GSoC 2025 Proposal: Gemini API Developer Workspace in Postman

Jevon Mao
Stanford University
jevon@cs.stanford.edu

March 26, 2025

Contact Information

- **GitHub:** https://github.com/jevonmao
- **Affiliation:** Stanford University
- **Country of Residence:** United States
- **Time Zone:** California (UTC-8)

Synopsis

This project proposes the creation of a comprehensive Postman Workspace tailored for Google's Gemini API suite. It will offer developers a robust, user-friendly hub to explore, integrate, and test Gemini's capabilities, significantly lowering the barrier to entry for new Gemini developers.

Deliverables

- **Phase 0 Deliverables**
  - Engage with the Gemini developer community via the Google AI Forum
  - Explore existing and related work on Gemini API exploration and documentation
  - Gather feedback on desired use cases and pain points
- **Phase 1 Deliverables:**
  - Postman collections for all major Gemini API endpoints
  - Parameterized request templates with documentation and comments
  - Initial environment setup for test and production usage
- **Phase 2 Deliverables:**
  - Integrated workspace documentation and tutorial walkthroughs
  - Example test scripts for validating success and error responses

  - Mock server configurations for local development
- **Final Deliverables:**
  - GitHub repo created with GitHub Action to auto-sync with Gemini API updates
  - Finalized and polished Postman Workspace with linked guides
  - Final technical documents covering project details for future long-term maintenance

Timeline

- **June 1 – June 2:** Community bonding, requirements gathering, and initial workspace setup
- **June 3 – July 14:** Develop core Postman collections with inline documentation and parameter customization
- **July 15 – July 21:** Midterm evaluation and mentor feedback integration
- **July 22 – August 26:** Add tutorials, test scripts, mock servers; implement GitHub Action automation
- **August 27 – August 31:** Final testing, documentation polish, and public workspace deployment

Technical Approach

The project will involve the following concrete steps:

- **Design Modular Postman Collections**
  - Identify and list all supported Gemini API endpoints (e.g., text generation, chat, multimodal, image generation, code generation)
  - Create separate folders for each capability within the Postman collection
  - For each endpoint, provide multiple pre-filled example requests covering different parameter configurations
- **Implement Secure and Flexible Environments**
  - Create two Postman environments: `Gemini-Testing` and `Gemini-Production`
  - Define variables such as `api_key`, `project_id`, `region`, and endpoint base URLs
  - Write instructions for how users can safely obtain, store, and switch API keys
- **Embed Inline Documentation and Use-Case Guides**
  - Add descriptions to every request explaining its purpose, required parameters, and expected responses
  - Use Postman's markdown editor to create integrated walkthroughs for common tasks (e.g., generating text from a prompt, uploading an image)
  - Cross-link relevant requests and variables for easier navigation
- **Add Automated Testing via Postman Scripts**

  - Write test scripts in JavaScript to check for response status codes, schema conformity, and common failure messages
  - Validate key response fields such as `candidates`, `text`, and `image_url` depending on endpoint
  - Export test results using Postman monitors or Newman CLI for CI workflows
- **Set Up Mock Servers for Local Testing**
  - Define mock endpoints in Postman to simulate Gemini responses
  - Use example responses from the official Gemini documentation and add success/error variants
  - Configure these for use with the `Gemini-Mock` environment to enable offline testing
- **Create GitHub Action for Workspace Maintenance**
  - Write a script to periodically check the official Gemini API schema/docs for updates
  - Auto-generate or flag changes in Postman collections based on updated endpoints
  - Deploy updated Postman collections and documentation via GitHub CI pipeline

Related Work

An existing third-party Postman Workspace for Gemini APIs is publicly available at `https://www.postman.com/ai-on-postman/google-gemini-apis/collection/1wnv7ft/gemini-api`. This collection covers most Gemini API routes, including text generation, image generation, and fine-tuning, and is well documented with concise instructions on obtaining and using an API key.

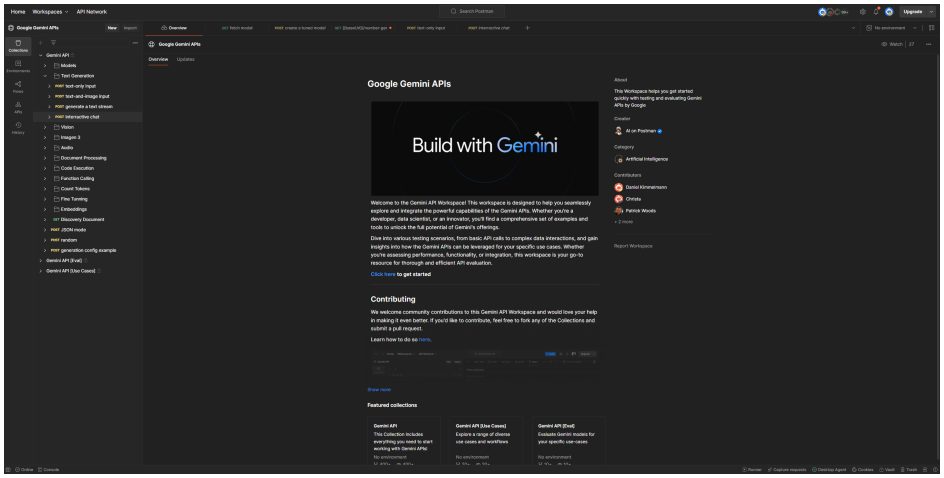

Figure 1: Screenshot of the Google Gemini APIs Postman Workspace

However, it has notable limitations:

- It lacks comprehensive parameter customization. While the endpoints are available, there is no clear or interactive method to modify parameters such as ‘temperature‘, ‘top_p‘, or ‘top_k‘, nor documentation explaining their purpose.

- There is no environment management for separating test vs. production usage, leading to higher risk during development.
- Documentation is limited to short one-sentence descriptions per endpoint, without tutorials or guided walkthroughs for common scenarios.
- It lacks testing scripts, validation logic, or mock server configurations. This means developers cannot easily verify expected API behavior or prototype without calling the real API.
- No CI/CD or update automation is provided, meaning the workspace can become outdated as Gemini evolves.

This proposal aims to significantly improve upon this foundation by providing:

- Fully customizable parameter templates with explanatory comments
- Separated and pre-configured environments for safe development and deployment
- Detailed documentation and embedded tutorials for tasks like chatbot setup, image generation, and prompt tuning
- Test scripts to validate response structures and values
- A Postman-based mock server and integrated GitHub Action for automatic updates

By learning from and building upon the existing workspace, my goal is to deliver a more powerful, reliable, and user-friendly toolkit for the Gemini developer community.

Benefits to the Community

This Workspace will accelerate the adoption of Gemini by providing developers with immediate, well-documented tools to experiment with. It aids onboarding, minimizes errors during setup, and supports efficient troubleshooting. By standardizing best practices and common configurations, this project will benefit the broader ecosystem of developers and researchers utilizing Gemini APIs.

About Me

- **Education:** B.S. in Computer Science, Stanford University (Expected 2027)
- **Work Experience:**
  - Mobile Engineer @ KOS AI — Led ML-integrated health monitoring app development using CNN-LSTM models, front end in React Native and SwiftUI
  - Cofounder @ CollegeBot.AI — Built an AI agent chatbot platform for college students, scaling it to over 200K users in 2 months after launch
- **Postman Experience:**
  - Reverse-engineered the Aeries gradebook app to build a private Postman collection for their internal APIs `https://documenter.getpostman.com/view/17518841/2sAYkKJyBk`
  - Used Postman extensively in *Project Democracy* to test and debug API endpoints from multiple external election info providers

  - Maintain a Postman server for the *SMHS App* to test backend routes (e.g., school announcements, calendar). Set up Postman monitors for scheduled endpoint testing

- **Open Source Projects:**
  - *SMHS App* — Sole developer of cross-platform campus app with 10K+ downloads `https://github.com/jevonmao/SMHS`
  - *PermissionsSwiftUI* — Open-source SwiftUI library with 1.5K+ stars `https://github.com/jevonmao/PermissionsSwiftUI`
  - *Project Democracy* — National award-winning iOS app connecting voters to local election info `https://github.com/itsliamdowd/Project-Democracy`
- **Contribution Experience:**
  - Contributor to Swift compiler — Authored protocol-oriented programming notes (PR #38253) `https://github.com/swiftlang/swift/pull/38253`
  - Ex-maintainer of Eul macOS system monitor (9K+ stars) `https://github.com/gao-sun/eul`
  - Ex-maintainer of SwifterSwift — Popular Swift extension library `https://github.com/SwifterSwift/SwifterSwift`
- **Research Experience:**
  - Stanford AI Lab — Advised by Prof. Fei-Fei Li and Karen Liu, researching diffusion-based 3D egocentric pose estimation with SLAM + point clouds
- **Why GSoC:** I'm passionate about improving developer tooling for LLMs and want to build a best-in-class Postman experience for Gemini. My extensive hands-on experience using Postman for reverse engineering, integration testing, and automated monitoring makes me confident in delivering a robust, production-grade workspace for the Gemini API.

Commitment

I expect to dedicate 25–30 hours per week during GSoC, with full-time availability after finishing my Stanford spring quarter. I will communicate regularly with mentors via GitHub, Slack, and email, or any other relevant channels.

Future Contributions

Post-GSoC, I plan to continue maintaining the workspace and contributing to related tooling. I'm also interested in extending it to other Google ML APIs to promote a unified developer interface. I hope to work at Google as a software engineering intern next year.