Course Project

Goal

Design, develop and deploy an innovative AI agent that autonomously solves a well-defined niche problem, leveraging state-of-the-art AI models with strong reasoning capabilities and appropriate AI agent architectures.

Requirements

1) Build the agent in an optimized way

Your implementation should be efficient:

- Avoid unnecessary LLM calls.
- Minimize prompt/context size (only what's needed).
- Stay within the project budget.

2) API Endpoints (Required)

Your system must expose the following HTTP endpoints (names must match exactly):

A) `GET /api/team_info`

Returns student details.

- **Purpose:** retrieve student names and emails.

Response format (JSON):

```
{
  "group_batch_order_number": "{batch#}_{order#}", // from
presentation list
  "team_name": "Your Team Name",
  "students": [
    { "name": "Student A", "email": "a@..." },
    { "name": "Student B", "email": "b@..." },
    { "name": "Student C", "email": "c@..." }
  ]
}
```

B) `GET /api/agent_info`

Returns agent meta + how to use it.
Must include:

- description
- purpose
- prompt templates (suggested way to work with the agent)
- prompt examples and full responses

Response format (JSON):

```
{
  "description": "…",
  "purpose": "…", // what this agent purpose
  "prompt_template": {
      "template": "…"
  },
  "prompt_examples": [
    {
      "prompt": "Example prompt 1…",
      "full_response": "Full response your agent returns…"
       “steps”: [full list of steps, see below]
    }
  ]
}
```

C) `GET /api/model_architecture`

Returns the architecture diagram as an image (PNG).

- **Purpose:** retrieve a PNG image of the model architecture.
- The architecture must be clear.
- **All sub-modules / sub-agents names must be consistent** across:
  - the architecture diagram
  - your `steps` logging (see `/api/execute`)
  - any descriptions you provide

Response:

- `Content-Type: image/png`
- Body: the PNG file

D) `POST /api/execute`

This is the main entry point.

- User sends an input prompt.
- Your API returns the agent response + the full traced steps.

Input format (JSON):

```
{
  "prompt": "User request here"
}
```

Response format (JSON) — must match exactly these top-level fields:

```
{
  "status": "ok",
  "error": null,
  "response": "…",
  "steps": []
}
```

If error:

```
{
  "status": "error",
  "error": "Human-readable error description",
  "response": null,
  "steps": []
}
```

`Steps:`

`steps` is an **array** describing every LLM call the agent did in order.

You must include:

- module/submodule name (must correlate to architecture diagram)
- prompt
- response

Required step object schema:

```
{
  "module": "…",// the module name according to your
architecture
  "prompt": {},
  "response": {},
}
```

3) Frontend/GUI (Required)

You must provide a minimal web UI to operate your agent.

GUI Requirements

- A text input (textarea) for entering a prompt/task.

- A **"Run Agent"** button that calls `POST /api/execute`.
- Display the **final agent response** (`response`).
- Display the **full steps trace** (`steps`), including:
  - `module`
  - `prompt`
  - `Response`

Optional (Only if supported by your agent)

- Support back-and-forth interaction (follow-up prompts).
- Display conversation history in the UI.

The UI should be simple and focused on interacting with the agent and inspecting its execution.

Deployment

Deploy your agent on **Render** (https://render.com). Keep your account active until receiving a grade.

Databases

- **Supabase**: primary database.
- **Pinecone**: for embedding / vector DB.

LLM Provider (LLMod.ai)

- Each group must create its own LLMod.ai API key.
  - **Note:** In the LLMod.ai platform, all group members share the same API key, if any member creates or rotates the key, it is updated and will be shown for the entire group.
- Budget: **$13 total**.

Submission Format

Submit your agent's **Render URL** and **GitHub repository link** in the following format:

Render URL: `{your url}`
GitHub Repo URL: `{your url}`

Deliverable & Deadline

Due date: **1/3/2026**

Final Note

These autonomous agents represent state-of-the-art technology with **advanced reasoning** capabilities. As you develop your agent, think about the broader implications and potential applications across various areas of life. Consider what new possibilities can be unlocked by harnessing these cutting-edge tools to solve real-world problems and enhance our daily experiences. Never it was possible to develop a fully autonomous agent so fast and with such quality - It's pretty cool 😊

Good luck!

Idan Hahn