Learn Smart AI

C4 Architecture Documentation

Author: Pranjal Parmar

September 1, 2025

Contents

Introduction

This document provides an architectural overview of Learn Smart AI using the C4 model. The C4 approach helps illustrate how the system works from a high-level context down to its internal components.

1 System Context Diagram

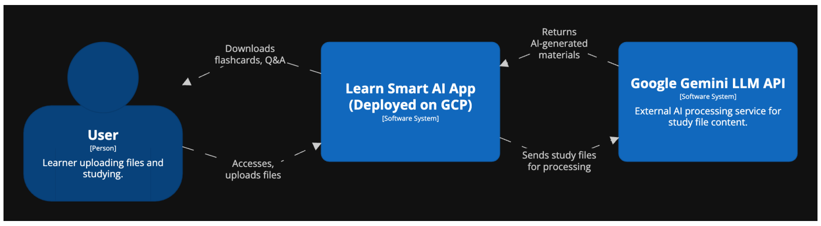

User
[Person]
Learner uploading files and studying.
Downloads flashcards, Q&A
Accesses, uploads files
Learn Smart AI App (Deployed on GCP)
[Software System]
Returns AI-generated materials
Sends study files for processing
Google Gemini LLM API
[Software System]
External AI processing service for study file content.

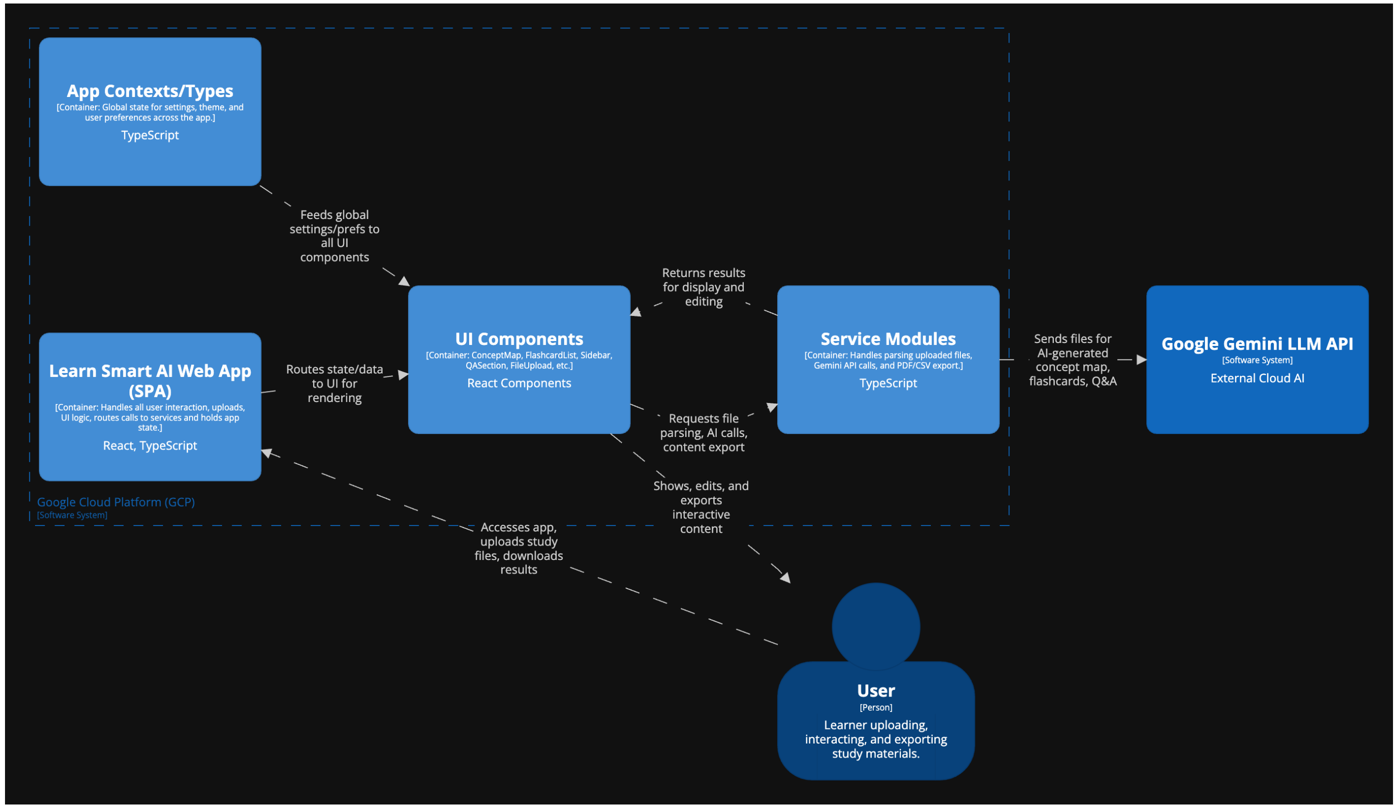
Figure 1: System Context Diagram

The context diagram gives a high-level view of Learn Smart AI. It highlights:

- The **User** (learner or student), who uploads study files, receives flashcards and Q&A, and interacts with the system.
- The **Learn Smart AI App**, deployed on Google Cloud Platform, which acts as the main interface and processor.
- The external **Google Gemini LLM API**, which analyzes files and generates all AI-powered study materials.

This diagram shows the boundaries of the system and the major ways in which users and services interact.

2 Container Diagram



App Contexts/Types
[Container: Global state for settings, theme, and user preferences across the app.]
TypeScript
Feeds global settings/prefs to all UI components
Learn Smart AI Web App (SPA)
[Container: Handles all user interaction, uploads, UI logic, routes calls to services and holds app state.]
React, TypeScript
Routes state/data to UI for rendering
UI Components
[Container: ConceptMap, FlashcardList, Sidebar, QASection, FileUpload, etc.]
React Components
Returns results for display and editing
Requests file parsing, AI calls, content export
Service Modules
[Container: Handles parsing uploaded files, Gemini API calls, and PDF/CSV export.]
TypeScript
Sends files for AI-generated concept map, flashcards, Q&A
Google Gemini LLM API
[Software System]
External Cloud AI
Google Cloud Platform (GCP)
[Software System]
Accesses app, uploads study files, downloads results
Shows, edits, and exports interactive content
User
[Person]
Learner uploading, interacting, and exporting study materials.

Figure 2: Container Diagram

The container diagram breaks Learn Smart AI into its major building blocks:

- **Learn Smart AI Web App (SPA):** Handles user interaction, file uploads, and all core logic. Built with React and TypeScript, running on Google Cloud Platform.
- **UI Components:** React components such as ConceptMap, FlashcardList, and FileUpload, which render the interface and manage user content.
- **App Contexts/Types:** TypeScript modules for managing global settings, state, and user preferences throughout the app.
- **Service Modules:** TypeScript modules responsible for file parsing, calling the Gemini API, and exporting data as PDF or CSV.
- **Google Gemini LLM API:** External service for AI processing of uploaded files.
- **User:** The learner who uploads files, interacts with materials, and downloads study results.

Arrows in the diagram show how data and requests flow between these containers—from file upload all the way to the receipt of AI-powered study tools.

3 Component Diagram

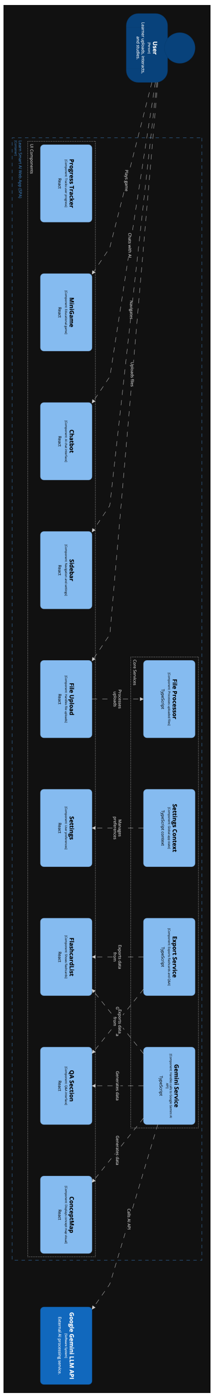

User
[Person]
Learner uploads, interacts, and studies.
Plays game
Chats with AI
Navigates
Uploads files
Learn Smart AI Web App (SPA)
[Container]
UI Components
Progress Tracker
[Component: Tracks user progress]
React
MiniGame
[Component: Educational game]
React
Chatbot
[Component: AI chat interface]
React
Sidebar
[Component: Navigation and settings]
React
File Upload
[Component: Handles file uploads]
React
Settings
[Component: User preferences]
React
FlashcardList
[Component: Shows flashcards]
React
QA Section
[Component: Q&A interface]
React
ConceptMap
[Component: Displays concept map visual]
React
Core Services
File Processor
[Component: Processes uploaded files]
TypeScript
Settings Context
[Component: Global app state]
TypeScript context
Export Service
[Component: Exports flashcards and Q&A]
TypeScript
Gemini Service
[Component: Handles calls to Google Gemini AI API]
TypeScript
Processes uploads
Manages preferences
Exports data from
Exports data from
Generates data
Generates data
Generates data
Calls AI API
Google Gemini LLM API
[Software System]
External AI processing service.

Figure 3: Component Diagram

The component diagram details the core building blocks inside the app:

- **UI Components:** Progress Tracker, MiniGame, Chatbot, Sidebar, FileUpload, Settings, FlashcardList, QA Section, and ConceptMap—all built as React components for interactive user experiences.
- **Core Services:** File Processor, Settings Context, Export Service, and Gemini Service—TypeScript modules that manage parsing, configuration, exporting, and AI-powered file processing.
- **Google Gemini LLM API:** The external AI service handling advanced content analysis and study aid generation.
- **User:** The student interacts with every component, submitting files, playing games, chatting with AI, and tracking progress.

Arrows show the flow of actions and data through the application, from the user's input to processing, AI calls, and back out as interactive study outputs.

4 Code/Class Diagram

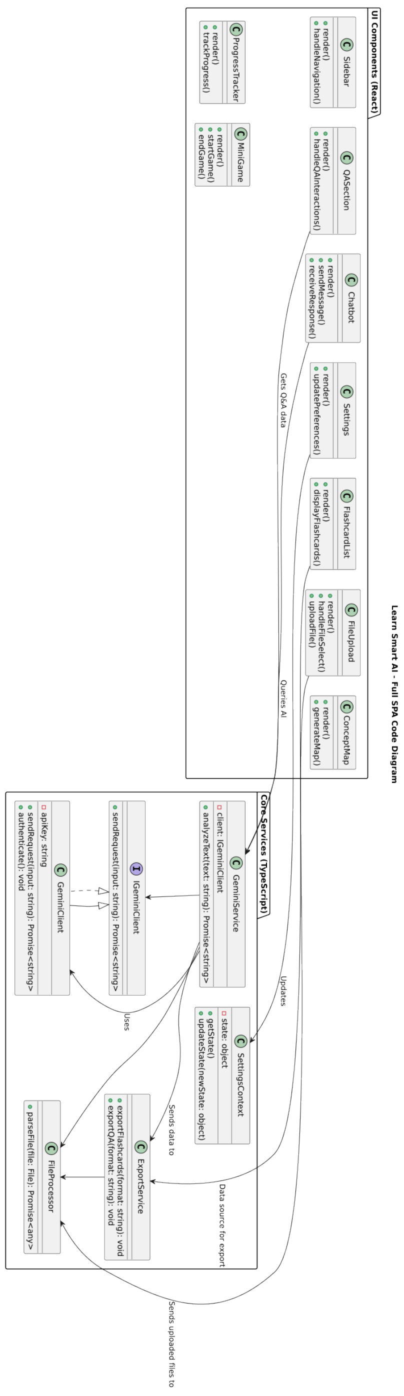


Figure 4: Code/Class Diagram

This code/class diagram reveals the structure of the core codebase:

- **UI Components (React):** Each box represents a key user-facing feature such as Sidebar, QASection, Chatbot, ProgressTracker, MiniGame, Settings, FlashcardList, FileUpload, and ConceptMap, each with crucial methods (e.g., `render()`, `handleNavigation()`, etc).
- **Core Services (TypeScript):** Classes and interfaces such as `GeminiService`, `IGeminiClient`, `GeminiClient`, `SettingsContext`, `ExportService`, and `FileProcessor`. These manage logic for AI communications, file processing, exporting data, and global state.
- Relationships and arrows show how components and services interact, passing data, invoking functions, and keeping the app integrated from user input to data export.

This level of detail helps engineers quickly understand, maintain, and extend the system.

Conclusion

This C4 documentation presents a clear, layered view of the Learn Smart AI system—from its high-level environment and technology choices down to its core components and code structure. These diagrams give developers, stakeholders, and future contributors everything they need to quickly understand how the platform works and where to extend or improve it.

Should the project grow, these diagrams will make onboarding new contributors easier and ensure sustainable, maintainable development.

References

1. Simon Brown, *The C4 Model for Visualising Software Architecture*, `https://c4model.com/`
2. Google AI Studio and Gemini API Documentation: `https://ai.google.dev/`
3. React Documentation: `https://react.dev/`
4. TypeScript Documentation: `https://www.typescriptlang.org/`
5. Structurizr: `https://structurizr.com/`
6. Learn Smart AI project repository (if public): `https://github.com/pranjalparmar/Learn-Smart-AI`