Université de Haute-Alsace

Projet Fin D'Annee

EDX : A Comprehensive Educational Management Platform

Hadjarsi Mohamed El Fateh
Boulaabi Ahmed

Supervisé par
Professeur Joel Heinis

Date de soumission : 12-06-2024

Abstract

This report presents the conception and development of EDX (EDucation eXpert), an educational management platform designed to improve administrative efficiency and the experience of pupils, teachers and academic administrators. EDX integrates a variety of functionalities including student and teacher management, course scheduling, seating arrangements, and real-time presence tracking. By utilising modern web technologies and a robust database architecture, EDX's goal is to streamline the processes of educational administration, providing intuitive interfaces for users. The platform's development followed a mix of agile and RAD (Rapid Application Development) methodologies, ensuring iterative improvement and responsiveness to feedback. Key features include creation of classroom seating plans, users, enrollment request and schedule management, student seating and presence, and Github repository management. The system has been constructed with technologies that ensure reliability, scalability, and ease of use. This report details the design, implementation, challenges faced, and potential future enhancements of the EDX platform.

Table des matières

1 Background and Objectives

A centralized, efficient, and user-friendly educational management platform has always been needed. Our project, EDX (EDucation eXpert), was initiated under the guidance of Professor Joel Heinis, who tasked and guided us throughout developing a comprehensive system to address the needs of educational institutions.

The global objective was to create a web platform that provides access for administrators, teachers and students to a crucial set of tools, thereby enhancing operational efficiency and user experience.

EDX aims to integrate features that are traditionally spread across various third-party tools. By grouping these functionalities into a single platform, EDX has the potential to streamline administrative processes into one locally hosted solution and reduce dependency on external software, and serve as a singular repository of educational tools.

EDX is simple, cohesive and efficient.

Figure 1 – EDX Logo

2 Overview of the Report Structure

This report is structured to provide a comprehensive overview of the development and implementation of the EDX platform. It includes the following sections :

- **System Overview :** A high-level description of EDX, highlighting its key features and functionalities.
- **Methodology :** An outline of the development process and the methodologies used, including the tools and technologies employed.
- **System Design :** Detailed descriptions of the system architecture, user interface design, and module interactions.
- **Implementation :** Insights into the implementation of key features, including code snippets.
- **Testing and Evaluation :** An explanation of the testing strategies, test cases, and results, as well as an evaluation of system performance.
- **Challenges and Solutions :** A discussion of the problems encountered during development and how they were addressed.
- **Results and Discussion :** A summary of the results achieved, compared to the initial objectives, and a discussion on the effectiveness and impact of EDX.
- **Future Work :** Potential improvements and additional features for the platform.
- **Conclusion :** A recap of the project objectives and achievements, and final thoughts on the significance of the project.

By the end of this report, we aim to demonstrate how EDX not only meets some of the current needs of educational administrators and teachers but also sets the foundation for future advancements.

3 Key Features

3.1 Student Management

The Student Management feature of EDX allows administrators and teachers to efficiently manage student information. This includes :

- **Student Profiles :** Creating and updating detailed student profiles with the ability for students to add personal, academic, and contact information.
- **Enrollment :** Handling student enrollment processes and maintaining records of enrolled students.
- **Attendance :** Tracking and managing student attendance records in real-time.

3.2 Teacher Management

The Teacher Management feature provides tools for managing teacher information and schedules. Key functionalities include :

- **Teacher Profiles :** Creating and maintaining profiles with personal, contact, and professional information.
- **Scheduling :** Managing teacher schedules and assigning classes.

3.3 Seating Arrangements

The Seating Arrangements feature helps in organizing classroom seating plans. It offers :

- **Dynamic Seating Plans :** Creating and adjusting seating plans based on classroom layouts.
- **Student Allocation :** Assigning seats to students manually or automatically, this is most useful for exams.
- **Interactive Maps :** Providing visual maps of seating arrangements for easy reference. As well as, the ability to print the seating as a PDF
- **Database Integration :** Storing seating plans in the database for future reference and adjustments.
- **Repository Management :** Managing the creation and access for repositories for seances and students.

3.4 Course Scheduling

Course Scheduling in EDX streamlines the process of planning and organizing course timetables. This feature includes :

- **Timetable Creation :** Ability to create timetables and schedules for the teachers.

— **Classroom Allocation :** Assigning courses to classrooms.

3.5 Real-Time Presence Tracking

Real-Time Presence Tracking ensures accurate monitoring of student presence. Features include :

— **Live Attendance :** Recording attendance in real-time during classes.

4 System Architecture

4.1 Overview of the System Architecture

We chose a to implement a modular architecture to ensure scalability, maintainability, and ease of integration.

4.2 The Components

The main components of the EDX system and their interactions are :

4.3 Next.js

Next.js is the foundation of EDX. Although It is considered to be a Backend framework, it provides control over both the backend and the frontend, it builds upon React.js a Javascript framework for building dynamic and reusuable UI components, which unlike PHP and J2EE that render HTML on the server and send it to the client, React.js handles all of that on the client-side, making the UI more interactive, responsive and dynamic. Next.js further enhances the DX (Developer Experience) and UX (User experience) of React.js by providing a backend layer allowing the full application to be built ontop of a single codebase. As well as, a plethora of optimizations that closely resemble how PHP and servers handle serving users, such as ISR (Incremental Static Regeneration) which renders the static parts of the website on the server before serving it to the user (Server-Side Rendering), and allows the dynamic parts of the website to run on the client for real-time reactivity and responsiveness (Client-Side Rendering).

Next.js marries the best of both worlds from traditional server-side rendering and modern client-side frameworks, providing the performance and SEO benefits of server-side rendering with the dynamic user experience and component-based architecture of client-side

4.4 TypeScript

TypeScript is a strongly typed superset of JavaScript that enhances the development process by adding static types. This helps catch errors at compile

time, rather than at runtime. Typescript ensures robust and maintainable code, allowing developers to build type-safe applications with ease. By integrating TypeScript into EDX, we ensure better code quality, improved developer productivity, and future-proofing.

FIGURE 2 – Next.js and TypeScript Logos

4.4.1 Client-Side (Frontend)

The client-side of EDX is built using modern web technologies to provide a responsive and intuitive user interface. It includes :

- **Tailwind CSS :** A utility-first CSS framework for styling the application.
- **Framer Motion :** Used for creating animations and transitions within the user interface.
- **D3.js :** A JavaScript library for producing dynamic, interactive data visualizations in web browsers.
- **Zod :** A TypeScript-first schema declaration and validation library for validating user inputs and ensuring data integrity.
- **Shadcn UI :** A component library that provides a set of accessible, customizable, and composable UI components.
- **Radix UI :** A set of low-level, accessible components for building high-quality, accessible web applications.

4.4.2 Server-Side (Backend)

The backend of EDX handles the business logic and data processing. It is composed of :

- **Supabase :** A backend-as-a-service providing authentication, database, and storage solutions through microservices.
- **PostgreSQL :** A robust relational database management system for storing and managing data.
- **Drizzle ORM :** An Object-Relational Mapping (ORM) tool that simplifies database interactions in a type-safe manner.

— **Jotai :** A state management library for React applications, used for managing global state in a simple and scalable way.
— **ESLint :** A static code analysis tool for identifying and fixing problems in JavaScript and TypeScript code.
— **Vercel :** A cloud platform for static sites and serverless functions that simplifies deployment and provides fast, reliable hosting.

4.4.3 Development and Build Tools

A set of development and build tools are used to streamline the development process and ensure code quality :

— **pnpm :** A fast, disk space-efficient package manager for JavaScript and TypeScript, used to manage project dependencies.
— **Node.js :** A JavaScript runtime used for building scalable server-side applications and running development scripts.
— **Bun :** An all-in-one JavaScript runtime and package manager that provides fast execution, comprehensive tooling, and built-in support for many modern JavaScript features.

4.4.4 CI/CD and Deployment

Continuous Integration and Continuous Deployment (CI/CD) processes are essential for maintaining a smooth development workflow. Key tools and scripts include :

— **Drizzle-Kit :** For database schema management and migration tasks.
— **Supabase CLI :** For managing Supabase projects and deployments.
— **Custom Scripts :** Scripts such as `changelog`, `lint`, `push`, `pull`, `generate`, `drop`, `check`, and `up` to automate various development tasks.

FIGURE 3 – Tailwind, Supabase, Drizzle-ORM, Framer-Motion, Jotai, Pnpm

5 Project Conception and Planning

The methodology behind the development of EDX began with the conception and planning phase, which included :

5.0.1 Overview of the Initial Project Idea

The initial idea for EDX was to create a comprehensive educational management platform that would centralize various functionalities needed by administrators, teachers, and students. The goal was to streamline administrative tasks, and improve the overall educational experience.

5.0.2 Discussions and Meetings with Professor Joel Heinis

Throughout the planning phase, regular discussions and meetings were held with Professor Joel Heinis. These meetings were crucial for refining the project idea and implementation, setting realistic goals, and ensuring that the platform would meet its milestones. Professor Heinis provided valuable insights and guidance that shaped the development of EDX.

Professor Heinis guided us through the first stages of development by writing an extensive document detailing all the necessary features and workflows for users. Additionally, he provided the base schema for various database tables, which served as a foundation for our data models and interactions.

5.0.3 Objectives and Goals Set During the Planning Phase

During the planning phase, several key objectives and goals were established :

- Develop a user-friendly interface for administrators, teachers, and students.
- Integrate core functionalities such as student and teacher management, seating arrangements, real-time presence tracking, and exam management with dynamic github repository creation for students.
- Ensure the platform is scalable, secure, and capable of handling a large number of users and data.
- Utilize modern web technologies and best practices to enhance the performance and responsiveness of the platform.

5.1 Development Process

5.1.1 Description of the Development Methodology Used

The development of EDX followed a combination of Agile and Rapid Application Development (RAD) methodologies. This approach allowed for iterative development, continuous feedback, and rapid prototyping.

5.1.2 Iterative Development Cycles and Sprints

The development process was organized into iterative cycles, each lasting a week. Each cycle, or sprint, focused on specific features or improvements. At the end of each sprint, we reviewed the progress, gathered feedback from professor Heinis, and planned the next steps. This iterative process ensured that the project stayed on track and could adapt to any changes or new requirements.

5.1.3 Tools and Technologies Employed

A variety of tools and technologies were employed throughout the development process to streamline workflows and ensure code quality :

- **Version Control :** Git and GitHub for version control and collaboration.
- **Project Management :** Github issues for tracking tasks and progress.
- **Development Tools :** Visual Studio Code as the primary code editor, with extensions for TypeScript, ESLint, and more.
- **Communication :** Discord for team communication and coordination.
- **CI/CD :** GitHub and Vercel for continuous integration and continuous deployment, ensuring that code changes were automatically tested and deployed.

5.1.4 Risk Management and Mitigation

During the development process, we identified several risks that could potentially impact the project. These included technical challenges, timeline constraints, and potential changes in requirements. To mitigate these risks, we adopted a proactive approach by scheduling regular meetings, maintaining clear communication, and using project management tools to track progress and address issues promptly.

5.2 Design Phase

5.2.1 Initial Design Meetings and Brainstorming Sessions

The design phase began with taking the requirements and base schema's and brainstorming challenges and workflows to best write a robust database schema from the start. During these sessions, we discussed the overall vision for the EDX platform, identifying key features, user interactions and technologies. Many ideas were proposed and evaluated to determine the best approach for creating an intuitive and efficient system.

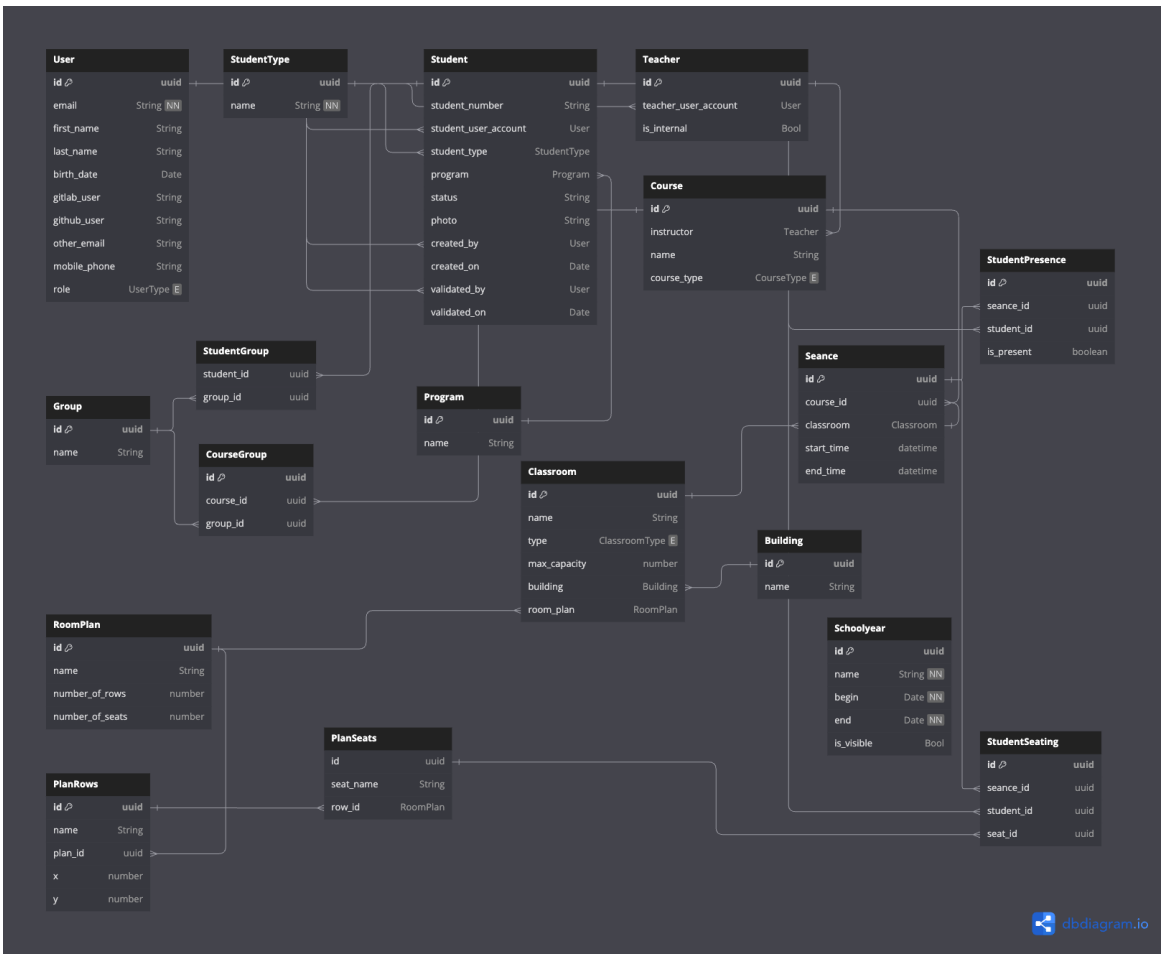

FIGURE 4 – Database schema

5.2.2 Design and Frontend

Following the brainstorming sessions, we proceeded with creating a UI design with figma for the main pages, as well as, the graphic design for EDX's branding. To save time and work faster, we did not make a design for each page, but we made a design system that would allow each page to have a cohesive look with the rest of the web app.

We opted to not do any wireframing, and decided to build the UI directly which saves us time and sets the constraints and guidelines for the backend.

Reusuable and dynamic components were made to further enhance the efficiency of our development, we also utilized shadcn's UI library to quickly prototype clean and responsive page layouts.

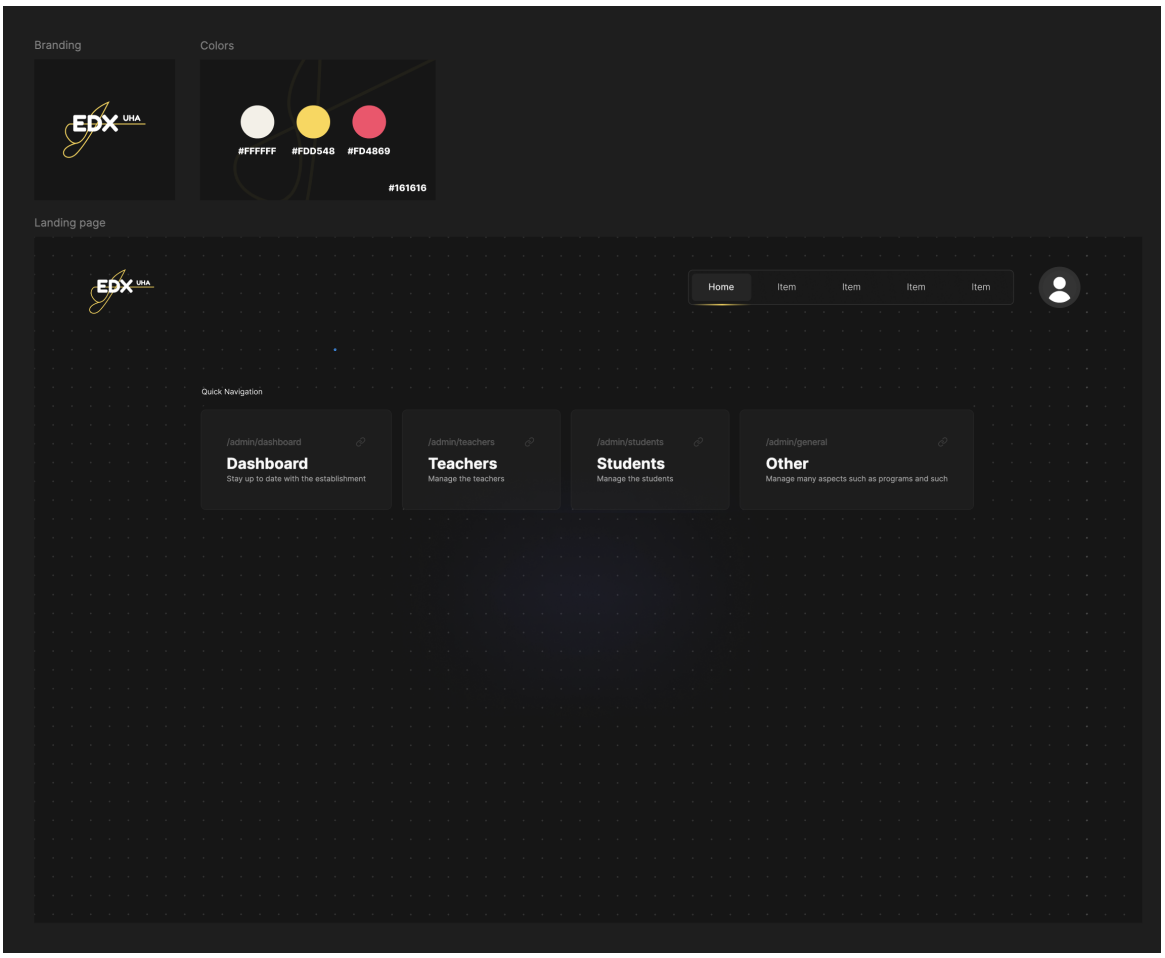

FIGURE 5 – Initial Design for the landing page and branding

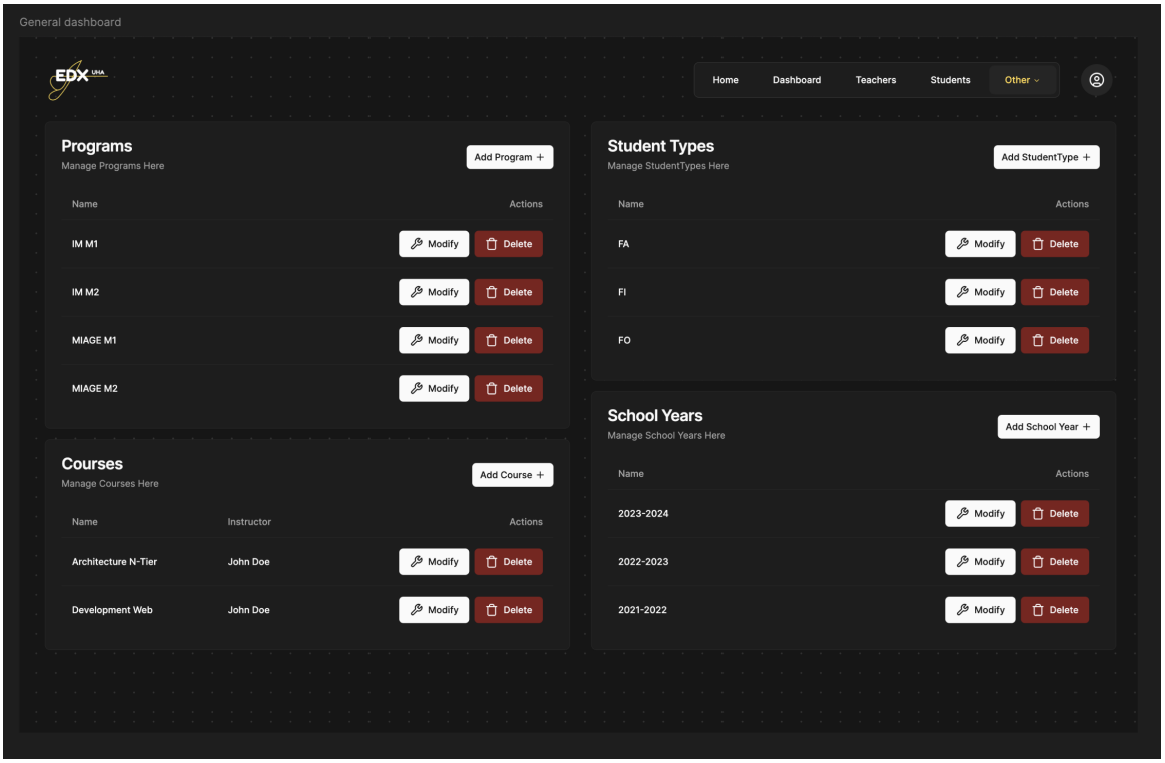

FIGURE 6 – Design for the admin dashboard page

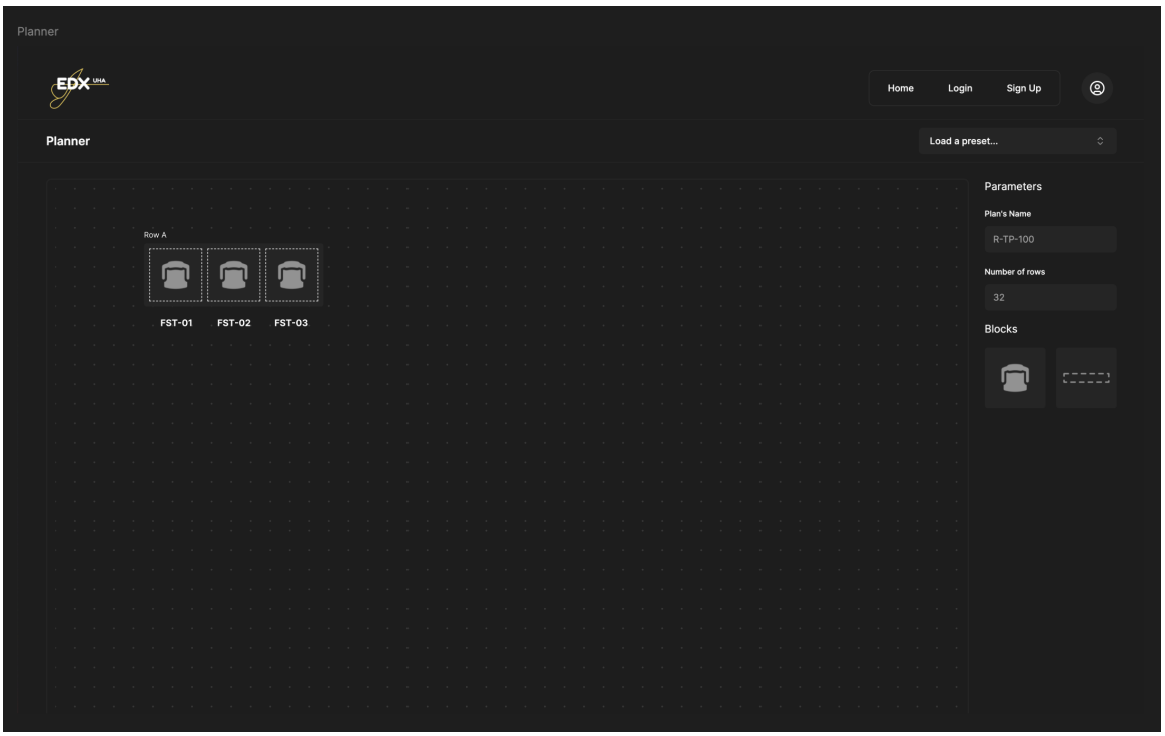

FIGURE 7 – Design for the Planner page

5.2.3 Feedback from Professor Joel Heinis and Adjustments Made

Throughout the project, Professor Joel Heinis provided ongoing feedback and guidance. He reviewed our progress, offering constructive criticism and suggestions for improvement. This iterative process of design and feedback ensured that the final product was both functional and user-friendly.

6 System Design

6.1 Detailed System Architecture

The system architecture of EDX is designed to ensure scalability, reliability, and ease of maintenance. The architecture includes multiple layers and components, each responsible for specific aspects of the system.

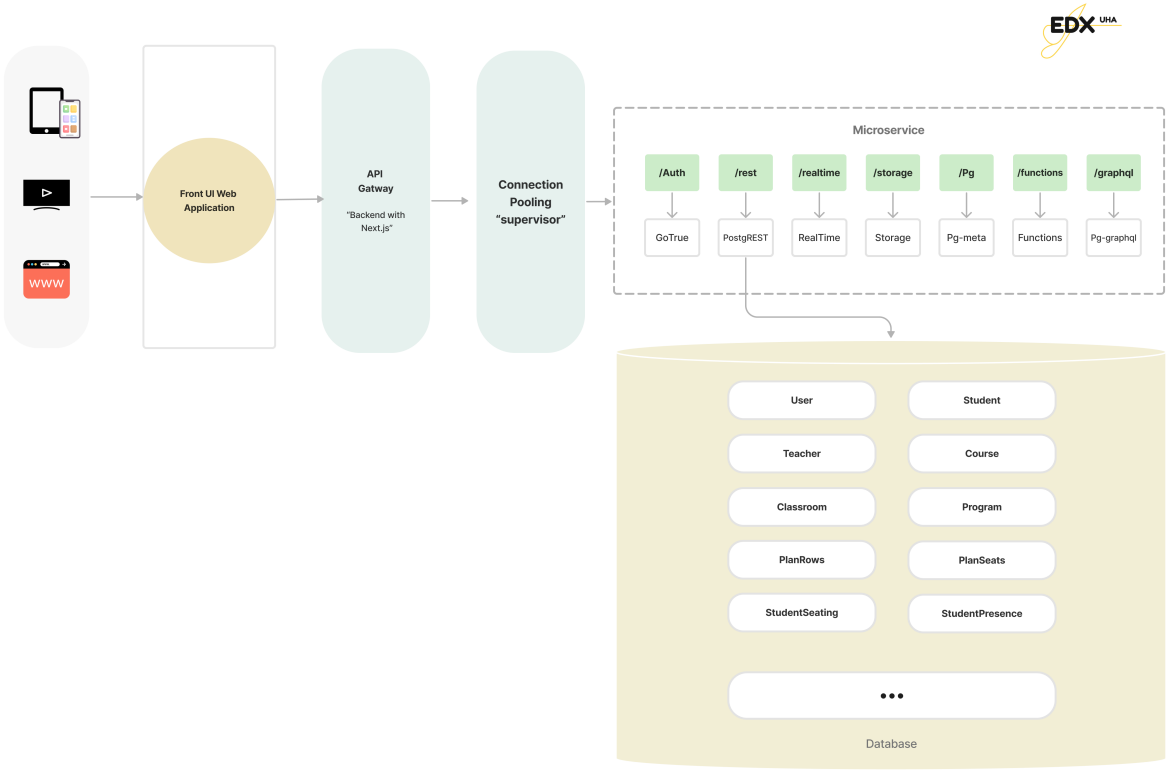

FIGURE 8 – Detailed System Architecture of EDX

6.1.1 Database Schema Design

The database schema, defined using Drizzle ORM, ensures data integrity and efficient querying with type safety. It includes tables for users, courses, attendance records, and seating arrangements, each with appropriate relationships and constraints to maintain consistency.

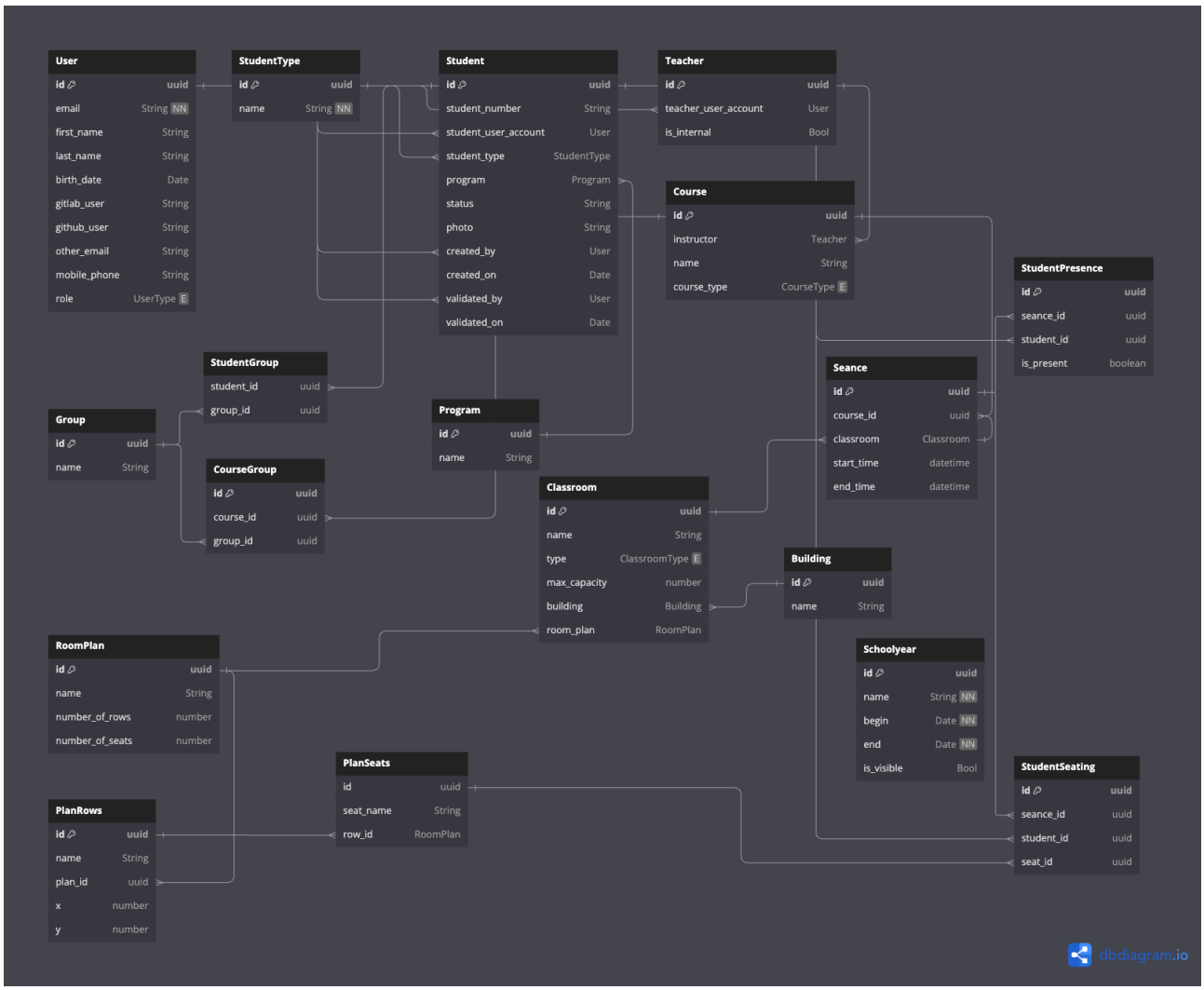

FIGURE 9 – Database Schema

```typescript
export const users = pgTable("users", {
  id: uuid("id").defaultRandom().primaryKey().notNull(),
  email: text("email").notNull().unique(),
  firstName: text("first_name"),
  lastName: text("last_name"),
  birthDate: date("birth_date"),
  gitlabUser: text("gitlab_user").unique(),
  githubUser: text("github_user").unique(),
  otherEmail: text("other_email"),
  mobilePhone: text("mobile_phone"),
  role: UserType("role"),
});

export const teachers = pgTable("teachers", {
  id: uuid("id").defaultRandom().primaryKey().notNull(),
  teacherUserAccount: uuid("teacher_user_account").references(() => users.id,
{   onDelete: "cascade",
  }),
  isInternal: boolean("is_internal"),
});
```

```typescript
export const studentTypes = pgTable("student_types", {
  id: uuid("id").defaultRandom().primaryKey().notNull(),
  name: text("name").notNull().unique(),
});

export const students = pgTable("students", {
  id: uuid("id").defaultRandom().primaryKey().notNull(),
  studentNumber: text("student_number").unique(),
  studentUserAccount: uuid("student_user_account").references(() => users.id,
{   onDelete: "cascade",
  }),
  studentType: uuid("student_type").references(() => studentTypes.id, {
    onDelete: "cascade",
  }),
  status: text("status"),
  photo: text("photo"),
  createdBy: uuid("created_by").references(() => users.id, {
    onDelete: "cascade",
  }),
  createdOn: date("created_on"),
  validatedBy: uuid("validated_by").references(() => users.id, {
    onDelete: "cascade",
  }),
  validatedOn: date("validated_on"),
});
```

FIGURE 10 – Example Schema definitions written in typescript

6.2 User Interface

The user interface focuses on providing an intuitive and responsive experience for all users.

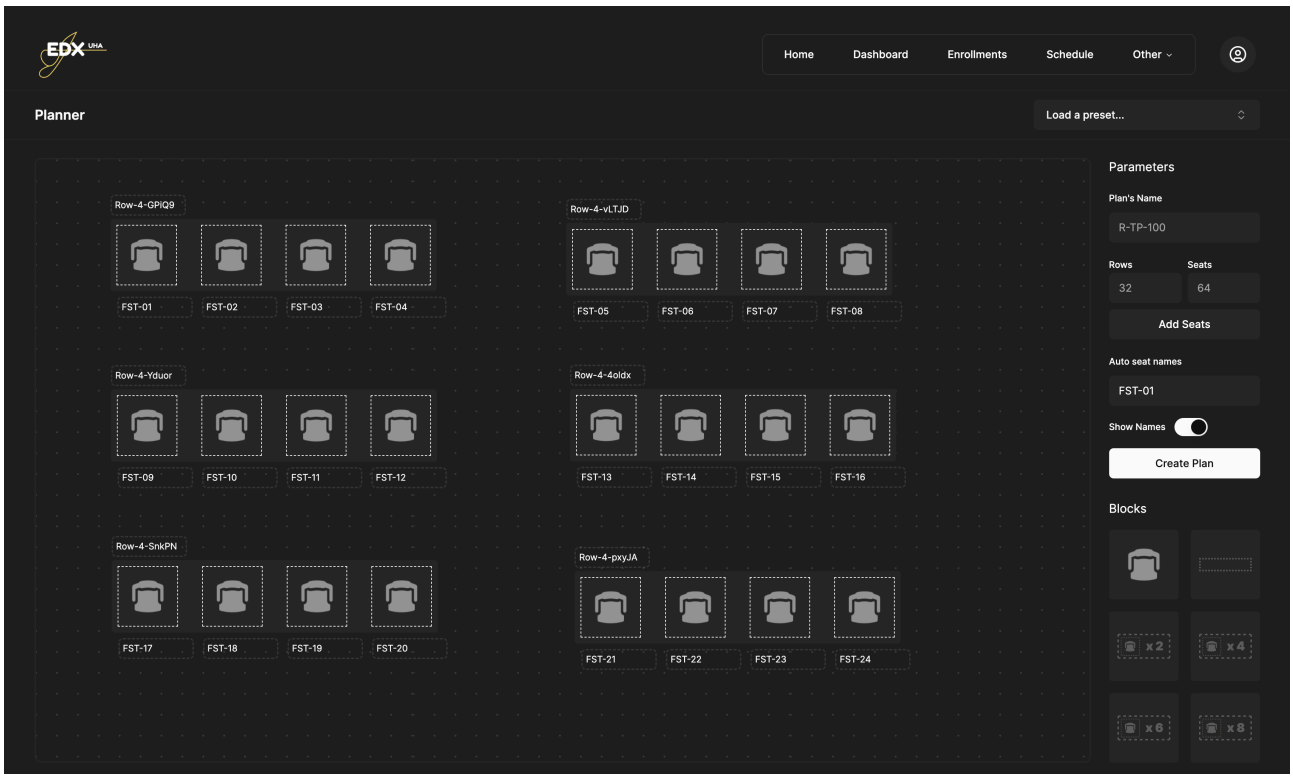

FIGURE 11 – Example finished page of the EDX User Interface

Key principles of the UI include :

- **Consistency :** Ensuring a consistent look and feel across all pages and components.
- **Intuitivness :** Focusing on easiness to use and intuitiveness across all pages.
- **Quality of life :** The importance of making the user experience as best as possible.
- **Responsiveness :** Making sure the interface adapts seamlessly to different screen sizes and devices.

6.2.1 Next.js App-dir

The ‘app‘ directory is a new introduction to Next.js allowing for managing routes, components, and API handlers. The following describes the key elements of the ‘app‘ directory structure as shown in the provided file structure :

Top-Level Directories The ‘src/app‘ directory is structured to include various top-level directories corresponding to different functionalities of the EDX platform :

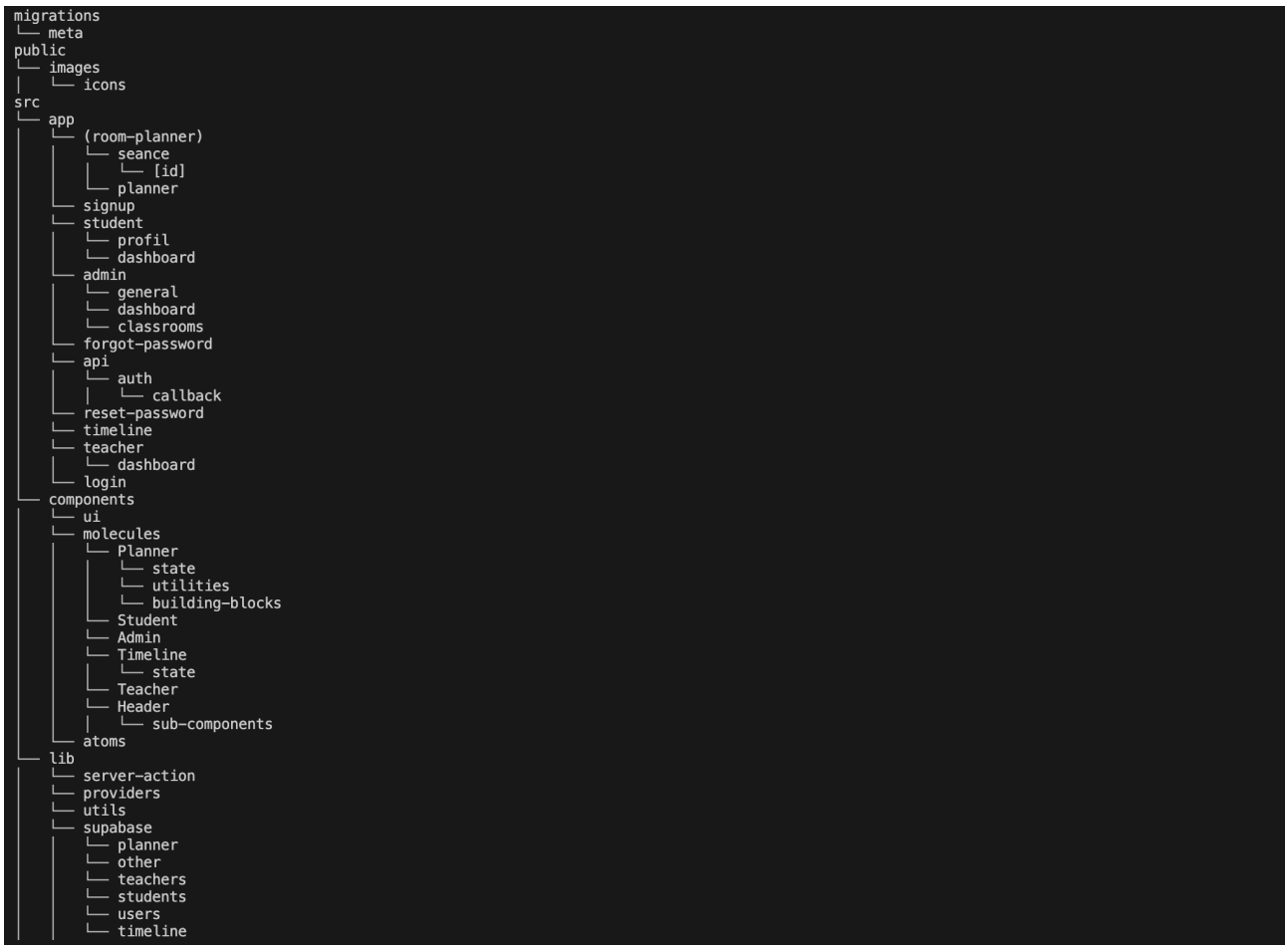

FIGURE 12 – Project file structure

— **room-planner :** Contains subdirectories for handling room planning functionality, including ‘seance‘ with dynamic routes like ‘[id]‘, and a ‘Planner‘ component.
— **signup :** Manages user signup functionalities.
— **student :** Includes ‘profil‘ and ‘dashboard‘ subdirectories to handle student profiles and dashboards.
— **admin :** Contains ‘general‘, ‘dashboard‘, and ‘classrooms‘ subdirectories for admin functionalities.
— **forgot-password :** Handles password recovery processes.
— **api :** Contains API route handlers, including an ‘auth‘ directory with a ‘callback‘ handler.
— **reset-password :** Manages password reset functionalities.
— **timeline :** Contains timeline management functionalities.
— **teacher :** Includes the teacher ‘dashboard‘ and ‘login‘ functionalities.

Components Directory The ‘components‘ directory under ‘src‘ is organized to enhance reusability and maintainability of UI components :
— **ui :** Contains common UI elements.
— **molecules :** Includes composite components like ‘Planner‘, ‘Student‘,

‘Admin‘, ‘Timeline‘, ‘Teacher‘, and ‘Header‘, each having its own state, utilities, and building blocks.

Lib Directory The ‘lib‘ directory contains utilities and actions that support the main functionalities of the application :

- **server-action :** Contains server-side actions.
- **providers :** Includes providers for various services.
- **utils :** General utility functions.
- **supabase :** This is a critical part of the backend integration, including :
  - **schema.ts :** Defines the database schema using Drizzle ORM, ensuring type-safe interactions with the database.
  - **hooks :** Custom hooks for communicating with the Supabase backend.
  - **queries and mutations :** Organized into subdirectories like ‘planner‘, ‘other‘, ‘teachers‘, ‘students‘, ‘users‘, and ‘timeline‘. These files contain queries and mutations written with Drizzle ORM to interact with Supabase.

Public Directory The ‘public‘ directory includes static assets such as images and icons, organized under ‘images‘ and ‘icons‘ subdirectories.

6.3 Backend

The backend of EDX is designed to provide a robust, scalable, and type-safe infrastructure that supports the platform's various functionalities and ensures a seamless user experience. The design principles and choice of components are aimed at achieving high performance, ease of development, and future-proofing the system.

6.3.1 Design Principles

The backend system design is guided by the following principles :

- **Scalability :** Ensuring the system can handle increasing amounts of work and accommodate growth.
- **Type Safety :** Leveraging type-safe technologies to reduce runtime errors and improve code quality.
- **Performance :** Utilizing efficient tools and technologies to provide a responsive and fast experience.
- **Ease of Development :** Choosing components that streamline the development process and enhance productivity.

6.3.2 Component Selection and Integration

The choice of backend components is crucial for achieving the desired design principles. Here is an overview of why specific components were selected and how they work together :

Supabase, PostgreSQL and Drizzle ORM We chose Supabase as it provides an incredible set of tools such as authentication, real-time databases, serverless functions and storage. It simplifies backend development by offering pre-built micro-services that integrate seamlessly with the rest of the system.

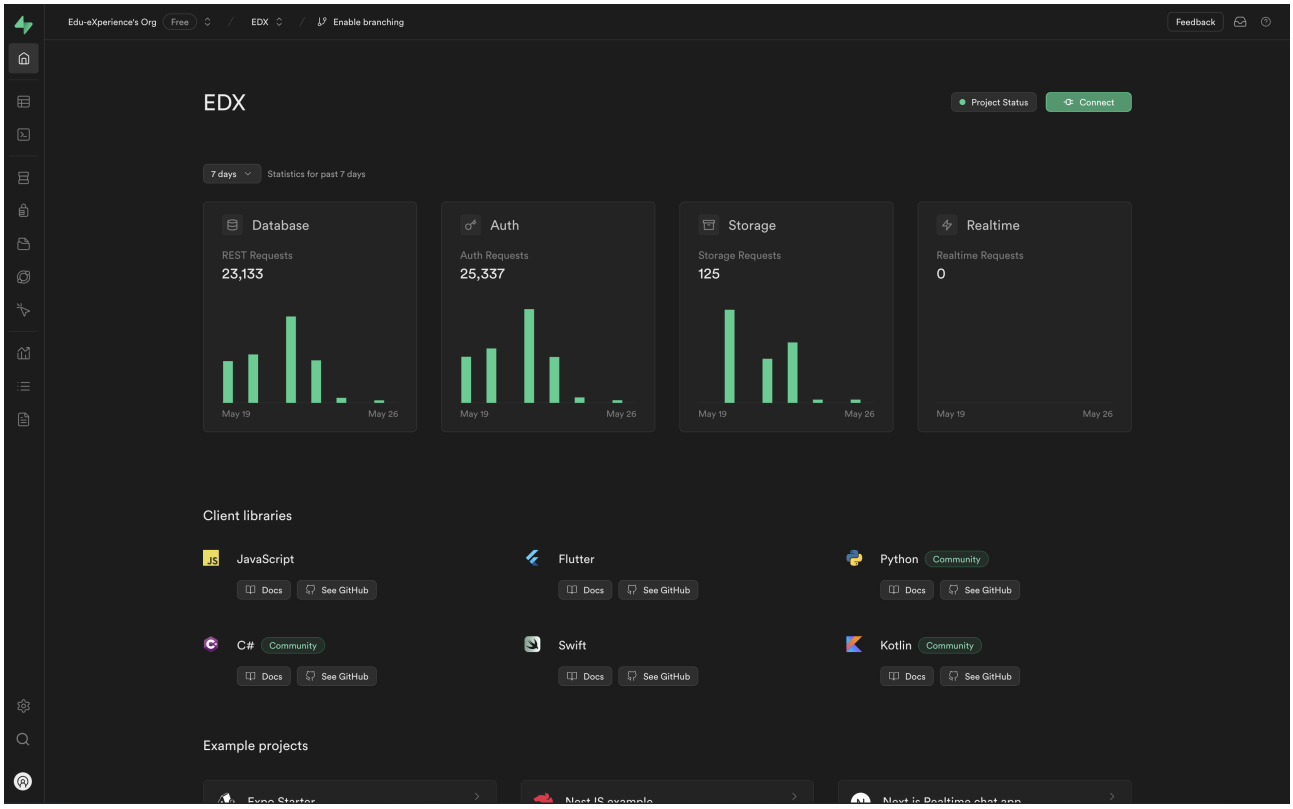

Figure 13 – Supabase Dashboard

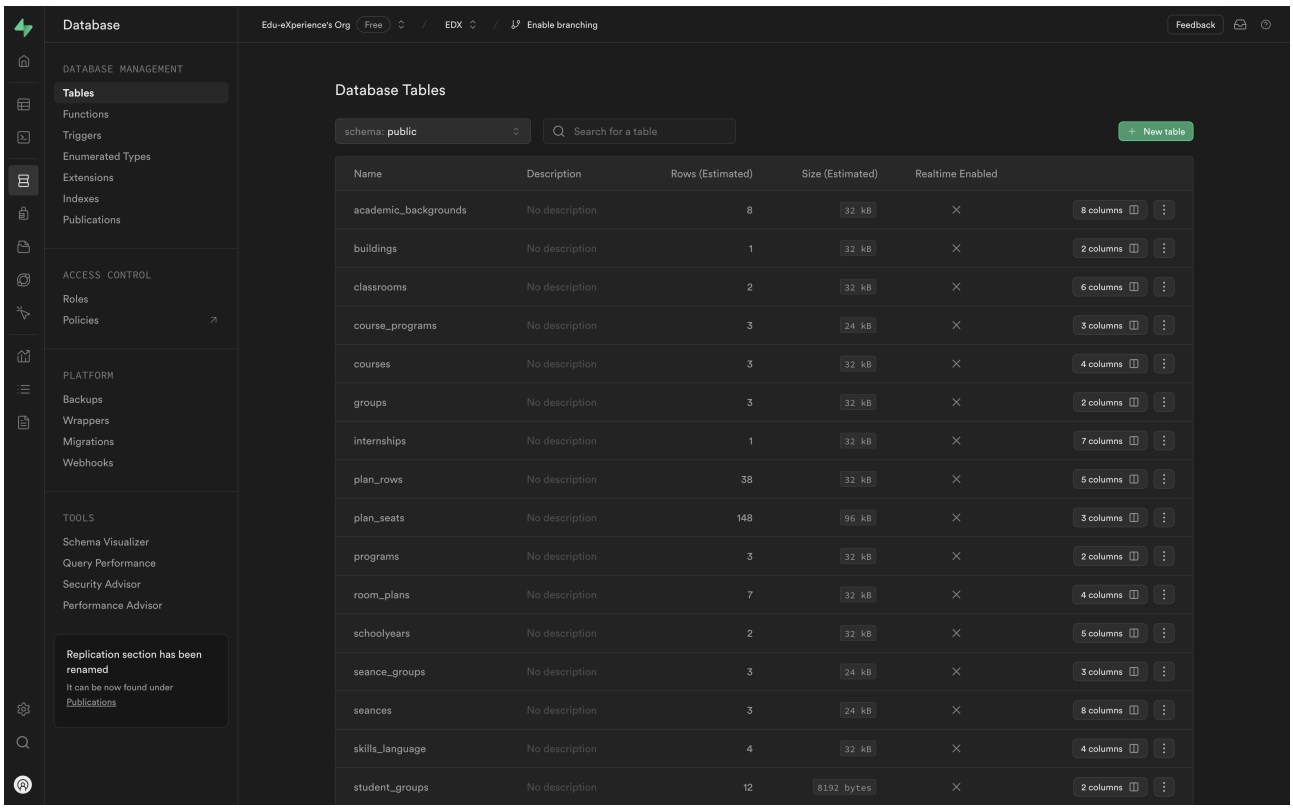

FIGURE 14 – Supabase Tables

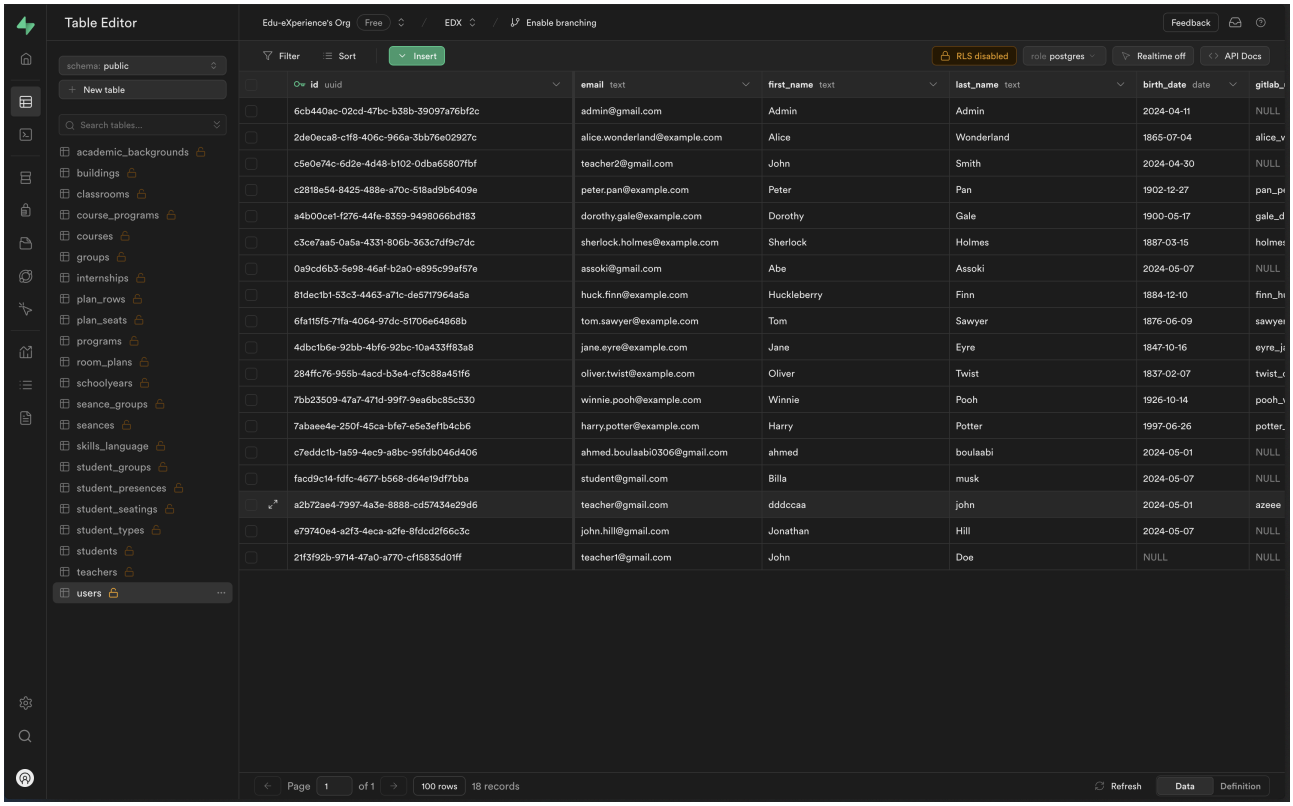

FIGURE 15 – Supabase Users Table

PostgreSQL is robust and scalable while being a relational database system.

We used Drizzle-orm to ensure end-to-end typesafety, and it also allows us to create complex queries in typescript, it builds prepared statements to interact with the database, which further improves the latency of our backend.

```
export async function createPlan(
  plan_name: string,
  cards: Card[],
  rowsData: RowsDataType
) {
  "use server";

  function countSeatsInRowsData(rowsData: RowsDataType): number {
    let totalSeats = 0;
    for (let key in rowsData) {
      if (rowsData.hasOwnProperty(key)) {
        totalSeats += rowsData[key].cards.length;
      }
    }
    return totalSeats;
  }

  const cleanCards = cards.filter((c) => c.type == "row");

  //   Mutation Prep
  const prepRoomPlans = db
    .insert(roomPlans)
    .values({
      name: sql.placeholder("plan_name"),
      numberOfRows: sql.placeholder("number_of_rows"),
      numberOfSeats: sql.placeholder("number_of_seats"),
    })
    .returning()
    .prepare("create-room-plan");

  const prepPlanRows = db
    .insert(planRows)
    .values({
      name: sql.placeholder("row_name"),
      planId: sql.placeholder("plan_id"),
      x: sql.placeholder("x"),
      y: sql.placeholder("y"),
    })
    .returning()
    .prepare("add-plan-row");

  const prepSeats = db
    .insert(planSeats)
    .values({
      rowId: sql.placeholder("row_id"),
      seatName: sql.placeholder("seat_name"),
    })
    .returning()
    .prepare("add-plan-seat");

  // Execution

  const roomPlan = await prepRoomPlans.execute({
    plan_name: plan_name,
    number_of_rows: cleanCards.length,
    number_of_seats: countSeatsInRowsData(rowsData),
  });

  for (let key in rowsData) {
    if (rowsData.hasOwnProperty(key)) {
      let R = rowsData[key];
      let RCard = cards.find((c) => c.id == R.id);
      let row = await prepPlanRows.execute({
        row_name: R.name.trim() != "" ? R.name : R.id,
        plan_id: roomPlan[0].id,
        x: RCard?.coordinates.x,
        y: RCard?.coordinates.y,
      });

      R.cards.map(async (card) => {
        await prepSeats.execute({ row_id: row[0].id, seat_name: card.text
});   });
    }
  }

  return roomPlan;
}
```

```
export async function saveStudentSeating(seance_id: string, seating: Seating)
{ const deleteSeating = db
    .delete(studentSeatings)
    .where(eq(studentSeatings.seanceId, sql.placeholder("seanceId")))
    .prepare("delete-student-seating");

  const insertSeating = db
    .insert(studentSeatings)
    .values({
      seanceId: sql.placeholder("seanceId"),
      seatId: sql.placeholder("seatId"),
      studentId: sql.placeholder("studentId"),
    })
    .prepare("insert-student-seating");

  await deleteSeating.execute({ seanceId: seance_id });

  for (const seatId in seating) {
    if (seating.hasOwnProperty(seatId)) {
      const seatInfo = seating[seatId];

      await insertSeating.execute({
        seanceId: seance_id,
        seatId: seatInfo.seat_id,
        studentId: seatInfo.student.id,
      });
    }
  }

  return { success: true };
}
```

FIGURE 16 – Example of a mutation file

Vercel We deploy on Vercel due to its fast, reliable hosting and continuous deployment capabilities.

7 Detailed Explanation of the Implementation of Key Features

7.1 Student Management

The Student Management feature was implemented to handle all student-related data and processes. This includes :

- **Student Profiles :** Profiles are created and updated using forms that validate inputs with Zod. The data is stored in the PostgreSQL database via Drizzle ORM. The schema includes fields for personal information, academic details, and contact information.
- **Enrollment :** The enrollment process involves multiple steps, including form submissions and approvals. The status of each enrollment request is tracked in the database.
- **Attendance :** Real-time attendance tracking is achieved using a combination of frontend interfaces and backend APIs. Teachers can mark attendance using a responsive UI, and the data is immediately updated in the database.

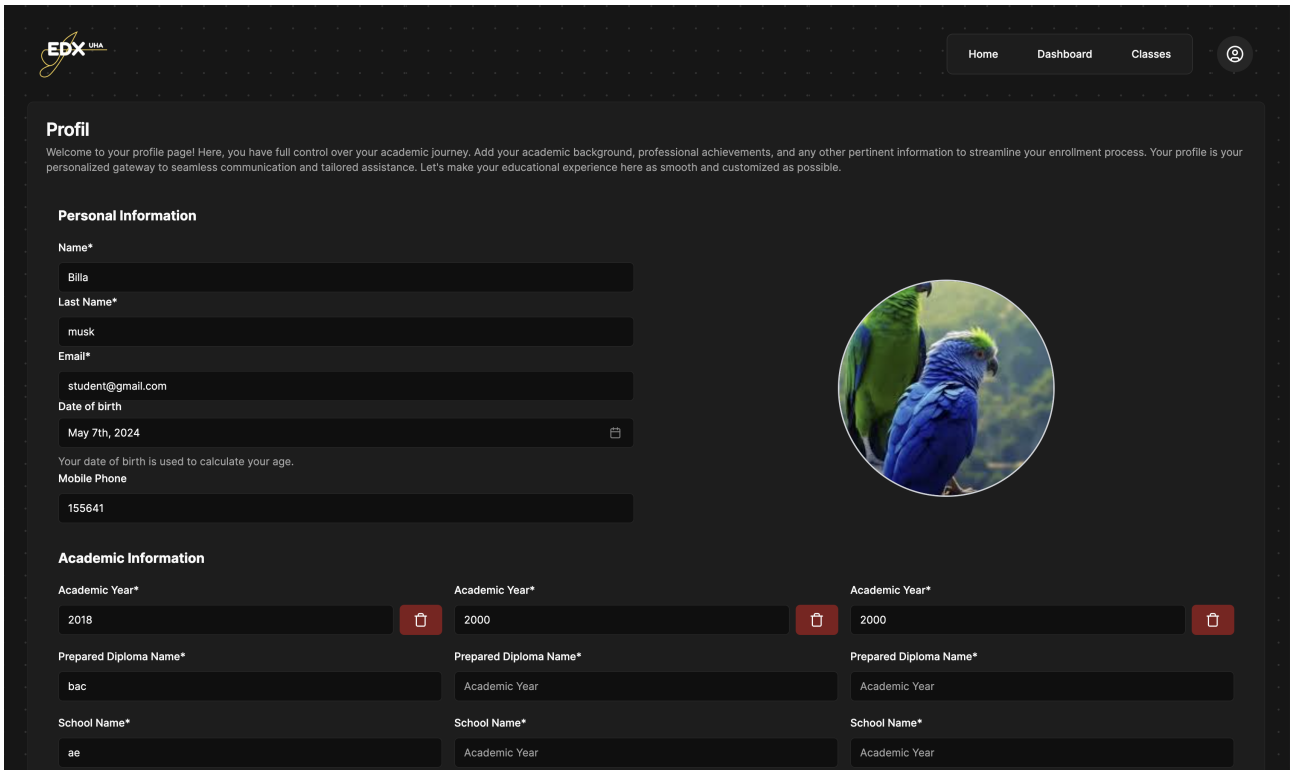

Figure 17 – Student Profile Page

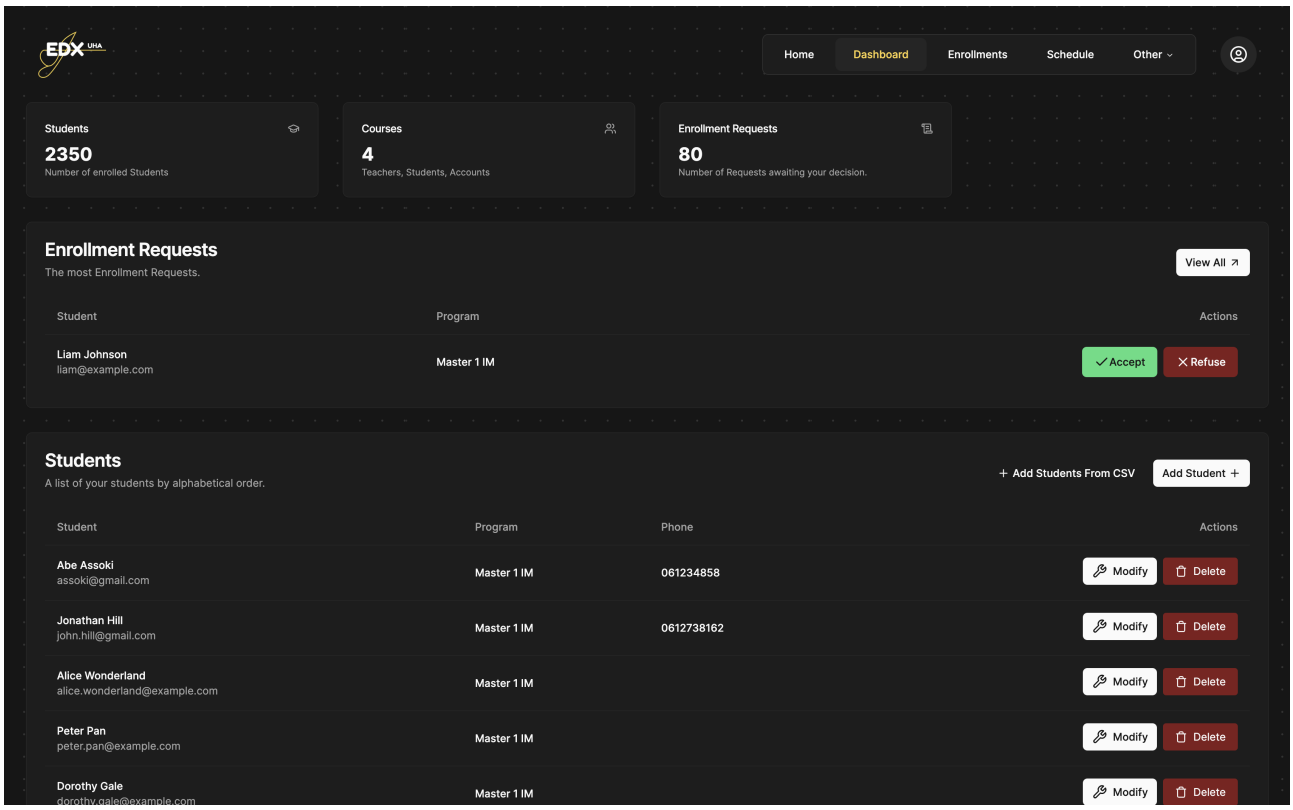

FIGURE 18 – Teacher Dashboard Page

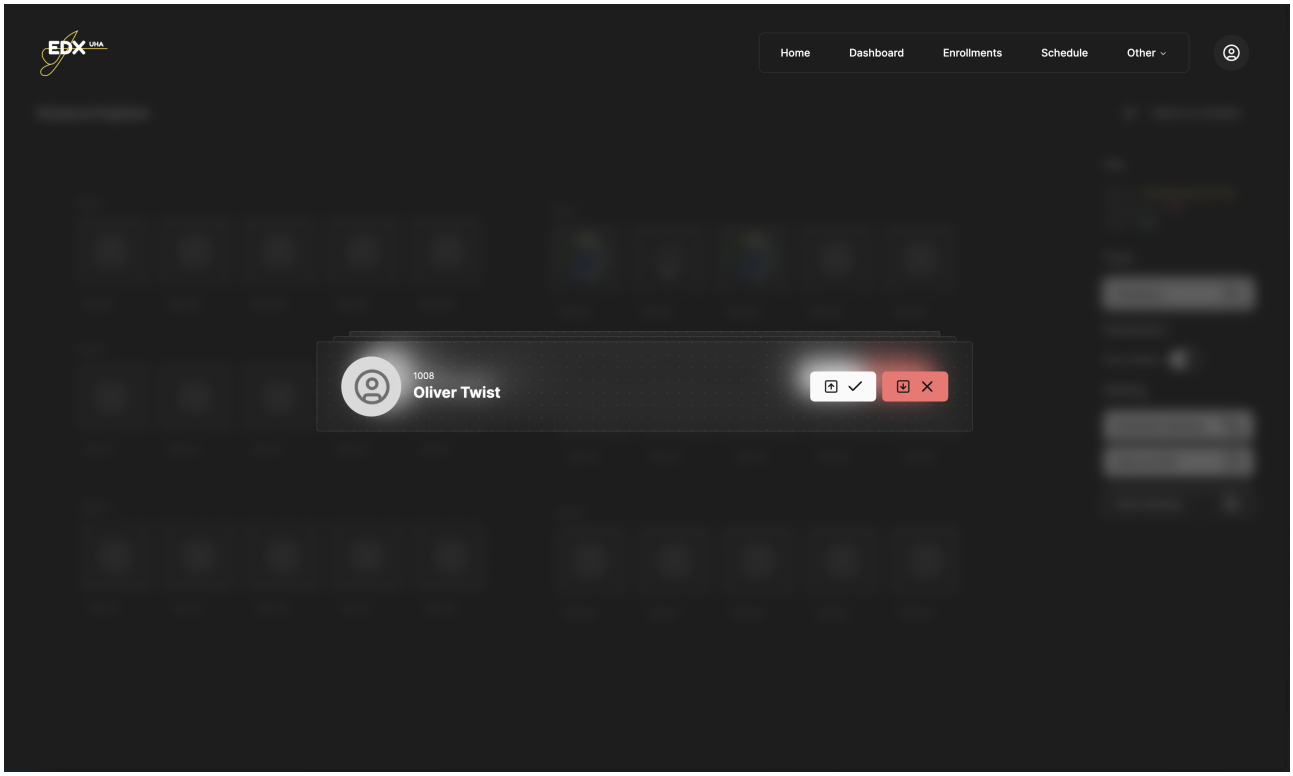

FIGURE 19 – Presence Tracking Page

7.2 Teacher Management

Teacher Management focuses on maintaining accurate records of teacher information and schedules :

— **Teacher Profiles :** Similar to student profiles, teacher profiles are managed using forms validated by Zod. Data includes professional information, subjects taught, and contact details.
— **Scheduling :** The scheduling system allows teachers to create seances. This feature provides a calendar view implemented with drag and drop containers.

7.3 Seating Arrangements

The Seating Arrangements feature was designed to facilitate the creation and management of classroom seating plans :

— **Dynamic Seating Plans :** Users can create and adjust seating plans through a drag-and-drop interface built with D3.js and dnd-kit. The layouts are stored in the database and can be printed as PDFs.
— **Student Allocation :** Students are allocated seats manually or automatically.
— **Interactive Maps :** Visual maps of seating arrangements are generated and updated in real-time, allowing easy adjustments and references.

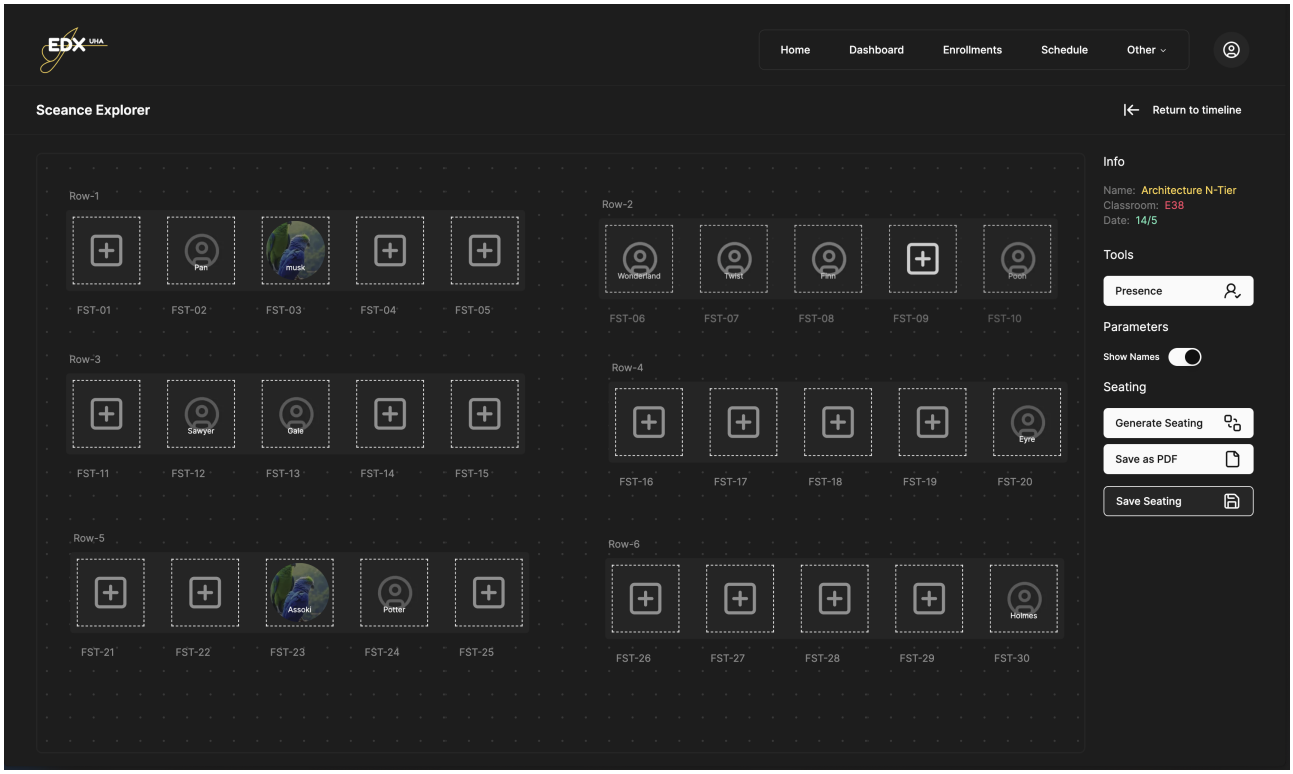

FIGURE 20 – Student Seating

7.4 Course Scheduling

Course Scheduling involves creating and managing course timetables :

— **Timetable Creation :** Teachers can create timetables using a user-friendly interface. Timetables are stored in the database and can be viewed and edited as needed.

— **Classroom Allocation :** Courses are assigned to specific classrooms.
— **Calendar Integration :** Future integration with personal and institutional calendars will allow for synchronized schedules.

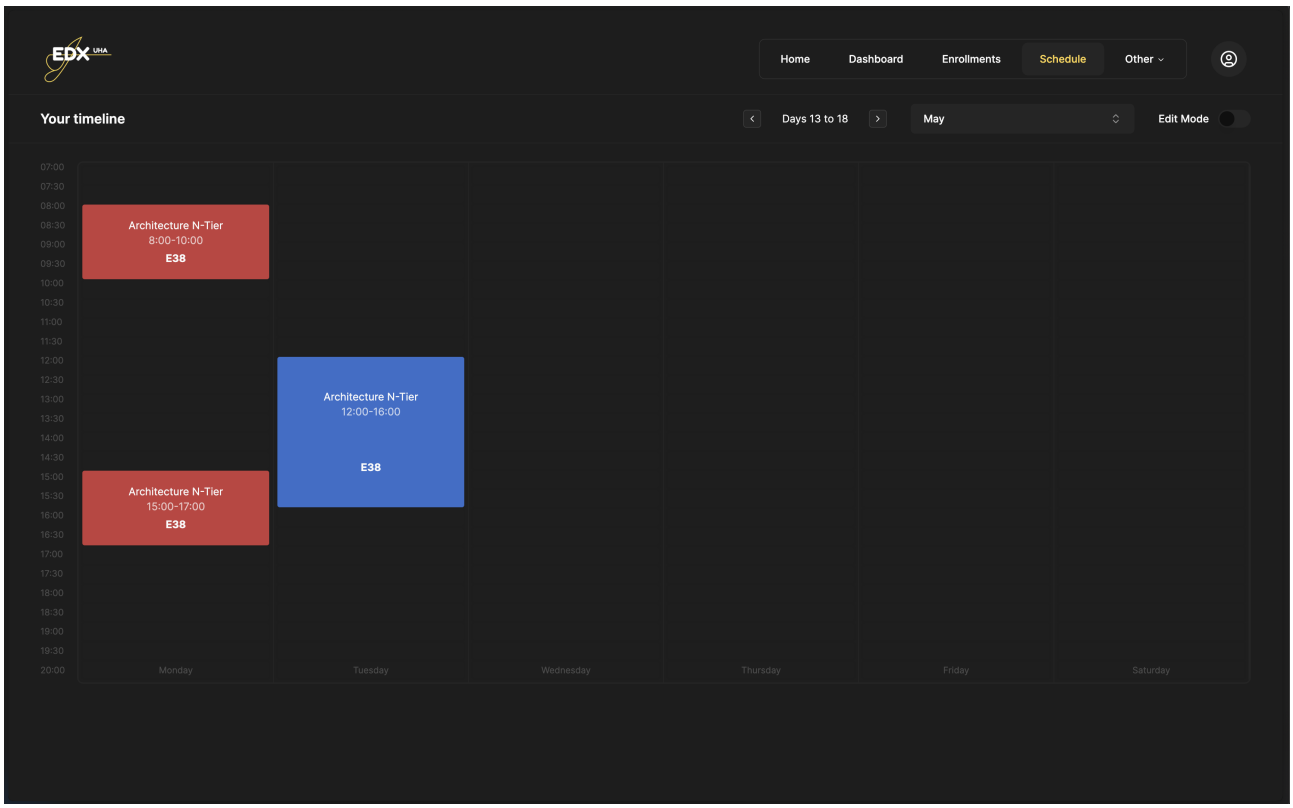

FIGURE 21 – Teacher's schedule page

7.5 Security and Authentication

Security is a crucial aspect of EDX, ensuring that data is protected and access is controlled :

— **Authentication :** Supabase's authentication service is used to manage user logins and registrations. JWT tokens are employed to secure API requests.
— **Authorization :** Role-based access control (RBAC) is important to guarantee that users can only access resources appropriate to their roles.
— **Data Protection :** As well as TLS encryption, sensitive data is encrypted, and secure coding practices are followed to prevent vulnerabilities.

7.6 Performance Optimization

Performance optimization ensures that EDX runs smoothly and efficiently :

— **Code Splitting :** Next.js's built-in code splitting ensures that only necessary code is loaded, improving load times.
— **Caching :** Strategic caching is implemented using Vercel's edge functions to reduce server load and speed up response times.

— **Database Optimization :** Indexing and query optimization techniques are used to ensure that database operations are efficient and fast.

7.7 Continuous Integration and Deployment

CI/CD processes are set up to streamline development and deployment :

— **Automated Testing :** Code tests and integration tests are run automatically on Vercel's end to ensure code quality.

— **Deployment Pipelines :** Vercel handles deployments, allowing for automatic updates and rollbacks as needed.

8 Testing and Evaluation

8.1 Testing Strategies and Methodologies

Given the scope and timeline of the EDX project, we employed several strategies to ensure the quality and performance of the platform. Instead of traditional unit and integration tests, we focused on automated tools and peer reviews.

8.1.1 Automated Tools

We used Google Lighthouse and GTmetrix to evaluate a few key metrics such as the performance, accessibility, best practices, and SEO of the EDX platform. These tools gave us a comprehensive insights and actionable recommendations to improve the quality of our web application.

8.1.2 Performance Testing

- **Page Load Times :** Both tools provided detailed metrics on page load times, highlighting areas where optimizations could reduce load times and improve user experience.
- **Accessibility :** Lighthouse audits included accessibility checks to ensure the platform is usable by individuals with disabilities. Recommendations were implemented to improve accessibility scores.
- **SEO :** Both tools provided insights into SEO best practices, helping us optimize the platform for search engines.
- **Best Practices :** Recommendations from Lighthouse and GTmetrix on web development best practices were implemented to enhance security, performance, and overall user experience.

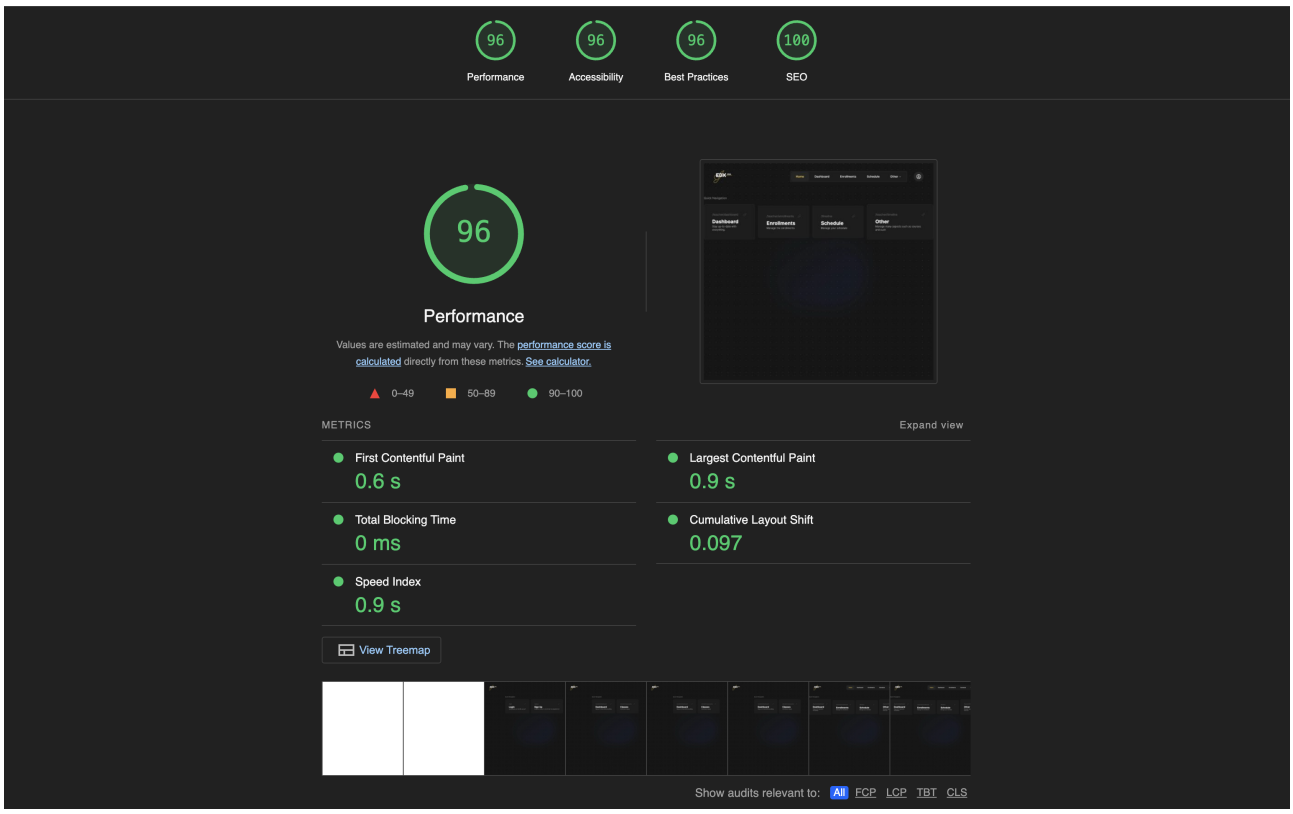

FIGURE 22 – Lighthouse report

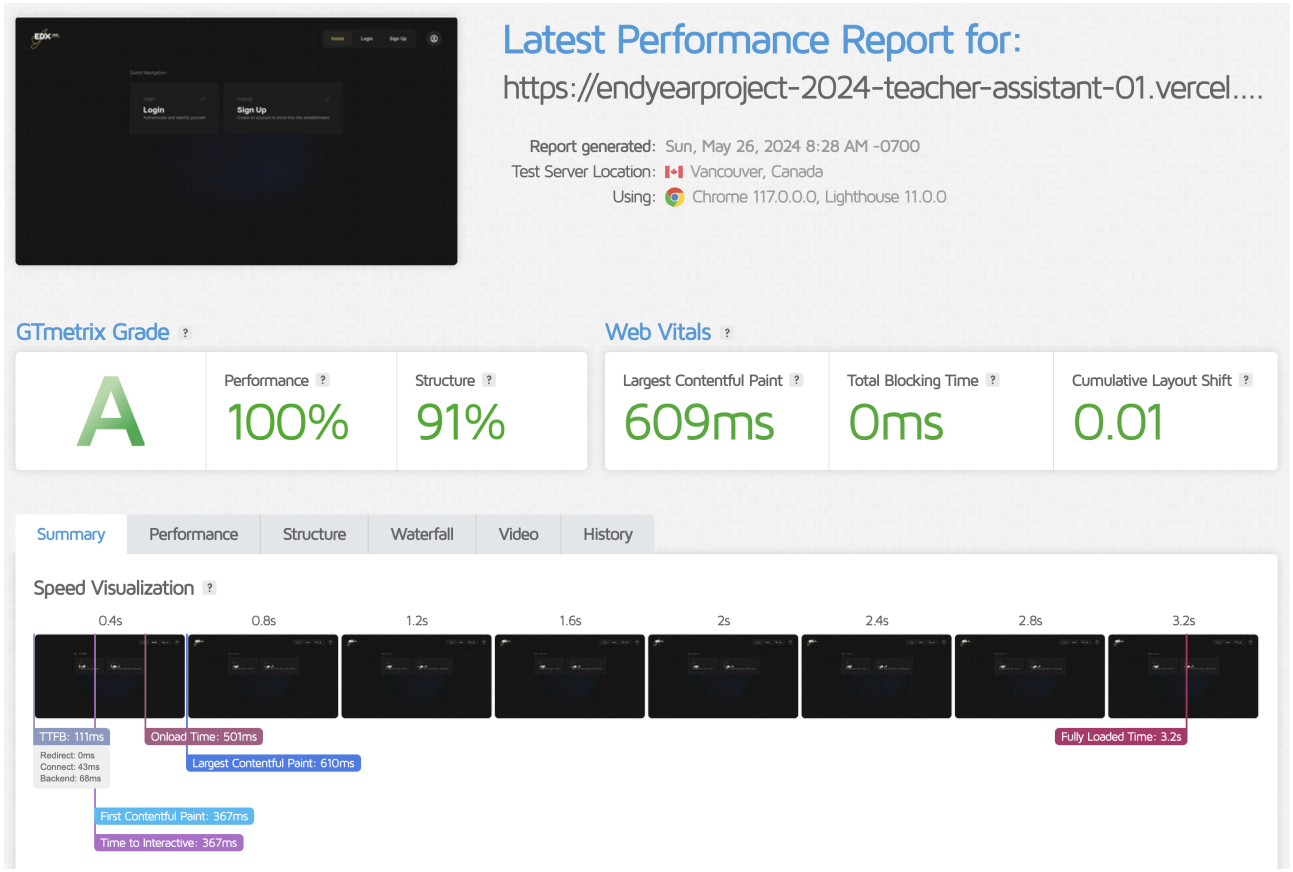

FIGURE 23 – GTMetrix report

You can find the live GTMetrix report here

8.1.3 Next.js Build Tests and Quality Control

Next.js provides built-in testing for build processes. We leveraged these features to ensure that our application builds correctly and performs well.

8.1.4 TypeScript and Peer Reviews

TypeScript played a crucial role in maintaining code quality and preventing runtime errors. By enforcing type safety, TypeScript helped catch errors early in the development process. Additionally, we conducted regular peer reviews to ensure code quality and consistency.

8.1.5 Build Tests and Quality Assurance

Next.js build tests were regularly executed to ensure that the application compiles without errors and that any build-time issues are identified and resolved promptly.

8.2 Evaluation of System Performance

Overall, the performance of the EDX platform was evaluated using the following criteria :

- **Load Times :** Evaluations showed acceptable load times with opportunities for further optimization.
- **Accessibility :** Accessibility scores were generally high, with continuous improvements based on audit recommendations.
- **SEO :** The platform's SEO performance was satisfactory, with all critical recommendations from the audits implemented.
- **Best Practices :** The platform adhered to best practices for web development, as indicated by high scores in Lighthouse and GTmetrix audits.

8.2.1 Future Testing Plans

While the current testing strategy provided valuable insights, future plans include :

- **Stress Testing :** To evaluate the platform's performance under load, stress testing will be conducted. This will require more time and resources than currently available.
- **Unit and Integration Tests :** Using Storybook and Jest, implement unit and integration tests to make sure every part fits together and works well within the system.
- **Automated End-to-End Testing :** Utilizing tools like Cypress or Selenium to automate end-to-end tests, ensuring that user interactions work as expected.

The goal of this testing and assessment is to make sure that when EDX develops further, it stays reliable, efficient, and easy to use.

9 Challenges and Solutions

9.1 Problems Encountered During Development

Throughout the development of EDX, we faced several challenges that required creative solutions and perseverance. Here are the key challenges and how we addressed them :

9.1.1 Time Constraints

We are very passionate about the project and have a strong desire to innovate, the time constraints were a significant challenge. We aimed to deliver a high-quality product within a limited timeframe, which required efficient time management and prioritization of tasks but still limited us in terms of goals we wanted to achieve.

9.1.2 Setting up Supabase and Drizzle ORM

Initially, setting up Supabase and Drizzle ORM was finicky. We encountered several issues related to configuration and stability, which necessitated multiple revisions of the code to ensure a stable and robust setup.

9.1.3 Supabase Auth Provider Issues

The Supabase auth provider occasionally caused issues, disrupting the authentication flow. To resolve this, we wrote new custom authentication hooks, which provided a more reliable and tailored solution to our needs.

9.1.4 Complexities with dnd-kit

dnd-kit, is powerful but lacks extensive documentation and only offers simple examples. This led us to spend considerable time understanding its intricacies and developing complex interactions without guides. Despite the initial time investment, the end result was highly effective and worth the effort.

9.1.5 Infinite Canvas for Room Planner

Creating an infinite canvas for the room planner page was particularly challenging. Integrating d3.js with dnd-kit led to conflicts between the two libraries. We experimented with numerous solutions, each introducing new limitations, until we developed a method that avoided these conflicts without imposing restrictions on functionality.

9.1.6 Drag and Drop schedule

We wanted to have a unique and interactive experience for the schedule management, this lead us to spend a considerable amount of time writing

complex functions to handle all kinds of different interactions to ensure the best experience to the users.

9.1.7 Prop Drilling in React

We quickly encountered the issue of prop drilling, a common problem in React.js development where props need to be passed through many layers of components. We resolved this by adopting Jotai for state management, which streamlined state handling and reduced complexity.

9.1.8 Animating with Framer Motion

Implementing complex animations with Framer Motion revealed limitations in their animation orchestrator. We had to experiment with various approaches, often through trial and error, to find solutions that met our requirements. Despite the initial challenges, we achieved the desired animations.

9.1.9 Deployment Issues

Deploying the application posed another challenge. Vercel requires a pro account for deployments directly from the UHA organization repository. To work around this, we forked the repository to a personal account and deployed from there.

9.2 How These Challenges Were Addressed and Resolved

- **Setting up Supabase and Drizzle ORM :** We conducted thorough testing and iteratively refined our setup until we achieved a stable configuration.
- **Supabase Auth Provider Issues :** By developing custom authentication hooks, we bypassed the limitations of the default auth provider and improved reliability.
- **Complexities with dnd-kit :** Extensive testing and exploration allowed us to understand dnd-kit deeply and implement advanced features without external guides.
- **Infinite Canvas for Room Planner :** Through persistent experimentation and iteration, we developed a solution that effectively integrated d3.js and dnd-kit without conflicts.
- **Prop Drilling in React :** Implementing Jotai for state management simplified our state handling and eliminated the prop drilling issue.
- **Animating with Framer Motion :** We navigated the limitations of Framer Motion by experimenting with different methods until we found effective solutions for complex animations.
- **Time Constraints :** Effective time management, task prioritization, and focusing on key features allowed us to meet our deadlines without

compromising quality.

— **Deployment Issues :** Forking the repository to a personal account enabled us to deploy the application on Vercel, circumventing the limitations of the UHA organization account.

9.3 Final Thoughts

Despite the numerous challenges we faced, we remained dedicated and passionate about the project. We are grateful for the opportunity provided by Professor Joel Heinis, which allowed us to learn, innovate, and create a platform we are proud of. The obstacles we encountered only strengthened our resolve and contributed to the development of a robust and effective educational management platform.

10 Results and Discussion

10.1 Summary of the Results Achieved

The development of EDX has led to the creation of a comprehensive educational management platform that integrates various functionalities into a cohesive system. Key achievements include :

- Implementation of student and teacher management features.
- Development of dynamic seating arrangements and course scheduling modules.
- Integration of real-time presence tracking.
- Utilization of modern web technologies to ensure a responsive and user-friendly interface.
- Establishment of a robust backend infrastructure with Supabase, PostgreSQL, and Drizzle ORM.

10.2 Comparison with Initial Objectives

Our initial objectives were to create a centralized platform that enhances operational efficiency and user experience for educational institutions. The results of our project align closely with these objectives :

- **Objective :** Develop a user-friendly interface for administrators, teachers, and students.
- **Result :** The platform's UI is intuitive, consistent, and responsive, meeting user needs effectively.
- **Objective :** Integrate core functionalities such as student and teacher management, seating arrangements, and real-time presence tracking.
- **Result :** All core functionalities have been successfully integrated, providing a seamless user experience.
- **Objective :** Ensure scalability, security, and performance.
- **Result :** The backend infrastructure is scalable and secure, with optimized performance.
- **Objective :** Utilize modern web technologies.
- **Result :** Technologies like Next.js, TypeScript, and Tailwind CSS have been effectively utilized to enhance the platform.

10.3 Discussion on the Effectiveness and Impact of EDX

The effectiveness of EDX is evident in its ability to centralize and streamline various administrative tasks. The platform has demonstrated significant potential in improving the operational efficiency of educational institutions by :

— Reducing the reliance on multiple third-party tools.
— Providing real-time data for better decision-making.
— Enhancing communication and coordination among administrators, teachers, and students.

The impact of EDX extends beyond immediate functionality, offering a scalable solution that can grow with the institution's needs. The use of modern technologies ensures that EDX remains adaptable and capable of integrating future advancements.

11 Future Work

11.1 Potential Improvements and Additional Features

While EDX has achieved its core objectives, there are several areas for potential improvement and additional features :

- **Enhanced Reporting :** Develop advanced reporting tools for detailed analytics on student performance, attendance, and other metrics.
- **Multilingual Support :** Add support for multiple languages to cater to a broader audience.
- **Calendar Integration :** Integrate with external calendar systems for better schedule management.
- **Stress Testing :** Conduct stress tests to ensure the platform can handle a large number of concurrent users.
- **Reporting :** Generating reports on attendance patterns and trends.
- **Conflict Resolution :** Detecting and resolving scheduling conflicts.
- **Notifications :** Sending alerts and notifications for attendance-related updates.

11.2 Long-term Vision for the Platform

In the long term we would like EDX to evolve into a full fledged educational ecosystem for all aspects of educational management. This includes :

- **Expanding Functionality :** Continuously add new features and improvements based on user feedback and emerging trends in education technology.
- **Integration with Other Systems :** Ensure seamless integration with other educational tools and platforms to provide a unified experience.

12 Conclusion

In conclusion, EDX has made significant strides towards becoming a vital tool for educational management. The foundation laid by this project paves the way for continuous improvement and expansion, with a focus on enhancing the educational experience for all.