Project - High Level Design

on

Multimodal Manufacturing Creator (ForgeVision AI)

Course Name: Generative AI

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
1.	Bhaskar Sharma	EN22CS301261
2.	Bhavesh Shadija	EN22CS301264
3.	Chitrak Jiyal	EN22CS301303
4.	Darshana Matade	EN22CS301309
5.	Dipanshi Yadav	EN22CS301337

Group Name: 08D3

Project Number: GAI-32

Industry Mentor Name: Mr. Suraj Nayak

University Mentor Name: Prof. Vineeta Rathore

Academic Year: 2025-26

Table of Contents

1. Introduction

1.1 Scope of the Document

This document details the architecture, data flow, and system design of the ForgeVision AI application, a specialized tool for industrial design synthesis using generative AI.

1.2 Intended Audience

- **System Architects:** To understand the decoupled model orchestration.

- **Frontend Developers:** To review Streamlit implementation and CSS injection.

- **DevOps Engineers:** To manage environment variables and API integrations.
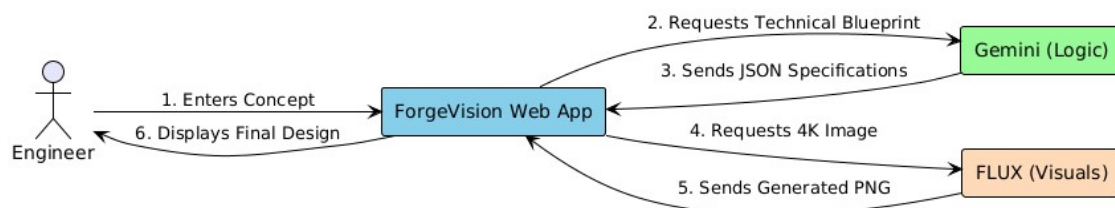
1.3 System Overview

ForgeVision AI is a "Precision Concept Engineering" dashboard. It leverages **Gemini 3 Flash** for technical reasoning and **FLUX.1-schnell** for visual rendering, transforming abstract manufacturing intents into structured engineering specifications and 4K-ready imagery.

---

2. System Design

2.1 Application Design

The application is built on a **Streamlit** framework, utilizing a "Controller-Service" pattern. The UI is heavily customized via CSS-in-JS to provide a high-fidelity industrial aesthetic.

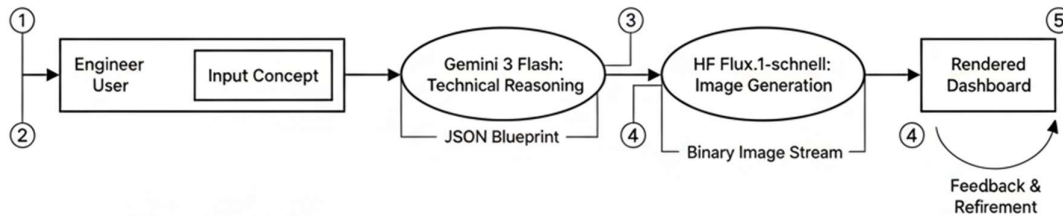**ForgeVision AI - Simple Connection Map**

2.2 Process Flow

1. **Input:** User provides a concept, industry vertical, and aesthetic driver.

2. **Logic Phase:** Gemini processes the prompt and generates a technical JSON schema.

3. **Visual Phase:** The image_prompt from the JSON is sent to the Hugging Face Inference API.

4. **Assembly:** The UI renders the resulting image and technical specs side-by-side.

2.3 Information Flow

The flow is strictly linear-sequential:

User Input -> Logic Engine (JSON) -> Vision Engine (PNG) -> State Storage -> UI Rendering.

2.4 Components Design

- **Sidebar Controller:** Manages global parameters and trigger actions.

- **Concept Workspace:** A multi-tab environment for visualization and blueprints.

- **Logic Engine Wrapper:** Contains robust JSON sanitization logic to ensure API stability.

2.5 Key Design Considerations

- **Resilience:** The system includes a multi-stage JSON parser to handle model formatting inconsistencies.

- **UX:** Mermaid.js is integrated to provide real-time architectural transparency to the user.

2.6 API Catalogue

Provider	Model	Purpose
Google	Gemini 3 Flash	Technical Reasoning & JSON Structuring
Hugging Face	FLUX.1-schnell	Latent Diffusion Image Generation

3. Data Design

3.1 Data Model

The system operates on a transient JSON-based data model:

JSON

```
{
  "name": "string",
  "philosophy": "string",
  "innovations": ["list"],
  "specs": { "materials": "string", "power": "string", "cost": "string" },
  "image_prompt": "string"
}
```

3.2 Data Access Mechanism

Data is fetched via RESTful API calls to Google Generative AI and Hugging Face Inference endpoints using secure environment tokens.
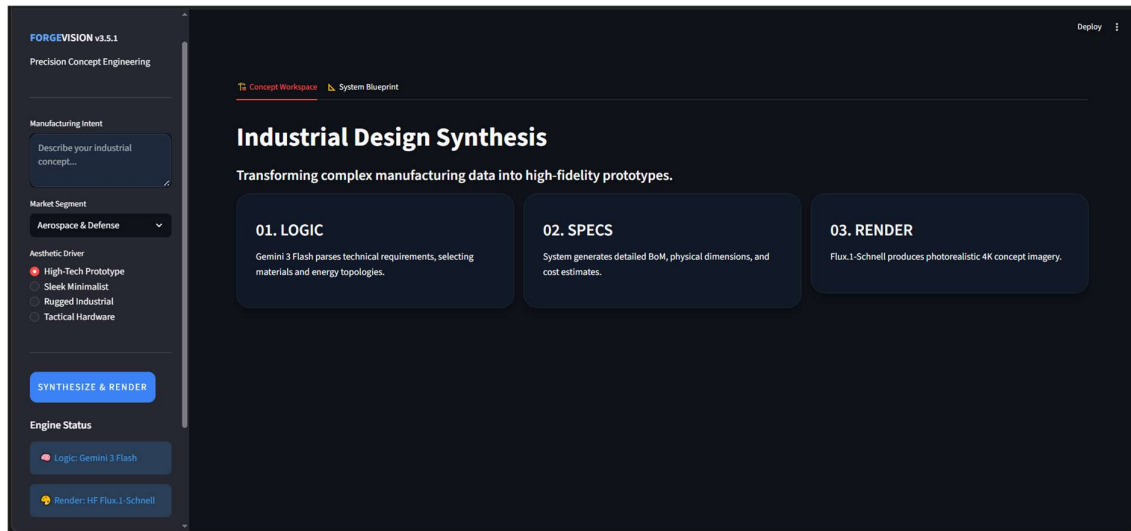
3.3 Data Retention Policies

Currently, the application uses Volatile Retention. Data exists only within the st.session_state and is cleared upon browser refresh or session termination.

3.4 Data Migration

N/A (Stateless application).

---

4. Interfaces

- **GUI:** Streamlit-based web interface.

- **External APIs:**

  - google.generativeai for LLM interaction.

  - huggingface_hub.InferenceClient for image generation.

---

5. State and Session Management

The application utilizes st.session_state to store the result dictionary. This ensures that when the UI reruns (a standard Streamlit behavior), the generated design and image are not lost until a new "Synthesis" is triggered.

---

6. Caching

The application does not currently implement @st.cache_data. However, the persistence of the result in session state acts as a manual cache for the duration of the user's visit.

---

7. Non-Functional Requirements

7.1 Security Aspects

- **Token Management:** API keys are retrieved via os.environ.get, ensuring no secrets are hardcoded.

- **Input Sanitization:** Uses unsafe_allow_html cautiously for UI styling, though user inputs are primarily handled as text data for the models.

7.2 Performance Aspects

- **Concurrency:** The application is synchronous; the user sees a loading spinner during the "Reasoning" and "Rendering" phases.

- **Latency:** Performance is dependent on the response times of the Gemini and Hugging Face remote endpoints.

---

8. References

- [Streamlit Documentation](#)

- [Google Gemini API Guide](#)

- [Hugging Face Inference API](#)