Test Buddy – Phase 3 Completion Report

Date: October 19, 2025

Prepared for: Test Buddy Project Team

Phase: 3 – Supabase Integration & Backend Validation

# 1. Overview

The primary objective of Phase 3 was to connect the FastAPI backend to a live cloud database using Supabase (PostgreSQL), ensure compatibility with Render hosting, and validate that all API routes successfully read and write data. This phase resolved all prior database connectivity and schema issues, resulting in a fully functional backend ready for frontend integration.

# 2. Architecture Recap

• FastAPI backend hosted on Render.  
• Supabase PostgreSQL as managed database.  
• SQLAlchemy ORM + psycopg3 driver for database communication.  
• Schema auto-generated from `db/models.py` using `Base.metadata.create\_all()`.  
• Environment variables securely managed in Render dashboard.

# 3. Key Fixes and Challenges Solved

|  |  |  |
| --- | --- | --- |
| Issue / Symptom | Root Cause | Resolution |
| 500 errors on /sessions | Old tables lacked required columns | Dropped tables → recreated schema automatically |
| OperationalError: connection failed | Render used IPv6 while Supabase defaulted to IPv4 | Switched to Render-compatible DB endpoint |
| ModuleNotFoundError: api.models | Incorrect import path for Pydantic models | Added dedicated api/models.py |
| Stale DB schema | Migrations from local env out-of-sync | Full schema reset via SQLAlchemy auto-create |
| Ping failures on startup | DB check ran before env vars loaded | Deferred db\_check to runtime |
| UndefinedColumn / Foreign key errors | Missing columns in schema | Fixed SQLAlchemy models and relationships |

# 4. Verification Results

All core API routes were validated successfully through Swagger and Supabase inspection:  
✅ /health and /\_debug/db-ping – returned success  
✅ /sessions – creates new session  
✅ /sessions/{sid} – retrieves session details  
✅ /sessions/{sid}/decision – adds decisions  
✅ /plan – adds plan data  
✅ /execute – queues suite  
✅ /author – saves artifacts  
✅ Tables created: sessions, decisions, plans, artifacts, runs

# 5. Testing Flow (Validated)

1. Create Session – POST /sessions  
2. Add Decision – POST /sessions/{sid}/decision  
3. Add Plan – POST /plan  
4. Add Artifact – POST /author  
5. Execute Suite – POST /execute  
6. Retrieve Summary – GET /sessions/{sid}  
  
All operations confirmed successful, with Supabase rows reflecting expected data.

# 6. Deployment Summary

• Render build and startup successful.  
• psycopg3 driver verified in runtime logs.  
• DATABASE\_URL and SUPABASE\_KEY environment variables confirmed.  
• Uvicorn server active and responding.  
• Swagger UI fully functional.  
• Masked connection format: postgresql+psycopg://<user>:<password>@<host>:5432/<database>

# 7. Recommendations / Next Steps

1. Add Supabase Auth (JWT or email-based login).  
2. Move from auto-create to Alembic migrations for production safety.  
3. Add API key or token-based authentication.  
4. Implement audit logging for API activity.  
5. Begin Phase 4 – Planner/Designer/Author/Executor/Curator flow integration.