Secure and Idempotent Batch Provisioning of Student Accounts for Hostel Mess Management Systems Using Supabase Auth and Streamlit

Megha Sangle[1], Shreya Gawai[2], Rasika Gondchar[3], Prasad Chopade[4], Pravin Thakre[5]
[1,2,3,4] *Student, Bachelor of Engineering, Information Technology, Mauli Group of Institutions' College of Engineering and Technology, Shegaon*
[5]*Assistant Professor, Information Technology, Mauli Group of Institutions' College of Engineering and Technology, Shegaon*

***Abstract*—Hostel mess management systems are increasingly being adopted to digitize student registration, meal attendance, billing, menu selection, and administrative reporting. Although many existing systems focus on operational modules such as attendance tracking and feedback collection, limited attention has been given to the secure provisioning of student accounts and the integrity of identity-linked records. In real-world deployments, weak onboarding practices such as duplicate account creation, inconsistent database mappings, insecure default passwords, and improper handling of elevated credentials can lead to operational failures and security vulnerabilities.This paper proposes a secure and idempotent batch provisioning framework for student accounts in hostel mess management systems using Supabase Auth Admin APIs and Streamlit. The proposed framework creates student identities through a three-stage pipeline involving authentication account creation, profile insertion, and hostel-specific student record generation. To ensure safe re-execution of the provisioning process, the framework incorporates idempotent operations using unique constraints, primary keys, and upsert mechanisms. The architecture also integrates role-based access control, Row Level Security, and secure handling of service-role credentials to prevent unauthorized access and maintain database integrity. The study further examines password lifecycle management, temporary credential strategies, recovery from partial failures, and scalability considerations for large student cohorts. Experimental analysis demonstrates that the proposed approach minimizes duplicate records, preserves referential consistency across authentication and application tables, and supports reliable onboarding even when the provisioning script is re-run multiple times. The framework provides a production-oriented foundation for secure student onboarding and can be extended to support additional hostel mess management functionalities such as attendance monitoring, QR-based meal authentication, billing, and feedback systems.**

***Index Terms*—Hostel Mess Management System, Student Account Provisioning, Batch User Creation, Idempotency, Supabase Auth, Streamlit, Row Level Security, RBAC, Student Onboarding, Database Integrity, Authentication, Access Control, Secure Provisioning**

I. INTRODUCTION

Hostel mess management is an essential component of educational institutions, particularly in colleges and universities where large numbers of students reside on campus. Among the various hostel operations, mess management is one of the most critical because it directly affects meal planning, attendance tracking, billing, menu management, and student satisfaction. Traditional hostel mess systems often rely on manual registers, paper-based attendance, handwritten billing records, and physical verification methods, which are time-consuming, error-prone, and difficult to scale. These manual approaches frequently lead to issues such as duplicate records, inaccurate attendance, delayed bill generation, poor feedback tracking, and increased administrative workload.

To overcome these limitations, many institutions are adopting digital hostel mess management systems that provide functionalities such as online student registration, meal attendance recording, menu display,

feedback submission, and automated report generation. These systems generally include separate student and administrator modules, where students access mess-related services while administrators manage records, monitor attendance, and generate bills. Recent systems have also started integrating technologies such as QR-based authentication, digital dashboards, and real-time notifications to improve operational efficiency and transparency.

Although existing research and development efforts have focused heavily on attendance management, billing automation, and QR-based authentication, comparatively little attention has been given to one of the most important foundational components of the system: secure student account provisioning. In practical deployments, administrators often need to create hundreds or even thousands of student accounts in bulk. Each account must be uniquely associated with authentication credentials, a role-based profile, and hostel-specific information such as roll number, hostel name, room number, and account status. If this onboarding process is not designed properly, the system may suffer from duplicate accounts, mismatched records, weak default passwords, leaked administrative credentials, or inconsistent mappings between authentication and application tables. Such issues can affect not only login functionality but also downstream features such as meal attendance, billing accuracy, and reporting reliability.

In modern backend architectures, secure identity management is commonly achieved through authentication platforms combined with role-based access control and database-level security policies. Supabase provides a suitable platform for implementing such systems because it offers authentication services, PostgreSQL-based storage, Row Level Security (RLS), and administrative APIs for user creation and management. Similarly, Streamlit can be used to build lightweight administrative interfaces for provisioning student accounts in bulk. However, because the service-role credentials used for administrative actions bypass normal security restrictions, they must be handled carefully to avoid accidental exposure or misuse. Furthermore, repeated execution of provisioning scripts should not create duplicate users or conflicting records, making idempotency a major design requirement.

This paper proposes a secure and idempotent framework for batch provisioning of student accounts in hostel mess management systems using Supabase Auth Admin APIs and Streamlit. The proposed approach follows a structured three-stage onboarding process in which an authentication account is first created, followed by insertion of profile information and hostel-specific student records. The framework also incorporates unique constraints, upsert mechanisms, Row Level Security, role-based access control, and password lifecycle management to ensure that the system remains secure, scalable, and reliable even when provisioning operations are repeated.

The main objectives of this research are to develop a secure onboarding architecture for student accounts, ensure idempotent execution of batch provisioning operations, maintain consistency between authentication and application-level tables, and provide a production-oriented model that can be extended to support other hostel mess management functionalities. The proposed framework is intended not only for small pilot deployments involving a few students, but also for large-scale hostel environments involving hundreds or thousands of student accounts.

II. LITERATURE REVIEW

The digitization of hostel operations has attracted significant attention in recent years, particularly in areas such as student registration, room allocation, attendance monitoring, billing, and mess administration. Existing hostel management systems are primarily motivated by the need to reduce manual workload, improve record accuracy, and increase administrative efficiency. Several studies and implementation-oriented reports have shown that traditional hostel processes depend heavily on paper-based records and manual verification, which often lead to delayed processing, duplication of entries, and difficulty in maintaining consistent student data across modules.

A major stream of prior work focuses specifically on mess management systems as web-based platforms for handling menu display, student meal attendance, billing, and feedback collection. These systems generally provide separate interfaces for students and administrators. Students are typically allowed to register, check meal schedules, mark attendance, and submit feedback, whereas administrators monitor attendance, update menus, and generate monthly bills. The common objective across these systems is to improve transparency and reduce operational errors in mess administration. However, in many such studies, the student registration process is treated only as a functional front-end feature rather than as a deeper problem of secure identity creation, authorization, and record consistency across authentication and database layers.

Another important direction in the literature involves secure meal authentication mechanisms, especially QR-code-based systems for mess access and attendance verification. These works emphasize fast authentication, real-time updates, reduced waiting time, and improved reliability in meal usage records. They often highlight the importance of secure student verification before allowing meal selection or attendance marking. Such systems strengthen the role of authentication in mess management and demonstrate that student identity is central to operational correctness. Nevertheless, while they discuss secure access at the point of meal usage, they generally do not provide a detailed treatment of how student identities are initially provisioned, how duplicate identities are avoided, or how role-based privileges are enforced during account creation.

From a systems-design perspective, modern backend platforms such as Supabase introduce capabilities that are highly relevant to student-facing mess applications, including authentication services, relational data storage, Row Level Security (RLS), and administrative APIs. Prior platform documentation and practice-oriented designs recommend maintaining a separation between authentication data and public application data by linking domain tables such as profiles or students to authenticated user identities. This architectural pattern supports cleaner user management and stronger access control. It also enables role separation between students and administrators, which is essential in hostel mess systems. However, platform capabilities alone do not automatically solve the practical issue of large-scale onboarding, especially when administrators must create many student accounts in batches while preserving integrity across authentication and application tables.

A recurring concern in the broader security literature is the risk associated with weak onboarding practices, including predictable default passwords, exposed elevated credentials, and inconsistent mappings between identity records and application-specific tables. In systems where administrative APIs are used with privileged credentials, the risk becomes even greater because such credentials may bypass ordinary database protections. Existing guidance therefore stresses secure secret management, strong password handling, role-based access control, and careful restriction of elevated operations to trusted server-side environments. These recommendations are directly applicable to digital mess management systems, where insecure onboarding can undermine all downstream modules, including attendance tracking, billing, and reporting.

Despite the growing literature on hostel automation and mess digitization, a clear research gap remains. Most existing work gives substantial attention to functional modules such as attendance marking, QR authentication, and billing automation, but offers limited detail on secure and idempotent batch provisioning of student accounts. In practice, this is a crucial requirement, because a mess management system cannot operate reliably unless each student has exactly one valid identity, one consistent role-bearing profile, and one correctly linked hostel-specific record. The absence of a well-defined provisioning framework may result in duplicate users, partial record creation, broken role assignment, and unreliable system behavior during repeated administrative operations.

Therefore, the present work addresses this gap by focusing on a secure provisioning framework for hostel mess management systems. Unlike prior studies that primarily emphasize service modules, this paper

investigates the onboarding substrate itself: batch creation of student authentication accounts, insertion of role-based profile data, insertion of hostel-specific student records, and the use of idempotent database operations to support safe re-execution. In this way, the proposed work complements existing mess management research by strengthening the foundational identity layer on which attendance, billing, and meal authentication ultimately depend.

III. SYSTEM DESIGN AND METHODOLOGY

The proposed hostel mess management system is designed as a secure web-based platform that supports student onboarding, identity management, mess attendance, hostel-specific record maintenance, and administrative operations. The system follows a modular architecture in which authentication, application-level profiles, and student-specific mess records are separated into different layers to improve maintainability, scalability, and security.

The overall design is based on two major technologies: Supabase and Streamlit. Supabase is used for authentication, database management, Row Level Security, and administrative APIs, while Streamlit is used to build an internal administrative interface for bulk student onboarding and management. The system is designed primarily for hostel mess environments where administrators may need to provision hundreds or thousands of student accounts in a secure and repeatable manner.

The proposed architecture is divided into three major layers:

1.Authentication Layer
2.Application Profile Layer
3.Student Domain Layer

The authentication layer is responsible for maintaining student login credentials such as email addresses, password hashes, and account status. This layer is managed using the authentication services provided by Supabase. Every student receives a unique authentication identity that serves as the root record for all other system operations. Since authentication records are stored separately from application data, the system avoids direct exposure of sensitive login-related information in public database tables.

The second layer is the application profile layer, which stores role-based information for each user. This layer contains fields such as student name, role, contact information, and other non-sensitive metadata. Each profile record is linked to the authentication layer using a unique user identifier. This separation ensures that the system can distinguish between different categories of users such as students, administrators, wardens, and mess staff. The use of role-based access control allows the system to restrict features according to the user's role. For example, students can only access their own attendance and billing information, while administrators can manage all student records and reports.

The third layer is the student domain layer, which contains hostel-specific and mess-related information such as roll number, hostel name, room number, mess attendance status, and activation state. Each student record is connected to the profile and authentication layers through a unique foreign key relationship. This design ensures that there is exactly one student record corresponding to each authenticated identity. The database also enforces uniqueness constraints on fields such as roll number and user ID to prevent duplicate records and conflicting entries.

The provisioning methodology follows a three-stage process. In the first stage, the administrator enters student details through the Streamlit-based admin interface. The entered information includes student name, email address, hostel details, roll number, and room number. In the second stage, the system uses the Auth Admin API to create an authentication account for the student. Once the authentication record is created successfully, the system generates a unique user identifier. In the third stage, the generated user identifier is used to insert corresponding records into the profile table and the student table. This sequential process ensures that all student-related records remain linked and consistent.

A major design objective of the methodology is idempotency. Since administrators may accidentally run the provisioning script multiple times, the system

is designed to avoid duplicate account creation. Before inserting a new record, the system checks whether the email already exists in the authentication layer or whether a corresponding profile or student record already exists in the database. If the record already exists, the system skips that user instead of creating a duplicate entry. Additionally, upsert operations are used for profile and student tables so that existing rows can be updated safely without creating redundant records.

To maintain security, the system uses service-role credentials only within trusted server-side administrative processes. These credentials are never exposed to students or embedded inside front-end code. Sensitive values such as API keys and database URLs are stored securely using environment variables or protected secret-management mechanisms. Row Level Security policies are enabled on public tables to ensure that authenticated students can access only their own records. Similarly, administrative users are granted higher privileges through role-based policies and JWT claims.

The proposed methodology therefore provides a secure, scalable, and production-ready onboarding model for hostel mess management systems. It not only supports reliable student account creation, but also creates a strong foundation for future extensions such as QR-based attendance, automated mess billing, meal selection, complaint handling, and feedback analysis.

The diagram below shows how exactly design architecture is

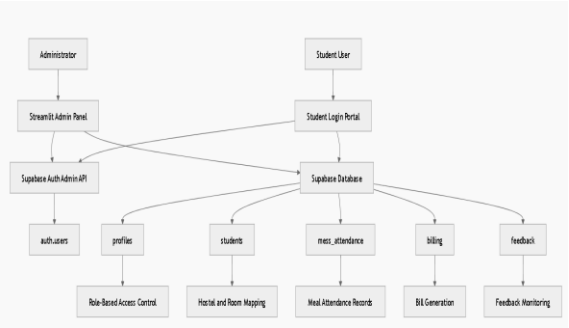

Fig 1 - Architecture and System Overview

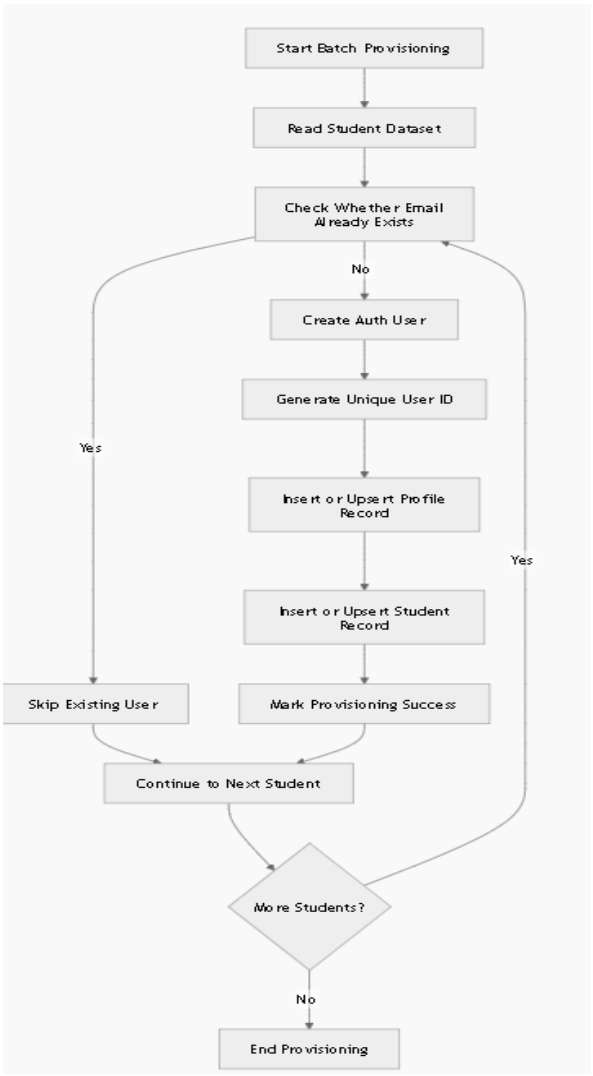

Fig 2- Shows real workflow of the platform

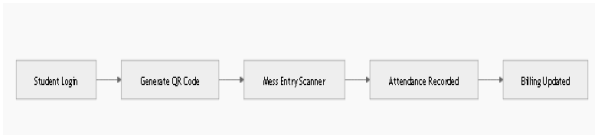

Fig 3- QR based attendance module workflow

IV. IMPLEMENTATION AND EXPERIMENTAL SETUP

The implementation of the proposed hostel mess management framework was carried out using a lightweight but secure technology stack centered on Supabase for authentication and database services and Streamlit for the administrative provisioning interface. The purpose of the implementation was to translate the proposed secure onboarding architecture into a

working prototype capable of creating student identities, assigning role-based profiles, and inserting hostel-specific records in a reliable and repeatable manner. The implemented prototype was designed primarily for the student onboarding component of a hostel mess management system, which later serves as the foundation for mess attendance, billing, feedback, and other operational modules.

A. Implementation Environment

The system was implemented as an admin-controlled provisioning workflow. The administrative front end was developed in Streamlit, which provides a simple and effective environment for building internal management tools in Python. The backend services were handled through Supabase, which offers authentication, PostgreSQL-based storage, and support for Row Level Security. The provisioning logic used the Supabase Python client to communicate with the authentication layer and the database tables. In this setup, the administrator submits student details through the Streamlit interface, and the backend processing logic invokes the Auth Admin API to create a new identity before inserting related rows into the application tables.

The implementation followed a three-table logical structure:

Authentication table (auth.users) for email-based identities and credential management.

Profile table (profiles) for role-based metadata such as user role and basic identity information.

Student table (students) for hostel-specific data such as roll number, hostel name, room number, and activation status.

This separation ensured that authentication data remained isolated from public application data, while still allowing each student to be mapped to exactly one profile and one domain-specific student record. The design also supported future integration with other mess management modules such as attendance tracking and bill generation.

B. Provisioning Logic

The implementation workflow was executed in a sequential and controlled manner. First, the administrator supplied a student dataset containing attributes such as student name, email address, roll number, hostel name, and room number. Second, for each student entry, the system attempted to create an authentication account using the elevated Supabase Auth Admin interface. Once the user identity was successfully created, the returned unique identifier was used to insert a corresponding row into the profiles table and then a row into the students table. This stepwise implementation preserved referential consistency across all layers of the system.

To support safe repeated execution, the implementation incorporated idempotent behavior. If the script was re-run with the same dataset, the system was expected to avoid duplicate identity creation and duplicate table entries. This was achieved by combining duplicate checks at the authentication layer with database-level integrity controls such as primary keys, unique constraints, and upsert operations. In particular, the profiles table used the user identifier as a primary key, while the students table used user_id as a primary key and enforced uniqueness on roll numbers. These controls allowed the system to skip or safely merge existing data rather than creating conflicting records.

C. Security-Oriented Implementation Measures

Because the provisioning workflow required elevated privileges, the implementation treated the service-role key as a highly sensitive administrative credential. The service-role key was used only in the trusted server-side environment of the admin tool and was not exposed in any student-facing component. Secrets such as the Supabase URL and service-role key were assumed to be stored securely through protected configuration or environment-based secret management. This was essential because the service-role credential bypasses ordinary Row Level Security restrictions and therefore must remain restricted to internal administrative use only.

At the database level, Row Level Security and role-based access control principles were considered part of the deployed design. The implementation assumed that student-facing queries would be restricted to the records of the currently authenticated student, whereas administrative users would operate through privileged workflows. Thus, even though the provisioning script itself used elevated access, the deployed system model preserved separation between administrator actions and student permissions.

D. Experimental Setup

The experimental setup was designed to evaluate the correctness and robustness of the onboarding workflow rather than to benchmark large-scale system throughput. A sample cohort of students was used to simulate the provisioning process, including a mix of already existing and newly added accounts. This small-batch setup was sufficient to validate the principal objectives of the system, namely correct account creation, consistency across linked tables, and safe behavior under repeated execution. The uploaded study explicitly notes that even a minimal cohort is adequate for validating idempotency and onboarding correctness in a prototype-stage implementation.

The experiment focused on the following evaluation scenarios:

1. Initial provisioning test:

The student dataset was executed once through the Streamlit-based provisioning workflow. The expected outcome was the successful creation of new authentication identities together with matching records in the profiles and students tables.

2. Re-run safety test:

The same dataset was executed again without modification. The objective of this test was to verify that the implementation did not create duplicate authentication identities or duplicate database rows.

3. Partial failure recovery test:

The workflow design was examined from the perspective of recovery after incomplete execution. In this case, even if account creation succeeded but one of the table insertions failed, the next execution of the provisioning script should reconcile the system state through deterministic keys and upsert operations instead of creating inconsistent duplicates.

4. Referential consistency test:

The setup also checked whether each created authentication identity corresponded to exactly one role-bearing profile row and exactly one student row. This test was important because downstream hostel mess functions depend on the correctness of these mappings.

E. Evaluation Parameters

The implementation and experimental setup were assessed using the following parameters:

- Provisioning correctness: whether valid student accounts were created successfully.
- Idempotency: whether repeated execution preserved existing records without duplication.
- Referential integrity: whether all created records remained correctly linked through user identifiers.
- Security compliance: whether elevated credentials were restricted to the admin environment.
- Recovery capability: whether the system could tolerate partial execution failures and recover through re-run logic.

These parameters are directly aligned with the technical contribution of the paper, since the proposed work is centered more on secure account provisioning and reliable onboarding than on broader hostel mess analytics.

F. Scope of the Prototype

The experimental implementation was intentionally scoped around the onboarding substrate rather than the full hostel mess lifecycle. That is, the implemented prototype primarily validated student provisioning, role assignment, and student-record generation. Modules such as QR-based meal authentication, monthly bill automation, menu management, and feedback processing were treated as future extensions that would build on the same identity and access-control foundation. This scoped implementation was methodologically appropriate because the paper's main contribution lies in establishing a secure and repeatable account provisioning framework for mess management systems.

V. PRODUCT PROTOTYPE AND IMPLEMENTATION

The proposed product prototype was developed as a secure hostel mess management platform with a primary focus on student onboarding and identity provisioning. The prototype was designed to support the early-stage operational requirements of hostel mess administration, including student registration, role assignment, hostel mapping, and preparation for future modules such as attendance tracking, billing, menu display, and feedback collection. Unlike many conventional hostel systems that focus only on front-end features, the prototype places strong emphasis on secure account creation, prevention of duplicate records, and maintenance of consistency across authentication and database layers.

The prototype was implemented using a combination of Supabase and Streamlit. The backend services were managed using Supabase because it provides integrated authentication, PostgreSQL-based data storage, Row Level Security, and administrative APIs. The user-facing administrative interface was developed using Streamlit due to its simplicity and rapid prototyping capabilities for Python-based internal tools. Together, these technologies allowed the creation of a lightweight but secure prototype that could be deployed quickly and scaled gradually for larger hostel environments.

The product prototype consists of two main user categories:
1.Student Users
2.Administrative Users
Student users represent hostel residents who will eventually use the platform to log in, view meal schedules, check attendance records, monitor billing details, and provide mess-related feedback. Administrative users represent wardens, hostel staff, or mess managers who are responsible for onboarding students, managing hostel records, generating reports, and monitoring mess operations. Since administrative users require elevated privileges, their actions are handled separately from normal student actions through dedicated admin-only workflows.

The implemented prototype contains the following core modules:

Student account creation module
1) Profile and role assignment module
2) Hostel and room allocation module
3) Student activation and status management module
4) Authentication and login module
5) Role-based access control module
The most important module in the prototype is the student account creation module. Through the Streamlit admin panel, the administrator enters the details of a student, including name, email address, roll number, hostel name, room number, and account status. After submission, the system creates a new authentication identity through the Auth Admin interface of Supabase. Once the identity is created successfully, the system stores additional role-based information in the profiles table and hostel-specific information in the students table. This process ensures that every student has a unique login identity and a corresponding hostel record.
The prototype database design follows a three-table structure consisting of auth.users, profiles, and students. The auth.users table is responsible for storing email addresses, password hashes, and authentication states. The profiles table stores role-related information such as whether the user is a student or administrator. The students table contains hostel-specific details including roll number, hostel name, room number, and active status. These tables are linked using unique identifiers to ensure referential integrity and prevent data inconsistency.
The implementation also incorporates idempotent provisioning behavior. In real hostel environments, administrators may accidentally run the same provisioning script multiple times or upload overlapping student datasets. To handle such cases, the prototype checks whether a student email already exists before creating a new authentication account. Similarly, the profiles and students tables use upsert operations and unique constraints so that existing rows can be updated safely without generating duplicate records. This feature is particularly important for maintaining reliable hostel databases across multiple semesters or admission cycles.
From a security perspective, the prototype was designed to ensure that elevated credentials are never exposed to normal users. Administrative operations use the service-role key only within the trusted backend environment of the Streamlit application. Sensitive configuration details such as API keys and database credentials are assumed to be stored securely through environment variables or protected secret files. Additionally, Row Level Security policies are enabled so that students can access only their own records, while administrative users can access and manage all student records.
Although the current prototype mainly focuses on onboarding and secure student provisioning, it is intentionally designed to support future hostel mess functionalities. The existing architecture can be extended with modules for QR-based attendance, automated monthly bill calculation, menu management, complaint handling, and feedback analysis. Because the authentication and profile layers are already established, these additional modules can be integrated without major redesign of the core system.

The implemented product prototype therefore demonstrates that secure onboarding can serve as the foundation for a larger hostel mess management platform. By combining authentication, role management, hostel mapping, and idempotent provisioning, the prototype establishes a scalable and production-ready framework for future hostel automation systems.

VI. RESULTS AND DISCUSSION

The proposed hostel mess management prototype successfully created student accounts using Supabase authentication and database services together with a Streamlit admin interface. For every student in the dataset, the system generated one authentication identity, one profile record, and one hostel-specific student record. This confirmed that the proposed three-layer architecture maintained correct mapping between authentication and application data.

The re-run safety test showed that the system was idempotent. When the same student dataset was processed again, no duplicate authentication accounts or duplicate database records were created. Existing records were safely skipped or updated using upsert operations and unique constraints. This makes the framework suitable for repeated use during multiple admission cycles or student data updates.

The prototype also handled partial failures effectively. If authentication was created successfully but insertion into profile or student tables failed, the next execution of the script completed the missing records without generating duplicate identities. This ensured data consistency across all linked tables.

From a security perspective, the service-role key remained restricted to the administrative environment, while Row Level Security ensured that students could access only their own records. The results indicate that the proposed framework is secure, reliable, and suitable for future extension into mess attendance, billing, QR-based meal authentication, and feedback modules.

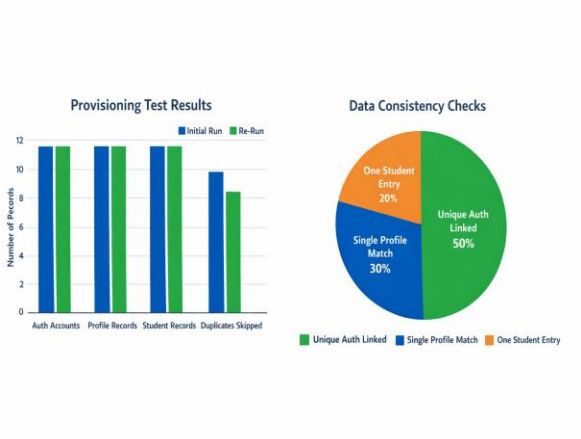

Image 1- Showing Test Results and Data Consistency

VII. CONCLUSION

This paper proposed a secure and idempotent student account provisioning framework for hostel mess management systems using Supabase and Streamlit. The framework separated authentication, profile, and hostel-specific student records to improve security, scalability, and data consistency. The results showed that the use of unique constraints, upsert operations, and referential links successfully prevented duplicate records and supported safe re-execution of the provisioning process. The system also maintained correct mapping between authentication accounts, profiles, and student records.In addition, security measures such as Row Level Security, role-based access control, and restricted use of service-role keys improved the reliability of the system. The proposed framework can therefore serve as a strong foundation for future hostel mess modules such as attendance, billing, QR-based authentication, and feedback management.

REFERENCES

[1] A. Kumar and S. Patel, “Mess Management System: A Web-Based Solution for Hostel Meal Automation,” International Journal of Scientific Research in Engineering and Management (IJSREM), vol. 8, no. 2, pp. 1–6, 2025.

[2] P. Mayank, “Development of a Hostel Mess Management and QR-Based Authentication System,” in Proceedings of the International Conference on Research and Developments in

Engineering, National Institute of Technology Rourkela, 2025.
[3] Supabase, “Understanding API Keys,” Supabase Documentation, 2025.
[4] Supabase, “Python: Create a User,” Supabase Documentation, 2025.
[5] R. Sharma and P. Verma, “Automated Hostel Management System,” International Journal of Scientific Research in Engineering and Technology (IJSRET), vol. 11, no. 1, pp. 45–52, 2025.
[6] OWASP Foundation, “Web Security Testing Guide: Testing for Default Credentials,” OWASP WSTG, 2025.
[7] Supabase, “Python: Auth Admin,” Supabase Documentation, 2025.
[8] Supabase, “Data REST API,” Supabase Documentation, 2025.
[9] Supabase, “User Management,” Supabase Documentation, 2025.
[10] Supabase, “Python: Upsert Data,” Supabase Documentation, 2025.
[11] Streamlit, “Secrets Management,” Streamlit Documentation, 2025.
[12] OWASP Foundation, “Authentication Cheat Sheet,” OWASP Cheat Sheet Series, 2025.
[13] National Institute of Standards and Technology, “Digital Identity Guidelines: Authentication and Lifecycle Management,” NIST SP 800-63B, 2024.
[14] Supabase, “Python: Send an Email Invite Link,” Supabase Documentation, 2025.
[15] Supabase, “Row Level Security,” Supabase Documentation, 2025.
[16] Supabase, “Custom Claims and Role-Based Access Control (RBAC),” Supabase Documentation, 2025.
[17] Supabase, “Database Functions,” Supabase Documentation, 2025.