Ashesi Insider

Web Technology

Fall 2025

Team Project

Seyram Awo Apreku

Nigel Nortey

Shaun Esua-Mensah

Eyram Divine Kwawukume

1 INTRODUCTION

1.1 Project Overview

Ashesi Insider is a platform designed to help students make informed decisions about courses, lecturers, restaurants, and hostels by offering a centralized space for reviews. It addresses information scarcity on campus by enabling students to view, submit, and compare ratings.

1.2 Problem Statement

Students often lack access to transparent and reliable information about:

- Which lecturers to choose
- Which courses align with their goals
- Which restaurants or hostels are best in quality, affordability, and safety

Ashesi Insider solves this by aggregating verified reviews, filtering tools, and admin moderation.

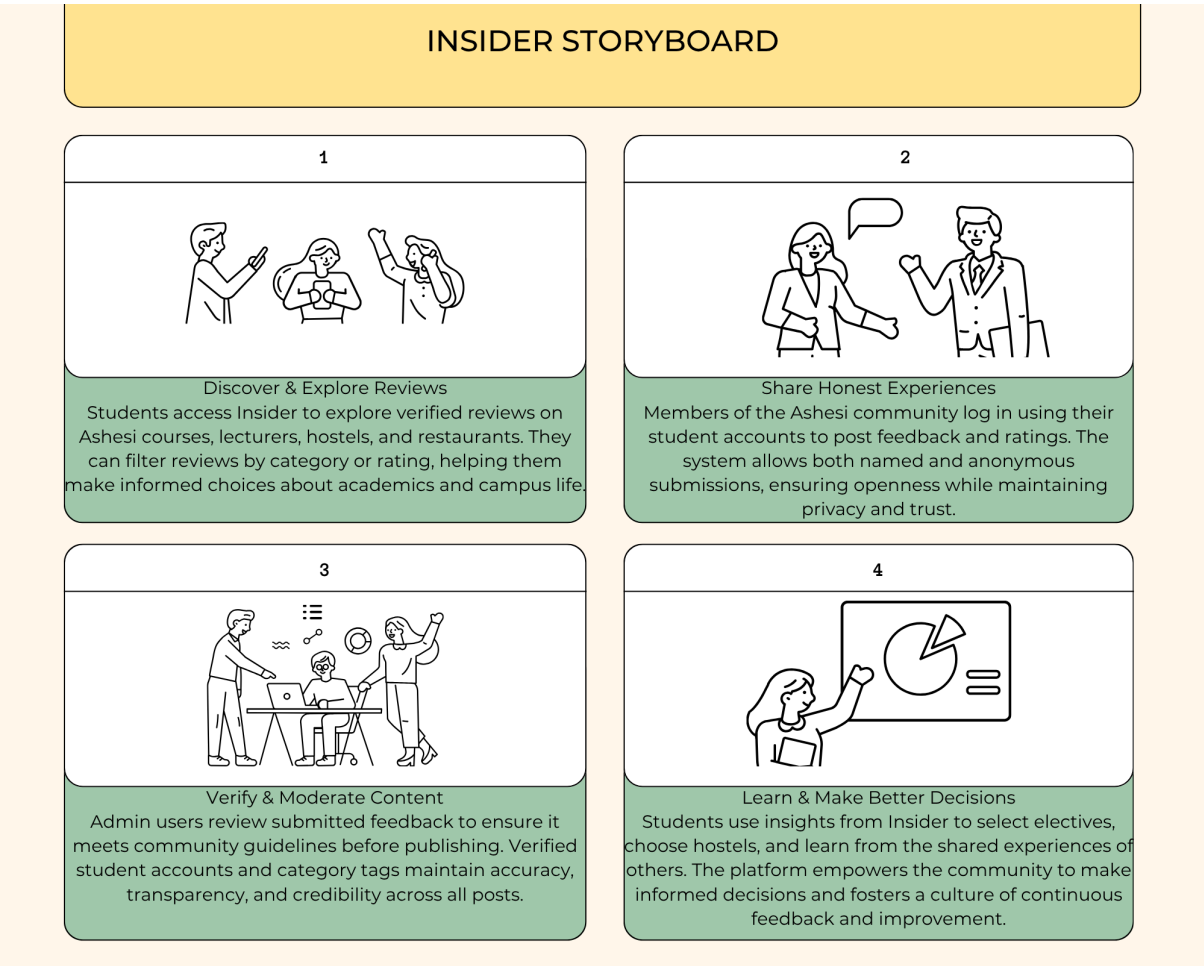
1.3 Target Users

- Students – submit and explore reviews
- Admins – moderate and manage the platform

1.4 Core Features

- Review submission for courses, lecturers, restaurants, and hostels
- Authenticated posting
- Browsing & filtering reviews
- Admin dashboard for content moderation

2 STORYBOARD



3 SYSTEM ARCHITECTURE

Ashesi Insider follows a three-tier architecture consisting of the frontend, backend, and database. This separation improves maintainability, performance, and security.

3.1 Architectural Layers

1. Frontend: React

The React interface handles all user-facing interactions, including:

- Browsing reviews for lecturers, courses, restaurants, and hostels
- Submitting new reviews and ratings
- Searching and filtering results
- Managing authentication state (login/logout)

React sends API requests to the backend and updates the UI based on responses.

2. Backend: Next.js API Routes (Replaced Express)

Our project originally began with a standalone Express.js backend. However, to simplify deployment and keep the project within a single unified codebase, the backend was migrated to Next.js API Routes.

These API routes handle all server-side logic, including:

- Validating incoming requests
- Authenticating users with JWT
- Protecting admin-only endpoints
- Performing CRUD operations for reviews and entities
- Returning structured JSON responses to the frontend

3. Database: PostgreSQL (Supabase)

Supabase hosts the PostgreSQL database and stores all platform data:

- Users and admins
- Courses, lecturers, restaurants, hostels
- Reviews for each category

The database enforces relationships, keeps data consistent, and supports efficient queries for filtering and search.

Supabase provides authentication, row-level security, and auto-generated APIs, ensuring consistency and efficient querying.

3.2 System Workflow

1. The user interacts with the React interface
2. Frontend sends a request to a Next.js API route
   (e.g., /api/reviews/add, /api/auth/signin)
3. The API handler validates input and checks permissions
4. The handler queries Supabase/PostgreSQL
5. Supabase returns data

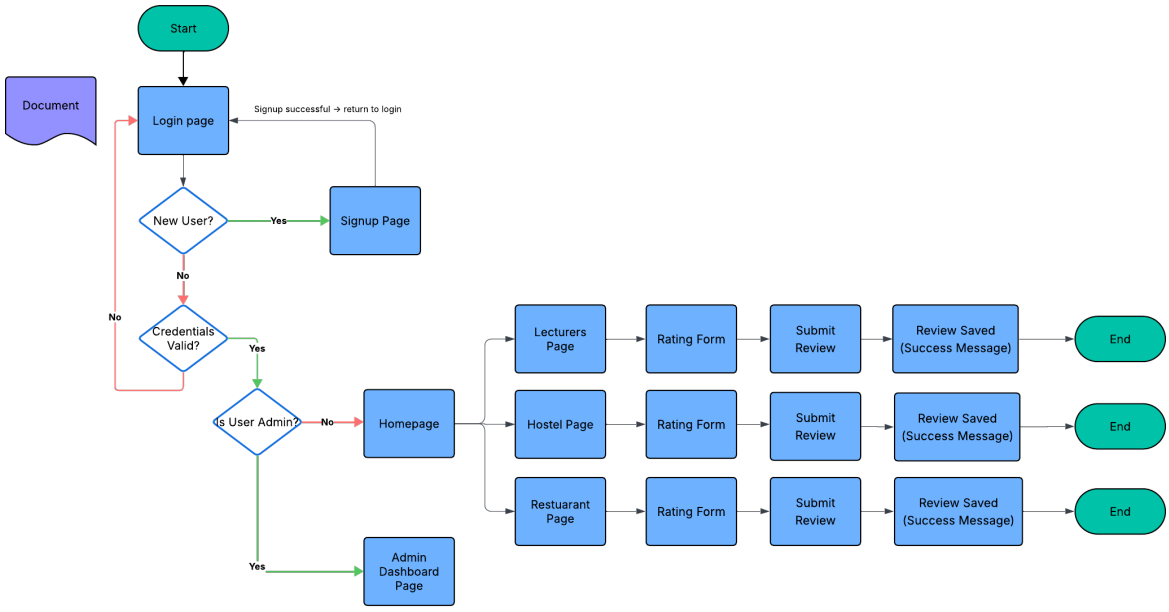
6. The API route responds with JSON
7. The UI updates accordingly

This cycle supports all core features: viewing reviews, submitting feedback, authentication, and admin actions.

3.3 Admin Workflow

- Admin logs in through restricted authentication
- Admin Dashboard loads protected data via secure API routes
- Admin can delete reviews, moderate users, or manage entities
- Changes are written to the database through API calls
- UI updates in real time using client-side rendering

4 SYSTEM FLOWCHART



5 DATABASE DESIGN

5.1 Entities

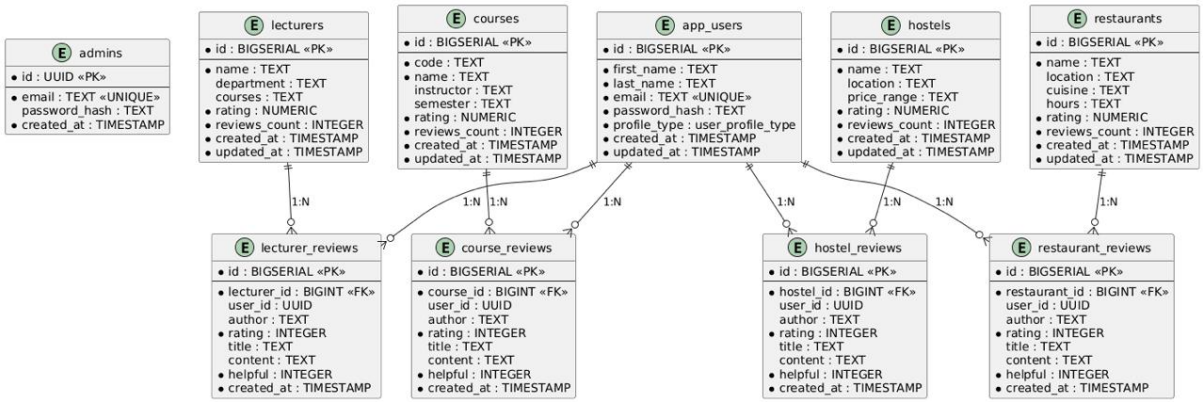
- Users
- Lecturers

- Courses
- Restaurants
- Hostels
- Reviews

5.2 Keys & Relationships

- Each review links to a user and a category-specific entity
- Users may post many reviews
- Admin roles defined within the users table

5.3 Entity–Relationship Diagram (ERD)



6 PROJECT MANAGEMENT SUMMARY

This section covers all sprint requirements **without copying sprint documents**.

6.1 Sprint 1 Summary

- Initial UI pages created
- Navigation and layout established
- Database schema drafted
- Basic React components implemented

6.2 Sprint 2 Summary

- ERD created
- Initial backend logic drafted using Express.js

- Team identified deployment challenges with a separate backend
- Migrated backend to **Next.js API Routes** with AI-assisted learning
- Successfully connected the frontend to Supabase
- Implemented initial review submission through unified API structure

6.3 Sprint 3 Summary

- Final architecture documented
- Flowchart created
- Admin dashboard (completed or nearly done)
- Remaining polish before final deployment

7 INDIVIDUAL CONTRIBUTIONS

7.1 Nigel Nortey – Authentication System & Hostels Feature

Key Contribution and Role

Nigel led the implementation of the authentication system and the development of the Hostels feature.

Features Worked On and Completed

- Implemented secure signup and login flows using Supabase authentication and custom JWT handling via Next.js API routes.
- Developed the full Hostels page, including search, rating filters, sorting, and collapsible review sections.
- Integrated rate limiting into the login process to prevent brute-force attacks.

Most Important Algorithm or Design

Nigel's most significant design is the **login rate-limiting algorithm**, built around an in-memory RateLimiter class. It tracks login attempts per IP, limits failed attempts to three, and enforces a five-minute cooldown, which is then integrated directly into the /api/auth/login route.

Challenges

- Handling token logic during the migration from Express to Next.js API routes.
- Extracting IP addresses consistently across environments for rate limiting.
- Managing complex UI states for dynamic hostel filtering and review expansion.

Lessons Learned

- Developed a deeper understanding of secure authentication patterns and session handling.
- Gained experience balancing performance with interactivity in filter-heavy pages.
- Learned how backend framework changes affect request handling and validation logic.

7.2 Shaun Esua-Mensah – Review Submission, Deletion & User Profile Page

Shaun created the system that allows users to add and delete reviews, ensuring that students

Key Contribution and Role

Shaun was responsible for core review interactions and for building the unified User Profile page.

Features Worked On and Completed

- Implemented review submission and deletion across all categories.
- Built the Profile page, which aggregates a user's reviews from all tables and organizes them with category filters.
- Added UI elements such as badges, review grouping, and confirmation dialogs for deletion.

Most Important Algorithm or Design

Shaun's key design is the **review aggregation system** in profile-page.tsx. It queries multiple review tables, maps them into a common structure, merges them into a single "activity feed," and sorts them by date to give users a unified view of their contributions.

Challenges

- Reconciling different review schemas across courses, lecturers, restaurants, and hostels.
- Ensuring that review deletion remained secure and only affected the correct user's data.

- Maintaining UI responsiveness while dealing with multiple review categories.

Lessons Learned

- Improved understanding of relational data modeling in multi-entity systems.
- Learned to design reusable, extensible logic for managing heterogeneous data sources.
- Strengthened skills in advanced React state management for dynamic, multi-tab interfaces.

7.3 Eyram Divine Kwawukume – Admin Dashboard, Admin API & Profile API

Key Contribution and Role

Divine developed the Admin Dashboard and the backend APIs that support moderation, statistics, and user review retrieval.

Features Worked On and Completed

- Built the multi-section Admin Dashboard interface for managing users and all review types.
- Implemented admin-only API routes for deleting content and fetching platform-wide statistics.
- Co-developed the Profile API that retrieves a user's reviews efficiently from all category tables.

Most Important Algorithm or Design

Divine's primary algorithmic contribution is the **parallel statistics-fetching system** in /api/admin/stats. Using Promise.all(), the route queries multiple tables at once, aggregates counts, and returns platform metrics with minimal delay.

Challenges

- Implementing secure role-based access control during the switch from Express to Next.js APIs.
- Ensuring that dashboard queries remained performant as multiple tables grew.
- Managing large datasets in a way that kept the Dashboard responsive.

Lessons Learned

- Gained strong experience in access control and backend security.
- Learned effective query optimization techniques with Supabase/PostgreSQL.
- Developed proficiency in building admin-facing interfaces that balance clarity with performance.

7.4 Seyram Apreku – Landing Page & Frontend Authentication (Signup/Signin)

Key Contribution and Role

Seyram designed the Landing Page and implemented the frontend authentication experience while collaborating closely with backend developers during the transition to Next.js API routes.

Features Worked On and Completed

- Developed the Landing Page design and layout, establishing the platform's visual identity.
- Implemented the signup and signin pages with validation and responsive UI.
- Integrated the forms with backend API routes and managed authentication state across protected pages.

Most Important Algorithm or Design

Seyram's most significant design is the **frontend authentication flow**, which coordinates secure form submission, handling JWT-bearing responses from API routes, storing session information, and guarding private pages based on user state.

Challenges

- Adjusting the frontend to accommodate changes in backend responses after migrating from Express to Next.js.
- Managing global authentication state across multiple components and routes.
- Ensuring a responsive, consistent layout while integrating dynamic logic.

Lessons Learned

- Learned how to build secure, user-friendly authentication flows on the frontend.

- Gained practical understanding of JWT handling, protected routing, and session persistence in Next.js.
- Improved skills in designing clean and intuitive UI experiences for user onboarding.

8 REQUIRED LINKS

- **GitHub Repository:** *https://github.com/divin081/AshesiInsider*
- **CMS Website:** *https://screeching-wheat.localsite.io*
- **Our Website:** *https://ashesi-insider-zeta.vercel.app/*

9 APPENDIX

Appendix A- Database schema