UPLB EASYPAY: A Web-Based Cashier and Queue Management System with Offline-First Transaction Support

Jemuel Juatco and Concepcion L. Khan

***Abstract*— The increasing reliance of school constituents on digital technologies has highlighted the inefficiencies of traditional cashiering methods. This study presents the development of a digital cashier application designed to streamline payment transactions in an academic setting. The system supports core cashier functions such as real-time queue tracking, transaction recording, and summary report generation. Built with an offline-first approach, the application remains functional even during internet interruptions, ensuring uninterrupted cashier operations. To ensure secure data handling, session-based transaction history and encryption techniques were applied. Usability and validation tests showed the application to be user-friendly and effective for its intended environment.**

***Index Terms*— cashier system, offline-first, DexieDB, Firebase, school payments, queue management**

I. Introduction

A. Background of the Study

As schools continue to grow and adapt to new technologies, having a reliable and efficient system for handling finances has become more important than ever. Managing payments and transactions smoothly play a big role in keeping an academic institution running well. A system that is secure, easy to use, and accurate can make a big difference in the daily operations of school cashiers and administrators. Traditionally, these transactions have been managed through manual methods—often involving handwritten official receipts and individually logged collections. While these conventional practices have served their purpose for years, they have also introduced several inefficiencies that hinder operational productivity [1].

Cashier administrators here in UPLB, in particular, face the brunt of this outdated system. On a typical day, they may process up to one hundred individual collections. Each transaction must be carefully written, reviewed, and entered into the institution's financial records. This manual repetition not only consumes a significant amount of time but also increases the risk of human error [2], [3]. Verifying the accuracy of all collection entries and ensuring that the total amounts align with daily reports becomes a laborious task—especially for administrators who must also manage other responsibilities throughout the day [4].

Moreover, the requirement to handwrite receipts and manually cross-check report data slows down service time, impacts client satisfaction, and contributes to physical and mental fatigue for the cashier staff [1], [5]. Given the increasing volume of transactions and the need for real-time reporting and accountability, there is a pressing need for a more intelligent and streamlined approach to managing financial collections.

The advent of modern digital technologies presents a compelling opportunity to address these challenges. By designing and implementing a centralized cashier application, the UPLB cashier's office can improve the efficiency, accuracy, and security of their financial processes [3], [6]. A well-designed system not only facilitates the smooth execution of transactions but also tackles relevant computer science challenges such as offline-first data management, real-time queue synchronization, secure record keeping, and automated report generation.

This research aims to bridge the gap between traditional cashiering methods and modern digital systems by developing a responsive, cross-platform cashier application that simplifies the daily workflow of cashier administrators while enhancing the overall reliability and transparency of financial operations within the university.

B. Statement of the Problem

The traditional way of financial transaction management within the cashier's office has proven to be increasingly difficult. As the university strives for efficiency, transparency, and security in financial operations, there is a need for the development and implementation of a specialized cashier administrator application. This application is intended to serve as the foundation for our financial administrative system of cashiers.

C. Significance of the Study

This study aims to streamline the administrative processes of the cashier's office by providing a digital application capable of managing and viewing cashier transactions, as well as

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

generating daily financial reports efficiently. The system is designed to reduce manual workload, minimize errors in transaction records, and improve overall workflow. Additionally, the mobile application allows UPLB students to conveniently schedule appointments and access their transaction history, promoting transparency and enhancing the student experience.

D. Objectives of the Study

The general objective of this study is to develop a cash register system for the UPLB Financial Administration. Specifically, it aims:

1) To develop a web-based application that focuses on easing the collection process and report generation for the cashier administrators,
2) To develop a mobile application that will be used by the UPLB students to make payment appointments to the cashier's office, and;
3) To evaluate the accuracy of the web-based application's output and assess the usability of both the web-based and mobile applications.

E. Scope and Limitations of the Study

This study focuses on the development of the cashier application designed for the UPLB cashier's office. The system consist of two primary components: a web-based cashier application and a mobile application for students. The implementation of the application focuses on managing collection transactions made at the cashier's office and generating daily reports. The mobile application focuses on creating appointment tickets set with an expiration date in days for students to have a reservation slot with the cashiers payment. The mobile application does not support integrations with banking systems or online payment features.

F. Date and Place of the Study

The study was conducted in the first semester of the academic year 2024-2025 at the Institute of Computer Science, University of the Philippines Los Baños.

II. Review of Related Literature

Managing payment transactions through traditional manual methods often results in delays, inaccuracies, and limited accessibility for both students and cashier personnel. As the demand for more efficient, real-time, and user-friendly systems grows, digital cashiering solutions have become a necessity. However, with the increasing reliance on web and mobile technologies, safeguarding sensitive transactional data has also become a critical concern [7]. This highlights the importance of securing databases to ensure data integrity, privacy, and continuity in service.

Securing financial data is crucial to maintaining trust, integrity and confidentiality to the clients. Database Management Systems (DBMS) are softwares for creating, handling data storage, and management. In today's digital age, the Advanced Encryption Standard (AES) remains as the standard when it comes to safeguarding sensitive information. This review aims to comprehensively summarize the existing literature on Database Management Systems in mobile and web development. Additionally, this review will also expound on the concept of AES

A. Database Management System

Database management system (DBMS) refers to the system software for creating and managing databases. A DBMS makes it possible for end users to create, protect, read, update and delete data in a database [8]. In today's age, several databases are versatile enough to be used for both mobile and web applications. Here are some of the databases that developers can use for both mobile and web applications [11]:

1) Firebase Real-Time Database: It is a NoSQL cloud-hosted database services offered by Firebase Incorporation. Data is stored as a JSON tree and synchronized in real-time to every connected client and remains available when the application goes offline.

2) Cloud Firestore: It is a NoSQL document-based cloud-host database services that is also offered by Firebase Incorporation. Each document is grouped into collections may further point to other subcollections.

3) MongoDB: It is a NoSQL document-based database services offered by MongoDB Inc. It is similar to Cloud Firestore where each document is grouped into collections. It is still to be said the most popular NoSQL DBMS according to many sources [12], [13].

4) PostgreSQL: PostgreSQL is an open-source relational database where data is organized in tables, columns, and rows. It offers deployment options either on a self-managed cloud server or through a fully managed cloud service [26].

5) Redis: Redis is key-value store type non-relational database that natively provides fast response time [28]. This open-source database is often dubbed as the "world's most loved" database for its extreme performance and scalability [13].

6) Cassandra: Cassandra is a column type non-relational database mainly built for storing large amounts of data and can easily scales as data increases [27].

7) Neo4j: It is a graph store type non-relational database mostly used for systems that are heavily reliant on relationships that exists between data and utilizes its graph architecture to easily optimize complex queries.

Each database services mentioned have various data types covered and can easily be integrated with mobile clients using the appropriate SDKs. However, Dahunsi et al. [11] summarized from multiple sources [14]–[16] that MongoDB's solution does not provide as wide of cloud database management tools compared to the other databases, and data storage size based on subscription plans are what limits Cloud Firestore and Firebase Real-Time Database.

B. Advanced Encryption Standard

Data security refers to the protection of digital information from unauthorized access. This is achieved through

mechanisms such as user authentication, authorization, and access control [9]. Advanced Encryption Standard (AES) was created by two Belgian cryptographers, Vincent Rjimen and Joan Daemen, replacing the old Data Encryption Standard [10] AES uses a symmetric block cipher, which means it can both encrypt and decrypt data. When data is encrypted, it is converted into a code called cipher text that cannot be easily understood. Decrypting the cipher text converts the data back into its original form, known as plaintext. AES allows for block sizes of 128, 168, 192, 224, and 256 bits. It can use keys of various lengths, such as 128, 192, and 256 bits, to encrypt and decrypt data in blocks of 128 bits [10].

AES is the most common and widely used symmetric encryption algorithm nowadays. It has been discovered to be at least six times faster than triple Data Encryption Standard – a substitute to Data Encryption Standrd, which hackers easily defeated [17]. AES is largely considered to be impenetrable against all attacks, except for brute force, which attempts to decipher messages using all possibilities in the 128, 192, or 256-bit cipher [18]. It is the algorithm trusted as the standard by the U.S. government in 2001 and eventually evolved to the encryption standard for numerous organizations ad private sector enterprises [18], [19]. AES 128-bit encryption has never been cracked. Moreover, it will take years to cover all possibilities generated by 128-bit encryption. Now, AES 256-bit encryption has been developed and theoretically, it is true that AES 256-bit is harder to crack than AES 128-bit encryption [19].

AES encryption is now widely used in devices and applications that consumers use. These include SSDs for data storage, Google Cloud storage services, internet browser progams such as Firefox and Opera, and security certificates for websites. Many popular apps (such as Snapchat and Facebook Messenger) use AES encryption to safely send information such as photographs and messages. File compression programs such as WinRAR, Winzip, and 7z also use the AES algorithm to prevent data breaches. It is also implemented in the libraries of programming languages such as Java, Python, and C++ [20].

C. Cloud Messaging

Cloud messaging plays a crucial role in modern application development, particularly for real-time communication between clients and servers. It enables systems to push updates, alerts, or transactional information instantly across devices and platforms. This technology is widely adopted in mobile and web applications for its ability to deliver messages efficiently, reduce server load, and improve user engagement [29].

Firebase Cloud Messaging (FCM), a service provided by Google, is one of the most widely used cloud messaging solutions. FCM allows developers to send push notifications and data messages to iOS, Android, and web applications. It supports various types of messaging, including topic-based, device-to-device and condition-based messaging, making it ideal for flexible and scalable notification systems [30].

III. Materials and Methods

The machine with the following specifications was used in developing the mobile and web-based application for the UPLB EasyPay System:

- Operating System: Windows 11 Home Single Language
- Processor: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz
- Memory: 24 GB DDR4

A. Development Tools

The following software development tools and technologies were used in developing the application:

- ***Visual Studio Code***
  A source-code editor used as the primary application development environment.
- ***Flutter***
  A UI software development kit that was used to develop the mobile application.
- ***Firebase***
  A backend-as-a-service platform that served as the system's backend.
- ***Cloud Firestore***
  A flexible, scalable NoSQL cloud database that served as the database.
- ***LocalStorage/DexieDB***
  Used to enable offline functionality and advanced local data querying.

B. Features

This section is divided into two parts according to the type of user.

1) Cashier Administrator:

- *Register*
  Register an official cashier administrator account. Once all cashier administrators have an account, the application will be modified so that only registered cashier administrators have access to the application.
- *Login*
  Cashier administrators are required to log in using the credentials they registered to use the application.
- *View and Manage Appointment Queues*
  Cashier administrators can view appointment request details, cancel the appointment, and confirm the appointment once the client has paid the payment.
- *Edit Receipt details before printing in the Official Receipt*
  Cashier administrators can edit receipt details such as the account code for the collection, or add more information such as bank information or ADA in the bottom of the receipt if the payment method is "Check" or "ADA/Bank Transfers".
- *Print receipt details on the Official Receipt*
  Once the client has completed their payment, the cashier administrators will print the receipt on the Official Receipt with the client's transaction details.

- *Add Collections*
  Aside from the appointment system for the students, the cashier administrators can add collections to the system by looking up the client's name or student number in the search bar. If there are no client details in the system, then the cashier administrators can add the client details when adding the collection.
- *View and Manage Collection History*
  Cashier administrators can view collection history per collection session. They can also edit the collection for the generation of the report. They cannot delete a collection since it has been marked with an official receipt. They can only cancel a collection with its amount reset to zero and the collection details removed.
- *Generate Financial Reports*
  Cashier administrators can generate their reports for the day after the collection session for the day ends.
- *View Client Information*
  Cashier administrators can view client information such as their transaction history by looking them up using their name. Additionally, if the client is a UPLB student, the cashier administrators can look up their student number.

2) Normal User (UPLB Student):

- *Register*
  UPLB Students must register with their email and input additional information to gain access to the system.
- *Make Payment Requests*
  Users can make a payment request and will have an expiration date on their queue, typically within days.
- *View Transaction History*
  Users can view their appointment and transaction history, even if the said appointment has expired.
- *View and Edit Profile*
  Users can view and edit their profile information
- *Receive Notification* Users can receive notifications about their appointment details.

C. Types of Users

The following are the types of users that will use the applications:

1) Cashier Administrator (Web-based Application):

- The Cashier Administrator manages the appointment queues and confirms payment requests submitted by the students.
- The Cashier Administrators can manually input walk-in transactions for UPLB students and non-student clients.
- The Cashier Administrators can generate official and financial reports.
- Cashier administrators can view, filter, and manage the transaction history.

2) UPLB Students (Mobile Application):

- The Students can request payment appointment, which is added to the cashier's appointment queues. The request has an expiration date which means they have to make an appointment again if they want to be added to the queue.
- The student receives real-time notification about their queue status via Firebase Cloud Messaging.
- The student can view their transaction history.
- The student may update their information, such as their College should they switch courses in the future.

D. Student Appointment Flowchart

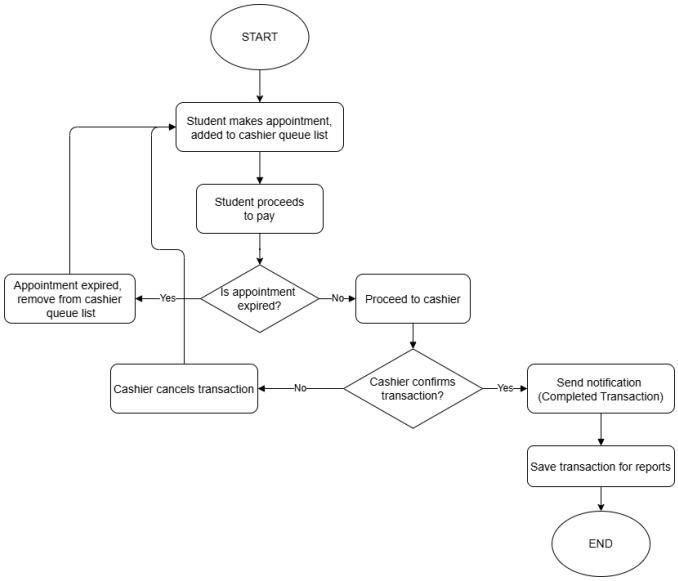

Fig. 1: *Simple Student Appointment Process*

Figure 1 illustrates a simple flowchart for the student appointment process from the mobile application to the cashier application.

E. Database Design

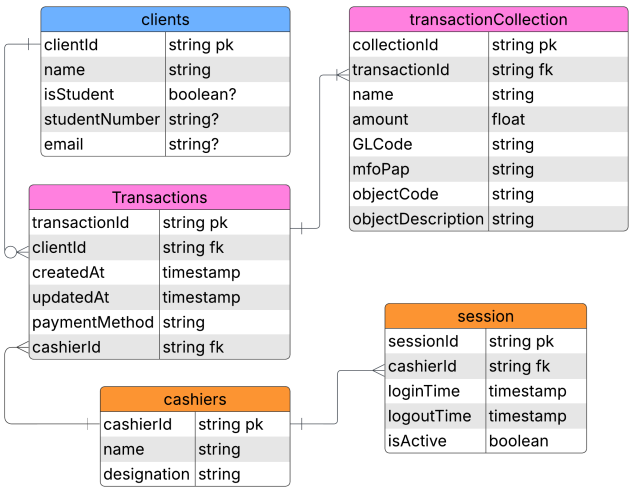

Fig. 2: *Entity Relationship Diagram*

Figure 2 illustrates the database design. It consists of four (4) entries: clients, transaction, cashier, and session. Dexie.js, a wrapper for indexedDB is used as the local database; and Firestore, a NoSQL database is used for syncing and

real-time syncing. While the ‘TransactionCollection‘ entity is depicted as a separate table to show the normalized, one-to-many relationship with ‘Transactions‘ (reflecting a collection of items), in the actual Dexie.js (IndexedDB) implementation, these collection items will be stored as an embedded array of objects directly within the ‘Transactions‘ record. This approach leverages IndexedDB’s flexibility for object storage while maintaining a clear logical model.

F. Authentication and Role-Based Access

Firebase Authentication will be used to manage user access and ensure role-specific permissions. User sessions will be authenticated during login, and each user’s identity will be verified before accessing relevant data. Firestore security rules will be configured so that users can only access documents that match their own identity. Only authenticated cashier administrators will be allowed to access, modify, or delete sensitive data, such as transactions and expired queues. The Firebase Rules Playground will be used to test and validate permission logic.

G. Offline Capability and Local Data Management

Offline capability for the cashier web app will be implemented using DexieDB, a wrapper around IndexedDB. IndexedDB is a low-level, client-side storage API that allows storing large amounts of structured data. DexieDB will simplify querying and managing data, enabling the administrator side of the system to function effectively even without an internet connection. Each transaction is encrypted using the Advanced Encryption Standard (AES) before being stored locally. An encryption key is used to securely encrypt and decrypt transaction data, reducing the risk of unauthorized access in case of data compromise. Session-based transaction history will be isolated per cashier login and will be cleared upon logout. Archived records will remain encrypted and accessible only through secure, authenticated access.

H. Cloud Storage and Synchronization

The Firebase Spark plan will be used throughout development. This plan includes quotas such as 50,000 document reads, 20,000 writes, 20,000 deletes, and 1 GB of storage per day on Cloud Firestore. These limitations are managed by optimizing query frequency and avoiding unnecessary reads and writes. For cloud-based storage, Firebase Cloud Firestore is used to backup and synchronize data. Sensitive fields are encrypted usign AES on the client side before transmission, ensuring data confidentiality even if the cloud is breached. Transaction logs, appointments, and queue data are backed up in Firestore, allowing mobile application users to view data across devices.

I. Appointment and Queue Management System

Students use the mobile application to submit appointment requests for cashier transactions. These requests have an expiration date and are stored in a “queues” collection in Firestore, with each document named using the format YYYY-MM-DD and containing a subcollection ”queue” for individual student requests. Cashier administrators can approve or cancel requests. Confirmed transactions are moved to the ’transactions’ collection with the status ’Completed’. Expired or canceled appointments are labeled ’Appointment Expired’ or ’Canceled’ and are automatically deleted from the queue based on server-side timestamp checks or the cashier cancels the appointment. The system sends real-time notifications using Firebase Cloud Messaging (FCM). Upon login, each user is assigned a unique FCM token that is stored in Firestore. When an appointment status changes, the system sends HTTP requests to the FCM API to notify the user via push notifications.

J. Performance Optimization and Cost Management

To minimize Firestore usage costs and improve efficiency:

- Expired appointment checks are deferred until the end of a cashier session
- Only essential data is queried or written
- All sensitive data is encrypted before being saved
- Firestore documents are designed to avoid excessive nesting or unnecessary fields

The combined use of DexieDB for local storage and Firestore for cloud synchronization ensures that the system is resilient, scalable, and suitable for the operational constraints of the cashier’s office.

K. User Evaluation

The web-based application was subjected to validation testing conducted by four (4) cashier administrators and six (6) non-cashier administrative personnel, for a total of ten (10) participants from the cashier’s office. This testing aimed to verify the usability and accuracy of the outputs generated by the system, such as reports and receipts intended for clients.

The evaluation form was divided into three sections: (1) ease of use and navigation, (2) output accuracy, and (3) system and functional behavior. Binary Yes/No checklists were employed to assess specific features. Each “Yes” response was interpreted as a positive outcome, while “No” indicated issues or areas for improvement. The total number of “Yes” responses was converted into a percentage to represent the level of positive feedback per feature. A higher percentage reflected greater usability, reliability, and user satisfaction [31]. Open-ended feedback was also collected for qualitative suggestions and improvements. The list of questions for this form is in Appendix I.

To assess the mobile application’s usability, a test was conducted with 20 UPLB students. They completed the System Usability Scale (SUS), a standardized 10-item questionnaire with responses ranging from “Strongly Disagree” to “Strongly Agree.” Participants also had the opportunity to offer feedback, sharing their personal experiences and ideas for improving the application.

IV. Results and Discussion

The resulting system consisted of a web-based application for cashier administrators and a mobile application for UPLB students.

A. Development of the Web Application

The Web Application was developed using React with TypeScript, styled with Tailwind CSS for a responsive and modern interface. Figure 3 illustrates the full view of the Cashier Administrators dashboard.

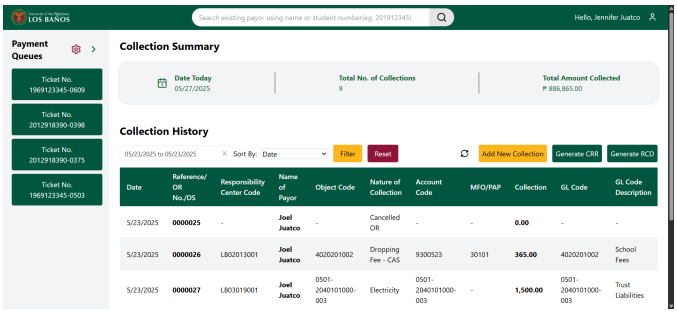

Fig. 3: *Cashier Administrator Dashboard*

The application features the Queue Sidebar, which displays all active queues submitted by the students through the mobile application. A separate tab displays the Pending Queues, which are appointments that have been received but the students have not shown up for the payment process and still within their valid request window. Expired queues are automatically removed from the Cloud Firestore to preserve storage space and maintain performance.

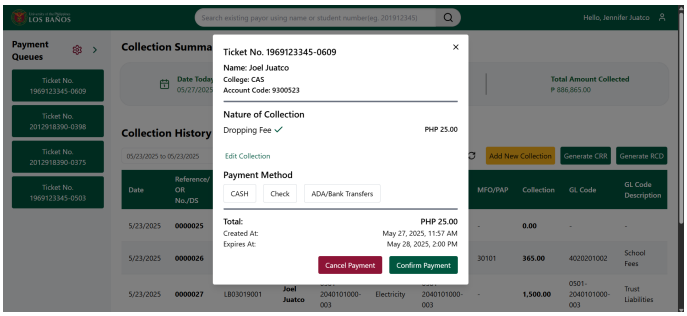

Fig. 4: Queue Item

When the cashier selects an appointment from the Queue sidebar, the appointment details modal, which is illustrated in Figure 4, presents the following information relevant to the transaction:

1) *Client Information*
   This includes the name, student number, college, and email in a compact header to quickly verify the identity of the student.
2) *Collection Items*
   Below the client information header is the list of collection items displayed with their respective amount provided by the student from the mobile application. This list can be modified if the student changes their mind or made an error while creating the appointment.
3) *Payment Method Selection*
   This section allows the cashier to choose between the allowed payment methods namely Cash, Check, or ADA/Bank Transfers.
4) *Total Amount Due*
   The total is displayed in bold, large font below the payment method selector. This ensures the cashier can immediately confirm the amount before proceeding.
5) *Action Buttons*
   The last section of the modal are the action buttons, styled for clarity and distinction. These are the Cancel payment and Confirm payment buttons. The Cancel payment, when clicked, simply cancels the appointment after the cashier confirms from the confirmation dialog box that will display. The Confirm payment action button displays another window that requires the cashier to input the OR number to successfully confirm the transaction after the client gives the payment.

Through this, Cashiers will no longer need to hand-write and track details of the OR; instead, the application streamlines each payment from client requests through official receipt issuance. Moreover, when the cashier confirms the payment, the application will prompt the cashier to print the receipt, a new modal will display the sample template with the receipt details. Cashier administrators can edit the receipt, such as adding collection codes or bank information, before printing in the official receipt.

Walk-in clients, including both non-student individuals and student clients who do not have access to the mobile application, are still accommodated by the system. The web application features an “Add New Collection” action button that, when clicked, opens a modal form requiring the necessary information for processing a transaction. Figure 5 illustrates an example which a walk-in client proceeds with their payment. If the client’s information already exists in the system, the cashier administrator can easily retrieve it using the search bar located in the navigation header or directly in the modal. Otherwise, the cashier can manually input the client’s details into the form. Once the transaction is successfully completed, the newly entered client information is automatically stored in the system for future reference.

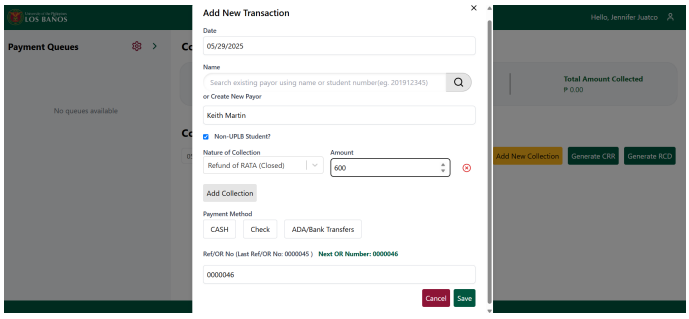

Fig. 5: *Adding Collection for walk-in client*

The cashier administrator can access and view client records through the search bar at the top of the interface. This functionality leads to the Client Information page, which contains the client's personal details along with a comprehensive transaction history. From this page, the cashier administrator also has the option to add new collection directly. Figure 6 shows the layout and features of the Client Information interface.

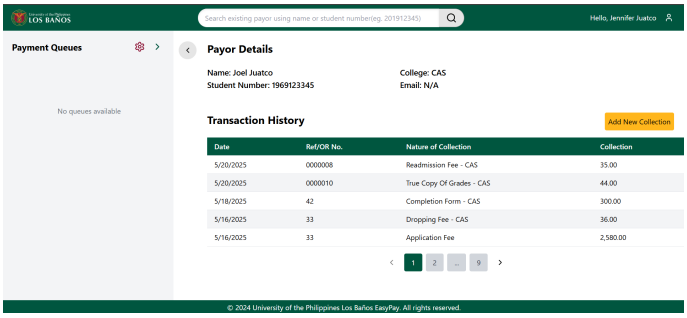

Fig. 6: *Client Details Page*

Beneath the header section of the dashboard is the Collection Summary panel for the current session. This summary provides an overview of the total number of collections made and the corresponding total amount received by the cashier administrator. Directly below the summary, the Transaction History section displays a detailed, filterable table of all collections recorded within the session. Each entry in the transaction history can be edited, and any modifications are logged and displayed in an editable history modal for audit and tracking purposes.

The system does not allow the deletion of transaction entries. Instead, if a transaction needs to be voided, the transaction can be edited with a status of "Canceled OR" and the associated amount is automatically adjusted to zero. This is done to maintain the integrity of the official receipt (OR) numbering system, which is critical for record-keeping and financial audits. Canceled ORs are still included in summary reports and logs to ensure that all receipt numbers are accounted for and that no discrepancies arise in the documentation. To preserve accountability and transparency, edited transaction's history can be viewed via the "Check History" button in the action bar. Figure 7 illustrates an example of an edited transaction through its history.

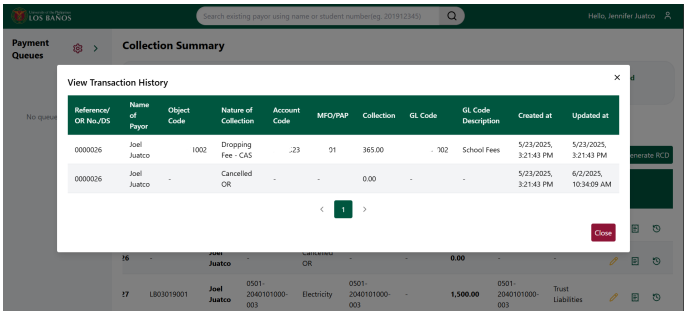

Fig. 7: Transaction History

The system includes a built-in feature that allows the cashier administrator to generate official report documents in PDF format. Two types of reports can be generated through the application: the Cash Receipts Record (CRR) and the Review of Collection and Deposit (RCD). These reports are generated automatically and formatted to follow the standard institutional layout. Upon generation, the system compiles all collection data recorded for the current session or selected day, including each transaction's details such as account code, object code, amount collected, and associated official receipt (OR) numbers. The system also calculates and includes the expected deposit total based on the accumulated collections. This deposit information is reflected in the RCD report, streamlining the process of documenting and reviewing cash deposits. The automated generation of these tables reduces manual paperwork and ensures consistency and accuracy in financial reporting, allowing the cashier to focus on transaction handling while minimizing administrative workload. Figures 8 and 9 shows the automatically generated reports.

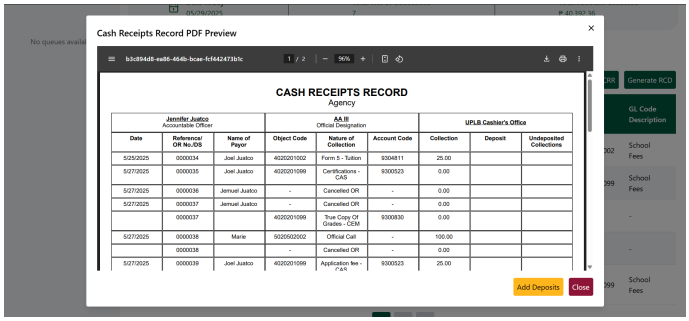

Fig. 8: *Generated Report 1*

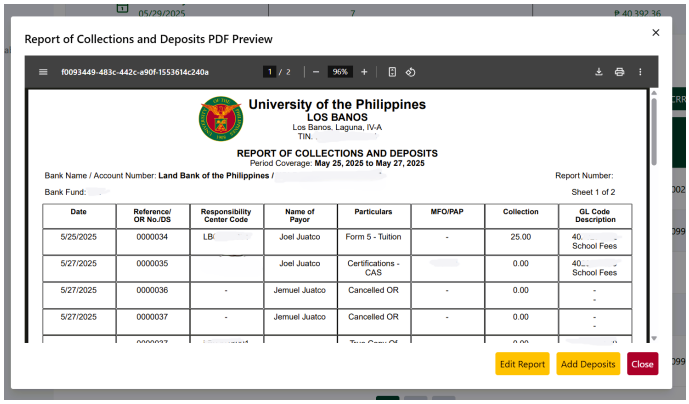

Fig. 9: *Generated Report 2*

B. Development of Mobile Application

The mobile application was developed using Flutter, a cross-platform framework that enabled seamless deployment on both Android and iOS devices. Upon launching the application, registered users are greeted with a dashboard that prominently displays important cashier information, including the designated office hours. This ensures that students are informed of service availability before initiating any transaction. Figure 10 shows the homepage of the mobile application.

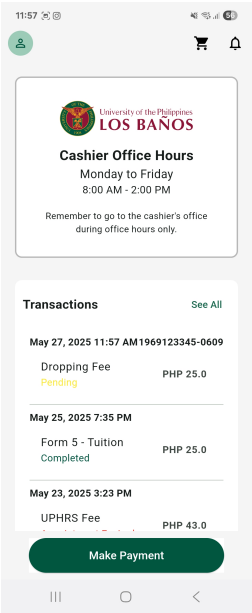

Fig. 10: *Mobile App Homepage*

Beneath the header, users are presented with a summary of their most recent transactions. Tapping on the View All text button in the top right corner of the widget redirects the user to a dedicated page that contains a view of their transaction history, allowing them to review past payments. At the bottom of the home screen is an action button labeled ”Make Payment”. Selecting this button navigates the user to a new page where they can choose the type of collection they wish to settle and input the corresponding amount.

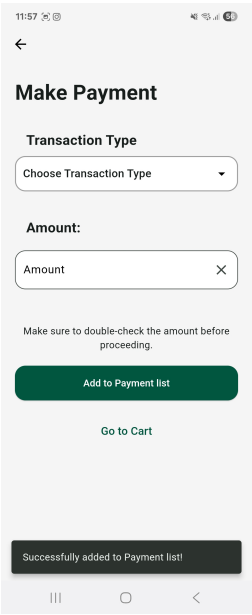

Fig. 11: *Mobile App Make Payment Page*

Figure 11 illustrates the make payment page. This modular form supports multiple entries, enabling users to add various collection types into a virtual cart. Once satisfied with their selections, users proceed to a Review Details page, where all payment items are summarized. This step ensures that users can verify the accuracy of the entered data before submitting their request. Figure 12 presents this page.

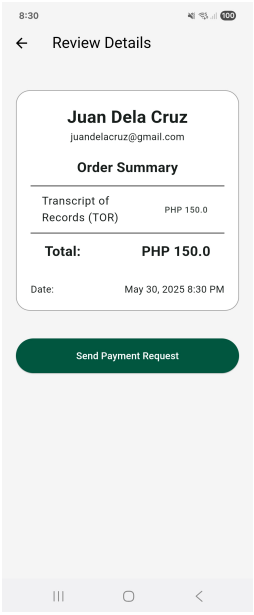

Fig. 12: *Mobile App Checkout Review*

After confirming the details and submitting the transaction request, the system generates a unique queue ticket number in the format of the student’s ID number followed by four randomly generated digits (e.g., 202512346-1234). The application also displays the expiration time for the ticket, measured in days, to inform the user of the request’s validity. Figure 13 shows the successful appointment

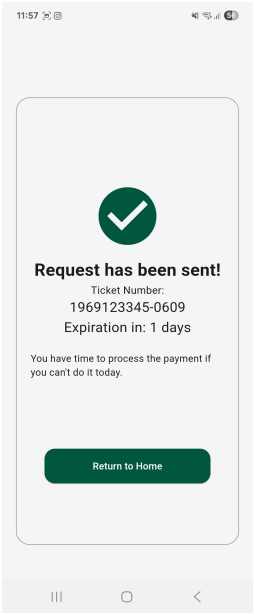

Fig. 13: *Mobile App Successful Appointment with ticket generation*

C. Testing of the Web-Based Application

The web-based application underwent validation testing to evaluate its usability, accuracy, and overall functionality. The

testing process consisted of three parts:

1) Task-Based Evaluation (Rating Scale 1–5):
   Participants were asked to complete a series of tasks within the application. After each task, they rated two criteria:
   - Difficulty of the task
   - Accuracy of the output

   The scoring system used for this section ranged from 1 to 5, where 5 is the best score (e.g., easiest or most accurate) and 1 is the worst.
2) Binary Evaluation (Yes/No):
   Participants responded to five binary questions related to system performance, interface clarity, and user satisfaction. These were answered with a simple Yes or No, where:
   - Yes indicates a positive response and marked "1" (e.g., feature is present or satisfactory)
   - No indicates a negative response or the absence of a feature and marked as "0"

   The final question in this section was intentionally framed so that the correct or expected answer was "No" (e.g., "Were there any bugs found during testing").
3) Open-Ended Feedback:
   The last part of the validation collected qualitative feedback. Participants were encouraged to share comments, suggestions, or any issues encountered while using the application. This section allowed testers to elaborate on their experience beyond structured ratings.

The second section of the validation testing comprised of open-ended feedback. The following are the questions in the open feedback section:

- Please describe all of the bugs and errors you found while testing the application below
- Please describe features that you expected but is missing in the application
- Please provide your suggestions for the the improvement of the application
- Overall impression of the application

The feedback form was completed by the participants after testing the application. Individual and average scores from all parts of the form were computed and normalized to ensure consistency. Table I presents the scores for the ease of use section. After calculating the mean for each question and averaging the results in all 10 items, the application received an overall score of 4.46. This indicates that the system is generally easy to use and navigate. One of the highest-rated items was the task of canceling an already punched receipt, suggesting that this was the easiest function to perform within the application.

TABLE I: Section 1A Scores on Perceived Difficulty (Higher = Easier)

Q1A	Q2A	Q3A	Q4A	Q5A	Q6A	Q7A	Q8A	Q9A	Q10A	Mean Score
4.4	4.4	4.4	4.5	4.5	4.5	4.6	4.4	4.5	4.4	**4.46**

Table II displays the mean scores for each question. The overall average score is 4.4, indicating that the application's outputs are generally accurate. The highest-rated item was the accuracy of the fund code assigned to each selected collection.

TABLE II: Section 1B Scores on Accuracy Level (Higher = More Accurate)

Q1B	Q2B	Q3B	Q4B	Q5B	Mean
4.4	4.4	4.3	4.4	4.5	**4.4**

Table III presents the average percentage of participant responses regarding the system's functionality and behavior. This section is based on binary Yes or No questions, which are listed in Table VII in Appendix I. Question 5 received the fewest "Yes" responses, indicating that some participants encountered bugs while testing the application. After calculating the average of all responses, the application received a 94

TABLE III: Section 1C Percentage on the Functional Behavior

Questions	YES	NO	Percentage
Q1B	10	0	100%
Q2B	10	0	100%
Q3B	10	0	100%
Q4B	10	0	100%
Q5B	7	3	70%
Average Percentage			**94**%

The resulting additional feedback from the cashier administrators indicated a generally positive experience with the application. The overall impression was that the system is user-friendly and will significantly assist in day-to-day cashier operations. Several constructive suggestions were provided to further improve the application's usability and efficiency. These included the addition of direct print and download buttons for generating reports and receipts, adjustments to the alignment of amount fields in the official receipt, and the inclusion of an automatic total computation per payment type (e.g., cash, check, ADA) directly in the generated report to facilitate easier verification.

Some of these suggested improvements were immediately implemented in the web application after feedback, demonstrating the system's adaptability and commitment to continuous enhancement based on end-user input. The remainder of the open feedback focused primarily on user interface refinements to improve clarity, layout consistency, and overall visual appeal.

D. Testing of the Mobile Application

The mobile application was tested and evaluated by twenty (20) UPLB students to assess the usability of the application. The participants were asked to complete the System Usability Scale (SUS), a standardized 10-item questionnaire that measures usability. Responses were rated on a scale from "Strongly Disagree" to "Strongly Agree." The items in the questionnaire are as follows:

After the participants completed the survey, the individual scores and the overall mean scores were calculated following

TABLE IV: System Usability Scale (SUS) Scores of Participants

Resp.	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Odd	Even	Score
1	5	1	5	1	5	1	5	1	5	1	20	20	100.00
2	4	2	4	2	4	2	4	2	3	2	14	15	72.50
3	5	1	5	1	5	1	5	1	5	2	20	19	97.50
4	5	1	5	1	5	1	5	1	4	4	19	17	90.00
5	5	2	5	1	5	1	5	1	5	1	20	19	97.50
6	3	2	5	1	4	2	4	4	5	1	16	15	77.50
7	5	1	5	1	5	1	5	1	5	1	20	20	100.00
8	5	1	5	1	5	1	5	1	5	1	20	20	100.00
9	5	1	5	1	5	1	5	1	5	1	20	20	100.00
10	5	1	4	2	4	1	4	2	4	2	16	17	82.50
11	4	1	5	1	4	1	4	1	5	2	17	19	90.00
12	5	1	5	1	5	1	5	1	5	3	20	18	95.00
13	5	2	5	1	5	2	4	2	4	1	18	17	87.50
14	5	2	3	3	5	2	5	3	5	3	18	12	75.00
15	5	2	4	1	5	1	5	1	5	1	19	19	95.00
16	5	2	5	2	5	2	5	1	4	2	19	16	87.00
17	5	1	5	1	5	1	4	1	4	1	18	20	95.00
18	5	1	5	1	5	1	5	1	5	1	20	20	100.00
19	5	2	5	1	5	2	5	1	5	2	20	17	92.50
20	5	1	5	1	5	2	5	2	4	2	19	17	90.00
											Average Score		**91.25**

the SUS computation rules. The individual scores are presented in Table IV. The mobile application received a score of 91.25 out of 100, which is considered above average. The results suggest that the mobile application is usable for its intended users. Participants also shared suggestions to further improve the application. Among the suggestions was to make the application an all-in-one platform by adding features such as online payment and direct notifications to the cashier and college secretaries.

V. Conclusion and Future Work

The study successfully developed a system that streamlines daily collections, report generation, and queue management for the UPLB cashier's office. The web-based application was able to provide essential features such as real-time transaction queueing, session-based collection tracking, generation of official receipts, and daily reports.

Validation testing conducted with multiple cashier administrators and personnel confirmed the application's usability, accuracy of outputs, and functional reliability. Their feedback highlighted that the system is user-friendly and beneficial for daily operations. Several suggestions were immediately integrated, further refining the user experience and efficiency. The mobile application was tested using SUS and got a 91.25 out of 100, indicating that the application is above average. The participants also shared their suggestions, the online payment feature is the most popular suggestion.

For future work, enhancements to the reporting interface, extended data analytics, and more flexible user management are recommended. Other features mentioned are already in the development stage and are planned to be included once the application is installed on the cashier's PC. The mobile application provides great potential to be an all-in-one application for student payment in transactions, further development of the application will greatly benefit the constituents of the university.

Appendix I
Questionnaire for the Validation Testing of the Web-Based Application

TABLE V: Section 1a: Feature-specific Evaluation according to Ease of use and Navigation

Test Item	Yes	No	Comments (optional)	Difficulty Level
Is the application easy to navigate?	☐	☐		
Is the user interface clear and understandable?	☐	☐		
Is the app responsive (works on different screen sizes)?	☐	☐		
Is confirming/canceling queued transactions easy?	☐	☐		
Is it easy to create and process walk-in transactions?	☐	☐		
Is it easy to edit transactions?	☐	☐		
Is it easy to cancel already punched ORs?	☐	☐		
Is it easy to generate reports?	☐	☐		
Is it easy to edit the generated report?	☐	☐		
Is it easy to navigate and change fund codes?	☐	☐		

TABLE VI: Section 1b: Feature-specific Evaluation according to accuracy of the output

Test Item	Yes	No	Comments (optional)	Accuracy Level
Is the correct transaction summary displayed?	☐	☐		
Are all computed totals accurate? (total amount, total deposit etc)	☐	☐		
Is the official receipt generated correctly?	☐	☐		
Are past transactions viewable and accurate?	☐	☐		
Does the application output the correct Fund codes according to the chosen nature of collection?	☐	☐		

TABLE VII: Section 1c: System and Functional Behavior

Test Item	Yes	No	Comments (optional)	
Is the transaction queue updating in real-time?	☐	☐		
Does the application correctly save each transaction?	☐	☐		
Is the cashier login/logout features working properly	☐	☐		
Does the application function offline as expected?	☐	☐		
Are there any errors or bugs encountered during use?	☐	☐		

Appendix II
SUS Questionnaire for the Mobile Application

1) I think I would like to use this tool frequently.
2) I found the tool unnecessarily complex.

3) I thought the tool was easy to use.
4) I think that I would need the support of a technical person to be able to use this system.
5) I found the various functions in this tool were well integrated.
6) I thought there was too much inconsistency in this tool.
7) I would imagine that most people would learn to use this tool very quickly.
8) I found the tool very cumbersome to use.
9) I felt very confident using the tool.
10) I needed to learn a lot of things before I could get going with this tool.

ACKNOWLEDGMENT

I would like to extend my sincere gratitude to my adviser, Assoc. Prof. Concepcion L. Khan, whose guidance has been invaluable. To my family and loved ones, for your love and belief in me have made all the difference. And finally, to my friends, thank you for your unwavering support and encouragement.

REFERENCES

[1] Wavetec. *How Manual Cash Handling Impacts Retailers Negatively*. Wavetec Blog, Nov. 2024. [Online]. Available: https://www.wavetec.com/blog/how-manual-cash-handling-impacts-retailers-negatively/
[2] Sesami. *6 reasons why manual cash handling is hurting retailers*. Sesami Blog, Apr. 2023. [Online]. Available: https://www.sesami.io/blog/6-disadvantages-of-manual-cash-handling
[3] MyEduComm. *9 Advantages of Digital Payment System in Educational Institutions*. MyEduComm, Aug. 2021. [Online]. Available: https://www.myeducomm.com/blog/9-advantages-of-digital-payment-system-in-educational-institutions/
[4] OpenEduCat Inc. *9 Benefits of Implementing Digital Payment Systems in Educational Institutions*. OpenEduCat Blog, Feb. 2024. [Online]. Available: https://openeducat.org/blog/our-blog-1/post/9-benefits-of-implementing-digital-payment-systems-in-educational-institutions-107
[5] Payment Pro. *Benefits of an automated bursar's office cashiering system*. Online. [Online]. Available: https://www.payment-pro.com/benefits.html
[6] Academia.edu. *Cashiering system*. Online. [Online]. Available: https://www.academia.edu/11264785/Cashiering_system
[7] PayMyTuition. *The Hidden Dangers: Exploring Data Threats in Educational Payment Systems*. 2023. [Online]. Available: https://www.paymytuition.com/resources/blog/the-hidden-dangers-exploring-data-threats-in-educational-payment-systems/
[8] C. S. Mullins. *What is a DBMS? Database Management System Definition*. Data Management, Dec. 2023. [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/database-management-system
[9] M. C. Murray. *Database Security: What Students Need to Know*. Digital-Commons@Kennesaw State University, Nov. 2022. [Online]. Available: https://digitalcommons.kennesaw.edu/facpubs/1389/
[10] D. Selent. *Advanced Encryption Standard*. Rivier Academic Journal, Fall 2010. [Online]. Available: https://www2.rivier.edu/journal/ROAJ-Fall-2010/J455-Selent-AES.pdf
[11] F. M. Dahunsi, A. J. Joseph, O. A. Sarumi, and O. O. Obe. *Database Management System for Mobile Crowdsourcing Applications*. Nigerian Journal of Technology, vol. 40, no. 4, pp. 713–727, 2021. doi:10.4314/njt.v40i4.18
[12] BairesDev. *4 Best & Most Popular NoSQL Databases*. Dec. 2023. [Online]. Available: https://www.bairesdev.com/blog/nosql-databases/
[13] M. Pandey. *8 Most Popular NoSQL Databases*. Analytics India Magazine, Dec. 2023. [Online]. Available: https://analyticsindiamag.com/8-most-popular-nosql-databases/
[14] Firebase. *Installation & Setup on Android — Firebase Realtime Database*. 2020. [Online]. Available: https://firebase.google.com/docs/database/android/start
[15] Firebase. *Firebase Pricing*. 2020. [Online]. Available: https://firebase.google.com/pricing
[16] MongoDB. *MongoDB Limits and Thresholds — MongoDB Manual*. 2020.
[17] H. Saini. *8 Strongest Data Encryption Algorithms in Cryptography*. Analytics Steps, Jan. 2024. [Online]. Available: https://www.analyticssteps.com/blogs/8-strongest-data-encryption-algorithms-cryptography
[18] Arcserve. *5 Common Encryption Algorithms and the Unbreakables of the Future*. Jan. 2024. [Online]. Available: https://www.arcserve.com/blog/5-common-encryption-algorithms-and-unbreakables-future
[19] IDERA. *AES-256-Bit Encryption*. Jan. 2024. [Online]. Available: https://www.idera.com/aes-256-bit-encryption/
[20] Sunny Valley Networks. *What is the Advanced Encryption Standard (AES)?* Jan. 2024. [Online]. Available: https://www.sunnyvalley.io/docs/network-security-tutorials/what-is-advanced-encryption-standard-aes
[21] Google. *Firebase Realtime Database*. [Online]. Available: https://firebase.google.com/docs/database
[22] Google. *Firestore — Firebase*. [Online]. Available: https://firebase.google.com/docs/firestore
[23] Neo4j. *Neo4j Graph Database & Analytics – The Leader in Graph Databases*. [Online]. Available: https://neo4j.com/
[24] Neo4j. *Neo4j Graph Database*. [Online]. Available: https://neo4j.com/product/neo4j-graph-database/
[25] MongoDB. *Why Use MongoDB and When to Use It?* [Online]. Available: https://www.mongodb.com/why-use-mongodb
[26] PostgreSQL. *About*. [Online]. Available: https://www.postgresql.org/about/
[27] TutorialsPoint. *Cassandra - Introduction*. [Online]. Available: https://www.tutorialspoint.com/cassandra/cassandra_introduction.htm
[28] Redis. *Redis*. [Online]. Available: https://redis.io/
[29] Sage IT. *What is Cloud Messaging?* [Online]. Available: https://sageitinc.com/reference-center/what-is-cloud-messaging
[30] Google. *Firebase Cloud Messaging Documentation*. [Online]. Available: https://firebase.google.com/docs/cloud-messaging
[31] American Intercontinental University. *Customer Analytics Marketing*. [Online]. Available: https://students.aiu.edu/submissions/profiles/resources/onlineBook/Q9C9A2_customer%20analytics%20marketing.pdf

Jemuel Juatco Jemuel is a BS Computer Science student in University of the Philippines Los Baños. He is from San Leonardo City, Nueva Ecija. He loves drawing, painting, and playing mobile and computer games.