# Test Buddy - Phase 1 Summary

## Phase Overview

Phase 1, titled “Functional Core,” focused on establishing seamless communication between the frontend (Next.js) and backend (FastAPI) for functional testing operations.   
This phase laid the foundation for UI–backend interaction by introducing structured API endpoints and dynamic frontend integration.

## Key Objectives

1. Create core backend API routes for the Functional Testing module.  
2. Enable frontend-to-backend connectivity using environment variables and dynamic fetch requests.  
3. Validate and visualize API results in real-time on the frontend.  
4. Deploy and verify communication between Render (backend) and Vercel (frontend).

## Activities Performed

1. \*\*Backend Development (FastAPI):\*\*  
 - Created a new router file `functional\_router.py` in `/apps/server/api/`.  
 - Added two routes:  
 • `/functional/test` – Returns structured dummy test data (passed/failed).  
 • `/functional/plan` – Accepts POST data and returns a draft test plan.  
 - Integrated the router into `main.py`.  
 - Successfully tested API routes via Render deployment.

2. \*\*Frontend Development (Next.js):\*\*  
 - Added a new page `/functional/page.js` to display functional testing UI.  
 - Created `FunctionalPanel.jsx` to fetch and visualize backend data.  
 - Implemented refresh and test plan generation via API POST request.  
 - Updated `ModuleCard.jsx` for navigational routing between modules.  
 - Verified live data rendering from the backend.

3. \*\*Deployment & Environment Setup:\*\*  
 - Configured `NEXT\_PUBLIC\_API\_BASE` in Vercel Environment Variables to point to the Render backend.  
 - Verified connection by visiting:  
 • Backend: https://test-buddy-b110.onrender.com/functional/test  
 • Frontend: https://test-buddy-4ozk.vercel.app/functional  
 - Deployment logs confirmed successful communication between services.

## Verification Results

✅ API connected successfully.   
✅ Frontend displays dynamic test data with status indicators.   
✅ Backend routes are fully functional and responding to live requests.   
✅ CORS configuration verified for cross-platform requests.   
✅ Functional module live and stable on both Render and Vercel.

## Tools & Platforms Used

- Frontend: Next.js (Deployed on Vercel)  
- Backend: FastAPI (Deployed on Render)  
- Version Control: GitHub  
- Communication: REST API over HTTPS  
- Additional Libraries: Tailwind CSS (for styling), Fetch API (for requests)

## Phase Outcome

The Functional Testing module is now fully operational with live API integration.   
The frontend can fetch, render, and interact with backend data in real time.   
This phase establishes the core communication pattern that will be reused across all upcoming modules (UI/UX, API Testing, Regression, etc.).