Progress Report 1	**300392818**
CSIS 4495	**Vidarshan A**

Summary

On the mobile and **frontend** side, I established a clean and scalable React Native (**Expo**) architecture. This included setting up a **global styling** system to reduce duplication, creating reusable UI components such as buttons, spacers, loaders, alerts, and typography components, and configuring consistent form controls to ensure platform **consistency** across iOS and Android. I also implemented **navigation** using Expo Router, resolved structural issues related to **layout** and routing, configured **custom fonts** to match client branding, and updated the UI to align with the latest iOS design guidelines. Several runtime and stability issues were addressed, including **SDK upgrades** and fixing infinite render.

On the **backend**, I designed and implemented the appointment **scheduling API** using **Next.js** and **Prisma**. This involved defining the appointment data model, handling appointment creation, updates, rescheduling, cancellation, and completion flows, and ensuring type safety throughout the API using **TypeScript**. I resolved **multiple Prisma migration and schema drift issues**, established correct handling of decimal values for pricing, and verified database state using **PostgreSQL** tools. A strong emphasis was placed on correct date and time handling by standardizing all appointment timestamps to UTC using **TIMESTAMPTZ** to avoid timezone and daylight-saving issues.

A major portion of my work focused on the appointment reminder and notification system. I designed and implemented a reliable, stateless reminder architecture based on **cron** jobs and database state rather than in-memory background jobs. Appointment reminder logic was implemented using day-based matching (e.g., 5-day and 1-day reminders) to prevent missed notifications due to timing drift. Reminder state flags were added to ensure idempotency and safe retries. I also implemented completion-based email notifications and ensured reminders reset correctly when appointments are rescheduled.

For email delivery, I evaluated transactional email services and integrated an email provider suitable for serverless environments. I created HTML email templates for appointment reminders and completion notifications and handled development-time constraints such as sender domain verification. I also implemented strategies to test email sending without a production domain.

In terms of testing and validation, I established a structured approach to testing appointment reminders by **injecting fake "current time" values**, **manually triggering cron endpoints**, verifying database state transitions, and ensuring duplicate emails are not sent. This allowed end-to-end verification of the scheduling and notification pipeline without waiting for real-time delays.

Changes to the original proposal

Since the original proposal, the project has evolved from a **Redis-based background job architecture** to a **cron-driven,** database state based scheduling system better suited for serverless environments. Date handling has been standardized using **UTC and TIMESTAMPTZ**, reminder logic has been refined to day-based matching to prevent missed notifications, and the backend has been implemented using **Next.js API routes with Prisma instead of a standalone Express server**. Additional practical considerations such as email domain verification, idempotent cron execution, and structured testing workflows were introduced during implementation, while remaining fully aligned with the original functional and non-functional requirements outlined in the proposal

Work Logs

Date	Date	Description of work done	Hours
20th January 2026	1:30pm-2:00pm	Had the initial meeting with Stephanie Riddel to understand the scope of the Riipen project.	0.5
22nd January 2026	9:30pm-11:30pm	Started research on technical feasibility upon discussing the requirements on a Teams meeting on 20th Tuesday.	2
23rd January 2026	5:30pm – 7:00pm	Setup the template for the word document (Proposal)	1.5
24th January 2026	6:30pm – 8:30pm 9:30pm – 10:30pm	Research + setup development tools needed for the mobile app including finding resources to learn technologies if there is a skill gap. These findings will be communicated to Upul as the development commences.	3
25th January 2026	12:00am – 1:30am	Compile and insert the information gained from the research of technologies.	1.5
27th January 2026	4:30pm-6:30pm	Implemented user get, edit, delete, create functions on the API	4
	8:30pm-10:30pm		
03rd February 2026	10:30pm-1:30pm	Connect mobile app to the web API	9.75
	1:30-2:30pm	Meeting with client to discuss the requirements (1h)	
	4:00-6:45pm	Persisting logged in status on React Native	
	8:30-11:30pm	Implemented authenticated navigation flow and rectify navigation control issues	
04th February 2026	4:30-6:30pm, 8:30-10:30pm,	Implemented appointments endpoint Send emails through cron jobs	4
05th February 2026	4:30-7:00pm	Implementing dashboard UI	2.5

Total		

AI Usage

AI		Prompt	Value Addition
ChatGPT	5.2, Plus	Research on scheduling related frameworks and utilities for Node JS	Improved scheduling system over a traditional cron job system.
ChatGPT	5.2, Plus	RN global style system	Reduced styling repetition across the app
ChatGPT	5.2, Plus	Styling form controls in React Native	Ensured consistent UI across iOS and Android
ChatGPT	5.2, Plus	RN reusable button component	Standardized button behavior and styling
ChatGPT	5.2, Plus	RN Spacer component	Avoided manual margin and padding usage
ChatGPT	5.2, Plus	RN title / text components	Removed redundant typography styling
ChatGPT	5.2, Plus	Expo navigation stack setup	Enabled navigation between screens
ChatGPT	5.2, Plus	Expo Router vs React Navigation	Established a single routing system
ChatGPT	5.2, Plus	Is it necessary to have _layout.tsx and index.tsx	Simplified and cleaned application structure
ChatGPT	5.2, Plus	Tab bar navigation for authenticated users	Controlled navigation based on authentication state
ChatGPT	5.2, Plus	Preserve login tokens in storage	Persisted login state across app restarts
ChatGPT	5.2, Plus	Bottom sheets for React Native Expo with gestures	Improved mobile user experience
ChatGPT	5.2, Plus	Avoid infinite loop in provider useEffect	Fixed application crash
ChatGPT	5.2, Plus	Upgrade Expo SDK to 55	Resolved runtime crashes and compatibility issues
ChatGPT	5.2, Plus	React Native list rendering	Improved performance when rendering task lists
ChatGPT	5.2, Plus	Configure fonts in Expo	Adapted typography to client branding
ChatGPT	5.2, Plus	iOS 26 app bar guidelines	Updated UI to match latest platform standards
ChatGPT	5.2, Plus	RN reusable alert component	Matched alert behavior between web and mobile apps
ChatGPT	5.2, Plus	RN loader / loading indicator component	Standardized loading states across screens
ChatGPT	5.2, Plus	TypeScript type for API error object	Improved error handling consistency
ChatGPT	5.2, Plus	Prisma schema design for appointments	Structured appointment lifecycle data
ChatGPT	5.2, Plus	Prisma migration to add multiple columns	Safely evolved database schema
ChatGPT	5.2, Plus	Prisma migration drift resolution	Fixed schema and migration mismatch
ChatGPT	5.2, Plus	Viewing PostgreSQL tables using psql	Verified database state and migrations
ChatGPT	5.2, Plus	Decimal vs number handling in Prisma	Prevented precision loss and type errors
ChatGPT	5.2, Plus	TIMESTAMPTZ vs TIMESTAMP in PostgreSQL	Ensured timezone-safe date storage
ChatGPT	5.2, Plus	UTC date storage strategy	Prevented timezone and DST-related bugs

ChatGPT	5.2, Plus	Create appointment API endpoint in Next.js	Implemented clean appointment creation logic
ChatGPT	5.2, Plus	Appointment rescheduling logic	Correctly reset reminder states on changes
ChatGPT	5.2, Plus	Appointment completion workflow	Triggered post-appointment email logic
ChatGPT	5.2, Plus	Cron-based appointment reminder system	Automated reminder email delivery
ChatGPT	5.2, Plus	Day-based reminder window logic	Prevented missed reminders due to timing mismatch
ChatGPT	5.2, Plus	Cron job design	Avoided duplicate emails on repeated runs
ChatGPT	5.2, Plus	Reminder flag architecture	Enabled safe retries and fault tolerance
ChatGPT	5.2, Plus	Cron setup in Next.js App Router	Enabled background automation in a serverless environment
ChatGPT	5.2, Plus	Vercel cron configuration	Enabled scheduled execution in production
ChatGPT	5.2, Plus	Testing cron jobs with fake dates	Allowed fast local testing without waiting real days
ChatGPT	5.2, Plus	Email service comparison for transactional emails	Selected Resend for reliable email delivery
ChatGPT	5.2, Plus	HTML email template design	Created branded and client-facing reminder emails
ChatGPT	5.2, Plus	Handling Resend domain verification errors	Identified correct approach for email sender setup
ChatGPT	5.2, Plus	Sending emails without a domain during development	Enabled continued development without production domain
ChatGPT	5.2, Plus	End-to-end appointment reminder testing	Validated full automation pipeline
ChatGPT	5.2, Plus	Appointment and reminder system architecture explanation	Designed a stateless, scalable, and maintainable system
ChatGPT	5.2, Plus	Step-by-step implementation plan for appointment reminders	Reduced integration errors and development time

Repo Check In

Work on the repository was carried out on a regular basis, aligned with the project timeline and development milestones. Development activities included frontend UI components, backend API implementation, database schema evolution, and automation logic.

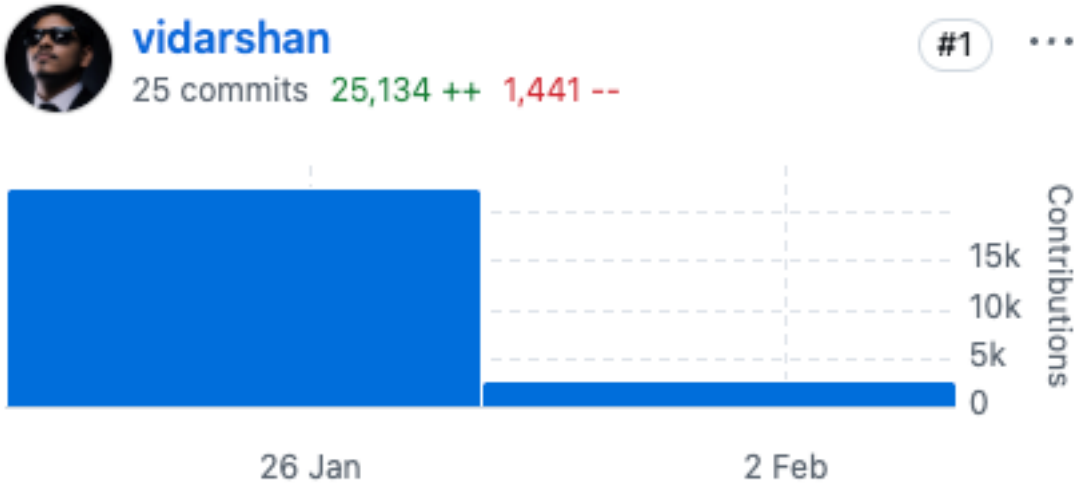
Majority of the work for the **eco-clean-web and eco-clean-mobile** was committed through the branch **web-api**, through multiple pull requests.

PR #1: https://github.com/UpulAtapattu/W26_4495_S2_UpulA/pull/1

PR #2: https://github.com/UpulAtapattu/W26_4495_S2_UpulA/pull/2

PR #3: https://github.com/UpulAtapattu/W26_4495_S2_UpulA/pull/3

PR #4: https://github.com/UpulAtapattu/W26_4495_S2_UpulA/pull/4



The work was checked into the repository almost every day, followed by a merge to main almost every 4 days.