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

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
38	"Implement the following plan: Add 6 New Features to Personal Assistant App"

39	"i need a ios widget as well"
40	"ok, could you add a feature where i call SARVIS from my phone it should answer"

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 + long-press copy/share + image
        ?   ??? ChatInput.tsx       # Camera button, image preview, mic button + voice
        ?   ??? SettingsModal.tsx   # Railway URL configuration
        ?   ??? HistoryModal.tsx    # Saved conversation list
        ??? utils/
        ?   ??? storage.ts          # Conversation save/load/list/delete (AsyncStorage)
        ?   ??? notifications.ts    # Push token registration + local scheduling
        ??? widgets/
            ??? AssistantWidget.tsx # Android home screen widget
```

Environment Variables Required

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

Phase 7 -- 6 New Features (Copy/Share, Chat History, Images, EAS Build, Voice, Push Notifications, Widget)

User Prompt

```
"Implement the following plan: Add 6 New Features to Personal Assistant App"
```

Features Added

Step 1 -- Message Copy/Share

- Long-press any message bubble -> ActionSheetIOS (iOS) or Alert (Android) with "Copy text" / "Share" / "Cancel"
- Uses expo-clipboard and react-native's Share API
- File: src/components/MessageBubble.tsx

Step 2 -- Chat History

- Auto-saves conversation when tapping x (if there are user messages)
- ? history button in header opens HistoryModal
- Conversations stored in AsyncStorage under CONV_<timestamp> keys with an index at CONV_INDEX
- Tap to reload, long-press or ? button to delete
- Files: src/utils/storage.ts (NEW), src/components/HistoryModal.tsx (NEW)

Step 3 -- Image Sharing

- ? camera button in input bar -- opens photo library or camera
- Selected image shown as thumbnail preview with x to remove
- Image sent as base64 in image_base64 field to /chat
- assistant.py builds multi-modal Claude content block (image + text) for the first turn; history remains text-only
- Backend: ChatRequest extended with image_base64 and image_mime_type fields

Step 4 -- EAS Build Setup

- Created mobile-fresh/eas.json with development / preview / production profiles
- Updated app.json with bundleIdentifier and package fields
- Build commands: eas build --platform ios --profile preview

Step 5 -- Voice Input

- ? mic button appears when text input is empty
- Hold to record, release to stop -- @react-native-voice/voice
- Pulsing red animation while recording
- Speech result populates text input

Step 6 -- Push Notifications

- On startup: registerForPushNotificationsAsync() -> Expo push token -> POST /register-device
- APScheduler runs every 1 minute on backend, checks reminders.json for due items, sends via Expo Push API
- New backend endpoint: POST /register-device saves token to DATA_DIR/devices.json
- Files: src/utils/notifications.ts (NEW)

Step 7 -- Android Home Screen Widget

- AssistantWidget.tsx using react-native-android-widget
- Shows last message and "Open app ->" tap target
- File: src/widgets/AssistantWidget.tsx (NEW)

Packages Installed

```
# Expo SDK packages
npx expo install expo-clipboard expo-image-picker expo-notifications expo-device

# Native packages (require EAS build, not Expo Go)
npm install @react-native-voice/voice react-native-android-widget
```

Python Dependencies Added

```
apscheduler>=3.10.0
httpx>=0.27.0
```

Files Modified

File	Change
`mobile-fresh/App.tsx`	History button, auto-save on clear, push init, load conversation, image send
`mobile-fresh/src/types.ts`	Add `imageUri?` to Message, add `ConversationSummary` type
`mobile-fresh/src/api.ts`	Image params in `sendMessage()`, add `registerDevice()`
`mobile-fresh/src/components/MessageBubble.tsx`	Long-press copy/share, render `imageUri`
`mobile-fresh/src/components/ChatInput.tsx`	Camera button + preview, mic button + voice
`mobile-fresh/src/utils/storage.ts`	NEW -- conversation save/load/list/delete
`mobile-fresh/src/utils/notifications.ts`	NEW -- push token, local scheduling
`mobile-fresh/src/components/HistoryModal.tsx`	NEW -- conversation history list
`mobile-fresh/src/widgets/AssistantWidget.tsx`	NEW -- Android home screen widget
`mobile-fresh/app.json`	Plugins for image-picker, notifications, voice, widget; permissions; dark UI
`mobile-fresh/eas.json`	NEW -- EAS build configuration
`server.py`	Image fields in ChatRequest, /register-device, APScheduler for reminders
`assistant.py`	Multi-modal content block support (image + text)
`requirements.txt`	apscheduler, httpx

Phase 8 -- iOS Home Screen Widget

User Prompt

> "i need a ios widget as well"

Approach

iOS widgets require native SwiftUI/WidgetKit code -- no pure-JS solution exists.

Used @bacons/apple-targets (by Evan Bacon / Expo team) as the Expo config plugin to inject the widget extension into the Xcode project during eas build.

Widget reads the last assistant reply from an App Group shared UserDefaults (group.com.saideep.personalassistant) so it stays in sync without network calls.

Files Created

File	Purpose
`targets/widget/index.swift`	SwiftUI widget -- small + medium layouts, dark theme matching app
`targets/widget/Info.plist`	Widget extension info plist
`targets/widget/expo-target.config.js`	Tells `@bacons/apple-targets` this is a WidgetKit extension
`modules/shared-defaults/ios/SharedDefaults.swift`	Native Expo module -- writes/reads App Group UserDefaults
`modules/shared-defaults/index.ts`	JS API: `updateWidgetLastMessage(msg)`
`modules/shared-defaults/expo-module.config.json`	Expo module config

Files Modified

File	Change
`app.json`	Added `@bacons/apple-targets` plugin; iOS App Group entitlement
`App.tsx`	Call `updateWidgetLastMessage(reply)` after each assistant response

How It Works

1. After every Claude reply: updateWidgetLastMessage(reply.slice(0,140)) writes to UserDefaults(suiteName: "group.com.saideep.personalassistant")
2. The SwiftUI widget reads this key on its 15-minute refresh cycle (or on WidgetKit's system schedule)
3. Small widget: emoji + title + message preview + "Open ->"
4. Medium widget: left branding column + right "last reply" column

Package Installed

```
npm install @bacons/apple-targets
```

Setup Required Before Building

1. In app.json, replace "REPLACE_WITH_YOUR_TEAM_ID" with your Apple Developer Team ID (find it at developer.apple.com -> Membership -- 10-character string like A1B2C3D4E5)
1. Register the App Group group.com.saideep.personalassistant in Apple Developer portal (Identifiers -> App Groups -> +)
1. Add the App Group to both the main app identifier AND a new widget extension identifier
2. Run: eas build --platform ios --profile preview

Phase 9 -- "Hey SARVIS" Voice Activation

User Prompt

```
"ok, could you add a feature where i call SARVIS from my phone it should answer"
```

What Was Built

"Hey Siri, SARVIS" (iOS) and "Hey Google, open SARVIS" (Android) open the app and automatically activate the microphone -- so the user can speak immediately.

How It Works

iOS -- Siri Shortcut

1. On first launch, donateSarvisShortcut() donates an NSUserActivity with suggested phrase "SARVIS" to Siri
1. iOS suggests adding it to Siri in Settings -> Siri & Search
2. User can also tap "Add SARVIS to Siri" in Settings ?? to set a custom phrase
3. When triggered: addShortcutListener fires -> autoVoice = true -> ChatInput auto-starts mic after 600 ms

Android -- Google Assistant

1. App is renamed "SARVIS" in app.json
2. "Hey Google, open SARVIS" opens the app (no code needed -- Google reads app name)
3. Android home screen shortcut: sarvis://voice deep link -> auto-starts mic

Deep Link fallback (sarvis://voice)

- Works on both platforms via Linking.getInitialURL() and Linking.addEventListener
- Any shortcut, NFC tag, or QR code pointing to sarvis://voice triggers auto-mic

Files Created

File	Purpose
`src/utils/shortcut.ts`	`donateSarvisShortcut()`, `addSarvisShortcutListener()`, `presentAddToSiriDialog`

Files Modified

File	Change
`app.json`	Renamed app to "SARVIS"; added `scheme: "sarvis"`; `NSUserActivityTypes`; Siri e
`App.tsx`	Donate shortcut on launch; Siri + URL deep link listeners; `autoVoice` state ->
`src/components/ChatInput.tsx`	`autoActivateMic` prop -- starts mic 600ms after mount if true
`src/components/SettingsModal.tsx`	"Add SARVIS to Siri" button (iOS) / Google Assistant instructions (Android)
`src/widgets/AssistantWidget.tsx`	Renamed to "SARVIS"
`targets/widget/index.swift`	Renamed widget to "SARVIS"

Package Installed

```
npm install react-native-siri-shortcut --legacy-peer-deps
npm install expo-linking --legacy-peer-deps
```

User Setup (iOS)

After installing the EAS build:

1. Open SARVIS -> tap ?? -> tap "Add 'SARVIS' to Siri"
2. Record your phrase (default suggestion: "SARVIS")
3. Say "Hey Siri, SARVIS" -- app opens and mic activates

User Setup (Android)

After installing the EAS build:

- Say "Hey Google, open SARVIS" -- app opens
- Or create a home screen shortcut with URL sarvis://voice for instant mic

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.