# SkillSync — Agentic Career Mentor (Day‑by‑Day Build Guide)

*A beginner‑friendly, code‑free process plan you can follow step by step. Stack: Python + Flask + CrewAI + Streamlit (UI) + MongoDB Atlas. Free hosting where possible.*

## 🍀 Project Scope (Finalized)

* **Agentic (not a chatbot):** remembers user state, plans goals, adapts daily, acts proactively (reminds, replans, nudges), and updates databases without being asked.
* **Core features now**
  1. Google or Email login (MongoDB‑backed)
  2. Profile intake (skills, goal role e.g., *Java Developer*, availability)
  3. **Roadmap builder** (tickable checklist; progress %; charts)
  4. **Adaptive planner** (if you miss a day → auto shift & re‑balance)
  5. **YouTube playlist → day‑wise plan** (based on your schedule)
  6. Notifications: **in‑app + email** (free) and **WhatsApp demo** (jugaad or trial)
  7. Optional **voice input** for marking tasks done / adding notes
  8. **Admin panel** (basic): view users, broadcast notice, manage templates
* **Future Phase**: **Job Aggregation & Redirection** (show curated/free jobs from external platforms once a user completes a roadmap; redirect to apply)

## 🧠 Agent Roles (CrewAI)

* **Profile Analyzer Agent**: normalizes skills, detects gaps for the chosen role
* **Roadmap Generator Agent**: creates a sequenced plan (weeks/days), maps YT videos/resources to days
* **Progress Orchestrator Agent**: listens to events (done/missed), replans tasks, updates calendar & notifications
* **Notification Agent**: prepares daily nudges (in‑app, email; optional WhatsApp) > Tip: Start rule‑based (no LLM), then switch to CrewAI + LLM once core flows work.

## 🗂️ Data Model (MongoDB Atlas)

**users**

{  
 "\_id": "ObjectId",  
 "auth\_provider": "google|email",  
 "email": "samiya@example.com",  
 "name": "Samiya",  
 "goal\_role": "Java Developer",  
 "skills": ["Java", "OOP"],  
 "availability": {"weekdays\_hours": 2, "weekend\_hours": 4},  
 "created\_at": "ISODate",  
 "last\_login": "ISODate"  
}

**roadmaps**

{  
 "\_id": "ObjectId",  
 "user\_id": "ObjectId",  
 "title": "Java Developer Roadmap",  
 "start\_date": "2025-08-22",  
 "plan": [  
 {"day": 1, "items": [{"id": "v1", "text": "Intro to Java", "yt\_url": "...", "done": false}]},  
 {"day": 2, "items": [...]}  
 ],  
 "progress\_pct": 18.0,  
 "last\_replanned": "ISODate"  
}

**events** (audit trail for agent decisions)

{  
 "\_id": "ObjectId",  
 "user\_id": "ObjectId",  
 "type": "DONE|MISSED|REPLAN|NOTIFY",  
 "payload": {"day": 3, "reason": "busy"},  
 "timestamp": "ISODate"  
}

**notifications**

{  
 "\_id": "ObjectId",  
 "user\_id": "ObjectId",  
 "channel": "inapp|email|whatsapp",  
 "message": "Today: Watch Java OOP video (12m)",  
 "status": "queued|sent|read",  
 "scheduled\_for": "ISODate",  
 "created\_at": "ISODate"  
}

**admin** (optional config)

{  
 "\_id": "ObjectId",  
 "type": "template|playlist|job\_link",  
 "label": "Java Beginner Playlist",  
 "data": {"urls": ["..."]},  
 "created\_by": "admin@skillsync",  
 "created\_at": "ISODate"  
}

## 🔐 Authentication (MongoDB‑friendly)

* **Option A – Google Login**: Use OAuth2 (Authlib/Flask‑Dance). Store Google sub google\_user\_id + email in users. Maintain a signed session/JWT.
* **Option B – Email/Password**: Store **bcrypt hash** only, never plain passwords. Login returns JWT (stored httpOnly cookie). Password reset by email link.

## 🔔 Notifications (Free‑first)

* **In‑app**: badge + feed (read/unread)
* **Email**: free via SMTP (e.g., Gmail app password) for low volume
* **WhatsApp (demo choices)**:
  + *Trial/official:* Twilio/Meta Cloud trial (few messages, safest)
  + *Jugaad:* WhatsApp Web automation script for demo (laptop must stay on)

## 🧭 Replanning Logic (simple first)

1. If user clicks **“I couldn’t study today”** → mark day as MISSED
2. Shift today’s unfinished items to next available day
3. Recompute progress\_pct = done\_items / total\_items
4. Notify: “I’ve moved today’s tasks to tomorrow. New finish date: …”

## 🧩 YouTube Playlist → Day‑wise Plan

* Input: playlist URL(s) OR paste list of video URLs
* Parse titles + durations (manual entry allowed at MVP)
* Use availability to pack ~X minutes per day (e.g., 120m weekdays)
* Save to roadmaps.plan[]

# 📆 Day‑by‑Day Build Plan (12 Days)

### **Day 1 — Project Kickoff & Repo**

* Create GitHub repo (MIT license, README skeleton)
* Decide exact MVP scope (login, dashboard, roadmap, replanning, email notif)
* Set up MongoDB Atlas cluster + .env template
* Sketch UI on paper: Login → Dashboard → Roadmap → Notifications → Admin

### **Day 2 — Flask API Skeleton + Atlas**

* Flask app with health route
* Connect to MongoDB (users, roadmaps)
* Create basic users CRUD (create/find by email)
* Save sample documents

### **Day 3 — Authentication**

* Pick **Google OAuth** OR **Email+Password**
* Implement login, logout, protected routes
* Store session/JWT, user doc creation on first login
* Add *Profile* page stub (goal role, availability)

### **Day 4 — Streamlit UI Shell**

* Deploy a minimal Streamlit app (free Streamlit Cloud)
* Pages: Login (link to Flask), Dashboard, Roadmap, Notifications
* Read profile from Flask API; render stub cards

### **Day 5 — Roadmap Builder (manual)**

* Form to add **goal role** and **availability**
* Input playlist or paste multiple URLs
* Generate day‑wise plan (simple packing by minutes)
* Save to roadmaps; render as **tickable checklist** per day

### **Day 6 — Progress & Charts**

* Mark item ✅ done / ❌ not done; auto recompute progress\_pct
* Add simple progress bar + weekly chart
* Insert events for DONE actions

### **Day 7 — Replanning & “I missed today”**

* Button: “Couldn’t study today” → shift items to tomorrow
* Update events with MISSED + REPLAN
* Show toast + new dates

### **Day 8 — Notifications (in‑app + email)**

* Build notifications queue
* Daily job (APScheduler) → enqueue today’s reminder
* Email sender for queued items (SMTP); mark sent
* In‑app feed + read/unread toggle

### **Day 9 — Voice Input (optional MVP)**

* Add **mic button** to mark tasks done / add note
* Easiest path: browser **Web Speech API** in a small HTML widget embedded in Streamlit; send text to Flask
* Store transcript in events

### **Day 10 — Admin Panel (basic)**

* Admin login (env list of admin emails)
* Views: users list, roadmaps count, notifications log
* Manage templates: add/edit *playlist templates* and *message templates*
* Broadcast tool: push a notice to all users (in‑app/email)

### **Day 11 — QA, Seed Data, Demo Script**

* Seed 1–2 example users + roadmaps
* Test flows end‑to‑end (login → plan → miss day → replan → email)
* Write a **demo script** (what you’ll click & say in viva)
* Add screenshots to README; record a 60–90s Loom video (optional)

### **Day 12 — Polish & Submission**

* UI tidy (icons, spacing, empty states)
* Error handling + helpful messages
* Final README: features, stack, how to run, screenshots
* Prepare a one‑pager PDF (problem, solution, agents, architecture)

## 🛠️ Endpoints (design before coding)

**Auth** - POST /auth/signup (email, password) → create user - POST /auth/login → JWT cookie - GET /auth/google/callback → create/find user

**Profile & Roadmap** - GET /me → user + roadmap summary - POST /profile → goal\_role, availability - POST /roadmap/generate → from playlist/urls - PATCH /roadmap/item → mark done/undone - POST /roadmap/missed\_today → replan

**Notifications** - GET /notifications → list - POST /notifications/read → mark read

**Admin** - GET /admin/metrics → counts - POST /admin/template → add playlist/message template - POST /admin/broadcast → send to all

## 🧱 Architecture (high‑level)

Streamlit (UI) ⇄ Flask API (auth, agents, scheduler) ⇄ MongoDB Atlas

CrewAI agents run **inside Flask worker** or a **separate worker** (Celery/RQ optional later). Scheduler (APScheduler) triggers daily tasks and Notification Agent.

## 🧪 Testing Checklist

* Can a new user sign in and create a roadmap?
* Does marking done update progress & charts?
* Does “missed today” replan correctly?
* Do email reminders send and toggle to sent?
* Do admin tools list users and send broadcast?

## 🧭 Future Enhancements (Phase 2)

* **Job Aggregation & Redirection**: after >80% completion, show curated jobs (RSS/APIs where allowed) with **Apply** buttons → redirect externally
* Smarter agents (LLM‑powered sequencing, difficulty pacing)
* Mobile‑friendly PWA; calendar sync (Google Calendar)
* Official WhatsApp Cloud API when budget allows

## 📎 Submission Tips

* Emphasize “**agentic**” behaviors: memory, proactivity, replanning, notifications
* Demo the **missed‑day → auto‑replan** moment; it sells the agent idea
* Keep costs near zero: Streamlit Cloud + Atlas free + SMTP email

### ✅ Deliverables You’ll Have

* Live Streamlit app (free)
* Flask API connected to Atlas
* Working login, roadmap, replanning, notifications
* Admin mini‑panel + README + demo script