Module #2 Report | CSE 310 – Applied Programming

Name	Date	Teacher
Moroni Motta	11/05/25	Bro McGary

Project Repository Link

Copy the link to your Github Repository

Module

Mark an X next to the module you completed

Module		Language
Cloud Databases		Java
Data Analysis		Kotlin
Game Framework		R
GIS Mapping		Erlang
Mobile App		Flutter
Networking		C#
Web Apps	X	Go
Language – C++		Rust
SQL Relational Databases	X	PostgreSQL

Fill Out the Checklist

Use this list to verify the module requirements were met.

Requirement	Your Response	Comments
Implemented a backend API for message delivery (email/SMS)	YES	Built with Go, Gin, GORM
Integrated with external providers (Twilio, SendGrid, Ntfy)	YES	Handlers and factories for each provider
Webhook endpoints implemented and tested	YES	SendGrid & Twilio webhook handlers; ngrok used for testing
Database schema with entities and relationships	YES	Entities: User, Plan, Integration, Message, MessageStatus

Requirement	Your Response	Comments
Repository, Usecase, and Handler layers implemented (clean architecture)	YES	Promotes separation of concerns and easy extension
README.md and example dashboard for manual testing	YES	Dashboard.html for sending messages and viewing statuses

Stretch Challenge (selected one)

- Implemented a small example dashboard and status priority system to surface the most relevant message status (error > delivered > sent > queued).

Record your time

How many hours did you spend on this module and the team project this Sprint?
Include all time including planning, researching, implementation, troubleshooting, documentation, video production, and publishing.

	Hours
Individual Module	12
Team Project	4

Retrospective

- What learning strategies worked well in this module?
    - Using a clean architecture (entities, repositories, usecases, handlers, factories) was especially effective. It made integrating new external APIs (SendGrid, Twilio, Ntfy) fast and low-risk because each provider implements a small, well-defined interface.
    - Iterative testing with ngrok for webhook development allowed rapid end-to-end verification of webhook flows.
- What strategies (or lack of strategy) did not work well?
    - Time estimation for integration-heavy tasks was optimistic. Handling provider-specific quirks (for example, SendGrid's different message ID formats and HTTP redirects) required more debugging and adjustments than initially planned.
    - Redirects and webhook routing (Gin's default trailing-slash/redirect behavior) caused unexpected 307 redirects during POST webhook delivery and consumed extra time to diagnose and fix.
- How can you improve in the next module?
    - Build in buffer time for third-party integration and webhook testing (add ~30–40% slack to estimates).
    - Create small, automated integration tests that validate both the send flow and webhook processing (use recorded or stubbed webhook payloads). This will reduce manual debugging

time.
- Use ngrok or a persistent tunnel in combination with detailed request logging early in the process so webhook routing issues are discovered sooner.