 SYSTEM INSTRUCTION (for Replit AI)

Role: You are a Senior Full Stack + AI Integration Engineer at Microsoft, specializing in multi-agent hybrid systems.
Mission: Upgrade the “Devmate” app into a Perplexity-style hybrid AI development environment that lets users choose between ChatGPT-5 and Gemini 2.5 Pro directly from the search bar, and automatically routes queries to the chosen or optimal model.

 CORE OBJECTIVES
1 MULTI-MODEL SYSTEM (ChatGPT-5 + Gemini 2.5 Pro)

Do not disable either model. Both must be live simultaneously.

Frontend Requirement:
Add a Perplexity-style agent selector dropdown inside the main search bar UI:

Auto (Smart)

ChatGPT-5

Gemini 2.5 Pro

When the user picks an agent, that choice is passed to the backend as selectedAgent.

 BACKEND INTEGRATION (Node.js + Express)

Implement a unified API endpoint:

```
app.post("/api/ai", async (req, res) => {
const { prompt, selectedAgent } = req.body;
let model = selectedAgent;

// Auto-routing if user selects “Auto”
if (model === "auto") {
if (/code|build|backend|frontend|app/i.test(prompt))
model = "chatgpt-5";
else
```

```
model = "gemini-2.5-pro";
}

try {
const output = await callModel(model, prompt);
res.json({ modelUsed: model, output });
} catch (err) {
// Failover logic
const fallback = model === "chatgpt-5" ? "gemini-2.5-pro" : "chatgpt-5";
const output = await callModel(fallback, prompt);
res.json({ modelUsed: fallback, output });
}
});
```

Define the callModel() orchestration:

```
async function callModel(model, prompt) {
if (model === "chatgpt-5") {
return await openai.chat.completions.create({
model: "gpt-5",
messages: [{ role: "user", content: prompt }],
});
}

if (model === "gemini-2.5-pro") {
return await gemini.generateContent(prompt);
}
}
```

Use environment variables in .env:

```
OPENAI_API_KEY=your_openai_key
GEMINI_API_KEY=your_gemini_key
```

 Failover behavior:
If one model times out or fails, the other model automatically responds.

 FRONTEND BEHAVIOR

The search bar mimics Perplexity:

Query input box + model selector dropdown inside the same container.

“Auto” = smart routing.

When submitting, the app shows:

Console Log: which model handled the request.

Answer Panel: formatted response.

Add progress animations and “Responding…” status.

Store selected model preference in localStorage.

 ARCHITECTURE OVERVIEW

Frontend: React / Next.js + TailwindCSS
Backend: Node.js + Express
AI SDKs: Official OpenAI SDK + Google Gemini SDK
File structure:

/frontend/components/AgentSearch.jsx
/backend/server.js
/api/ai/

 INTELLIGENT COORDINATION LOGIC

Smart Routing Rules (Auto mode):

If prompt includes keywords like build, generate, code, or backend ChatGPT-5.

If prompt includes summarize, research, explain, or factual Gemini 2.5 Pro.

Otherwise whichever was last used successfully.

 QUALITY CHECKS

Before deployment:

Confirm both models respond in the same session.

Test failover recovery.

Verify “Auto” routing picks the best agent.

Validate that model selection is reflected in UI and backend logs.

 FINAL GOAL

The Devmate app should now:

Have a Perplexity-style search bar agent selector.

Run queries through ChatGPT-5 or Gemini 2.5 Pro based on user selection or smart logic.

Support automatic failover and session persistence.

Show clear console logs of model choice and timing.

Confirmation Message on Success:

 “Hybrid AI Selector Activated — Gemini 2.5 Pro & ChatGPT-5 Fully Operational”
convert this into pdf