Final Workflow: IdeaGenie AI Project Generator (Tech-Mapped)

Tech Stack Breakdown

Functionality	Tool/Library
Full-stack App	Next.js (App Router)
Auth	Firebase Auth
Database	MongoDB + Mongoose
Web Scraping	Cheerio / Puppeteer
Problem Storage (Scraped)	Google Sheets API
Group Skill Extraction	GitHub API
AI Idea Generation	Gemini API
Hosting	Vercel

Flow Structure

Step 1: User/Group Onboarding (Frontend + Backend)

- **Frontend:** Next.js form UI (Team Name, Description, GitHub usernames, manual skills entry)
- **Backend:** API route `/api/group/register`
  - Store team info in MongoDB
  - Use GitHub API to fetch top repos & extract stack via languages/topics
  - Combine with manual skill input
- **Auth:** Firebase Authentication (Google login recommended)

Step 2: Problem Scraping (Backend/Worker Function)

- Use Cheerio or Puppeteer to scrape:
  - Reddit (e.g., r/learnprogramming, r/startups)
  - StackOverflow recent questions
  - GitHub issues (popular repos)
- Extract post title, content, tags, author, emotion cue words
- Use Gemini API to:
  - Regenerate into a concise **Problem Statement**
  - Detect **Emotion** (e.g., frustration, confusion, urgency)

Store Problems To:

1. **MongoDB Collection:** `problems`

```
{
  title: String,
  emotion: String,
  source: String,
  link: String,
  tags: [String],
  generatedAt: Date
}
```

2. **Google Sheets API Integration:**
    - Append each problem into a Google Sheet (`problems-log`)
    - Use `googleapis` npm package to connect and write rows

Step 3: Problem Recommendation to Team

- API Route: `/api/problems/suggest?groupId=xyz`
    - Fetch group tech stack from DB
    - Use Gemini to match problems from MongoDB based on skills + emotion context
    - Return top 10–15 matched problem statements
- **Frontend:** Display problems with:
    - Source
    - Tags
    - Emotion
    - Relevance score

Step 4: User Selects Problems

- Users pick 4–5 problems from the suggestions
- Submit via `/api/ideas/generate`

Step 5: Gemini Project Idea Generation

- Input: Selected problems + group stack
- Output:
    - Project title
    - Problem it solves
    - Features
    - Suggested tools
    - Use case / target user
- Store this idea in `MongoDB > ideas` collection for the team

Step 6: View/Export Project Idea

- Frontend UI to show generated idea with markdown formatting
- Export options:
    - Copy as markdown

    - Download PDF
    - Shareable public page via `slug`

Optional Features

- Add webhook or CRON to auto-scrape problems daily
- Tag problems by domain: health, productivity, fintech, etc.
- Let groups vote internally on their favorite ideas
- Add feature to request collaborators with missing skills

Deployment

- Host everything (Next.js frontend/backend) on **Vercel**
- Store problems in **Google Sheets** + **MongoDB Atlas**
- Run scrapers as Vercel cron jobs or use background runners (e.g., Railway or Google Cloud Functions)