functions

The use cases in the proposed information system are illustrated in Figure 1. All personnel have their login credentials wherein the system will provide registration and login functionalities to ensure access.

Most of the transactions involve the management of the parcel details. Newly received parcels can be enrolled in the system with necessary information including parcel ID, weight, seller region, seller city, buyer region, and buyer city. Then, the status can be changed depending on the process step the parcel is in wherein the location and update timestamp will also be noted. In real-life operations, this can be done by scanning barcodes or QR codes. For simplicity, the information system will involve manual updating of the details through the web application.

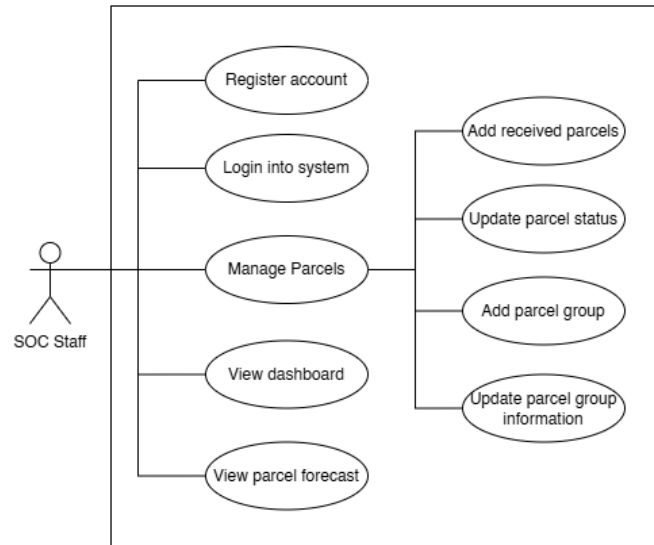


Figure 1. *Use Case Diagram of APMIS*

There are some additional features of the system that enable analytical analysis.

2.1.1 Dashboards

As the logistics process requires updating the parcel status, dashboards with interactive and dynamic visualizations can be implemented regarding the actual parcel counts going into the sorting center. It can also display the current metric performances even in the middle of the day to enable operation adjustments.

2.1.2 Predictive Parcel Count Forecasting

Based on historical data, a predictive algorithm based on a Long Short-Term Memory (LSTM) recurrent neural network (RNN) was used to forecast the expected parcel count for a certain hub or sorting center at a particular date. This technique analyzes the past data and gets patterns to allow accurate prediction which can help the operations in resource allocation, capacity planning, and operation process optimizations.

2.2 ERD

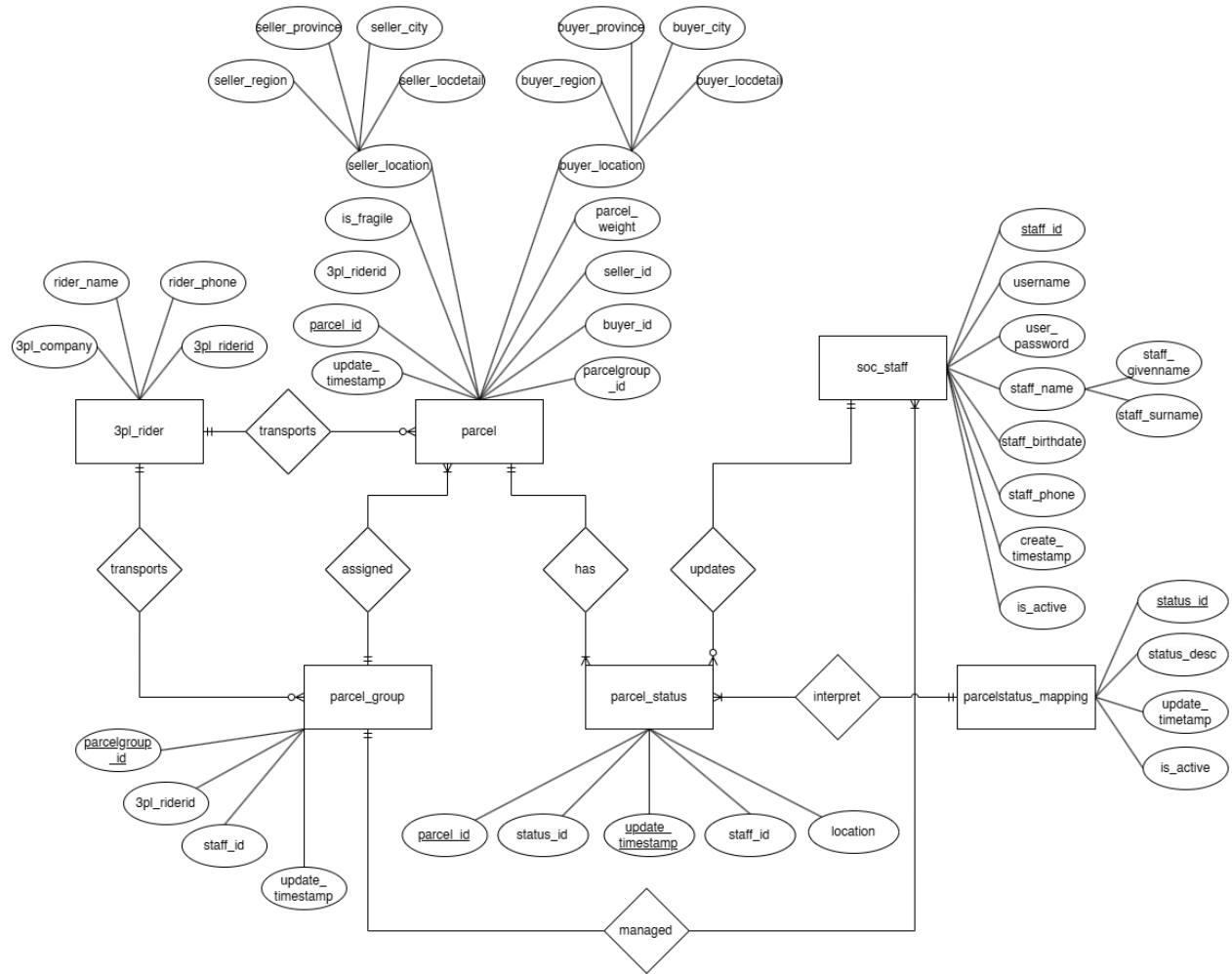


Figure 2. Entity Relationship Diagram of APMIS

2.3 Relational Model

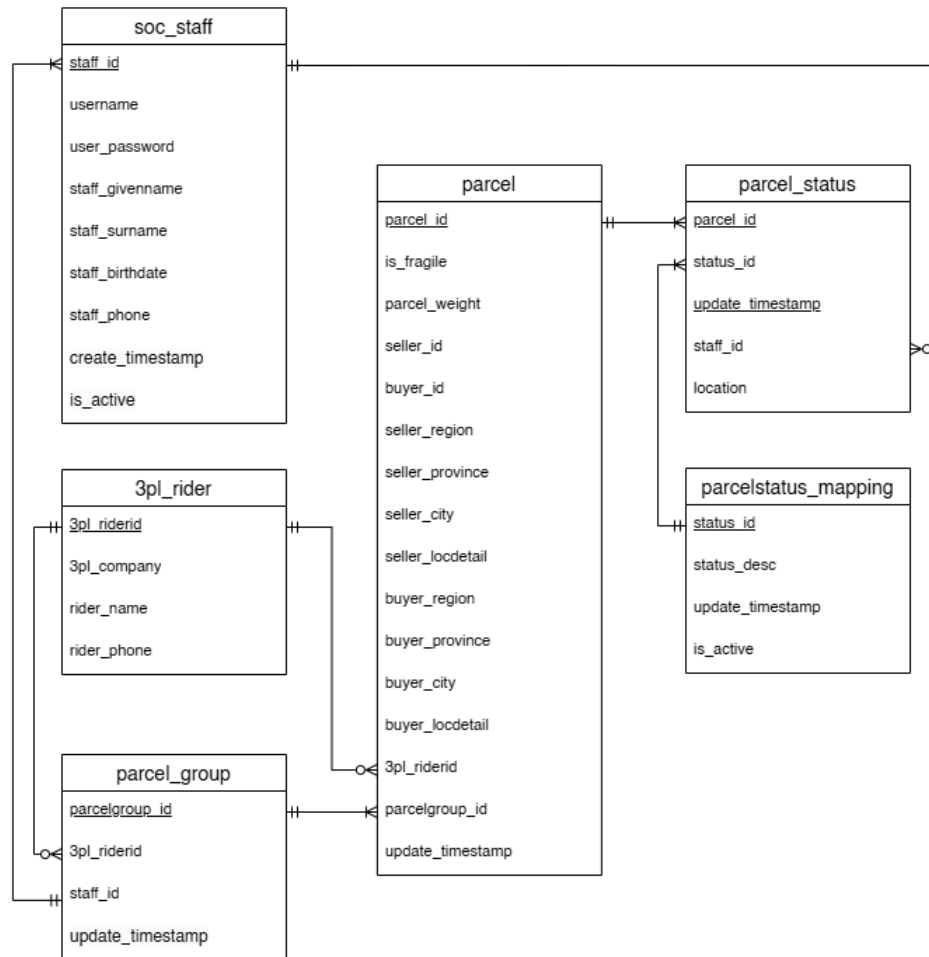


Figure 3. Relational Model of APMIS

3 IMPLEMENTATION

3.1 Database Implementation

3.1.1 Database Schema

For APMIS, PostgreSQL 16 was used as the relational database management system. The SQL queries for the database creation can be browsed in Appendix 5.2.

The schema involves various entities including SOC staff, 3PL rider, parcel group, parcel, parcel status, and parcel status mapping.

3.1.1.1 *parcel table*

This table contains information specific to each parcel such as weight, fragility, parcel group, and 3PL rider. It also contains the user IDs of the buyer and seller and their corresponding location details which can be retrieved from the database of the e-commerce business through API.

Table 1. *Details of the parcel table*

Column Name	Data Type	Constraints	Description
parcel_id	SERIAL	PRIMARY KEY	Unique identifier for parcels.
is_fragile	INT	CHECK (0, 1)	Value is 1 if fragile, 0 if not fragile.
parcel_weight	NUMERIC		Weight of the parcel in g.
seller_id	INT		Identifier for the seller.
buyer_id	INT		Identifier for the buyer.
seller_region	VARCHAR(64)		Region of the seller's address.
seller_province	VARCHAR(64)		Province of the seller's address.
seller_city	VARCHAR(64)		City of the seller's address.
seller_locdetail	TEXT		Detailed address of the seller. Can include barangay, street, house number, etc.
buyer_region	VARCHAR(64)		Region of the buyer's address.
buyer_province	VARCHAR(64)		Province of the buyer's address.
buyer_city	VARCHAR(64)		City of the buyer's address.
buyer_locdetail	VARCHAR(64)		Detailed address of the buyer. Can include barangay, street, house number, etc.
pl3_riderid	INT	REFERENCES pl3_rider(pl3_riderid)	Identifier of the 3PL rider who delivered the parcel.
parcelgroup_id	INT		Identifier for parcel group.
update_timestamp	TIMESTAMP	DEFAULT NOW()	Timestamp indicating last update time.

3.1.1.2 *parcel_status table*

This table shows the historical changes in the status per parcel_ID. This shows the timestamp per change in status_id and location. The staff_id is also noted to reference the staff who handled each part of the process. The composite key for this table is the combination of parcel_id and update_timestamp

Table 2. *Details of the parcel_status table*

Column Name	Data Type	Constraints	Description
parcel_id	INT	COMPOSITE KEY 1	Identifier for the parcel.
status_id	INT		Identifier for the status.
update_timestamp	TIMESTAMP	COMPOSITE KEY 2	Timestamp when the status is pushed.
staff_id	INT	REFERENCES soc_staff(staff_id)	Identifier of the staff member involved in the status transaction
location	VARCHAR(100)		Location information for the parcel status such as storage rack and warehouse numbers

3.1.1.3 *parcelstatus_mapping table*

This mapping shows the corresponding descriptive status of each status ID. It has an updated timestamp attribute to document when the status was first introduced. It also has another column to show if the status ID is still used for possible changes in the process.

Table 3. *Details of the parcelstatus_mapping table*

Column Name	Data Type	Constraints	Description
status_id	SERIAL	PRIMARY KEY	Unique identifier for each part of the SOC process.
status_desc	VARCHAR(64)	NOT NULL	Description of the status.
update_timestamp	TIMESTAMP	DEFAULT NOW()	Status ID creation timestamp.
is_active	INT	DEFAULT 1, CHECK (0, 1)	1 if status_id is still used, 0 if not updated and not used anymore.

3.1.1.4 *parcel_group table*

The parcel_group table takes into account the staff_id involved in combining parcels which will be taken into the same delivery hub with the corresponding 3PL rider and update timestamp.

Table 4. *Details of the parcel_group table*

Column Name	Data Type	Constraints	Description
parcelgroup_id	SERIAL	PRIMARY KEY	Unique identifier for parcel groups.
pl3_riderid	INT	REFERENCES pl3_rider(pl3_riderid)	Identifier of the 3PL rider who received the parcel group and will deliver to the delivery hub.
staff_id	INT	REFERENCES soc_staff(staff_id)	Identifier of the staff member who dispatched the parcel group (must be registered in the pl3_rider table)
update_timestamp	TIMESTAMP	DEFAULT NOW()	Timestamp indicating last update time.

3.1.1.5 *pl3_rider table*

This contains basic information regarding the 3PL riders. Information is ideally taken from the external databases with rider ID as a key. Note that '3pl' is spelled as 'pl3' in the implementation due to restrictions on the character format.

Table 5. *Details of the pl3_rider table*

Column Name	Data Type	Constraints	Description
pl3_riderid	SERIAL	PRIMARY KEY	Unique identifier for PL3 riders.
pl3_company	VARCHAR(64)		Company associated with the rider.
rider_name	VARCHAR(64)		Name of the rider.
rider_phone	VARCHAR(20)		Phone number of the rider.

3.1.1.6 soc_staff table

The soc_staff table contains the login and personal information of all enrolled staff. It contains the login credentials with the encrypted password stored in the database. For the initial version of the information system, the roles (eg., admin) are not yet defined.

Table 6. Details of the soc_staff table

Column Name	Data Type	Constraints	Description
staff_id	SERIAL	PRIMARY KEY	Unique identifier for staff members.
username	VARCHAR(32)	UNIQUE, NOT NULL	Username for staff login.
user_password	VARCHAR(64)	NOT NULL	Password for staff login.
staff_givenname	VARCHAR(64)	NOT NULL	Given name of the staff member.
staff_surname	VARCHAR(64)	NOT NULL	Surname of the staff member.
staff_birthdate	DATE	NOT NULL	Birthdate of the staff member.
staff_phone	VARCHAR(20)	NOT NULL	Phone number of the staff member.
create_timestamp	TIMESTAMP	DEFAULT NOW()	Timestamp of account creation.
is_active	INT	DEFAULT 1, CHECK (0, 1)	1 if active employee, 0 if resigned employee.

3.1.2 Database Connection

The web application was primarily built on Python (3.11.5) language and Dash (2.16.1) framework connected to the PostgreSQL database through the psycopg2 library.

There is a dbconnect.py script containing the functionalities to interact with the database. The getdblocation function contains the connection details.

```
def getdblocation():
    db = psycopg2.connect(
        host='localhost',
        database='271casedb',
        user='postgres',
        port=5432,
        password='xxxx'
    )
    return db
```

There are also 2 additional helper functions. The modifydatabase function calls the database credential, creates a cursor object to execute the SQL query with the corresponding value inputs, and commits the changes before closing the connection. This can be used to connect to the database and insert or update records in the tables such as inputting parcel information in the system and adding and updating parcel_status. On the other hand, the querydatafromdatabase function works almost the same way as the former but is used to fetch data from the database and store it in a data frame instead of making changes in the database. This is used to get the relevant data that will appear in the dropdown buttons and data to be used in the dashboard visualizations.


```

def modifydatabase(sql, values):
    db = getdblocation()
    cursor = db.cursor()
    cursor.execute(sql, values)
    db.commit()
    db.close()

def querydatafromdatabase(sql, values, dfcolumns):
    db = getdblocation()
    cur = db.cursor()
    cur.execute(sql, values)
    rows = pd.DataFrame(cur.fetchall(), columns=dfcolumns)
    db.close()
    return rows

```

3.2 Walkthrough of Application Features

3.2.1 Login Page

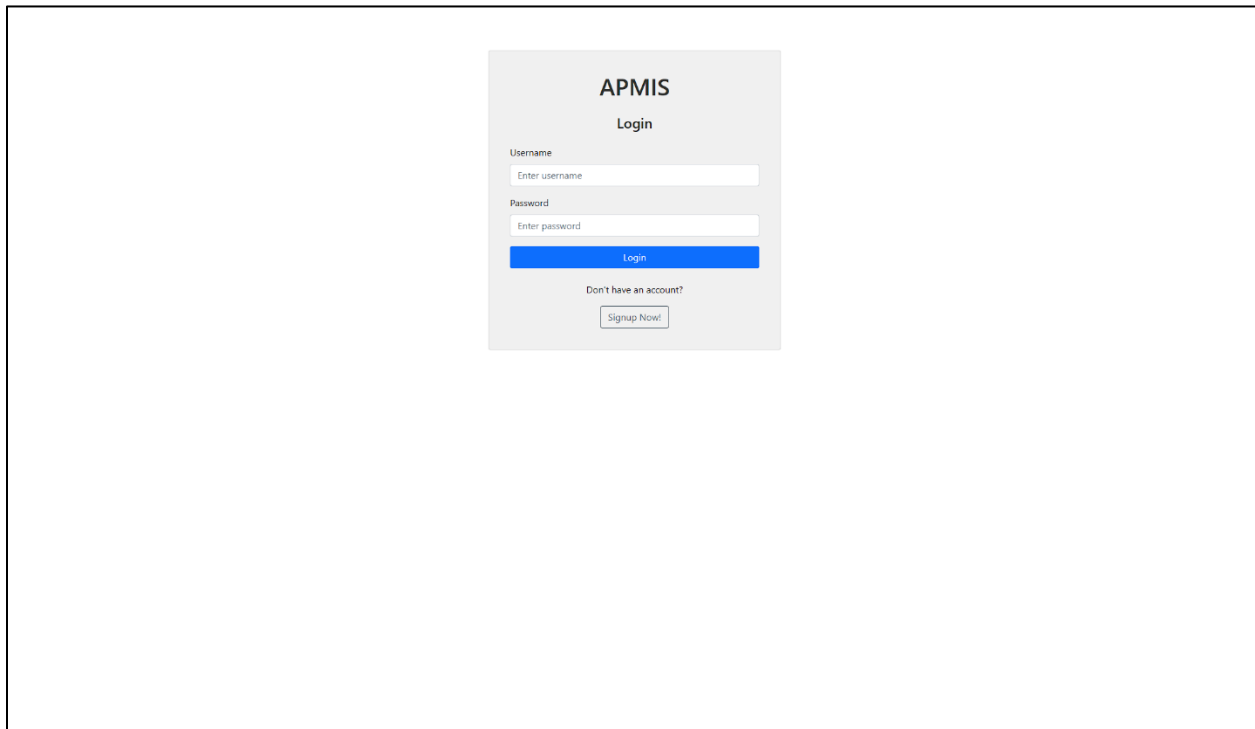
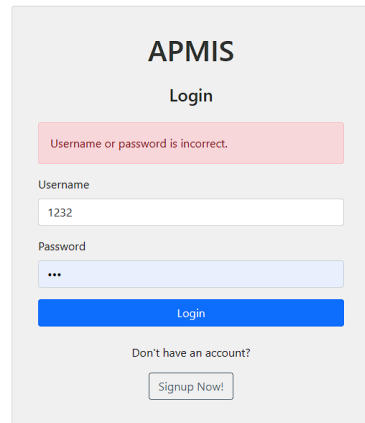


Figure 4. Login Page of APMIS

The first page that will be displayed in APMIS is the login page serving as the gateway for the users to access the system. This page allows users to

- enter username and password in the respective fields,
- click the login button to trigger the authentication procedure,

- receive an alert message if the input credentials are incorrect or unregistered, and

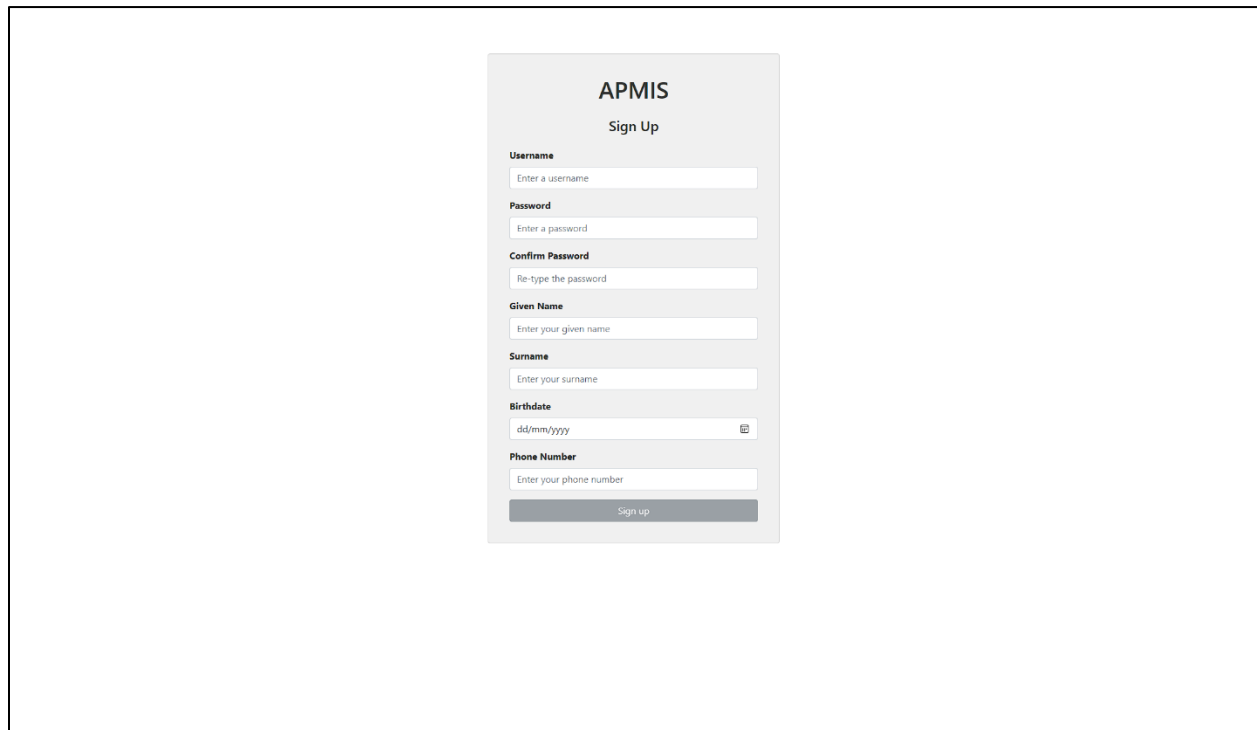


The screenshot shows the APMIS Login page. At the top, it says "APMIS" and "Login". Below this, there is a red error message box that says "Username or password is incorrect." Underneath the error message, there are two input fields: "Username" with the value "1232" and "Password" with three dots indicating a masked password. Below these fields is a blue "Login" button. At the bottom, there is a link that says "Don't have an account?" and a button that says "Signup Now!".

- proceed to the signup page by clicking on the Signup Now button. Note that the system requires this for new employees to record their login credentials and personal information in the system

This page has a function that compares the username and password input to the entries in the soc_staff table. If the credentials are valid, it will update the value of the current user ID (which will be used in transactions in other pages to note the staff_id of the user) and will proceed to the home page. For password storage and authentication, Hashlib is being used for encryption and decryption.

3.2.2 Sign Up Page



The screenshot shows the APMIS Sign Up page. At the top, it says "APMIS" and "Sign Up". Below this, there are several input fields with labels: "Username" (with placeholder "Enter a username"), "Password" (with placeholder "Enter a password"), "Confirm Password" (with placeholder "Re-type the password"), "Given Name" (with placeholder "Enter your given name"), "Surname" (with placeholder "Enter your surname"), "Birthdate" (with placeholder "dd/mm/yyyy" and a calendar icon), and "Phone Number" (with placeholder "Enter your phone number"). At the bottom, there is a grey "Sign up" button.

Figure 5. Signup Page of APMIS

This page can be used by new employees to make a new record of their information in the soc_staff table. This page allows users to:

- input their login credentials and personal information,
- validate the provided information (the signup button will turn blue if all information is complete (left photo), but will still be greyed out if there is a missing field (middle photo) or if the password does not match (right)), and

The image shows three variations of the APMIS Sign Up form. Each form has the following fields: Username, Password, Confirm Password, Given Name, Surname, Birthdate, and Phone Number. The 'Sign up' button is at the bottom.

- Left Form (Complete):** All fields are filled. Username: 'juan', Password: '****', Confirm Password: '****', Given Name: 'Juan', Surname: 'Dela Cruz', Birthdate: '01/01/19990', Phone Number: '12345'. The 'Sign up' button is blue.
- Middle Form (Missing Field):** The 'Given Name' field is empty. The 'Sign up' button is greyed out.
- Right Form (Password Mismatch):** The 'Password' field is '****' and the 'Confirm Password' field is '*****'. The 'Sign up' button is greyed out.

- submit their account creation form through the Sign-Up button and receive a feedback message if successful.

The image shows a feedback message overlaying the Sign Up form. The message is titled 'User Saved' and says 'User has been saved'. There is an 'Okay' button. The form fields are visible in the background but are dimmed.

For every successful account registration, the account credentials with other personal details are inserted in the soc_staff table as a new record. It should be noted that the passwords are encrypted using Hashlib for security purposes.

3.2.3 Home Page

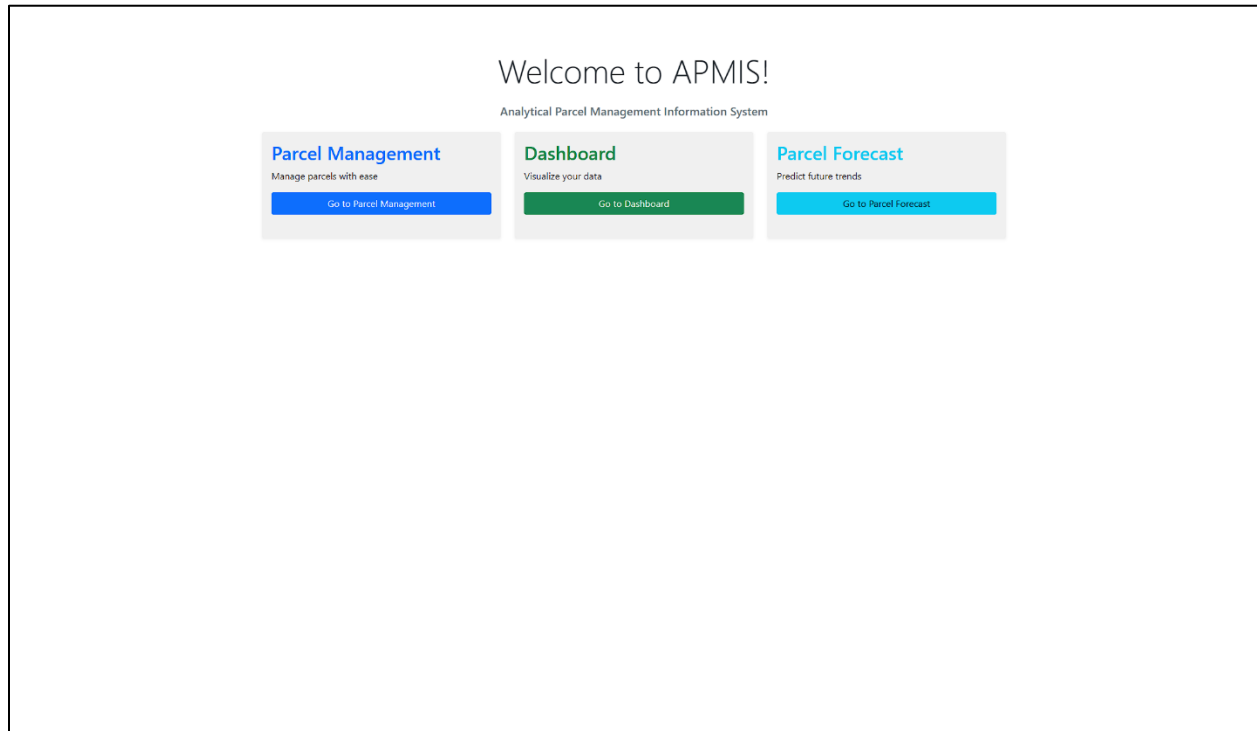


Figure 6. Home Page of APMIS

The home page serves as quick navigation for the 3 main functionalities of APMIS in separate navigation cards. The user can choose either the Parcel Management button if the user needs to add/update a parcel, the Dashboard button if the user needs to access data visualizations, or the Parcel Forecast button to check forecasted order counts in the next 30 days for process improvement planning. Page redirection will occur upon clicking any of the 3 buttons. It should also be noted that clicking the APMIS text in the navigation bar on the other pages leads to this home page.

3.2.4 Parcel Management Main Page

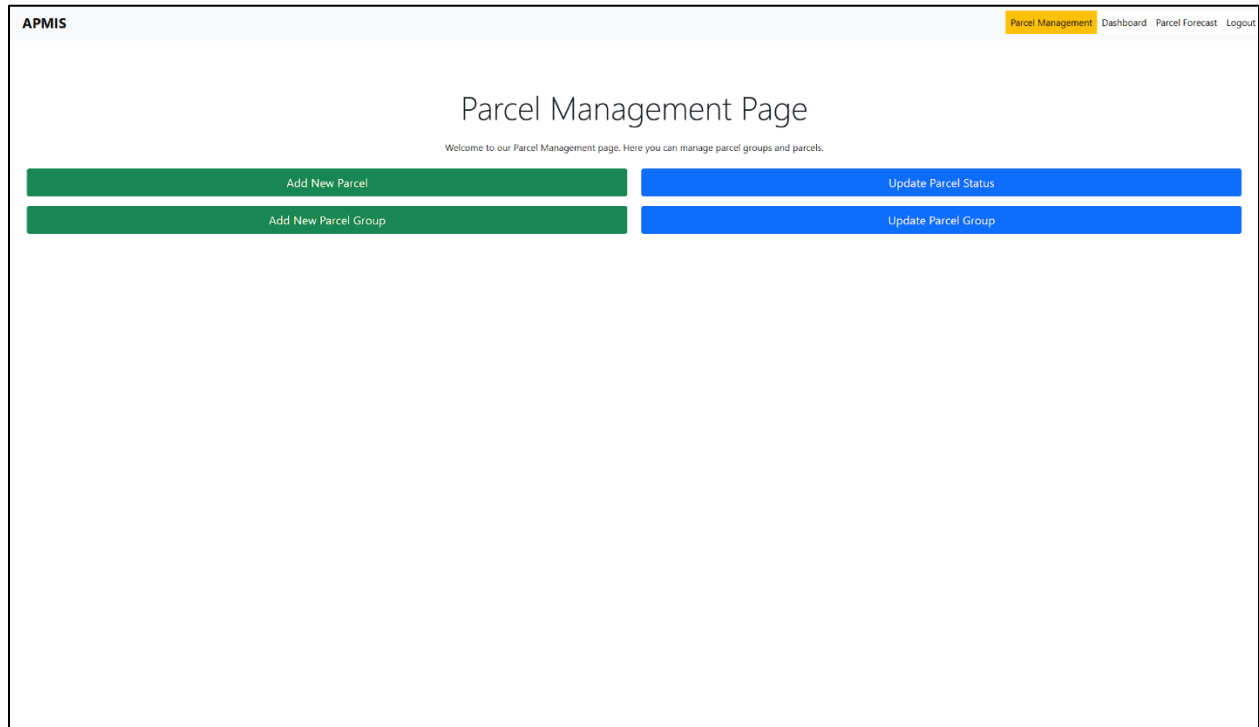


Figure 7. *Parcel Management Page of APMIS*

The Parcel Management page of APMIS serves as the main portal for 4 possible actions regarding the parcels processed at the sorting center. The user can choose any of the 4 buttons to be directed to the respective page.

3.2.5 Add New Parcel Page

APMIS

Parcel Management Dashboard Parcel Forecast Logout

Add New Parcel

Parcel ID*

Is Fragile? ☐ Yes ☐ No

Parcel Weight*

Location*

Seller ID*

Buyer ID*

3PL Rider ID*

Seller Information

Region

Province

City

Location Detail

Buyer Information

Region

Province

City

Location Detail

Figure 8. Add New Parcel Page of APMIS

In the process cycle of a parcel within the sorting center, the first functionality needed is to add the newly received parcel into the database to start tracking its status changes. When the parcel is received, it must be registered on this page to gather the required information such as parcel ID, fragility, weight, current location, and 3PL rider ID. The other details are not required since this information can be fetched via API from partner 3PL and e-commerce companies.

This page allows the user to:

- fill out necessary information regarding the parcel to be registered,
- receive error alerts such as if there is a missing required field, and

Add New Parcel

Parcel ID*

Is Fragile? ☒ Yes ☐ No

Parcel Weight*

Location*

Seller ID*

Buyer ID*

3PL Rider ID*

Seller Information

Region

Province

City

Location Detail

Buyer Information

Region

Province

City

Location Detail

Check your inputs. Please fill in all the required fields.

- receive a successful message if all queries were successfully performed.

Add New Parcel

Parcel ID*
11111

Is Fragile? ☒ Yes ☐ No

Parcel Weight*
123

Location*
Warehouse A

Seller ID*
123

Buyer ID*
12345

3PL Rider ID*
1

Seller Information

Region
City

Province
Location Detail

Buyer Information

Region
City

Province
Location Detail

Submit

New parcel added successfully.

For every parcel registration, there are always 2 insert queries performed at every successful transaction. First, the parcel-ID and parcel information will be enrolled in the parcel table. Second, this should also be accompanied by adding a record in the parcel_status table with status 1 (received). The current timestamp during form submission always be the basis of the update_timestamp and then the corresponding staff_id of the user logged in will be recorded. Note that the staff_id is a stored value that was set on the login page.

3.2.6 Update Parcel Status Page

APMIS
Parcel Management Dashboard Parcel Forecast Logout

Update Parcel Status

Parcel ID*
Select Parcel ID

Status*
Select Status

Location

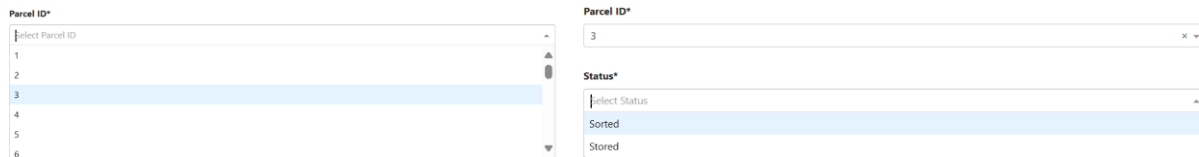
Update Status

Figure 9. Update Parcel Status Page of APMIS

It can be recalled that the process of parcels in sorting centers involves receiving (status=1), sorting (status=2), storing (status=3), and dispatching (status=4). This page is concerned with updating the parcel status after it is either sorted or stored (status 1 is done on the Add New Parcel page while status 4 is performed on the Update Parcel Group page)

This page allows the user to:

- select the parcel ID of the parcel to be updated and the corresponding parcel_status from the dropdown buttons (the options for the dropdown fields were automatically filled with the available parcel_id from the parcel table),



Parcel ID*

Select Parcel ID

1

2

3

4

5

6

Parcel ID*

3

Status*

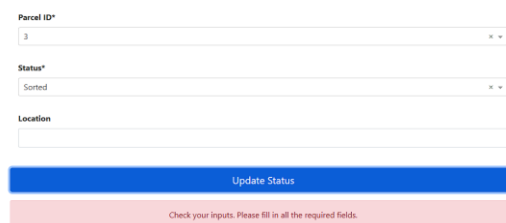
Select Status

Sorted

Stored

- receive error alerts such as if there is a missing required field, and

Update Parcel Status



Parcel ID*

3

Status*

Sorted

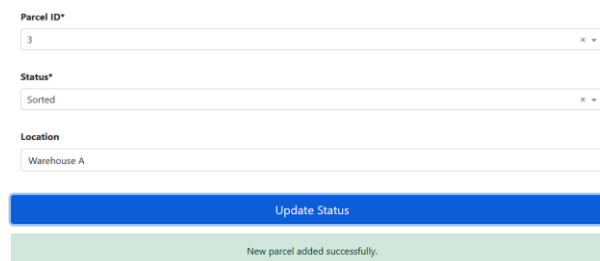
Location

Update Status

Check your inputs. Please fill in all the required fields.

- receive a successful message if all queries were successfully performed.

Update Parcel Status



Parcel ID*

3

Status*

Sorted

Location

Warehouse A

Update Status

New parcel added successfully.

For every successful transaction, a new record is inserted in the parcel_status table containing the parcel ID, status, and location inputs with the user ID of the staff who pushed and the timestamp when the button is triggered.

3.2.7 Add New Parcel Group Page

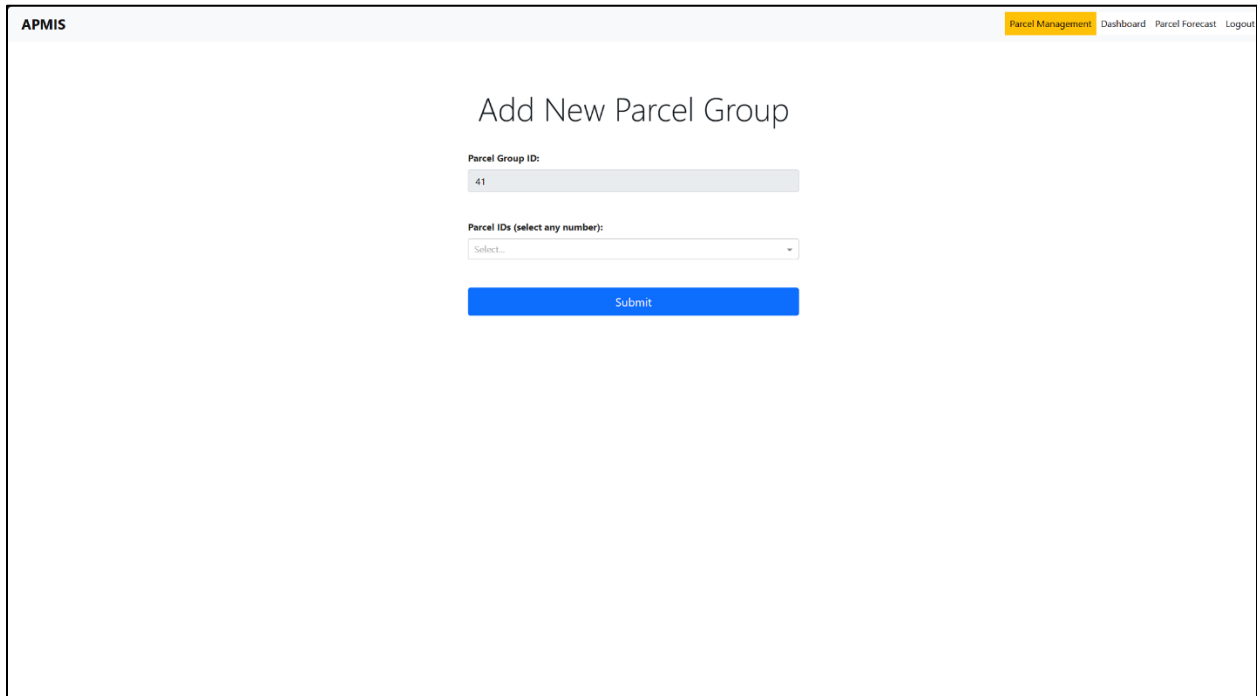
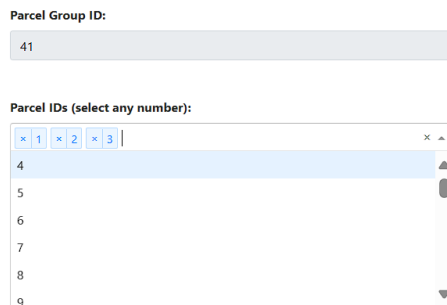


Figure 10. Add New Parcel Group Page of APMIS

One process in the sorting center involves grouping parcels to be delivered at the same last-mile hub. This page enables recording the parcels grouped together before handling to the 3PL riders. The page allows the users to

- create a new parcel group ID and indicate the parcels grouped (the parcel group ID field automatically shows the ID that will be assigned to the next group as this attribute is of serial data type. As the maximum parcel group ID in the current database is 40, the next parcel group ID is 41 as displayed in the field. All available parcel IDs are also shown in the dropdown menu)

Add New Parcel Group



- receive error alerts such as if there is a missing required field (at least one parcel ID should be chosen for each parcel group), and

Add New Parcel Group

Parcel Group ID:
41

Parcel IDs (select any number):
Select...

Submit

No parcel IDs selected!

- - receive a successful message if all queries were successfully performed (if the previous transaction is successful, the next parcel group ID will be automatically refreshed with an increment of one and the user can indicate the parcel IDs of the next group).

Add New Parcel Group

Parcel Group ID:
42

Parcel IDs (select any number):
Select...

Submit

Parcel IDs inserted successfully into the database!

○ Whenever the submit button is triggered, there are always 2 queries to be performed. The first one inserts a new record in the parcel_group table to indicate the newly created parcelgroup_id, the user ID of the staff who pushed the update, and the corresponding timestamp. Then, the parcelgroup_id attribute in the parcel table of all involved parcels will also be updated accordingly.

3.2.8 Update Parcel Group Page

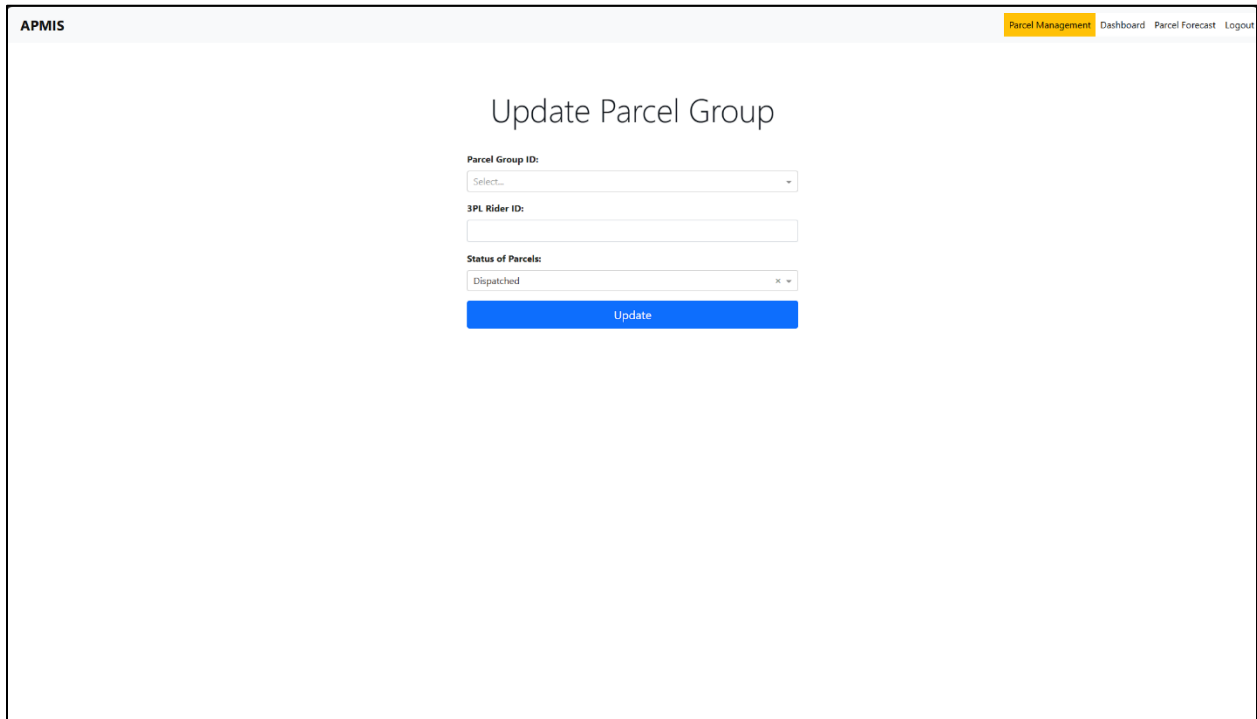
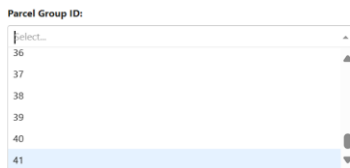


Figure 11. Update Parcel Group Page of APMIS

This page is used when a parcel group is dispatched to a 3PL rider. Note that on this page, the status field is automatically labeled as 'Dispatched'. This allows users to:

- choose the parcel group ID to be dispatched from the dropdown button (all parcel group ID from the `parcel_group` table is shown in the choices),

Update Parcel Group



- indicate the 3PL rider ID to receive the parcel group,

- receive an error when the 3PL rider ID is not recognized in the system (Note that this transaction in the app requires identification of 3PL rider ID that should be already enrolled in the system or can be fetched through API in real practice. The app does not allow registration of 3PL riders as this process should be done on the 3PL side.), and

Update Parcel Group

Parcel Group ID:
41

3PL Rider ID:
123

Status of Parcels:
Dispatched

Update

Error updating parcel group: insert or update on table "parcel_group" violates foreign key constraint "parcel_group_pl3_riderid_fkey" DETAIL: Key (pl3_riderid)=(123) is not present in table "pl3_rider".

- receive a successful message if the transaction is successfully performed.

Update Parcel Group

Parcel Group ID:
41

3PL Rider ID:
1

Status of Parcels:
Dispatched

Update

Parcel Group updated successfully!

Once the user clicks the Update button, the app will perform 2 operations. First, it will update the parcel_group table to indicate the ID of the 3PL rider to whom the parcel group was dispatched to. Then, the next operation needs to update the records in the parcel_status. For this operation, the system will first query all the parcel IDs belonging to the indicated group ID. Then, a for loop will be executed to insert records in the parcel_status table for all concerned parcel IDs indicating the status_id of 4 (for dispatched status), the timestamp, and the user ID of the staff. Note that the location column is blank in this case.

3.2.9 Dashboard Page

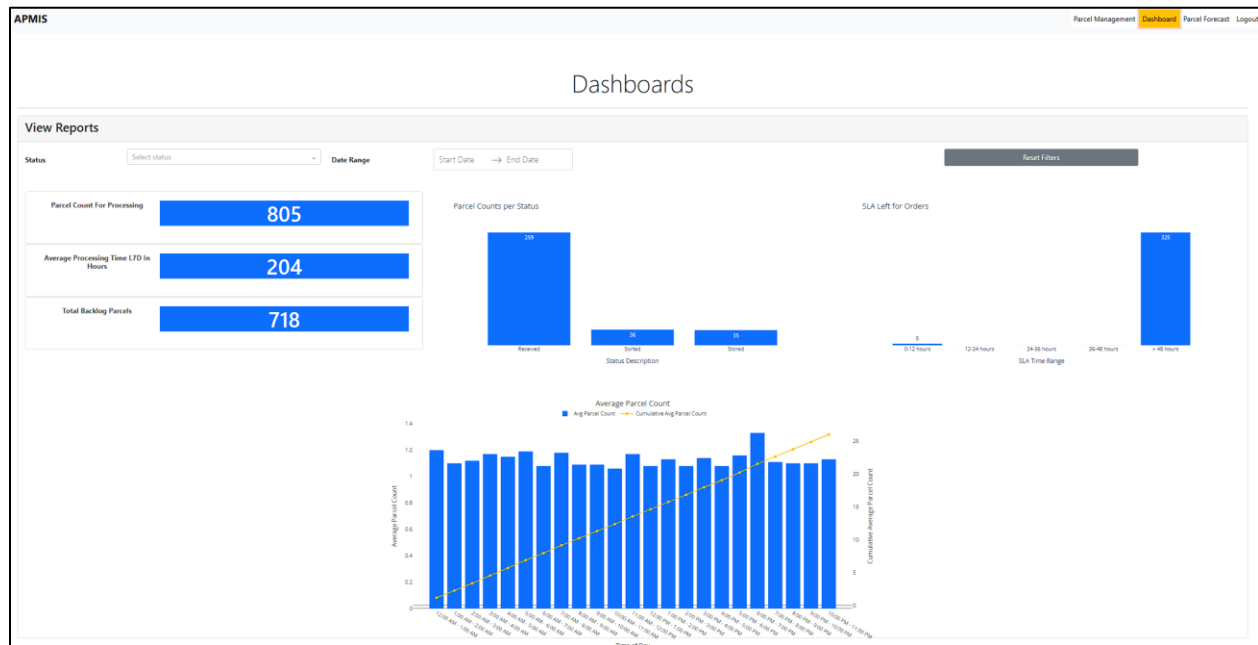


Figure 12. Dashboard Page of APMIS

The Dashboard Page is the second main functionality of APMIS wherein visualizations are presented to get quick updates on the status of parcels in the sorting center. On the left side are card visualizations to indicate the number of parcels needed to be processed, the average time difference (in hours) between the received and dispatched status of the parcels for the last 7 days, and the total parcels that are considered backlog (parcels not yet dispatched after 24 hours of reception in the sorting center). The bar chart in the upper middle portion indicates the count of parcels that are not yet dispatched across different statuses while the visualization in the upper right shows the count of parcels for each service level agreement (SLA) time left group (number of hours left before the 24 hours deadline for the parcel to be dispatched). The combination chart at the bottom part shows the average (bar) and cumulative percentage of (line) parcel count over time.

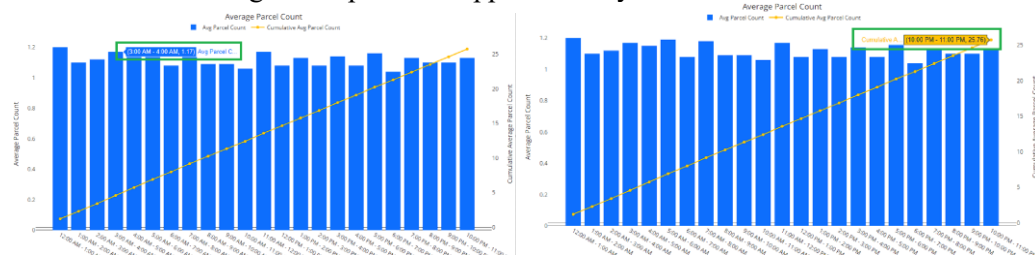
The page allows the user to

- View the important sorting center metrics displayed on the cards and bar charts
- Filter the visualizations according to status and date range (the visualizations, except the cards, will be filtered accordingly)

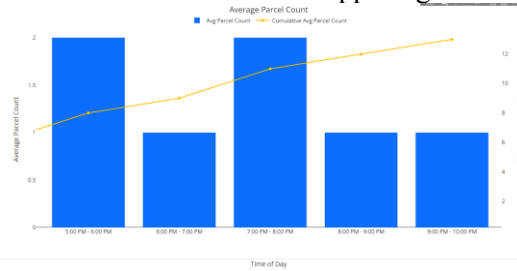


- Reset the filters through the grey button at the upper right corner

- Get the exact numbers through tooltips which appear when you hover at the bar or line



- Zoom into specific parts (by highlighting a portion of the plot. The view can be reverted back through the reset view button at the upper right of each chart)



3.2.10 Parcel Forecast

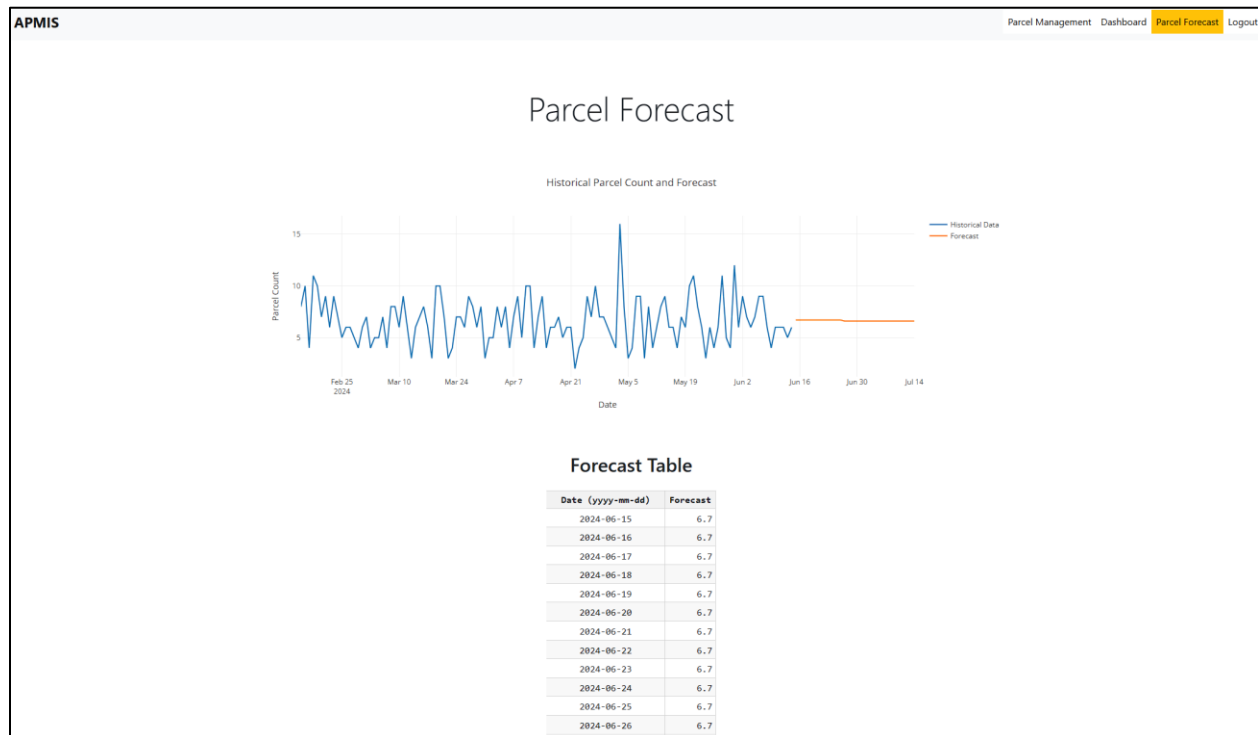
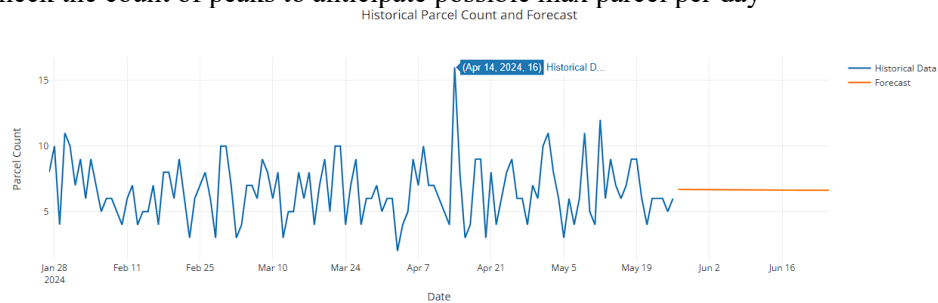


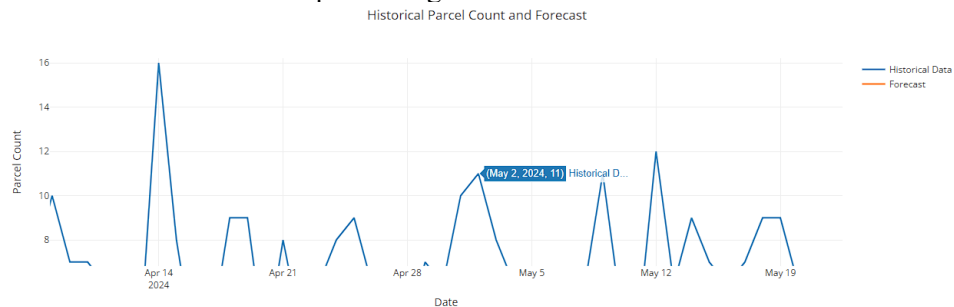
Figure 13. Parcel Forecast Page of APMIS

This page is the third main functionality which provides a 30-day forecast of the parcel count to be received. This allows the users to

- Check the next 30-day forecast of parcel count as displayed in the line chart
- Interact with the chart
 - Check the count of peaks to anticipate possible max parcel per day



- Zoom the data to focus on specific segments



The forecasting method is based on long short-term memory (LSTM) neural networks. In creating the model, a 5-month sample dataset with 500 records was used wherein 80% of the data was used for training. For data processing, the daily parcel count was calculated before it was normalized using MinMaxScaler. A helper function was used to create a sequence with a length of 60 and is used for training. The LSTM model was constructed with a single LSTM layer, a dropout layer, and a dense output layer. Adam optimizer was used to train for 200 epochs with mean squared error as the loss function. The model was programmed to always give a 30-day forecast wherein the final output will be inverse-transformed to the original scale.

3.3 Significance of the Information System

Based on industry experience, there are still a lot of opportunities for system improvement including the use of artificial intelligence (AI) and automated data processing tasks to improve operational processes and decision-making.

The information system enables sorting centers to have data-enabled operations which can lessen paper-based and manual processes. Using AI and analytics, the operations personnel can also create data-informed processes and resource improvements as they can view automated metric measurements through the forecasting feature and dashboards.

3.3.1 Predictive Parcel Count Forecasting

The parcel demand forecast uses a machine learning model to estimate the number of parcels that will be processed by the sorting center allowing the operations to plan resources and adjust certain processes. This can enhance the operation efficiency by enabling them to request additional manpower or delivery vehicles, especially on campaign days when there is an expected increase in parcel volume. This also allows for improved resource allocation by allocating manpower, transportation assets, and warehouse space among hubs and sorting centers months in advance to avoid resource underutilization and overcapacity.

3.3.2 Dashboards

As there is a possible deviation in the forecasted numbers, resource adjustments can still be done in time depending on the actual numbers given by dashboards. With this capability, logistics can make adjustments to meet their daily metrics. This is particularly beneficial when the actual operation cannot keep up with the actual demand due to unforeseen operational and logistics issues. The metrics can also allow the cross-functional team to assist operations in identifying process bottlenecks depending on the hour-to-hour performance.

4 REFERENCES

- Candelon, F., Reichert, T., Duranton, S., Charme, R., Carlo, D., & De Bondt, M. (2020). *The Rise of The AI-Powered Company in the Postcrisis World*.
- Dragomirov, N. (2020). E-Commerce Platforms and Supply Chain Management – Functionalities Study. *Economic Alternatives*, 26(2), 250–261. <https://doi.org/10.37075/EA.2020.2.04>
- Fabri, L., & Valverde Márquez, I. (2020). Will e-commerce dominate physical store? *International Journal of Technology for Business*, 2(1), 23–20. <https://doi.org/10.5281/zenodo.3894442>

5 APPENDIX

5.1 *Github Repository*

The entire code of the APMIS web application can be browsed at [jcmhernandez/APMIS \(github.com\)](https://github.com/jcmhernandez/APMIS)

5.2 *SQL Query for Table Creation*

Note that sample data can be generated by executing the code at [APMIS/IE271caseapp/db_creation_commands.sql at may26 · jcmhernandez/APMIS \(github.com\)](#)

```
CREATE TABLE IF NOT EXISTS soc_staff (  
  staff_id SERIAL PRIMARY KEY,  
  username VARCHAR(32) UNIQUE NOT NULL,  
  user_password VARCHAR(64) NOT NULL,  
  staff_givename VARCHAR(64) NOT NULL,  
  staff_surname VARCHAR(64) NOT NULL,  
  staff_birthdate DATE NOT NULL,  
  staff_phone VARCHAR(20) NOT NULL,  
  create_timestamp TIMESTAMP DEFAULT NOW(),  
  is_active INT DEFAULT 1 CHECK (is_active IN (0, 1))  
);
```

```
CREATE TABLE IF NOT EXISTS pl3_rider (  
  pl3_riderid SERIAL PRIMARY KEY,  
  pl3_company VARCHAR(64),  
  rider_name VARCHAR(64),  
  rider_phone VARCHAR(20)  
);
```

```
CREATE TABLE IF NOT EXISTS parcel (  
  parcel_id SERIAL,  
  is_fragile INT CHECK (is_fragile IN (0, 1)),  
  parcel_weight NUMERIC,  
  seller_id INT,  
  buyer_id INT,  
  seller_region VARCHAR(64),  
  seller_province VARCHAR(64),  
  seller_city VARCHAR(64),  
  seller_locdetail TEXT,  
  buyer_region VARCHAR(64),  
  buyer_province VARCHAR(64),  
  buyer_city VARCHAR(64),  
  buyer_locdetail VARCHAR(64),  
  pl3_riderid INT REFERENCES pl3_rider(pl3_riderid),  
  parcelgroup_id INT,  
  update_timestamp TIMESTAMP DEFAULT NOW(),  
  PRIMARY KEY (parcel_id, update_timestamp)  
);
```

```
CREATE TABLE IF NOT EXISTS parcelstatus_mapping (  
    status_id SERIAL PRIMARY KEY,  
    status_desc VARCHAR(64) NOT NULL,  
    update_timestamp TIMESTAMP DEFAULT NOW(),  
    is_active INT DEFAULT 1 CHECK (is_active IN (0, 1))  
);
```

```
CREATE TABLE IF NOT EXISTS parcel_group (  
    parcelgroup_id SERIAL PRIMARY KEY,  
    pl3_riderid INT REFERENCES pl3_rider(pl3_riderid),  
    staff_id INT REFERENCES soc_staff(staff_id),  
    update_timestamp TIMESTAMP DEFAULT NOW()  
);
```

```
CREATE TABLE IF NOT EXISTS parcel_status (  
    parcel_id INT,  
    status_id INT,  
    update_timestamp TIMESTAMP,  
    staff_id INT REFERENCES soc_staff(staff_id),  
    location VARCHAR(100),  
    PRIMARY KEY (parcel_id, update_timestamp)  
);
```

5.3 Forecasting Model Script

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from datetime import timedelta

# Load data
raw = pd.read_csv('data-1715427392544.csv')

# Convert update_timestamp to datetime
raw['update_timestamp'] = pd.to_datetime(raw['update_timestamp'])

# Filter the DataFrame where status_id is 1
df_filter = raw[raw['status_id'] == 1]

# Group by update_timestamp and count the number of parcels for each date
df = df_filter.groupby(df_filter['update_timestamp'].dt.date).size().reset_index(name='parcel_count')

# Prepare the data
df['update_timestamp'] = pd.to_datetime(df['update_timestamp'])
df.set_index('update_timestamp', inplace=True)

print(df)

# Normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df[['parcel_count']])

# Function to create sequences for LSTM
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        end_ix = i + seq_length
        seq_x, seq_y = data[i:end_ix], data[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)

# Set sequence length
seq_length = 60 # Increased sequence length

# Create sequences
X, y = create_sequences(scaled_data, seq_length)
```

```

# Split data into train and test sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Define the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(seq_length, 1)))
model.add(Dropout(0.2)) # Dropout layer to reduce overfitting
model.add(Dense(1))
model.compile(optimizer='adam', Loss='mse')

# Train the model
model.fit(X_train, y_train, epochs=200, verbose=0) # Increased epochs to 200

# Forecast for the next 30 days
forecast = []
last_sequence = scaled_data[-seq_length:].reshape((1, seq_length, 1))

for _ in range(30):
    next_pred = model.predict(last_sequence)
    forecast.append(next_pred[0, 0])
    last_sequence = np.roll(last_sequence, -1, axis=1)
    last_sequence[0, -1, 0] = next_pred

# Inverse transform forecast
forecast = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))

# Print the forecasts
forecast_dates = pd.date_range(start=df.index[-1] + timedelta(days=1), periods=30)
forecast_df = pd.DataFrame({'update_timestamp': forecast_dates, 'forecasted_parcel_count':
forecast.flatten()})
forecast_df.set_index('update_timestamp', inplace=True)

print(forecast_df)

```