# TestBuddy – Agentic AI-Powered Testing Platform (Next.js + FastAPI Edition)

## 1. Project Overview & Vision

TestBuddy is an agentic AI-powered testing platform built using the Model Context Protocol (MCP). In this updated edition, the frontend is powered by Next.js (hosted on Vercel), and the backend leverages FastAPI (Python) for intelligent orchestration of agents. The platform serves as an intelligent assistant capable of planning, designing, authoring, executing, and analyzing software tests from multiple sources such as documents, Figma designs, and URLs. The aim is to automate the complete testing lifecycle while maintaining human-in-the-loop checkpoints for transparency and control.

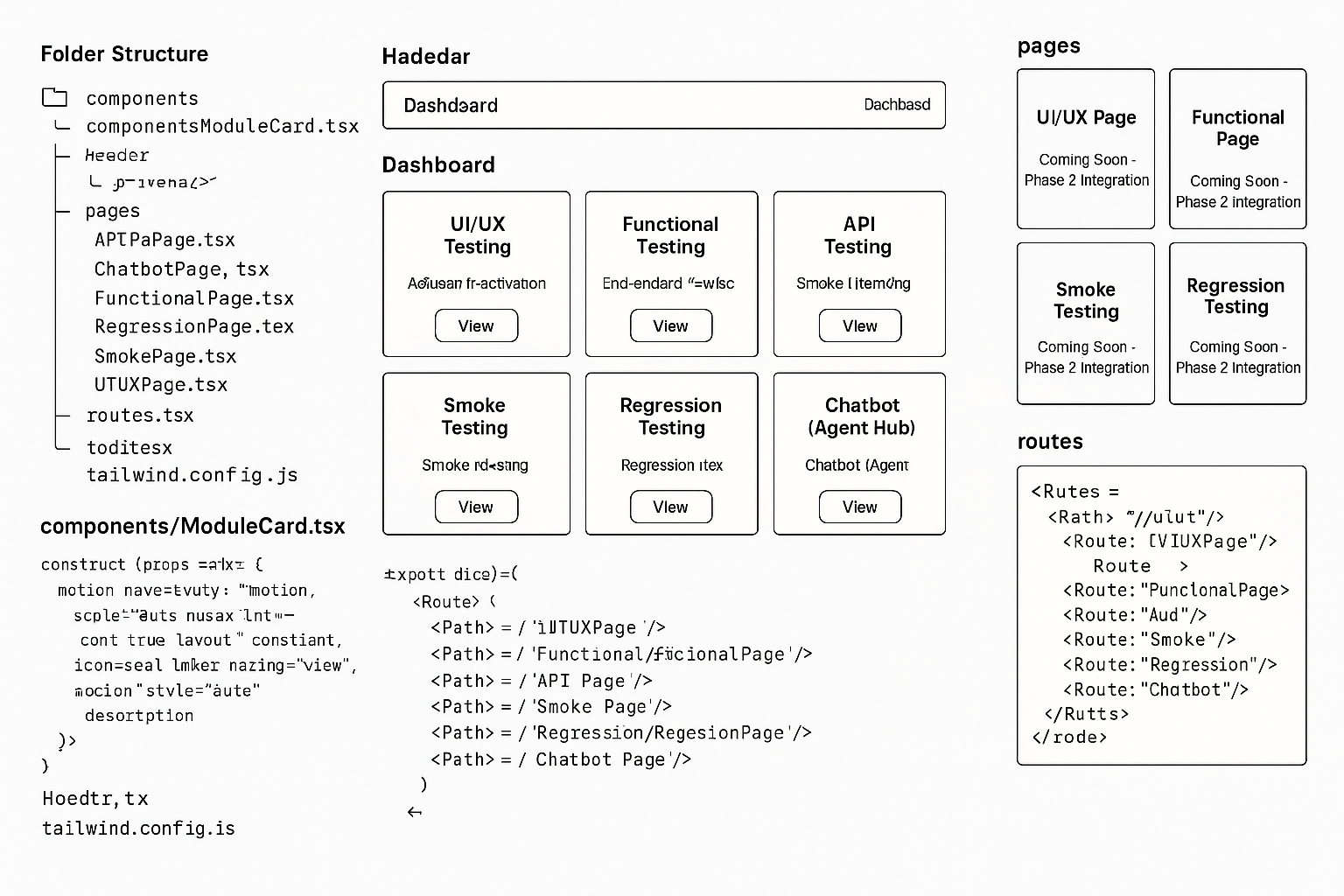
## 2. Tech Stack Summary

|  |  |  |
| --- | --- | --- |
| Layer | Technology | Purpose |
| Frontend | Next.js + Tailwind + shadcn/ui + Framer Motion | Interactive dashboard & ChatDock |
| Backend | FastAPI (Python) | MCP Agent Orchestration and AI logic |
| Database | Supabase / PostgreSQL | Session and user data storage |
| Hosting | Vercel (Frontend) + Render (Backend) | Free-tier scalable deployment |
| AI | OpenAI API | Agentic reasoning and test scenario generation |

## 3. System Architecture

The architecture combines a server-rendered Next.js frontend and a FastAPI backend connected via REST and WebSockets. The backend hosts five core MCP agents — Planner, Designer, Author, Executor, and Curator — each performing a distinct step in the testing lifecycle. The frontend manages user interactions, chat sessions, and human validation checkpoints.

Below is the high-level architecture diagram:



## 4. Multi-Agent Workflow

The workflow remains consistent with the MCP-based agentic design, now orchestrated through FastAPI endpoints:  
1. Planner Agent – Extracts insights from documents, URLs, or Figma inputs.  
2. Designer Agent – Converts plans into structured Gherkin test scenarios.  
3. Author Agent – Generates executable test code.  
4. Executor Agent – Runs tests and streams logs to the frontend via WebSocket.  
5. Curator Agent – Analyzes results, detects flakiness, and refines regression suites.

## 5. Deployment Plan

Frontend: Hosted on Vercel (free tier) – provides SSR, CDN caching, and global delivery.  
Backend: Hosted on Render (FastAPI app) or Railway/Fly.io – supports persistent compute.  
Database: Supabase or Neon (Postgres) – stores sessions, decisions, and chat logs.  
File Storage: Supabase Storage or S3-compatible storage for document uploads.  
CI/CD: GitHub → Vercel (Next.js) and GitHub → Render (FastAPI).

## 6. Deployment Workflow

Deployment pipeline showing flow from code to hosted services:

GitHub → Vercel (Next.js build & deploy) → https://testbuddy.vercel.app  
GitHub → Render (FastAPI deploy) → https://testbuddy-api.onrender.com  
Frontend environment variable: NEXT\_PUBLIC\_API\_BASE=https://testbuddy-api.onrender.com

## 7. Phase-wise Development Plan

The 12-phase roadmap remains unchanged but now explicitly mentions Next.js and FastAPI technologies in the Tech Focus column.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Goal | Key Deliverable / Demo | Tech Focus | Depends On |
| Phase 0 | Setup & Core Skeleton | Repo structure + Docker setup | Next.js + FastAPI + MCP Scaffolding | — |
| Phase 1 | UI Framework + Box Layout | Interactive dashboard (6 modules) | Next.js + Tailwind + shadcn/ui | Phase 0 |
| Phase 2 | MCP Core & Agent Framework | FastAPI base agents | FastAPI + MCP classes | Phase 1 |
| Phase 3 | Document & Figma Ingestion | Upload → Generate Test Plan | FastAPI + Figma API + embeddings | Phase 2 |
| Phase 4 | Scenario Generation | Structured Gherkin scenarios | LLM prompt + MCP workflow | Phase 3 |
| Phase 5 | Test Code Authoring | Cypress/Playwright code | LLM code gen + analyzer | Phase 4 |
| Phase 6 | Test Execution Engine | Run suite + log streaming | Dockerized runners + WebSocket | Phase 5 |
| Phase 7 | Result Analysis & Insights | AI-driven report | FastAPI + OpenAI insights | Phase 6 |
| Phase 8 | Regression Manager & Smart Recommender | Regression suite selector | Stats engine + run history | Phase 7 |
| Phase 9 | Integration Hub | Jira + GitHub + Jenkins connectors | API connectors + MCP providers | Phase 8 |
| Phase 10 | UX/Visual & Accessibility Testing | Visual diff + UX scoring | ImageMagick + custom AI | Phase 9 |
| Phase 11 | Full Agentic Orchestration | End-to-end flow automation | MCP message bus + state manager | All prior |
| Phase 12 | Final Polish & Presentation | Deployed prototype + report | Vercel + Render + visualization | All prior |

## 8. Hosting & Environment Configuration

Vercel:  
 - Connect GitHub repo for frontend auto-deploy.  
 - Environment variable: NEXT\_PUBLIC\_API\_BASE.  
Render:  
 - Deploy FastAPI via GitHub.  
 - Environment variables: OPENAI\_API\_KEY, SUPABASE\_URL, SUPABASE\_KEY.  
Supabase:  
 - Provides free-tier database and storage.

## 9. Future Enhancements

• Plugin ecosystem for integrating additional testing frameworks.  
• Advanced self-healing test maintenance via AI.  
• Predictive analytics for test prioritization.  
• Persistent chat memory for the ChatDock.  
• Support for multi-user collaboration and role-based access.

## 10. Appendices

Includes:

• MCP Architecture Diagram

• Deployment Pipeline Diagram

• API Endpoint Reference (FastAPI)

• Environment Variable Configurations

Test Buddy/

├─ apps/

│ ├─ web/ # Next.js frontend (JS)

│ │ ├─ app/

│ │ │ ├─ layout.js

│ │ │ ├─ page.js # Dashboard landing (Phase 0 health + Phase 1 cards)

│ │ │ ├─ ui-ux/

│ │ │ │ └─ page.js

│ │ │ ├─ functional/

│ │ │ │ └─ page.js

│ │ │ ├─ api-testing/

│ │ │ │ └─ page.js

│ │ │ ├─ smoke/

│ │ │ │ └─ page.js

│ │ │ ├─ regression/

│ │ │ │ └─ page.js

│ │ │ └─ chatbot/

│ │ │ └─ page.js

│ │ ├─ components/

│ │ │ ├─ Header.jsx # Top nav: Dashboard + 6 modules

│ │ │ ├─ ChatDock.jsx # Persistent right-side chat on all pages

│ │ │ ├─ ModuleCard.jsx # Reusable card for each module on dashboard

│ │ │ ├─ ReviewBar.jsx # Fixed bottom bar: Accept / Reject (+note later)

│ │ │ ├─ SourceSelector.jsx # Intake forms (Figma, URL, Docs, Sheet)

│ │ │ ├─ ResultPanel.jsx # Renders outputs (planner/designer/author/...)

│ │ │ └─ DecisionLog.jsx # Timeline of decisions for traceability

│ │ ├─ lib/

│ │ │ ├─ api.js # fetch helpers (uses NEXT\_PUBLIC\_API\_BASE)

│ │ │ └─ session.js # front-end session state (localStorage for now)

│ │ ├─ public/

│ │ ├─ styles/ # keep Tailwind globals here

│ │ │ └─ globals.css

│ │ ├─ tailwind.config.js

│ │ ├─ postcss.config.js

│ │ └─ .env.local # NEXT\_PUBLIC\_API\_BASE=http://localhost:8000

│ │

│ └─ server/ # FastAPI backend (Python)

│ ├─ main.py # FastAPI app + CORS + /health

│ ├─ api/

│ │ ├─ \_\_init\_\_.py

│ │ └─ routers/

│ │ ├─ \_\_init\_\_.py

│ │ ├─ health.py # /health route

│ │ ├─ sessions.py # create session, intake, decisions

│ │ ├─ planner.py # /plan

│ │ ├─ designer.py # /design

│ │ ├─ author.py # /author

│ │ ├─ executor.py # /execute + WS logs

│ │ └─ curator.py # /curate

│ ├─ core/

│ │ ├─ \_\_init\_\_.py

│ │ ├─ models.py # Pydantic models for Session/Steps

│ │ └─ services/ # agent services (stub now, logic later)

│ │ ├─ \_\_init\_\_.py

│ │ ├─ planner\_service.py

│ │ ├─ designer\_service.py

│ │ ├─ author\_service.py

│ │ ├─ executor\_service.py

│ │ └─ curator\_service.py

│ └─ requirements.txt # fastapi, uvicorn, python-multipart

│

└─ .gitignore