

# Personal Assistant App -- Build Log

---

A complete record of all prompts, decisions, commands, and fixes used to build this multi-agent personal assistant from scratch.

---

## Table of Contents

---

1. Project Overview
  2. Phase 1 -- Python Backend
  3. Phase 2 -- Railway Deployment
  4. Phase 3 -- GitHub Setup
  5. Phase 4 -- React Native Mobile App
  6. Phase 5 -- Bug Fixes & Debugging
  7. Phase 6 -- UI Polish
  8. All User Prompts (Chronological)
  9. All Shell Commands Executed
  10. Files Created
  11. Environment Variables Required
  12. Architecture Decisions
- 

## Project Overview

---

Goal: A multi-agent personal assistant accessible from an iOS/Android phone.

Stack:

- AI: Claude Opus 4.6 (Anthropic) with adaptive thinking
  - Backend: Python + FastAPI, deployed on Railway
  - Mobile: Expo (React Native) app tested via Expo Go
  - Storage: Railway persistent volume at /data
  - Web Search: Tavily API (replaced DuckDuckGo)
  - Version Control: GitHub (SSH authentication)
- 

## Phase 1 -- Python Backend

---

### User Prompts

```
"create a folder"
"personal_assistant"
"build a personal assistant app in this folder which is a multi agent app
with orchestrator + sub-agents (web search, calendar, notes, reminders,
memory, file management) and persistent memory"
```

### Files Created

**tools/config.py**

Centralizes data directory paths. Reads DATA\_DIR and WORKSPACE\_DIR from environment variables so Railway can override them to use a persistent volume.

```
def get_data_dir() -> Path:
    env = os.environ.get("DATA_DIR")
    if env:
        return Path(env)
    return Path.home() / ".personal_assistant" / "data"
```

**tools/web\_search.py**

Initial version used duckduckgo\_search. Later switched to Tavily API (see Phase 5).

**tools/calendar\_tool.py**

JSON-backed calendar. Operations: add, list, delete, update events.

**tools/notes\_tool.py**

Markdown files stored in DATA\_DIR/notes/. Operations: create, list, read, update, delete.

**tools/reminders\_tool.py**

JSON-backed reminders with due dates. Operations: set, check, complete, delete.

**tools/memory\_tool.py**

Long-term key-value memory in DATA\_DIR/memory.json. Operations: remember, recall, forget.

**tools/file\_tool.py**

File manager in WORKSPACE\_DIR/. Operations: list, read, write, delete.

**tools/research\_agent.py**

Sub-agent: a separate Claude instance that runs its own web-search agentic loop and returns a synthesized research report. Called via the research\_task tool.

**tools/\_\_init\_\_.py**

Registers all 22 tools and dispatches via execute\_tool(name, inputs).

Full tool list:

```
web_search, add_calendar_event, list_calendar_events, delete_calendar_event,
update_calendar_event, create_note, list_notes, read_note, update_note,
delete_note, set_reminder, check_reminders, complete_reminder, delete_reminder,
remember, recall, forget, list_files, read_file, write_file, delete_file,
research_task
```

**assistant.py**

Orchestrator with two entry points:

- run\_turn() -- CLI mode with Rich console output
- run\_turn\_headless() -- API mode (no console output), returns (reply, history)

Key implementation details:

-

Uses `client.messages.stream()` for real-time streaming

- Full `response.content` (including thinking blocks) preserved during tool use loop
- Explicit API key: `anthropic.Anthropic(api_key=os.environ.get("ANTHROPIC_API_KEY"))`
- Adaptive thinking: `thinking={"type": "adaptive"}`

### main.py

CLI entry point. Runs a REPL loop calling `run_turn()`.

### requirements.txt

```
anthropic>=0.40.0
rich>=13.0.0
python-dateutil>=2.8.0
fastapi>=0.115.0
uvicorn[standard]>=0.30.0
```

## Phase 2 -- Railway Deployment

### User Prompts

```
"how can i host this as an app on my phone"
"Railway"
"lets get started"
```

### Files Created

#### server.py

FastAPI app wrapping the assistant for HTTP access.

```
@app.get("/health")
async def health():
    return {"status": "ok"}

@app.post("/chat", response_model=ChatResponse)
async def chat(req: ChatRequest):
    reply, updated_history = await asyncio.to_thread(
        run_turn_headless, req.message, history
    )
    return ChatResponse(reply=reply, history=updated_history)
```

Uses `asyncio.to_thread` to bridge the synchronous `run_turn_headless` into FastAPI's async event loop.

### Procfile

```
web: uvicorn server:app --host 0.0.0.0 --port $PORT
```

### railway.toml

```
[build]
builder = "nixpacks"

[deploy]
healthcheckPath = "/health"
healthcheckTimeout = 30
restartPolicyType = "on_failure"
```

## Railway Setup Steps

1. Created Railway account and new project
2. Connected GitHub repo (kottesaideep-eng/personal-assistant)
3. Added environment variables (separate Key / Value fields -- not KEY=VALUE format):
  - ANTHROPIC\_API\_KEY -- Anthropic API key
  - TAVILY\_API\_KEY -- Tavily search API key
  - DATA\_DIR -- /data (Railway persistent volume mount)
1. Added a Volume in Railway and mounted it at /data
2. Deployed -- Railway URL: <https://web-production-e4f17.up.railway.app>

## Phase 3 -- GitHub Setup

### User Prompts

```
"kottesaideep-eng"  (GitHub username)
"set up SSH keys"
"kotte.saideep@gmail.com"  (email for SSH key)
"done, now test the SSH connection"
```

### Commands Executed

```
# Generate SSH key
ssh-keygen -t ed25519 -C "kotte.saideep@gmail.com" -f ~/.ssh/id_ed25519 -N ""

# Add to SSH agent
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519

# Display public key (added to GitHub -> Settings -> SSH Keys)
cat ~/.ssh/id_ed25519.pub

# Test connection
ssh -T git@github.com

# Switch remote URL to SSH
git remote set-url origin git@github.com:kottesaideep-eng/personal-assistant.git

# Push
git push origin main
```

## Phase 4 -- React Native Mobile App

### User Prompts

```
"lets get started on step 2" (mobile app)
"make it look better on the phone"
```

### Scaffold Command (final, working)

```
npx create-expo-app@latest mobile-fresh --template blank-typescript
```

This produced React Native 0.81.5, React 19.1.0, Expo SDK 54 -- matching the version installed in Expo Go.

### Files Created

#### mobile-fresh/src/types.ts

```
export interface Message {
  id: string;
  role: "user" | "assistant";
  content: string;
  timestamp: number;
}

export interface HistoryItem {
  role: "user" | "assistant";
  content: string;
}
```

#### mobile-fresh/src/api.ts

```
export async function sendMessage(
  backendUrl: string,
  message: string,
  history: HistoryItem[]
): Promise<{ reply: string; history: HistoryItem[] }> { ... }

export async function checkHealth(backendUrl: string): Promise<boolean> { ... }
```

#### mobile-fresh/src/components/SettingsModal.tsx

Bottom-sheet modal for configuring the Railway backend URL.

Includes a Test Connection button that calls /health.

#### mobile-fresh/src/components/MessageBubble.tsx

Chat bubble component using react-native-markdown-display for rich text.

- User bubbles: blue (#2563eb), right-aligned
- Assistant bubbles: dark slate (#1e293b), left-aligned with robot avatar
- Renders: bold, italic, code, code blocks, lists, blockquotes, links

### mobile-fresh/src/components/ChatInput.tsx

- Animated send button: bounces on press via Animated.sequence
- Button active (blue) vs inactive (grey) based on text content
- Multiline input, max 4000 chars

### mobile-fresh/App.tsx

Main screen with:

- TypingIndicator component -- 3 animated dots using Animated.loop
- Suggestion chip grid on empty chat: "Search the web", "Add calendar event", "Create a note", "Set a reminder"
- Header: robot emoji avatar + title + online/offline status dot
- x button to clear conversation
- ?? button to open Settings
- FlatList auto-scrolls to bottom on new messages
- State: messages (display), apiHistory (sent to backend), backendUrl

### Package Installed

```
npm install react-native-markdown-display
```

## Phase 5 -- Bug Fixes & Debugging

### Bug 1: expo-router Plugin Error

Error: PluginError: Cannot find module 'expo-router/plugin'

Fix: Removed expo-router from plugins array in app.json.

Changed main from expo-router/entry to node\_modules/expo/AppEntry.js.

### Bug 2: Missing Expo Packages

Error: Cannot find module 'expo-asset'

Fix:

```
npm install expo-asset expo-font expo-modules-core
```

### Bug 3: Expo Go SDK Mismatch

Error: SDK 52 app trying to run on Expo Go SDK 54

Fix:

```
npx expo install --fix
# Full reinstall after package.json update
rm -rf node_modules && npm install
```

### Bug 4: Missing babel-preset-expo

Error: Cannot find module 'babel-preset-expo'

Fix:

```
npm install --save-dev babel-preset-expo
```

### Bug 5: TurboModule PlatformConstants Error

Error: Invariant Violation: TurboModuleRegistry.getEnforcing(...): 'PlatformConstants' could not be found

Root Cause: Deep version mismatch between React Native packages

Fix: Scaffolded a completely fresh project:

```
npx create-expo-app@latest mobile-fresh --template blank-typescript
```

### Bug 6: Railway Environment Variable Format

Error: invalid key-value pair '= DATA\_DIR=/data': empty key

Root Cause: User entered KEY=VALUE in a single field instead of separate Key/Value fields

Fix: In Railway dashboard -> Variables -> add Key in one field, Value in another

### Bug 7: ANTHROPIC\_API\_KEY Not Found on Railway

Error: AuthenticationError: API key not found

Root Cause: SDK auto-detection failed on Railway

Fix: Explicitly pass key in assistant.py:

```
_client = anthropic.Anthropic(api_key=os.environ.get("ANTHROPIC_API_KEY"))
```

### Bug 8: DuckDuckGo Search Returns No Results on Railway

Root Cause: Railway server IPs are blocked by DuckDuckGo

Fix: Switched to Tavily API (free tier: 1,000 searches/month)

tools/web\_search.py rewritten to use Tavily:

```
def web_search(query: str, max_results: int = 5) -> str:
    api_key = os.environ.get("TAVILY_API_KEY")
    payload = {"api_key": api_key, "query": query, "max_results": max_results}
    # POST to https://api.tavily.com/search
```

### Bug 9: GitHub Push Authentication Failures

Root Cause: HTTPS tokens had expired or insufficient permissions

Fix: Set up permanent SSH key authentication (see Phase 3)

---

## Phase 6 -- UI Polish

---

### User Prompt

```
"make it look better on the phone"
```

## Changes Made

Component	Before	After
MessageBubble	Plain text only	Markdown rendering (bold, lists, code blocks)
ChatInput	Basic button	Animated bounce, active/inactive states
App	Simple list	Typing indicator, suggestion chips, polished header

## Commit

```
Redesign mobile UI with polished chat interface
- MessageBubble: react-native-markdown-display, shadows, border
- ChatInput: animated bounce on send, active/inactive button states
- App: animated 3-dot typing indicator, suggestion chips, online status dot
```

## All User Prompts (Chronological)

#	Prompt
1	"create a folder"
2	"personal_assistant"
3	"build a personal assistant app in this folder which is a multi agent..."
4	"try it out" (interrupted)
5	"how can i host this as an app on my phone"
6	"Railway"
7	"lets get started"
8	"lets get started on step 2" (mobile app)
9	"kottesaideep-eng" (GitHub username)
10	"it says no repo found" (Railway GitHub access issue)
11	"cant find volume in the tabs"
12	"I FOUND IT" (found Railway volume)
13	"its online" (Railway deployed)



14	"the build shows a failure message"
15	"ERROR: invalid key-value pair '= DATA_DIR=/data': empty key"
16	"it's deployed now"
17	"do i have to purchase some credits, or come back in sometime for it to work?"
18	"ok I've added credits"
19	"it works now"
20	"shows an error cannot find module babel preset expo"
21	"invariant violation: turbomodule registry..." (first occurrence)
22	"it shows a HTTP response error 404: endpoint is offline"
23	"invariant violation: turbomodule registry..." (second occurrence)
24	"in server setting it asks for railway backend url now"
25	"it's connected"
26	"it says web search isn't returning results"
27	"done" (Tavily key added)
28	"push the mobile app changes to github"
29	"set up SSH keys"
30	"kotte.saideep@gmail.com" (email for SSH key)
31	"done, now test the SSH connection"
32	"test it by sending a message from the phone"
33	"it works on my phone"
34	"make it look better on the phone"
35	"it's connected" (new UI connected to Railway)
36	"push the changes to github"
37	"Could you document all the prompts used to build this app and the queries execu

## All Shell Commands Executed

### Project Setup

```
mkdir -p /Users/sdk/personal_assistant
cd /Users/sdk/personal_assistant
python3 -m venv venv
source venv/bin/activate
pip install anthropic rich python-dateutil
```

### Git Setup

```
git init
git add .
git commit -m "Initial commit: multi-agent personal assistant"
git remote add origin https://github.com/kottesaideep-eng/personal-assistant.git
```

### SSH Key Setup

```
ssh-keygen -t ed25519 -C "kotte.saideep@gmail.com" -f ~/.ssh/id_ed25519 -N ""
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
cat ~/.ssh/id_ed25519.pub
ssh -T git@github.com
git remote set-url origin git@github.com:kottesaideep-eng/personal-assistant.git
git push origin main
```

### Expo App -- Initial Scaffold Attempts (failed)

```
# Attempt 1 -- manual setup (SDK mismatch issues)
mkdir mobile && cd mobile
npm init -y
npm install expo react react-native

# Attempt 2 -- create-expo-app with SDK 52 (version mismatch with Expo Go)
npx create-expo-app mobile --template blank-typescript
```

### Expo App -- Final Working Scaffold

```
npx create-expo-app@latest mobile-fresh --template blank-typescript
cd mobile-fresh
npm install react-native-markdown-display
npx expo install --fix
```

### Expo Dev Server

```
# Start with tunnel (for Expo Go on phone)
npx expo start --tunnel

# Start fresh (clear cache)
npx expo start --tunnel --clear
```

## Dependency Fixes

```
npm install expo-asset expo-font expo-modules-core
npm install --save-dev babel-preset-expo
npx expo install --fix
rm -rf node_modules && npm install
```

## Files Created

```
personal_assistant/
??? assistant.py          # Orchestrator (CLI + headless API mode)
??? main.py              # CLI entry point
??? server.py            # FastAPI server for Railway
??? Procfile             # Railway process definition
??? railway.toml         # Railway build/deploy config
??? requirements.txt      # Python dependencies
??? BUILD_LOG.md         # This file
?
??? tools/
?   ??? __init__.py      # 22 tool definitions + dispatcher
?   ??? config.py        # Centralized path config (DATA_DIR, WORKSPACE_DIR)
?   ??? web_search.py    # Tavily API web search
?   ??? calendar_tool.py # JSON-backed calendar
?   ??? notes_tool.py    # Markdown note files
?   ??? reminders_tool.py # JSON-backed reminders
?   ??? memory_tool.py   # Long-term key-value memory
?   ??? file_tool.py     # Workspace file manager
?   ??? research_agent.py # Research sub-agent (separate Claude instance)
?
??? mobile-fresh/
??? App.tsx              # Main screen (typing indicator, suggestions, header)
??? app.json             # Expo config
??? package.json         # Node dependencies
??? src/
    ??? types.ts         # Message + HistoryItem interfaces
    ??? api.ts           # sendMessage() + checkHealth()
    ??? components/
        ??? MessageBubble.tsx # Markdown chat bubbles
        ??? ChatInput.tsx    # Animated input + send button
        ??? SettingsModal.tsx # Railway URL configuration
```

## Environment Variables Required

Variable	Where	Description
<code>`ANTHROPIC_API_KEY`</code>	Railway	Anthropic API key for Claude
<code>`TAVILY_API_KEY`</code>	Railway	Tavily API key for web search
<code>`DATA_DIR`</code>	Railway	<code>`/data`</code> -- path to Railway persistent volume

## Architecture Decisions

### Why Multi-Agent?

The orchestrator handles all 22 tools directly. The `research_task` tool delegates to a separate Claude instance (`research_agent.py`) that runs its own web-search agentic loop, allowing deeper research without polluting the main conversation context.

### Why Stateless API?

The full conversation history is stored client-side (in the mobile app's state) and sent with each request. The backend is completely stateless, making it simple to deploy and scale on Railway.

### Why Tavily Instead of DuckDuckGo?

Railway's server IPs are rate-limited/blocked by DuckDuckGo's scraping protection. Tavily is a proper search API designed for LLM applications, with a free tier of 1,000 searches/month.

### Why Expo SDK 54?

The Expo Go app on the App Store ships with a specific SDK version. Building against the wrong SDK version causes a version mismatch error at runtime. Using `npx create-expo-app@latest` ensures the scaffold matches the latest Expo Go.

### Why SSH Keys for GitHub?

HTTPS personal access tokens expire or require re-entry. SSH keys are persistent and don't require entering credentials on every push.