Aung Moe Myint Thu

Full-Stack Web Developer

SUMMARY

Aspiring and highly motivated full-stack web developer with a strong foundation in front-end and back-end technologies. Skilled in building responsive and dynamic web applications through personal and academic projects. Proficient in database management and experienced in using tools like Git for version control. Committed to continuous learning and excited to contribute innovative solutions to real-world challenges while growing within a collaborative development environment.

CONTACT INFORMATION

- Location : Hmawbi, Yangon, Myanmar
- Email : aungmoemyintthu@gmail.com
- Phone : +959751200822
- Github : https://github.com/aungmoe32
- LinkedIn : https://www.linkedin.com/in/aung-moe-myint-thu-679884258
- Portfolio : https://aungmoe32.github.io/

TECHNICAL SKILLS

Languages

- JavaScript
- Typescript
- PHP
- Java
- SQL

Frameworks/Libraries

- Laravel, Nextjs, React Native (Expo)
- Laravel Filament, InertiaJs, Passport
- React, Vue.js, TailwindCSS
- React Query, React Hook Form, Zod
- Vite, Mix

Tools

- Git (Gitflow Technique)
- Github
- Postman
- Laravel Sail
- Vercel

Databases

- PostgreSQL
- MySQL

Others

- Authentication/Authorization (JWT, OAuth)
- RESTful API Design and Implementation
- Web APIs, Web Push, WebRTC
- Secure Notification System
- Web Security (CORS, XSS, SQL Injection)
- MVC design pattern
- Firebase, Supabase
- Stripe, Paypal payments integration
- Responsive Design

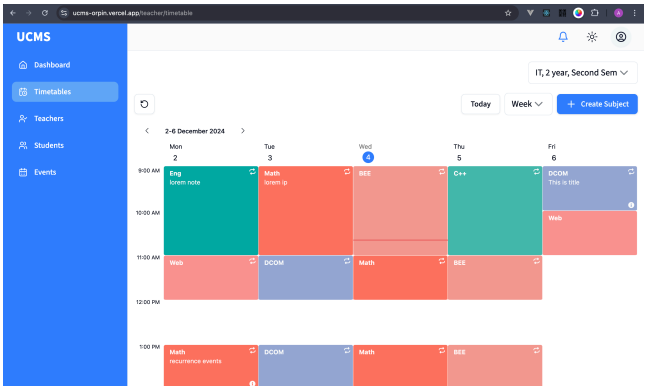
PROJECTS

University Campus Management System (UCMS)

- **Role** : Full-Stack Developer
- **App URL** : https://ucms-orpin.vercel.app
- **Demo Video** : YouTube
- **Github Repo** : https://github.com/aungmoe32/ucms

Description

The **University Campus Management System (UCMS)** is a web-based application designed to simplify campus operations, including timetable management, event notifications, and CRUD operations for students and teachers. It ensures a responsive, efficient, and user-friendly experience for both students and teachers.



Problem Statement

Universities always face timetable problems. It would be more convenient if students could get live details (date, location, room, etc.) about changes in class times, tutorials, assignments, and other events. The system should be restricted to only university students (no outsiders).

Solution

I've developed a web app with two roles:

- Student
- Teacher (with administration access)

A semester consists of year, major, and term.
Students are grouped by semester.
Each semester has a timetable and subjects.
Students can view subject timings in the Dashboard's Timetable according to their semester.
When teachers make timetable changes, students receive web push notifications immediately.

Teachers can:

- Modify timetables for subjects they teach in their respective semesters
- Teach multiple subjects
- Easily view their specific teaching times in the Dashboard Timeline
- Perform CRUD operations on resources (students, teachers, events, subjects, etc.)

Registration is closed to prevent outside access (login only).
Teachers must create student accounts.
Each Event on the Timetable is categorized by:

- Title
- Description
- Start Date
- End Date
- Subject

- Event type (tutorial, assignment, etc.)
- Repeat (for recurrence events)
- Color (based on subject color)

When a teacher modifies a semester's timetable, students in that semester receive web push notifications.

Technologies Used

Frontend:

- **Framework**: React, Next.js
- **Styling**: TailwindCSS, Shadcn UI (Dark/Light mode support)
- **State Management**: React Query
- **Libraries** : Devextreme UI Components (For Scheduler), React Hook Form, Zod (for validation)

Backend:

- **Framework**: Next.js API
- **Database**: PostgreSQL with Drizzle ORM
- **Authentication**: JWT, NextAuth

Implementations

- **Next.js Features**: Enhances user experience by utilizing Server-Side Rendering (SSR) for faster initial loads, Lazy Loading to optimize resource usage, and Prefetching to preload data for smooth navigation.
- **Loading UI with Skeletons**: Improves user experience by displaying skeleton loaders while data is being fetched, providing a visual cue during loading.
- **Infinite Scrolling** : Automatically loads more content as users click load more button.
- **Debouncing Search**: Delays the search input processing until the user stops typing, reducing unnecessary API calls and improving efficiency.
- **Timetable Implementation**: Built a timetable using DevExtreme React Scheduler for efficient scheduling and event management.
- **Array Field with React Hook Form**: Enables managing dynamic form fields as an array, making it easier to handle lists of inputs (selecting multiple teaching subjects).
- **Zod Validation**: Ensures consistent form validation on both client and server, providing type-safe and schema-based validation.
- **NextAuth with JWT**: Handles authentication securely using JWT for stateless session management in Next.js.
- **Vercel Edge Middleware**: Optimizes performance by running custom logic at the edge, allowing for faster response times.

Challenges Overcome

- **Recurring Event System**: Built a system to manage recurring events with flexible schedules and optimized database storage for scalability.
- **Performance Optimization in Next.js**: Implemented lazy loading, SSR, and prefetching to improve page load speed and navigation for dynamic content.

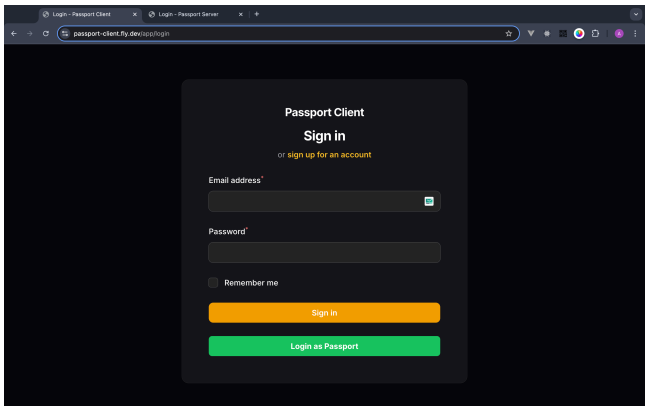
- **React Query Integration**: Used React Query for efficient data fetching, caching, and synchronization, simplifying state management and improving user experience.
- **Database Query Optimization**: Optimized database queries for complex timetable relationships.
- **Web Push Notifications for iOS**: Added push notifications for iOS devices.
- **Optimized APIs**: Created APIs with sorting, filtering, and pagination for efficient data retrieval.
- **React Hook Form**: Used React Hook Form to manage forms and validation efficiently.
- **GitFlow**: Used GitFlow for managing branches during development, ensuring organized workflows for feature development, releases, and bug fixes.

Future additions

- **Notification Log** : A feature that will track and store all notifications sent within UCMS, allowing easy access to past notifications for better management.
- **Exam Results and Grades** : A feature that will organize and display exam results and grades based on academic years, allowing both students and teachers to easily track performance over multiple years.

Passport OAuth Server

The **Passport-Server** project is an OAuth 2.0 authentication server built with Laravel Passport and Laravel Filament, providing secure login and token management for client applications like Passport-Client.



Demo

- Explore the demo: YouTube (Timestamps on description)
- passport server : https://passport-server.fly.dev/
- passport client : https://passport-client.fly.dev/
- reference

Source Code

- Github repo : https://github.com/aungmoe32/passport-server

Technologies

- Laravel

- Passport
- Laravel Filament

Workflow Overview

1. **Login Button Click** (Passport-Client):
    - User clicks the “Login as Passport-Server” button on the Passport-Client login page.
    - The client redirects the user to the Passport-Server’s authorization endpoint, including a query string with the **client ID**, **redirect URI**, and **scope**.
2. **Authenticate on Passport-Server**:
    - If the user is **not already logged in** on Passport-Server, they are prompted to log in (e.g., email and password form).
    - Once logged in, Passport-Server asks for user consent (if applicable) to share their profile with Passport-Client.
3. **Authorization Code Issued**:
    - Upon successful login and consent, Passport-Server redirects the user back to the Passport-Client’s **redirect URI**, along with an **authorization code**.
4. **Authorization Code Exchanged for Token**:
    - Passport-Client sends the authorization code, along with its client ID and client secret, to Passport-Server’s token endpoint.
    - If valid, Passport-Server responds with an **access token** (and optionally a refresh token).
5. **User Logged In**:
    - Passport-Client uses the access token to fetch the user profile from Passport-Server’s user info endpoint.
    - Passport-Client creates a session for the user, completing the login process.

Features

On Passport-Server:

- OAuth 2.0 authentication services for secure login.
- A **Posts** section for users to write and manage their content.
- A **Dashboard** to manage users, clients, and tokens.
- Ability to **revoke client tokens** for enhanced security and control.

On Passport-Client:

- Integration with Passport-Server for seamless user authentication.
- A **Products** section for users to manage product listings.

Implementations

- **Laravel Filament** : Using Laravel Filament to provide an elegant admin interface for managing OAuth clients, users, and tokens.

- **On Passport-Server:**

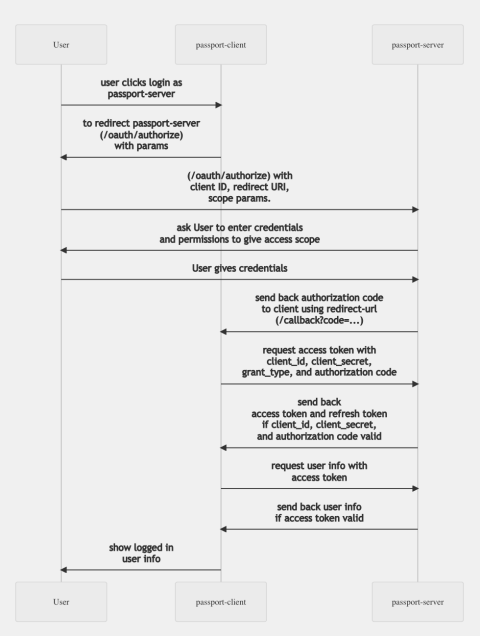
    - **Authorization Endpoint**: Validates the client, authenticates the user, and issues an authorization code.
    - **Token Endpoint**: Exchanges the authorization code for an access token.
    - **User Info Endpoint**: Provides user details when queried with a valid access token.

- **On Passport-Client:**

    - **Login Flow**: Initiates the OAuth flow by redirecting to Passport-Server.
    - **Callback Handling**:
        - Processes the authorization code and retrieves the access token.
        - Uses the token to fetch user details and establish a session.

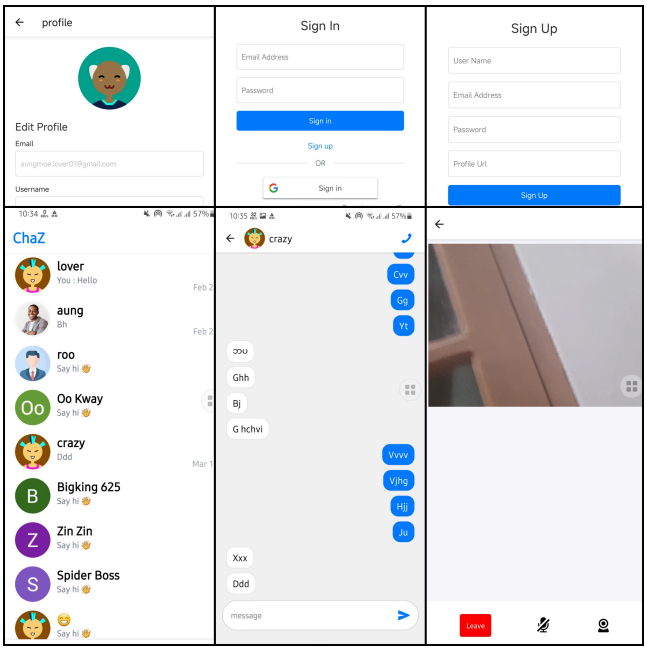
Challenges

1. **Token Security**: Ensuring the secure storage and handling of tokens on both server and client sides.
2. **OAuth Flow Complexity**: Implementing and debugging the complete OAuth 2.0 Authorization Code Grant flow.
3. **User Experience**: Maintaining a seamless user experience during redirections between Passport-Client and Passport-Server.
4. **Token Revocation**: Properly handling token revocation and ensuring affected sessions are invalidated.
5. **Admin Panel Customization**: Adapting Laravel Filament for specialized management tasks like token monitoring and user actions.



Chaz

A feature-rich react native chat application designed for seamless and real-time communication. The app supports multiple functionalities such as instant messaging, video calling, and personalized profiles, providing a user-friendly and interactive experience.



Source Code

- Github repo : https://github.com/aungmoe32/Chaz

Demo Video

- YouTube

Download apk

- https://drive.google.com/file/d/1LmfkDXhVcBwL5nnau9K6_9coyEnO6IC0/view?usp=sharing

Technologies Used

- **Frameworks and Libraries :**
    - React Native
    - Expo
    - Nativewind
    - React Navigation
    - Expo Router
    - Lottie (for animations)
    - Expo Packages (various)
    - Typescript
- **Backend and Database**: Firebase services, Firestore
- **Third-Party Services**: VideoSDK
- **Development Tools**:
    - Expo Go

Features

- Google Authentication
- Password Reset
- Push Notifications
- Video And Audio Calls
- User Profiles

> Note : Only available for android currently

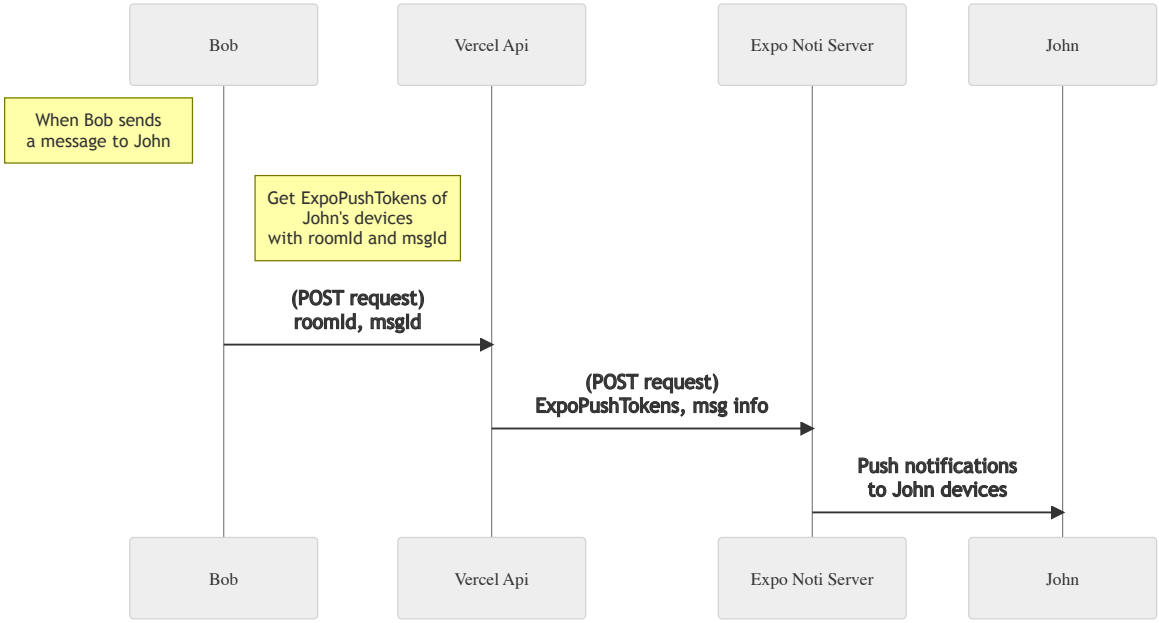
Implementations

- **Navigation**: Implemented using Expo Router's tab and stack navigation for seamless routing.
- **Video and Audio Calls**: Enabled using VideoSDK for real-time communication.
- **Authentication**: Implemented email and Google authentication using Firebase Auth.
- **Database**: Used Firestore for real-time, cloud-based data storage.
- **Password Reset**: Built a forgot password feature using Firebase Auth.
- **Push Notifications**: Implemented using Expo Push Notifications with a Vercel serverless API for real-time updates.
- **Animations**: Used Lottie for loading animations.
- **Development**: Used Expo Go for development on mobile devices.

Challenges

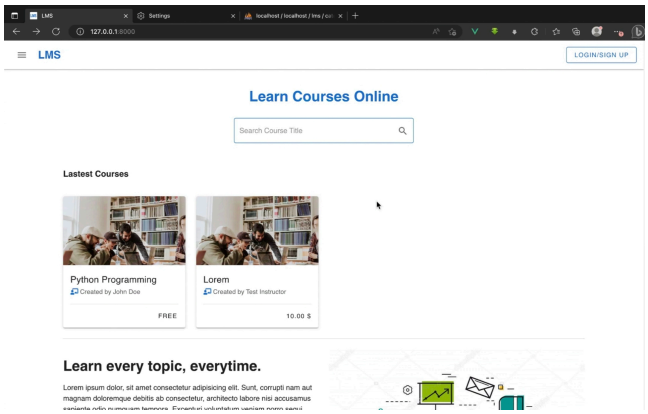
- **Build Process**: Faced challenges with EAS builds and local build configurations.
- **Security**: Used Expo Secure Store for environment variables.
- **Push Notification Challenges**: First time using Expo Push Notifications, faced learning curve in setup and configuration.
- **Notification System**: Faced challenges implementing a secure notification system using Expo Push Notifications.
- **Responsive Design**: Ensured responsive design across devices for user experience.

Secure Push Notification System



LMS

A comprehensive Learning Management System (LMS) built to simplify online learning and course management.



Source Code

- Github repo : https://github.com/aungmoe32/lms

Demo

- Explore the demo: YouTube.

Tech Stack

- **Backend**: Laravel 8

- **Frontend**: Vue 2 with Vuetify for UI
- **State Management**: Vuex
- **Routing**: Vue Router
- **API Security**: Laravel Sanctum

Features

- **User Authentication**: Login and register functionality with role-based permissions.
- **Course Management**:
  - Courses have sections and lectures with support for video, audio, text, and documents.
  - Includes filtering and searching by categories, price, and levels.
- **User Roles**:
  - **Student**: Enroll in courses, rate/comment, and learn from curriculums.
  - **Instructor**: CRUD operations on courses and sections, and monetize courses.
  - **Admin**: Manage users, categories, and roles.
- **Course Lecture File Support**:
  - Video: MP4, QuickTime
  - Audio: MP3, WAV, M4A
  - Document: PDF
- **Ratings & Comments**: Students can rate and review courses.
- **Payment Integration**: Uses PayPal sandbox for course payments.

Implementations

- **SPA with Laravel Sanctum API**: Built Vue.js SPA connected to a Laravel Sanctum API backend for secure authentication and data handling.

- **Vuetify for UI and Responsive Design**: Built with Vuetify to create UI components like curriculums and tables, ensuring a clean and responsive design for all devices.

- **State Management in Vue**: Implemented state management using Vuex, allowing centralized management of application state.

- **Role Permissions**:

  - Students, Instructors, and Admins have distinct access levels.

- **Laravel Eloquent Polymorphic Relationship for Course Lecture Types**: Implemented in Laravel Eloquent that links various lecture content types (video, audio, document, text) to a course lecture.

- **API Protection with Middlewares**: Secured the API using Laravel Auth middlewares and implemented custom middleware such as `IsSubscribed` to ensure users have an active subscription and `IsCourseOwner` to verify course ownership before allowing access to specific routes.

- **Laravel Sanctum Session-Based Authentication**: Using Laravel Sanctum's session-based cookie authentication, which provides CSRF protection and secures credentials against XSS attacks.

- **Eloquent Events for Relationship Models**: I used Laravel Eloquent events to handle updates and deletions of related models, ensuring the integrity and proper management of relationships.

- **PayPal Integration**: Integrated PayPal using its sandbox environment for testing payments, allowing users to make secure transactions for courses within the application.

Challenges

- **State Management in Vue**: Managing the application's state efficiently with Vuex.
- **Responsive UI**: Ensuring that the user interface works well across different screen sizes and devices.
- **Optimized Eloquent Relationships**: Improving database queries and relationships in Laravel's Eloquent ORM for better performance.
- **PayPal Integration**: Integrating PayPal for payment processing, including handling transactions and testing with the sandbox environment.

PAM (Tiny PHP MVC Framework)

A custom lightweight PHP MVC framework inspired by Laravel, designed to implement clean architecture principles and simplify web development.

Key Features

- **Database & Models**: Integrated Eloquent ORM for seamless database operations.
- **Routing System**: Dynamic routing with middleware support and controller binding.
- **Templating**: Implemented Blade templating engine for rendering dynamic views.
- **Session Management**: Secure session handling and state management.
- **File Storage**: Centralized file system with configuration-driven storage capabilities.
- **Helper Functions**: Utilities like `view()`, `router()`, `storage()` for efficient coding.

Implementation Highlights

- **Bootstrap Process**: Developed a centralized `bootstrap/app.php` file to handle application loading, including environment variables and instance bootstrapping.
- **Middleware Pipeline**: Built a global and route-specific middleware pipeline to filter and validate requests before routing them.
- **Configuration Management**: Centralized configuration files for database, session, and file handling for enhanced maintainability.
- **Helper Utilities**: Created custom helper functions for common tasks, streamlining the developer experience.

Outcome

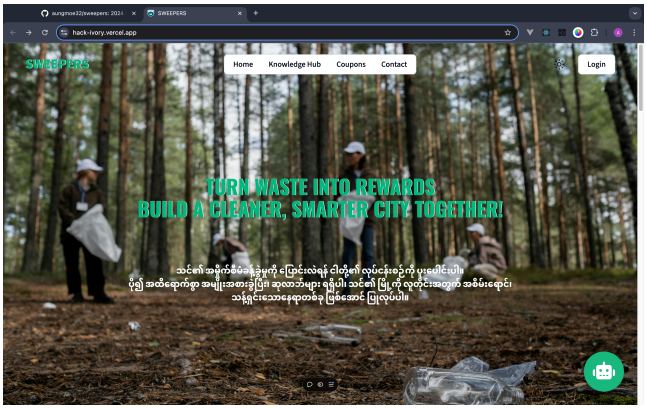
Demonstrated PAM's capabilities by developing a fully functional **Todo List App**. The project showcased the framework's ability to handle routing, database operations, session management, and Blade templating with ease.

- Demo : YouTube
- Github Repo : https://github.com/aungmoe32/pam

Technologies Used

- PHP, Illuminate Components (Eloquent, Blade, Filesystem), Composer

Sweepers



Sweepers is a waste management web app designed to encourage eco-friendly practices by integrating AI-powered hardware with a user-friendly platform. It features a smart bin equipped with a camera sensor that identifies waste types and weights, generating coupons redeemable for reward points. These points can be exchanged for cash, promoting sustainable waste disposal. The app also includes educational waste articles, a chatbot for assistance, and responsive design, making it accessible across devices.

This is the project that competed in the Hackathon at the 100th Anniversary Engineering Festival at NSPU.

Source Code

- Github repo : https://github.com/aungmoe32/sweepers

Demo

- App url : https://hack-ivory.vercel.app/
- Video : YouTube

Features

- User registration and mock authentication.
- AI-powered hardware bin with a camera sensor for waste detection by type and weight.
- Generation of 4-digit coupons on the bin's LCD screen.
- Redeem coupons for reward points, exchangeable for cash, based on waste type and weight.
- Integrated chatbot for user assistance.
- Waste articles section for educational content.
- User profile editing.
- Reward exchange form for using coupons.
- Fully responsive design for seamless usage on all devices.

Tech Stack

- **Frontend Framework**: Next.js
- **UI Components**: Shadcn
- **Styling**: TailwindCSS

Implementations

- AI-powered waste type and weight detection using a hardware-integrated bin.
- Real-time coupon generation and validation on the app.
- Points calculation logic based on waste properties.
- Responsive UI/UX for cross-platform compatibility.
- Chatbot integration for enhanced user interaction.
- Mock authentication for demonstration and testing.

Hobbies & Interests

- Exploring music through fingerstyle guitar.
- Enjoying social gaming experiences with friends.
- Building meaningful connections by meeting new people.

Conclusion

“Looking forward to contributing my skills and learning new technologies in a professional environment.”