PeiDocker Terminal GUI - Application Overview Design

Claude Code

July 23, 2025

1 Application Architecture Overview

This document provides comprehensive visual designs for the PeiDocker Terminal GUI using the Textual framework. The GUI provides two interaction modes: Simple (wizard-based) and Advanced (form-based) for creating Docker container configurations.

1.1 High-Level Application Flow

1.2 Application Components Architecture

1.3 File Structure Organization

1.4 Data Flow Architecture

1.5 Mode Selection Interface

2 Technical Implementation Notes

2.1 Textual Framework Integration

The GUI will be built using the Textual framework with the following key components:

- **App Class**: Main application controller inheriting from `textual.app.App`
- **Screen Classes**: Separate screens for different phases (startup, mode selection, configuration)
- **Widget Classes**: Custom widgets for specialized input types (SSH keys, port mappings, etc.)
- **Reactive Variables**: For real-time validation and state management
- **CSS Styling**: For consistent visual appearance across all components

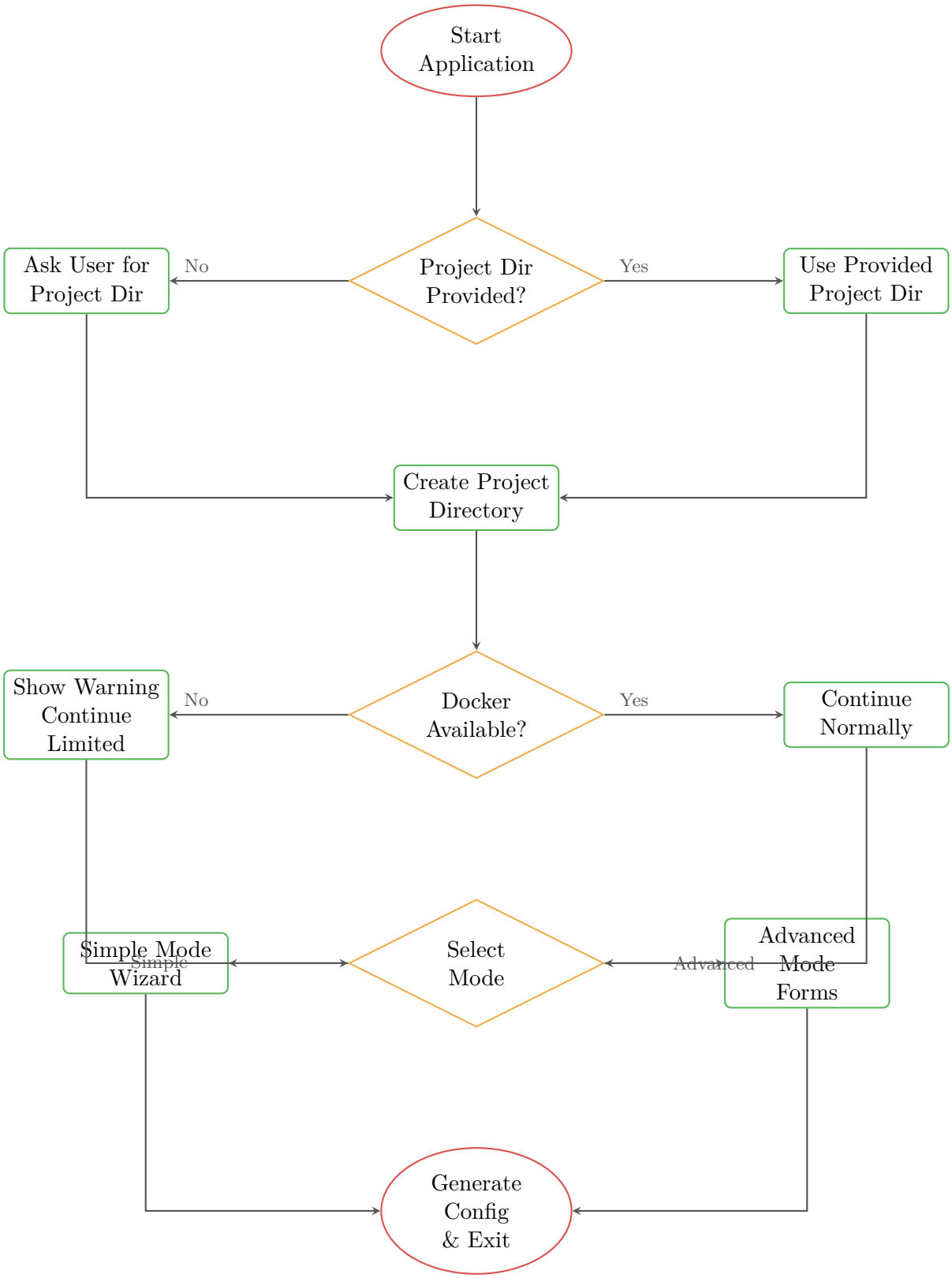


Figure 1: Main Application Flow

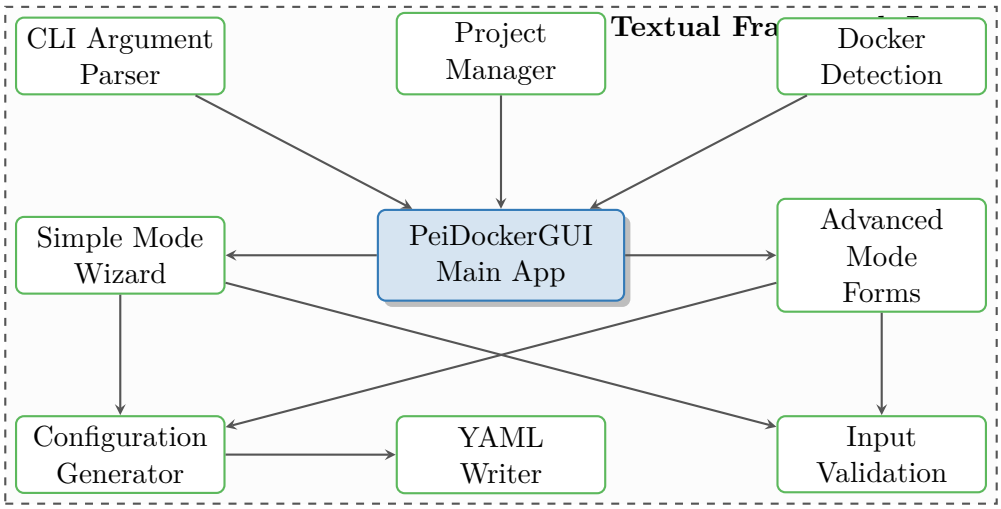


Figure 2: Application Components Architecture

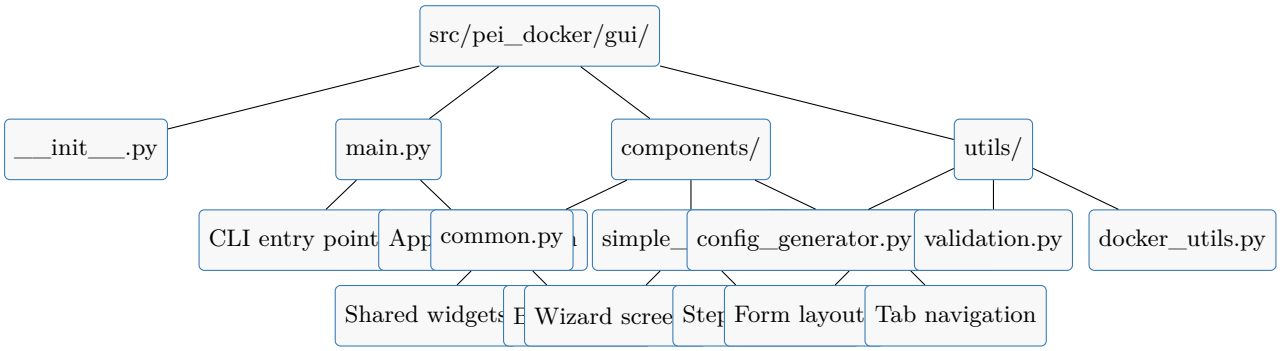


Figure 3: Proposed File Structure

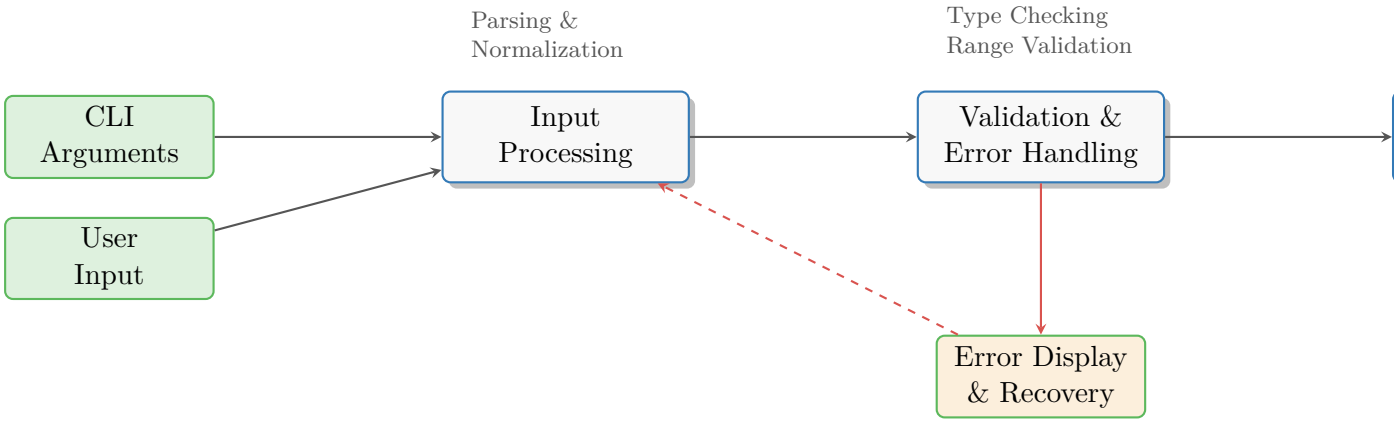


Figure 4: Data Flow Through Application

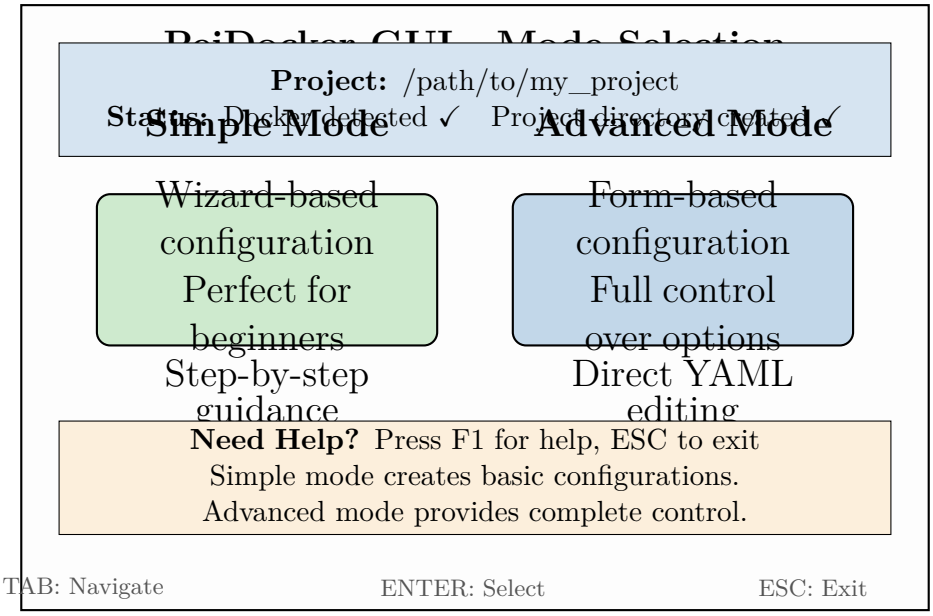
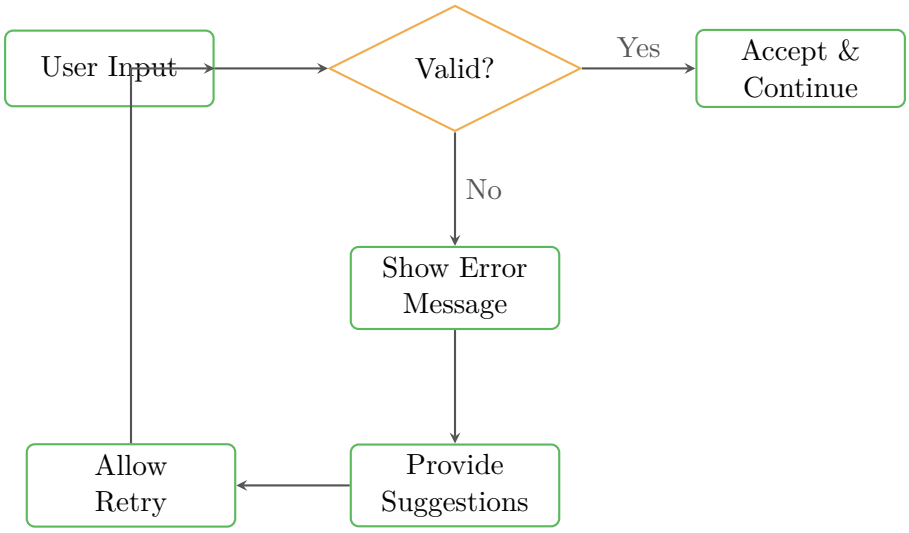


Figure 5: Mode Selection Screen Layout

2.2 Error Handling Strategy



2.3 Configuration Generation Pipeline

The application follows a structured pipeline for converting user input into the final YAML configuration:

1. **Input Collection**: Gather all user inputs through GUI forms/wizards
2. **Validation**: Check data types, ranges, and relationships
3. **Transformation**: Convert GUI data structures to config objects
4. **Template Application**: Apply user data to configuration templates
5. **YAML Generation**: Serialize final configuration to YAML format
6. **File Writing**: Save configuration to project directory