

Project Verify

Overview

Project Verify is a multi-model specification synthesis system built for the Secure Program Synthesis Hackathon. It takes unstructured inputs such as source code, documentation, or system requirements and transforms them into structured formal specifications using large language models. The system integrates outputs from multiple models, compares their interpretations, and produces a consolidated view of correctness-related artifacts such as invariants, preconditions, postconditions, hidden assumptions, and testable properties.

Objective

The primary objective of Project Verify is to improve the reliability of AI-generated interpretations of software behavior by introducing a structured verification layer over large language models. Instead of relying on a single model output, the system leverages multiple models to surface inconsistencies and reduce specification ambiguity.

System Design

Project Verify is implemented as a client-side React application built with Vite. It operates entirely in the browser and interacts directly with external model APIs when API keys are provided by the user.

The architecture consists of two main workflows:

1. Specification Elicitation Workflow

This workflow processes a single input (code, documentation, or requirements) and sends it to selected language models. Each model returns a structured JSON response containing:

- A natural language summary of behavior
- A list of formal specifications including preconditions, postconditions, invariants, and properties
- Hidden assumptions that may affect correctness
- Areas that are underspecified
- Suggested properties for testing

The system parses and renders these outputs into a structured interface for comparison and review.

2. Cross-Model Validation Workflow

This workflow evaluates outputs from multiple models on the same input. It performs a secondary analysis step using a designated model to:

- Identify agreements between models
- Detect disagreements and explain their significance
- Highlight missing or inconsistent specifications
- Generate a consensus specification
- Provide a confidence rating for the synthesis

This enables structured comparison of reasoning differences across models.

Model Integration

Project Verify supports three models: Claude (Anthropic), GPT-4o (OpenAI), Gemini 1.5 Pro (Google)

API keys are optional and supplied by the user at runtime. If a key is not provided for a model, that model is disabled in the interface. Claude is treated as the default available model, while OpenAI and Gemini require user-provided authentication.

Security and Key Handling

The system is designed with a client-first security model. API keys are:

- Stored only in browser memory during runtime
- Never persisted in local storage or external databases
- Sent only directly to the respective model providers when making requests

This ensures that sensitive credentials remain under user control and are not handled by any backend service within the project.

Output Structure

All model outputs are required to follow a strict JSON schema. This enables deterministic rendering of:

- Structured specification blocks
- Risk-tagged hidden assumptions
- Categorized underspecification areas
- Testable property definitions

The structured format allows systematic comparison rather than free-form text analysis.

User Interface Design

The frontend uses an earth tone visual system optimized for readability and technical focus. The interface is divided into two primary tabs:

- Specification Elicitation
- Cross-Model Validation

Each model output is displayed in a dedicated panel with color-coded identification and structured breakdowns of generated artifacts. Disagreements and consensus outputs are visually separated for clarity.

Limitations

The system depends on external API availability and user-provided keys for full functionality. Model outputs may vary in structure quality, and JSON parsing robustness is required to handle inconsistent responses. Additionally, all computation is client-side, which limits scalability for heavy batch processing.

Future Development Direction

Project Verify is intended to evolve into a privacy-preserving edge-native library for multi-model orchestration. The planned direction includes:

- Moving core logic into a reusable library layer
- Supporting execution in edge environments such as Cloudflare Workers and Deno
- Introducing local model inference support through WebGPU and lightweight runtime frameworks
- Eliminating reliance on centralized backend systems
- Enhancing secure multi-model routing without persistent credential storage

The long-term goal is to establish a framework for secure, verifiable, and decentralized specification synthesis.

Conclusion

Project Verify demonstrates a multi-model approach to formal specification extraction and comparison. By combining structured prompting, strict output schemas, and cross-model validation, it provides a foundation for improving reliability in AI-assisted software understanding and early-stage verification workflows.