Semester-1_Final_Stack

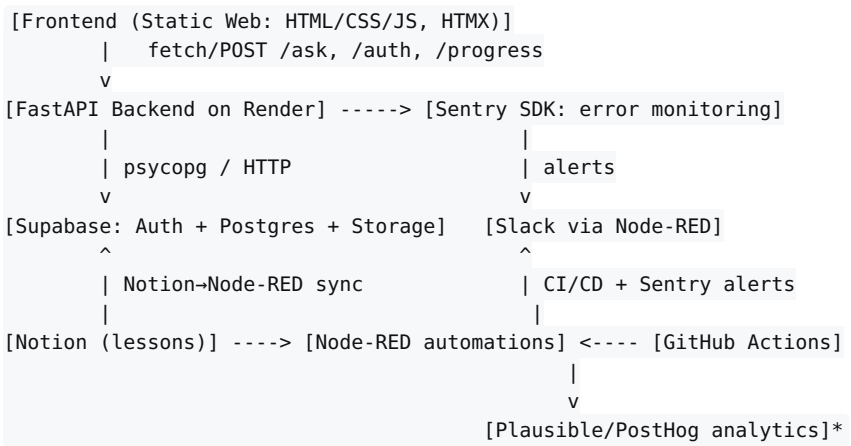
Semester 1 — Final Stack (Avatar-AI-Educator)

Date: 2025-08-10

Purpose

This document is a print-friendly map of the full toolchain for Semester 1. It shows what each tool does, how they connect, the deployment flow, required environment variables, and a setup checklist. Keep this as your desk reference while you build.

High-Level Architecture



*Analytics optional for Semester 1, recommended if you want usage data from day

one.

Tools and Roles

- **GitHub** – Source control, Issues, Pull Requests, GitHub Actions CI.
- **ChatGPT Plus** – On-demand architecture help, code explanations, prompts.
- **GitHub Copilot** *(optional but recommended)* – Inline coding assistance in Codespaces.
- **Replit** – Quick prototypes/scratch experiments outside the main repo.
- **Node-RED** – Automation glue (Notion → Supabase sync; Slack notifications).
- **Slack** – Notifications (CI status, deploys, errors).
- **Supabase** – Auth (email magic link), Postgres DB, optional file storage.
- **Pytest** – Unit/integration tests (API handlers, quiz grading).
- **Flake8** – Linting/style checks for Python backend.
- **Prettier** – Formatting for HTML/CSS/JS/JSON.
- **Sentry** – Runtime error tracking for FastAPI.
- **Render** – Hosting for FastAPI (web service) + Static Site for frontend.
- **Dependabot** – Automated dependency update PRs in GitHub.
- **Plausible/PostHog** *(optional)* – Privacy-friendly analytics.

CI/CD Pipeline (One Workflow)

1. **Prettier Check** (Node 20): `npm ci` → `npm run format:check`
2. **Python Lint/Tests** (Py 3.11): `flake8 .` → `pytest -q`
3. **Deploy (optional)**: Trigger Render deploy on `main` after both checks pass.

File: `.github/workflows/ci.yml`
- Guard deploy with repo variable `RENDER_DEPLOY=true` and secrets `RENDER_API_SERVICE_ID`, `RENDER_API_KEY`.

Environment Variables (Render → Backend Web Service)

- `SUPABASE_URL` – Your Supabase project URL
- `SUPABASE_ANON_KEY` – Public anon key (client) or service role in server context
- `LLM_PROVIDER` – e.g., `groq` or `openai`
- `LLM_API_KEY` – Key for the selected provider
- `SENTRY_DSN` – Sentry project DSN (optional until enabled)

GitHub Actions Variables/Secrets (if using optional deploy): - Variable: `RENDER_DEPLOY = true` - Secrets: `RENDER_API_SERVICE_ID`, `RENDER_API_KEY`

Database Schema (Supabase)

Key Tables

- **profiles** (id, username, created_at)
- **courses** (slug, title, lang, is_published)
- **lessons** (course_id, slug, title, body_md, quiz JSON)
- **progress** (user_id, lesson_id, completed, score, updated_at)
- **chat_logs** (user_id, lesson_id, role, content, created_at)

Repository Structure

```
avatar-edu/
├── app/
│   ├── __init__.py
│   ├── main.py          # FastAPI routes (/ask, /health, etc.)
│   └── llm.py           # Provider switch: Groq/OpenAI
├── web/
│   └── index.html       # HTMX + TTS avatar front end
├── requirements.txt
├── Procfile             # uvicorn start command for Render
├── render.yaml          # blueprint for Render services
├── .prettierrc          # frontend formatting rules
├── .prettierignore
├── .editorconfig
├── .gitattributes
├── .gitignore
└── .github/workflows/ci.yml
```

Setup Checklist (Quick)

1. **Repo & Codespaces**
   - Add `.gitignore`, `.editorconfig`, `.gitattributes`.
   - Install Prettier: `npm i -D prettier`; add scripts `format`, `format:check`.
2. **Backend**
   - `pip install fastapi uvicorn[standard] sentry-sdk`
   - Create `app/main.py` + `app/llm.py` scaffolds.
3. **Frontend**
   - Create `web/index.html` with HTMX + browser TTS.
4. **Supabase**
   - Create project; run schema; configure Auth.
5. **CI**
   - Add `ci.yml` (Prettier + flake8 + pytest); push.
6. **Render**
   - Deploy FastAPI Web Service + Static Site; set env vars.
7. **Observability/Automation**
   - Add Sentry DSN (optional), Node-RED flows for Notion sync + Slack alerts.
8. **Dependabot**

    - Enable in GitHub → Security → Code security and analysis.

Security & Secrets

- Never commit API keys; store them in **Render Env Vars** or **GitHub Secrets**.
- Restrict Supabase policies (RLS) as you expose tables to the client.
- Rotate keys periodically; keep `.env` and `.env.*` in `.gitignore`.

Success Criteria (Semester 1)

- Deployed app: static frontend + FastAPI API on Render
- Supabase Auth working; lessons and progress stored
- LLM Q&A endpoint functional
- CI passes: Prettier + flake8 + pytest
- Sentry capturing errors from production
- Node-RED pushes deploy/test alerts to Slack
- Readme explains setup in under 5 minutes

Notes

- Keep **Analytics** optional until you want usage insights.
- Add **tests** incrementally (quiz grader, lesson CRUD, auth guard).
- Use **Copilot** to accelerate boilerplate, but review diffs carefully.