

**BPP\_AGENT LLD**

***Version 1.1***

*November 2025*

**

|  |
| --- |
| **Revisions** |

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1.1 | Siddharth Bhatia | Bpp\_Agent LLD | 07-11-2025 |

|  |
| --- |
| **Review & Approval** |

**Requirements Document Approval History**

| **Approving Party** | **Version Approved** | **Signature** | **Date** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Requirements Document Review History**

| **Reviewer** | **Version Reviewed** | **Signature** | **Date** |
| --- | --- | --- | --- |
| **Jeevan Kumar** | 0.5 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
|  |

Contents

[Mini Agent Low-Level Design (LLD) Document 4](#_Toc213774798)

[Device Registration Tool (Windows & Linux) 4](#_Toc213774799)

[Document Information 4](#_Toc213774800)

[Table of Contents 4](#_Toc213774801)

[1. Introduction 5](#_Toc213774802)

[1.1 Purpose 5](#_Toc213774803)

[1.2 Scope 5](#_Toc213774804)

[1.3 Audience 5](#_Toc213774805)

[1.4 Key Differences from BPP Agent 5](#_Toc213774806)

[2. System Overview 5](#_Toc213774807)

[2.1 Application Type 5](#_Toc213774808)

[2.2 Supported Platforms 6](#_Toc213774809)

[2.3 Key Features 6](#_Toc213774810)

[2.4 User Journey 6](#_Toc213774811)

[3. Technology Stack 7](#_Toc213774812)

[3.1 Core Technologies 7](#_Toc213774813)

[3.2 External Dependencies 7](#_Toc213774814)

[3.3 Build Configuration 7](#_Toc213774815)

[4. Architecture Design 8](#_Toc213774816)

[4.1 High-Level Architecture 8](#_Toc213774817)

[4.2 Component Interaction Flow 8](#_Toc213774818)

[5. Module Design 9](#_Toc213774819)

[5.1 Device Information Collection Module 9](#_Toc213774820)

[5.2 Browser Detection Module 10](#_Toc213774821)

[5.3 User Input Module 11](#_Toc213774822)

[5.4 Registration Module 11](#_Toc213774823)

[6. Component Design 12](#_Toc213774824)

[6.1 Data Structures 12](#_Toc213774825)

[6.2 Validation Functions 13](#_Toc213774826)

[6.3 System Information Functions 14](#_Toc213774827)

[7. Platform-Specific Implementation 15](#_Toc213774828)

[7.1 Windows-Specific Features 15](#_Toc213774829)

[7.2 Linux-Specific Features 18](#_Toc213774830)

[8. User Interface Design 20](#_Toc213774831)

[8.1 Windows GUI Design 20](#_Toc213774832)

[8.2 Linux CLI Design 21](#_Toc213774833)

[9. Data Collection 22](#_Toc213774834)

[9.1 Device Information Fields 22](#_Toc213774835)

[9.2 Browser Information 22](#_Toc213774836)

[9.3 Data Collection Flow 23](#_Toc213774837)

[10. API Integration 23](#_Toc213774838)

[10.1 Backend Endpoint 23](#_Toc213774839)

[10.2 Request Format 23](#_Toc213774840)

[10.3 Response Handling 24](#_Toc213774841)

[10.4 Error Handling 24](#_Toc213774842)

[11. Validation & Error Handling 25](#_Toc213774843)

[11.1 Input Validation 25](#_Toc213774844)

[11.2 Error Categories 25](#_Toc213774845)

[11.3 Error Recovery 26](#_Toc213774846)

[12. Security Implementation 26](#_Toc213774847)

[12.1 Data Privacy 27](#_Toc213774848)

[12.2 Input Sanitization 27](#_Toc213774849)

[12.3 Secure Communication 28](#_Toc213774850)

[12.4 Security Best Practices 28](#_Toc213774851)

[13. Deployment Architecture 28](#_Toc213774852)

[13.1 Build Process 28](#_Toc213774853)

[13.2 Distribution 29](#_Toc213774854)

[13.3 System Requirements 29](#_Toc213774855)

[13.4 Deployment Scenarios 30](#_Toc213774856)

[13.5 Update Process 30](#_Toc213774857)

[14. Appendix 31](#_Toc213774858)

[14.1 Glossary 31](#_Toc213774859)

[14.2 Constants Reference 31](#_Toc213774860)

[14.3 Allowed Email Domains 31](#_Toc213774861)

[14.4 API Endpoint Summary 31](#_Toc213774862)

[14.5 Error Codes 32](#_Toc213774863)

[14.6 File Locations 32](#_Toc213774864)

[14.7 Browser Detection Commands 32](#_Toc213774865)

[14.8 System Information Commands 32](#_Toc213774866)

[14.9 Troubleshooting Guide 33](#_Toc213774867)

[14.10 Performance Metrics 34](#_Toc213774868)

[14.11 Version History 34](#_Toc213774869)

[14.12 Future Enhancements 34](#_Toc213774870)

[14.13 Known Limitations 34](#_Toc213774871)

[14.14 Comparison with BPP Agent 35](#_Toc213774872)

[14.15 Dependencies 35](#_Toc213774873)

[Document Revision History 35](#_Toc213774874)

[Approval 35](#_Toc213774875)

# Mini Agent Low-Level Design (LLD) Document

## Device Registration Tool (Windows & Linux)

## Document Information

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 2025-01-07 | System Analysis | Initial LLD Document |

## Table of Contents

1. [Introduction](#Xe3d0fc0bea9a42ce7605565d0964033d7f6ee47)
2. [System Overview](#X5508b397684bf4192a0d8d0a48692b0135f807f)
3. [Technology Stack](#Xd4be06774eb8fef4566a4cdca4cb8ee01c79793)
4. [Architecture Design](#X47d58dc3c1c5a47155d99cdbd1f9bf296cc11c8)
5. [Module Design](#Xf1c5de4ef166a2e585c6d43a6ae34496e5e09cf)
6. [Component Design](#X1985a58bf911c4ec67977ca16fec294b39d4afe)
7. [Platform-Specific Implementation](#Xc017135340a4ae5419c7654aa8facd6c253b038)
8. [User Interface Design](#X28e215ea9c8001e472d2eb9205424b2cce90e5a)
9. [Data Collection](#X30c2c4ac6053aed566ac1a00115573218483251)
10. [API Integration](#Xc33b41bcca5255013a31164cd981e78e976c006)
11. [Validation & Error Handling](#X88cad0fe03c9181ddac42d498fe26bed4715a5b)
12. [Security Implementation](#X258f489e907798d8d20a6efd0d09b0520228797)
13. [Deployment Architecture](#Xbc9cbaa365f06b69738fef395b81baa8c94e238)
14. [Appendix](#X2178c9a55ac239f72e7388d4578fc59683c7999)

## 1. Introduction

### 1.1 Purpose

This document provides a comprehensive low-level design for the Mini Agent, a lightweight device registration tool that collects system information and registers devices with the backend server for license allocation.

### 1.2 Scope

The Mini Agent is a Go-based standalone application that provides: - Device information collection (UUID, MAC address, serial number, etc.) - Browser detection (Chrome, Firefox, Edge) - User information collection (email, phone number) - Device registration with backend server - Cross-platform support (Windows with GUI, Linux with CLI) - Input validation and error handling

### 1.3 Audience

* Backend Developers
* System Administrators
* End Users
* Technical Support
* QA Engineers

### 1.4 Key Differences from BPP Agent

| **Feature** | **Mini Agent** | **BPP Agent** |
| --- | --- | --- |
| Purpose | Device registration | License management & extension installation |
| Installation | One-time execution | System service installation |
| User Interaction | Required (email, phone) | Minimal (setup wizard) |
| Service | No service installation | Windows Service / systemd service |
| Extensions | No extension management | Full extension management |
| License | Registration only | Verification & enforcement |
| Size | Lightweight (~5 MB) | Full-featured (~15 MB) |

## 2. System Overview

### 2.1 Application Type

* **Windows**: GUI application with step-by-step wizard
* **Linux**: Command-line interface (CLI) application

### 2.2 Supported Platforms

* **Windows**: Windows 10/11 (32-bit and 64-bit)
* **Linux**: Ubuntu, Debian, CentOS, RHEL, Fedora

### 2.3 Key Features

#### 2.3.1 Core Features

1. **Device Information Collection**
   * System UUID
   * MAC address
   * Serial number
   * OS information
   * Hostname
   * Architecture
2. **Browser Detection**
   * Chrome version detection
   * Firefox version detection
   * Edge version detection
   * Installation status
3. **User Information Collection**
   * Email address (with domain validation)
   * Phone number (8-10 digits)
   * Real-time validation
4. **Device Registration**
   * POST data to backend server
   * Error handling and retry
   * Success/failure feedback
5. **User Interface**
   * Windows: GUI wizard with 5 steps
   * Linux: Interactive CLI prompts
   * Input validation feedback
   * Progress indicators

### 2.4 User Journey

#### 2.4.1 Windows User Journey

Launch Application  
 ↓  
Step 1: Welcome Screen  
 ↓  
Step 2: Collect Device Info  
 ↓  
Step 3: Enter Email & Phone  
 ↓  
Step 4: Send to Server  
 ↓  
Step 5: Complete

#### 2.4.2 Linux User Journey

Launch Application  
 ↓  
Enter Email (with validation)  
 ↓  
Enter Phone (with validation)  
 ↓  
Collect Device Info  
 ↓  
Send to Server  
 ↓  
Display Result

## 3. Technology Stack

### 3.1 Core Technologies

| **Category** | **Technology** | **Version** | **Purpose** |
| --- | --- | --- | --- |
| Language | Go (Golang) | 1.18+ | Core programming language |
| Windows API | golang.org/x/sys/windows | Latest | Windows GUI and system calls |
| HTTP Client | net/http | Built-in | API communication |
| JSON | encoding/json | Built-in | Data serialization |
| Network | net | Built-in | MAC address detection |

### 3.2 External Dependencies

import (  
 "golang.org/x/sys/windows" // Windows API (Windows only)  
)

### 3.3 Build Configuration

**Windows Build**:

# GUI version (no console)  
go build -ldflags="-H windowsgui" -o mini\_agent.exe main\_gui\_window.go  
  
# Console version (for debugging)  
go build -o mini\_agent\_console.exe main\_gui\_window.go

**Linux Build**:

# Standard build  
go build -o mini\_agent main.go  
  
# Optimized build  
go build -ldflags="-s -w" -o mini\_agent main.go

## 4. Architecture Design

### 4.1 High-Level Architecture

┌─────────────────────────────────────────────────────────────┐  
│ Mini Agent Process │  
│ ┌───────────────────────────────────────────────────────┐ │  
│ │ Main Application │ │  
│ │ ┌─────────────────────────────────────────────────┐ │ │  
│ │ │ UI Layer │ │ │  
│ │ │ - GUI Wizard (Windows) │ │ │  
│ │ │ - CLI Prompts (Linux) │ │ │  
│ │ │ - Input Validation │ │ │  
│ │ │ - Progress Display │ │ │  
│ │ └─────────────────────────────────────────────────┘ │ │  
│ │ ┌─────────────────────────────────────────────────┐ │ │  
│ │ │ Business Logic Layer │ │ │  
│ │ │ - Device Info Collection │ │ │  
│ │ │ - Browser Detection │ │ │  
│ │ │ - Input Validation │ │ │  
│ │ │ - Data Formatting │ │ │  
│ │ └─────────────────────────────────────────────────┘ │ │  
│ │ ┌─────────────────────────────────────────────────┐ │ │  
│ │ │ Data Access Layer │ │ │  
│ │ │ - HTTP Client │ │ │  
│ │ │ - JSON Serialization │ │ │  
│ │ │ - System Commands │ │ │  
│ │ └─────────────────────────────────────────────────┘ │ │  
│ └───────────────────────────────────────────────────────┘ │  
└─────────────────────────────────────────────────────────────┘  
 ↕  
┌─────────────────────────────────────────────────────────────┐  
│ Backend Registration Server │  
│ (Django REST Framework) │  
└─────────────────────────────────────────────────────────────┘

### 4.2 Component Interaction Flow

┌──────────────┐  
│ User Launch │  
└──────┬───────┘  
 │  
 ▼  
┌──────────────────┐  
│ Initialize UI │  
└──────┬───────────┘  
 │  
 ▼  
┌──────────────────┐  
│ Collect Email │  
│ (with validation)│  
└──────┬───────────┘  
 │  
 ▼  
┌──────────────────┐  
│ Collect Phone │  
│ (with validation)│  
└──────┬───────────┘  
 │  
 ▼  
┌──────────────────┐  
│ Collect Device │  
│ Information │  
└──────┬───────────┘  
 │  
 ▼  
┌──────────────────┐  
│ Format JSON │  
│ Payload │  
└──────┬───────────┘  
 │  
 ▼  
┌──────────────────┐  
│ POST to Backend │  
└──────┬───────────┘  
 │  
 ├─ Success ──────┐  
 │ │  
 ▼ ▼  
┌──────────────┐ ┌──────────────┐  
│ Show Error │ │ Show Success │  
└──────────────┘ └──────────────┘

## 5. Module Design

### 5.1 Device Information Collection Module

#### 5.1.1 Purpose

Collects comprehensive system information for device identification and registration.

#### 5.1.2 Key Functions

**getDeviceIdentifiers()**

func getDeviceIdentifiers() DeviceIdentifiers {  
 hostname, \_ := os.Hostname()  
   
 return DeviceIdentifiers{  
 UUID: getFullUUID(),  
 MacAddress: getMacAddress(),  
 SerialNumber: getSerialNumber(),  
 OSType: runtime.GOOS,  
 OSPlatform: runtime.GOOS,  
 OSRelease: getOSRelease(),  
 HostName: hostname,  
 Architecture: runtime.GOARCH,  
 CurrentVersion: "1.0.0",  
 Browsers: getBrowserVersions(),  
 }  
}

#### 5.1.3 Data Collection Methods

| **Field** | **Windows Method** | **Linux Method** |
| --- | --- | --- |
| UUID | WMI query | dmidecode, /sys/class/dmi/id/product\_uuid |
| MAC Address | net.Interfaces() | net.Interfaces() |
| Serial Number | WMI query | dmidecode, /sys/class/dmi/id/product\_serial |
| OS Type | runtime.GOOS | runtime.GOOS |
| OS Release | ver command | /etc/os-release, lsb\_release |
| Hostname | os.Hostname() | os.Hostname() |
| Architecture | runtime.GOARCH | runtime.GOARCH |

### 5.2 Browser Detection Module

#### 5.2.1 Purpose

Detects installed browsers and their versions for compatibility checking.

#### 5.2.2 Browser Detection Logic

**Windows**:

func getBrowserVersions() Browsers {  
 browsers := Browsers{  
 Chrome: Browser{Installed: false, Version: ""},  
 Firefox: Browser{Installed: false, Version: ""},  
 Edge: Browser{Installed: false, Version: ""},  
 }  
   
 // Chrome detection via PowerShell  
 chromeVersion := executeCommand(`powershell -command "(Get-Item 'C:\Program Files\Google\Chrome\Application\chrome.exe').VersionInfo.ProductVersion"`)  
 if chromeVersion != "" {  
 browsers.Chrome = Browser{Installed: true, Version: chromeVersion}  
 }  
   
 // Similar for Firefox and Edge...  
   
 return browsers  
}

**Linux**:

func getBrowserVersions() Browsers {  
 browsers := Browsers{  
 Chrome: Browser{Installed: false, Version: ""},  
 Firefox: Browser{Installed: false, Version: ""},  
 Edge: Browser{Installed: false, Version: ""},  
 }  
   
 // Chrome detection  
 chromeVersion := executeCommand("google-chrome --version 2>/dev/null")  
 if chromeVersion != "" && !strings.Contains(chromeVersion, "not installed") {  
 version := strings.Replace(chromeVersion, "Google Chrome ", "", 1)  
 browsers.Chrome = Browser{Installed: true, Version: version}  
 }  
   
 // Similar for Firefox and Edge...  
   
 return browsers  
}

### 5.3 User Input Module

#### 5.3.1 Email Collection

**Windows (GUI)**: - Text input control with real-time validation - Visual feedback (green checkmark / red X) - Debounced validation (2 seconds after typing stops) - Domain whitelist validation

**Linux (CLI)**: - Interactive prompt with validation loop - Immediate feedback on invalid input - Retry until valid email provided

#### 5.3.2 Phone Number Collection

**Windows (GUI)**: - Numeric-only input control - 8-10 digit validation - Real-time character filtering - Visual validation feedback

**Linux (CLI)**: - Interactive prompt with validation loop - 8-10 digit validation - Retry until valid phone provided

### 5.4 Registration Module

#### 5.4.1 Purpose

Sends collected device and user information to backend server for registration.

#### 5.4.2 Registration Flow

func postDeviceInfo(email, phoneNumber string, deviceInfo DeviceIdentifiers) error {  
 // Create registration request  
 registrationRequest := DeviceRegistrationRequest{  
 Email: email,  
 PhoneNumber: phoneNumber,  
 UUID: deviceInfo.UUID,  
 MacAddress: deviceInfo.MacAddress,  
 SerialNumber: deviceInfo.SerialNumber,  
 OSType: deviceInfo.OSType,  
 OSPlatform: deviceInfo.OSPlatform,  
 OSRelease: deviceInfo.OSRelease,  
 HostName: deviceInfo.HostName,  
 Architecture: deviceInfo.Architecture,  
 CurrentVersion: deviceInfo.CurrentVersion,  
 Browsers: deviceInfo.Browsers,  
 }  
   
 // Convert to JSON  
 jsonData, \_ := json.Marshal(registrationRequest)  
   
 // Create HTTP client with timeout  
 client := &http.Client{Timeout: 30 \* time.Second}  
   
 // Create POST request  
 req, \_ := http.NewRequest("POST", LicenseVerificationURL, bytes.NewBuffer(jsonData))  
 req.Header.Set("Content-Type", "application/json")  
   
 // Send request  
 resp, err := client.Do(req)  
 if err != nil {  
 return handleNetworkError(err)  
 }  
 defer resp.Body.Close()  
   
 // Check response  
 if resp.StatusCode >= 200 && resp.StatusCode < 300 {  
 return nil // Success  
 }  
   
 return parseErrorResponse(resp)  
}

## 6. Component Design

### 6.1 Data Structures

#### 6.1.1 Core Structures

type Browser struct {  
 Installed bool `json:"installed"`  
 Version string `json:"version,omitempty"`  
}  
  
type Browsers struct {  
 Chrome Browser `json:"chrome"`  
 Firefox Browser `json:"firefox"`  
 Edge Browser `json:"edge"`  
}  
  
type DeviceIdentifiers struct {  
 UUID string `json:"uuid"`  
 MacAddress string `json:"macAdress"` // Note: typo matches backend  
 SerialNumber string `json:"serialNumber"`  
 OSType string `json:"osType"`  
 OSPlatform string `json:"osPlatform"`  
 OSRelease string `json:"osRelease"`  
 HostName string `json:"hostName"`  
 Architecture string `json:"architecture"`  
 CurrentVersion string `json:"current\_version"`  
 Browsers Browsers `json:"browsers"`  
}  
  
type DeviceRegistrationRequest struct {  
 Email string `json:"email"`  
 PhoneNumber string `json:"phone\_number"`  
 UUID string `json:"uuid"`  
 MacAddress string `json:"macAdress"`  
 SerialNumber string `json:"serialNumber"`  
 OSType string `json:"osType"`  
 OSPlatform string `json:"osPlatform"`  
 OSRelease string `json:"osRelease"`  
 HostName string `json:"hostName"`  
 Architecture string `json:"architecture"`  
 CurrentVersion string `json:"current\_version"`  
 Browsers Browsers `json:"browsers"`  
}

### 6.2 Validation Functions

#### 6.2.1 Email Validation

func isValidEmail(email string) bool {  
 if len(email) == 0 {  
 return false  
 }  
   
 // Check for @ symbol  
 atIndex := strings.Index(email, "@")  
 if atIndex <= 0 || atIndex == len(email)-1 {  
 return false  
 }  
   
 // Extract domain  
 domain := email[atIndex+1:]  
   
 // Allowed domains  
 allowedDomains := []string{  
 "gmail.com",  
 "yahoo.com",  
 "outlook.com",  
 "gov.in",  
 "ekvayu.com",  
 }  
   
 // Check domain whitelist  
 for \_, allowedDomain := range allowedDomains {  
 if strings.ToLower(domain) == strings.ToLower(allowedDomain) {  
 return true  
 }  
 }  
   
 return false  
}

#### 6.2.2 Phone Number Validation

func isValidPhone(phone string) bool {  
 if len(phone) == 0 {  
 return false  
 }  
   
 // Extract digits only  
 digits := ""  
 for \_, char := range phone {  
 if char >= '0' && char <= '9' {  
 digits += string(char)  
 }  
 }  
   
 // Must be 8-10 digits  
 return len(digits) >= 8 && len(digits) <= 10  
}

### 6.3 System Information Functions

#### 6.3.1 MAC Address Detection

func getMacAddress() string {  
 interfaces, err := net.Interfaces()  
 if err != nil {  
 return "00:00:00:00:00:00"  
 }  
   
 for \_, iface := range interfaces {  
 // Skip loopback and down interfaces  
 if iface.Flags&net.FlagLoopback != 0 || iface.Flags&net.FlagUp == 0 {  
 continue  
 }  
   
 addrs, \_ := iface.Addrs()  
 for \_, addr := range addrs {  
 if ipnet, ok := addr.(\*net.IPNet); ok && !ipnet.IP.IsLoopback() {  
 if ipnet.IP.To4() != nil {  
 return iface.HardwareAddr.String()  
 }  
 }  
 }  
 }  
   
 return "00:00:00:00:00:00"  
}

#### 6.3.2 UUID Detection (Linux)

func getFullUUID() string {  
 // Try dmidecode with sudo  
 output := executeCommand("sudo dmidecode -s system-uuid 2>/dev/null")  
 if output != "" && !strings.Contains(output, "Permission denied") {  
 return output  
 }  
   
 // Try without sudo  
 output = executeCommand("dmidecode -s system-uuid 2>/dev/null")  
 if output != "" {  
 return output  
 }  
   
 // Try /sys filesystem  
 output = executeCommand("cat /sys/class/dmi/id/product\_uuid 2>/dev/null")  
 if output != "" {  
 return output  
 }  
   
 // Fallback to machine-id  
 output = executeCommand("cat /etc/machine-id 2>/dev/null")  
 if output != "" {  
 return output  
 }  
   
 return "Unknown"  
}

## 7. Platform-Specific Implementation

### 7.1 Windows-Specific Features

#### 7.1.1 GUI Window Creation

func createMainWindow() uintptr {  
 className, \_ := windows.UTF16PtrFromString("MiniAgentClass")  
 windowName, \_ := windows.UTF16PtrFromString("EKVAYU - Device Registration")  
   
 // Register window class  
 wc := WNDCLASSEX{  
 CbSize: uint32(unsafe.Sizeof(WNDCLASSEX{})),  
 LpfnWndProc: windowProcPtr,  
 HInstance: hInstance,  
 HCursor: loadCursor(IDC\_ARROW),  
 HbrBackground: COLOR\_WINDOW,  
 LpszClassName: className,  
 }  
 procRegisterClassExW.Call(uintptr(unsafe.Pointer(&wc)))  
   
 // Create window  
 hwnd, \_, \_ := procCreateWindowExW.Call(  
 0,  
 uintptr(unsafe.Pointer(className)),  
 uintptr(unsafe.Pointer(windowName)),  
 WS\_OVERLAPPEDWINDOW|WS\_VISIBLE,  
 CW\_USEDEFAULT, CW\_USEDEFAULT,  
 WINDOW\_WIDTH, WINDOW\_HEIGHT,  
 0, 0, hInstance, 0,  
 )  
   
 return hwnd  
}

#### 7.1.2 GUI Steps

**Step 0: Welcome Screen** - Application title - Version information - Welcome message - “Next” button to proceed

**Step 1: Data Collection** - Progress indicator - Status messages - Automatic device info collection - Auto-advance to next step

**Step 2: Review Information** - Display collected device info - UUID, MAC, Serial Number - OS information - Browser detection results - “Previous” and “Next” buttons

**Step 3: User Information** - Email input control with validation - Phone input control (numeric only, 8-10 digits) - Real-time validation feedback - Visual indicators (green checkmark / red X) - Debounced validation (2 seconds)

**Step 4: Complete** - Success/failure message - Backend response display - “Finish” button to close

#### 7.1.3 Input Controls

**Email Input Control**:

func createEmailInputControl(parentHwnd uintptr) {  
 editClass, \_ := windows.UTF16PtrFromString("EDIT")  
 editStyle := WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_LEFT | ES\_AUTOHSCROLL  
   
 emailInputControl, \_, \_ = procCreateWindowExW.Call(  
 0x00000200, // WS\_EX\_CLIENTEDGE  
 uintptr(unsafe.Pointer(editClass)),  
 0,  
 uintptr(editStyle),  
 uintptr(contentX),  
 uintptr(contentY),  
 uintptr(inputWidth),  
 uintptr(inputHeight),  
 parentHwnd,  
 ID\_EMAIL\_INPUT,  
 0,  
 0,  
 )  
   
 // Set focus  
 procSetFocus.Call(emailInputControl)  
}

**Phone Input Control** (with numeric filtering):

func phoneInputProc(hwnd uintptr, msg uint32, wParam, lParam uintptr) uintptr {  
 switch msg {  
 case WM\_CHAR:  
 char := rune(wParam)  
   
 // Allow backspace and control characters  
 if char == 8 || char == 127 || char < 32 {  
 break  
 }  
   
 // Only allow digits  
 if char < '0' || char > '9' {  
 return 0 // Block character  
 }  
   
 // Check length (max 10 digits)  
 length, \_, \_ := procSendMessageW.Call(hwnd, WM\_GETTEXTLENGTH, 0, 0)  
 if length >= 10 {  
 return 0 // Block if already 10 digits  
 }  
 }  
   
 // Call original procedure  
 ret, \_, \_ := procCallWindowProcW.Call(originalPhoneProc, hwnd, uintptr(msg), wParam, lParam)  
 return ret  
}

#### 7.1.4 Validation Feedback

func drawValidationIndicator(hdc uintptr, x, y int, isValid bool) {  
 var bgColor uintptr  
 var symbol string  
   
 if isValid {  
 bgColor = 0x0000AA00 // Green  
 symbol = "✓"  
 } else {  
 bgColor = 0x000000AA // Red  
 symbol = "✗"  
 }  
   
 // Draw circle background  
 bgBrush, \_, \_ := procCreateSolidBrush.Call(bgColor)  
   
 // Draw ellipse  
 procEllipse.Call(hdc, uintptr(x), uintptr(y),  
 uintptr(x+20), uintptr(y+20))  
   
 // Draw symbol  
 procSetTextColor.Call(hdc, 0x00FFFFFF) // White text  
 procDrawTextW.Call(hdc, symbolPtr, symbolLen, &rect, DT\_CENTER|DT\_VCENTER)  
}

### 7.2 Linux-Specific Features

#### 7.2.1 CLI Interface

**Main Flow**:

func main() {  
 fmt.Println("🔧 Device Registration Tool")  
 fmt.Println(strings.Repeat("=", 50))  
   
 // Get email  
 email, err := getUserEmail()  
 if err != nil {  
 log.Fatal(err)  
 }  
 fmt.Printf("✅ Email registered: %s\n", email)  
   
 // Get phone  
 phoneNumber, err := getUserPhoneNumber()  
 if err != nil {  
 log.Fatal(err)  
 }  
 fmt.Printf("✅ Phone number registered: %s\n", phoneNumber)  
   
 // Collect device info  
 fmt.Println("Collecting Device Information...")  
 deviceInfo := getDeviceIdentifiers()  
 printDeviceInfo(deviceInfo)  
   
 // Send to server  
 err = postDeviceInfo(email, phoneNumber, deviceInfo)  
 if err != nil {  
 fmt.Printf("❌ %v\n", err)  
 } else {  
 fmt.Println("✅ Device registration sent successfully!")  
 }  
   
 // Wait for user  
 fmt.Println("Press Enter to exit...")  
 fmt.Scanln(&input)  
}

#### 7.2.2 Interactive Prompts

**Email Prompt**:

func getUserEmail() (string, error) {  
 reader := bufio.NewReader(os.Stdin)  
   
 for {  
 fmt.Print("Please enter your email address: ")  
 email, err := reader.ReadString('\n')  
 if err != nil {  
 return "", err  
 }  
   
 email = strings.TrimSpace(email)  
   
 if !isValidEmail(email) {  
 fmt.Println("❌ Please enter a valid email address from allowed domains")  
 fmt.Println(" (gmail.com, yahoo.com, outlook.com, gov.in, ekvayu.com)")  
 continue  
 }  
   
 return email, nil  
 }  
}

**Phone Prompt**:

func getUserPhoneNumber() (string, error) {  
 reader := bufio.NewReader(os.Stdin)  
   
 for {  
 fmt.Print("Please enter your phone number: ")  
 phoneNumber, err := reader.ReadString('\n')  
 if err != nil {  
 return "", err  
 }  
   
 phoneNumber = strings.TrimSpace(phoneNumber)  
   
 // Validate  
 digitCount := 0  
 for \_, char := range phoneNumber {  
 if char >= '0' && char <= '9' {  
 digitCount++  
 }  
 }  
   
 if digitCount < 8 || digitCount > 10 {  
 fmt.Println("❌ Phone number must contain 8-10 digits")  
 continue  
 }  
   
 return phoneNumber, nil  
 }  
}

#### 7.2.3 System Commands

**Safe Command Execution**:

func executeCommand(command string) string {  
 cmd := exec.Command("bash", "-c", command)  
 output, err := cmd.Output()  
 if err != nil {  
 return "" // Silent failure  
 }  
 return strings.TrimSpace(string(output))  
}

**Privileged Commands** (with sudo):

// Try with sudo first  
output := executeCommand("sudo dmidecode -s system-uuid 2>/dev/null")  
if output != "" && !strings.Contains(output, "Permission denied") {  
 return output  
}  
  
// Fallback without sudo  
output = executeCommand("dmidecode -s system-uuid 2>/dev/null")

## 8. User Interface Design

### 8.1 Windows GUI Design

#### 8.1.1 Window Layout

┌─────────────────────────────────────────────────────────┐  
│ EKVAYU - Device Registration [\_][□][X]│  
├─────────────┬───────────────────────────────────────────┤  
│ │ │  
│ Sidebar │ Main Content Area │  
│ │ │  
│ Step 0 │ Welcome Screen │  
│ Step 1 │ - Title │  
│ Step 2 │ - Description │  
│ Step 3 │ - Instructions │  
│ Step 4 │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ │  
│ │ [Previous] [Next] │  
└─────────────┴───────────────────────────────────────────┘

#### 8.1.2 Color Scheme

| **Element** | **Color** | **Hex Code** |
| --- | --- | --- |
| Primary | Blue | #3749A6 |
| Secondary | Light Blue | #5A6FB8 |
| Success | Green | #00AA00 |
| Error | Red | #AA0000 |
| Background | White | #FFFFFF |
| Text | Dark Gray | #333333 |
| Border | Light Gray | #CCCCCC |

#### 8.1.3 Typography

| **Element** | **Font** | **Size** | **Weight** |
| --- | --- | --- | --- |
| Title | Segoe UI | 24pt | Bold |
| Subtitle | Segoe UI | 14pt | Normal |
| Body Text | Segoe UI | 11pt | Normal |
| Button Text | Segoe UI | 11pt | Normal |
| Input Text | Segoe UI | 11pt | Normal |

### 8.2 Linux CLI Design

#### 8.2.1 Output Format

🔧 Device Registration Tool  
==================================================  
Please enter your email address: user@gmail.com  
✅ Email registered: user@gmail.com  
Please enter your phone number: 1234567890  
✅ Phone number registered: 1234567890  
  
--------------------------------------------------  
Collecting Device Information...  
--------------------------------------------------  
Device Identifiers:  
UUID: 12345678-1234-1234-1234-123456789012  
MAC Address: 00:11:22:33:44:55  
Serial Number: ABC123  
OS Type: linux  
OS Platform: linux  
OS Release: Ubuntu 22.04.3 LTS  
Hostname: user-pc  
Architecture: amd64  
Current Version: 1.0.0  
Browsers:  
 Chrome: Installed=true, Version=120.0.6099.109  
 Firefox: Installed=true, Version=121.0  
 Edge: Installed=false, Version=N/A  
  
--------------------------------------------------  
--------------------------------------------------  
✅ Device registration sent successfully!  
  
==================================================  
Press Enter to exit...  
==================================================

#### 8.2.2 Error Messages

❌ Please enter a valid email address from allowed domains  
 (gmail.com, yahoo.com, outlook.com, gov.in, ekvayu.com)  
  
❌ Phone number must contain 8-10 digits  
  
❌ Server is down or check your internet connection, try again later  
  
❌ Error: [specific error message from backend]

## 9. Data Collection

### 9.1 Device Information Fields

| **Field** | **Description** | **Example** | **Required** |
| --- | --- | --- | --- |
| UUID | System UUID | 12345678-1234-1234-1234-123456789012 | Yes |
| MacAddress | Primary MAC address | 00:11:22:33:44:55 | Yes |
| SerialNumber | System serial number | ABC123 | Yes |
| OSType | Operating system type | Windows\_NT / linux | Yes |
| OSPlatform | OS platform | windows / linux | Yes |
| OSRelease | OS release version | Windows 11 / Ubuntu 22.04 | Yes |
| HostName | Computer hostname | USER-PC / user-pc | Yes |
| Architecture | CPU architecture | amd64 / arm64 | Yes |
| CurrentVersion | Agent version | 1.0.0 | Yes |
| Email | User email | user@gmail.com | Yes |
| PhoneNumber | User phone | 1234567890 | Yes |
| Browsers | Browser info | {chrome, firefox, edge} | Yes |

### 9.2 Browser Information

type Browser struct {  
 Installed bool `json:"installed"`  
 Version string `json:"version,omitempty"`  
}

**Example**:

{  
 "chrome": {  
 "installed": true,  
 "version": "120.0.6099.109"  
 },  
 "firefox": {  
 "installed": true,  
 "version": "121.0"  
 },  
 "edge": {  
 "installed": false  
 }  
}

### 9.3 Data Collection Flow

Start Collection  
 ↓  
Get UUID  
 ↓  
Get MAC Address  
 ↓  
Get Serial Number  
 ↓  
Get OS Information  
 ↓  
Get Hostname  
 ↓  
Get Architecture  
 ↓  
Detect Chrome  
 ↓  
Detect Firefox  
 ↓  
Detect Edge  
 ↓  
Format Data Structure  
 ↓  
Return DeviceIdentifiers

## 10. API Integration

### 10.1 Backend Endpoint

**URL**: http://3.109.178.115:10101/plugin/agent-requests/

**Method**: POST

**Content-Type**: application/json

### 10.2 Request Format

{  
 "email": "user@gmail.com",  
 "phone\_number": "1234567890",  
 "uuid": "12345678-1234-1234-1234-123456789012",  
 "macAdress": "00:11:22:33:44:55",  
 "serialNumber": "ABC123",  
 "osType": "Windows\_NT",  
 "osPlatform": "windows",  
 "osRelease": "Windows 11",  
 "hostName": "USER-PC",  
 "architecture": "amd64",  
 "current\_version": "1.0.0",  
 "browsers": {  
 "chrome": {  
 "installed": true,  
 "version": "120.0.6099.109"  
 },  
 "firefox": {  
 "installed": false  
 },  
 "edge": {  
 "installed": true,  
 "version": "120.0.2210.91"  
 }  
 }  
}

### 10.3 Response Handling

#### 10.3.1 Success Response (200-299)

{  
 "status": "success",  
 "message": "Device registered successfully",  
 "data": {  
 "device\_id": "12345",  
 "registration\_date": "2025-01-07T10:30:00Z"  
 }  
}

#### 10.3.2 Error Response (400-599)

{  
 "status": "error",  
 "message": "Device already registered",  
 "code": "DUPLICATE\_DEVICE"  
}

### 10.4 Error Handling

func postDeviceInfo(email, phoneNumber string, deviceInfo DeviceIdentifiers) error {  
 // ... create request ...  
   
 resp, err := client.Do(req)  
 if err != nil {  
 // Network errors  
 if strings.Contains(err.Error(), "network is unreachable") ||  
 strings.Contains(err.Error(), "connection refused") ||  
 strings.Contains(err.Error(), "no such host") ||  
 strings.Contains(err.Error(), "timeout") {  
 return fmt.Errorf("server is down or check your internet connection")  
 }  
 return fmt.Errorf("error sending request: %v", err)  
 }  
 defer resp.Body.Close()  
   
 // Success  
 if resp.StatusCode >= 200 && resp.StatusCode < 300 {  
 return nil  
 }  
   
 // Parse error response  
 var errorResponse map[string]interface{}  
 if json.Unmarshal(responseBody, &errorResponse) == nil {  
 if message, exists := errorResponse["message"]; exists {  
 return fmt.Errorf("%s", message)  
 }  
 }  
   
 return fmt.Errorf("server returned error: %s", resp.Status)  
}

## 11. Validation & Error Handling

### 11.1 Input Validation

#### 11.1.1 Email Validation Rules

1. **Format**: Must contain @ symbol
2. **Domain**: Must be from whitelist
   * gmail.com
   * yahoo.com
   * outlook.com
   * gov.in
   * ekvayu.com
3. **Length**: At least 3 characters before @
4. **Case**: Case-insensitive domain matching

#### 11.1.2 Phone Validation Rules

1. **Length**: 8-10 digits
2. **Characters**: Only digits (0-9)
3. **Format**: Can include spaces, hyphens, parentheses (stripped for validation)

### 11.2 Error Categories

#### 11.2.1 Network Errors

| **Error** | **Message** | **Action** |
| --- | --- | --- |
| No Internet | “Check your internet connection” | Retry |
| Server Down | “Server is down, try again later” | Retry later |
| Timeout | “Request timeout, try again” | Retry |
| DNS Failure | “Cannot reach server” | Check network |

#### 11.2.2 Validation Errors

| **Error** | **Message** | **Action** |
| --- | --- | --- |
| Invalid Email | “Please enter valid email from allowed domains” | Re-enter |
| Invalid Phone | “Phone must contain 8-10 digits” | Re-enter |
| Empty Field | “Field cannot be empty” | Fill field |

#### 11.2.3 Server Errors

| **Error** | **Message** | **Action** |
| --- | --- | --- |
| Duplicate Device | “Device already registered” | Contact support |
| Invalid Data | “Invalid device information” | Retry |
| Server Error | “Server error occurred” | Retry later |

### 11.3 Error Recovery

// Retry mechanism for network errors  
func postDeviceInfoWithRetry(email, phone string, deviceInfo DeviceIdentifiers, maxRetries int) error {  
 var lastErr error  
   
 for i := 0; i < maxRetries; i++ {  
 err := postDeviceInfo(email, phone, deviceInfo)  
 if err == nil {  
 return nil // Success  
 }  
   
 lastErr = err  
   
 // Check if error is retryable  
 if isNetworkError(err) {  
 time.Sleep(time.Duration(i+1) \* 2 \* time.Second)  
 continue  
 }  
   
 // Non-retryable error  
 return err  
 }  
   
 return lastErr  
}

## 12. Security Implementation

### 12.1 Data Privacy

#### 12.1.1 Sensitive Data Handling

* Email and phone number collected with user consent
* No password or financial information collected
* Data transmitted over HTTP (should be HTTPS in production)
* No local storage of sensitive data

#### 12.1.2 Network Information Sanitization

func cleanNetworkInfo(message string) string {  
 // Remove IP addresses  
 message = regexp.MustCompile(`\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`).  
 ReplaceAllString(message, "[IP\_HIDDEN]")  
   
 // Remove port numbers  
 message = regexp.MustCompile(`:\d{4,5}`).  
 ReplaceAllString(message, ":[PORT]")  
   
 // Remove URLs  
 message = regexp.MustCompile(`https?://[^\s]+`).  
 ReplaceAllString(message, "[URL\_HIDDEN]")  
   
 return message  
}

### 12.2 Input Sanitization

#### 12.2.1 Email Sanitization

func sanitizeEmail(email string) string {  
 // Trim whitespace  
 email = strings.TrimSpace(email)  
   
 // Convert to lowercase  
 email = strings.ToLower(email)  
   
 // Remove any control characters  
 email = strings.Map(func(r rune) rune {  
 if r < 32 || r == 127 {  
 return -1  
 }  
 return r  
 }, email)  
   
 return email  
}

#### 12.2.2 Phone Sanitization

func sanitizePhone(phone string) string {  
 // Extract digits only  
 digits := ""  
 for \_, char := range phone {  
 if char >= '0' && char <= '9' {  
 digits += string(char)  
 }  
 }  
 return digits  
}

### 12.3 Secure Communication

#### 12.3.1 HTTP Client Configuration

client := &http.Client{  
 Timeout: 30 \* time.Second,  
 Transport: &http.Transport{  
 TLSClientConfig: &tls.Config{  
 MinVersion: tls.VersionTLS12,  
 // In production, enable certificate verification  
 },  
 },  
}

#### 12.3.2 Request Headers

req.Header.Set("Content-Type", "application/json")  
req.Header.Set("User-Agent", "Device-Info-Collector/1.0.0")  
req.Header.Set("Accept", "application/json")

### 12.4 Security Best Practices

1. **Use HTTPS**: All production communication should use HTTPS
2. **Certificate Validation**: Enable TLS certificate verification
3. **Input Validation**: Validate all user inputs before processing
4. **Error Messages**: Don’t expose sensitive information in error messages
5. **Timeout**: Set reasonable timeouts for network requests
6. **No Credentials**: Don’t store or transmit passwords
7. **Minimal Permissions**: Request only necessary system permissions

## 13. Deployment Architecture

### 13.1 Build Process

#### 13.1.1 Windows Build

# GUI version (recommended for end users)  
go build -ldflags="-H windowsgui -s -w" -o mini\_agent.exe main\_gui\_window.go  
  
# Console version (for debugging)  
go build -o mini\_agent\_console.exe main\_gui\_window.go  
  
# With icon embedding  
go build -ldflags="-H windowsgui -s -w" -o mini\_agent.exe main\_gui\_window.go

**Build Flags**: - -H windowsgui: Hide console window - -s: Strip symbol table - -w: Strip DWARF debugging information - Reduces binary size by ~30%

#### 13.1.2 Linux Build

# Standard build  
go build -o mini\_agent main.go  
  
# Optimized build  
go build -ldflags="-s -w" -o mini\_agent main.go  
  
# Static binary (no external dependencies)  
CGO\_ENABLED=0 go build -ldflags="-s -w" -o mini\_agent main.go

### 13.2 Distribution

#### 13.2.1 Windows Distribution

**Package Contents**:

mini\_agent\_windows.zip  
├── mini\_agent.exe  
├── README.txt  
└── LICENSE.txt

**Installation Instructions**: 1. Download mini\_agent\_windows.zip 2. Extract to any folder 3. Double-click mini\_agent.exe 4. Follow on-screen instructions 5. No installation required

#### 13.2.2 Linux Distribution

**Package Contents**:

mini\_agent\_linux.tar.gz  
├── mini\_agent  
├── README.txt  
└── LICENSE.txt

**Installation Instructions**:

# Extract  
tar -xzf mini\_agent\_linux.tar.gz  
  
# Make executable  
chmod +x mini\_agent  
  
# Run  
./mini\_agent

### 13.3 System Requirements

#### 13.3.1 Windows Requirements

* **OS**: Windows 10 or later
* **RAM**: 50 MB
* **Disk**: 10 MB
* **Network**: Internet connection required
* **Privileges**: No administrator rights required

#### 13.3.2 Linux Requirements

* **OS**: Any modern Linux distribution
* **RAM**: 50 MB
* **Disk**: 10 MB
* **Network**: Internet connection required
* **Privileges**: No root required (except for dmidecode)
* **Dependencies**: None (static binary)

### 13.4 Deployment Scenarios

#### 13.4.1 Individual User Deployment

1. User downloads executable
2. User runs executable
3. User enters email and phone
4. Device registered automatically
5. User closes application

#### 13.4.2 Bulk Deployment (Enterprise)

**Windows (Group Policy)**:

# Deploy via GPO startup script  
\\server\share\mini\_agent.exe

**Linux (Ansible)**:

- name: Deploy Mini Agent  
 copy:  
 src: mini\_agent  
 dest: /usr/local/bin/mini\_agent  
 mode: '0755'  
  
- name: Run Mini Agent  
 command: /usr/local/bin/mini\_agent  
 environment:  
 EMAIL: "{{ user\_email }}"  
 PHONE: "{{ user\_phone }}"

### 13.5 Update Process

#### 13.5.1 Version Check

const CurrentVersion = "1.0.1"  
  
func checkForUpdates() (string, error) {  
 resp, err := http.Get("https://server/api/version")  
 if err != nil {  
 return "", err  
 }  
 defer resp.Body.Close()  
   
 var versionInfo struct {  
 LatestVersion string `json:"latest\_version"`  
 DownloadURL string `json:"download\_url"`  
 }  
   
 json.NewDecoder(resp.Body).Decode(&versionInfo)  
   
 if versionInfo.LatestVersion != CurrentVersion {  
 return versionInfo.DownloadURL, nil  
 }  
   
 return "", nil // No update available  
}

## 14. Appendix

### 14.1 Glossary

| **Term** | **Definition** |
| --- | --- |
| Mini Agent | Lightweight device registration tool |
| Device Registration | Process of registering device with backend |
| UUID | Universally Unique Identifier for system |
| MAC Address | Media Access Control address for network interface |
| dmidecode | Linux tool for reading system DMI/SMBIOS information |
| WMI | Windows Management Instrumentation |
| CLI | Command-Line Interface |
| GUI | Graphical User Interface |

### 14.2 Constants Reference

const (  
 LicenseVerificationURL = "http://3.109.178.115:10101/plugin/agent-requests/"  
 AppTitle = "EKVAYU"  
 AppVersion = "1.0.1"  
 WINDOW\_WIDTH = 850  
 WINDOW\_HEIGHT = 600  
 SIDEBAR\_WIDTH = 200  
)

### 14.3 Allowed Email Domains

allowedDomains := []string{  
 "gmail.com",  
 "yahoo.com",  
 "outlook.com",  
 "gov.in",  
 "ekvayu.com",  
}

### 14.4 API Endpoint Summary

| **Endpoint** | **Method** | **Purpose** | **Auth** |
| --- | --- | --- | --- |
| /plugin/agent-requests/ | POST | Register device | No |

### 14.5 Error Codes

| **Code** | **Message** | **Description** |
| --- | --- | --- |
| 200 | Success | Device registered successfully |
| 400 | Bad Request | Invalid data format |
| 409 | Conflict | Device already registered |
| 500 | Server Error | Internal server error |
| 503 | Service Unavailable | Server is down |

### 14.6 File Locations

**Windows**:

C:\Users\<username>\Downloads\mini\_agent.exe

**Linux**:

/home/<username>/Downloads/mini\_agent  
/usr/local/bin/mini\_agent (system-wide)

### 14.7 Browser Detection Commands

**Windows**:

# Chrome  
(Get-Item 'C:\Program Files\Google\Chrome\Application\chrome.exe').VersionInfo.ProductVersion  
  
# Firefox  
(Get-Item 'C:\Program Files\Mozilla Firefox\firefox.exe').VersionInfo.ProductVersion  
  
# Edge  
(Get-Item 'C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe').VersionInfo.ProductVersion

**Linux**:

# Chrome  
google-chrome --version  
  
# Firefox  
firefox --version  
  
# Edge  
microsoft-edge --version

### 14.8 System Information Commands

**Windows**:

# UUID  
wmic csproduct get UUID  
  
# MAC Address  
getmac  
  
# Serial Number  
wmic bios get serialnumber  
  
# OS Version  
ver

**Linux**:

# UUID  
sudo dmidecode -s system-uuid  
cat /sys/class/dmi/id/product\_uuid  
  
# MAC Address  
ip link show  
  
# Serial Number  
sudo dmidecode -s system-serial-number  
cat /sys/class/dmi/id/product\_serial  
  
# OS Version  
cat /etc/os-release  
lsb\_release -a

### 14.9 Troubleshooting Guide

#### 14.9.1 Common Issues

**Issue**: Application won’t start

Solution:  
- Windows: Check if antivirus is blocking  
- Linux: Check if executable permission is set (chmod +x)  
- Both: Verify system requirements

**Issue**: Cannot collect device information

Solution:  
- Windows: Run as administrator for full access  
- Linux: Use sudo for dmidecode commands  
- Both: Check if system commands are available

**Issue**: Network error when sending data

Solution:  
- Check internet connection  
- Verify firewall settings  
- Check if backend server is accessible  
- Try again later if server is down

**Issue**: Email validation fails

Solution:  
- Use email from allowed domains:  
 gmail.com, yahoo.com, outlook.com, gov.in, ekvayu.com  
- Check email format (must have @ symbol)  
- Remove any spaces or special characters

**Issue**: Phone validation fails

Solution:  
- Enter 8-10 digits only  
- Remove any letters or special characters  
- Don't include country code

### 14.10 Performance Metrics

| **Metric** | **Target** | **Actual** |
| --- | --- | --- |
| Startup Time | < 2 seconds | ~1 second |
| Data Collection | < 3 seconds | ~2 seconds |
| Network Request | < 5 seconds | ~3 seconds |
| Total Time | < 10 seconds | ~6 seconds |
| Memory Usage | < 50 MB | ~30 MB |
| Binary Size (Windows) | < 10 MB | ~8 MB |
| Binary Size (Linux) | < 10 MB | ~7 MB |

### 14.11 Version History

| **Version** | **Date** | **Changes** |
| --- | --- | --- |
| 1.0.0 | 2024-12-01 | Initial release |
|  |  | - Device info collection |
|  |  | - Email and phone input |
|  |  | - Backend registration |
| 1.0.1 | 2025-01-07 | Improvements |
|  |  | - Enhanced validation |
|  |  | - Better error messages |
|  |  | - UI improvements |

### 14.12 Future Enhancements

1. **HTTPS Support**: Implement HTTPS for secure communication
2. **Offline Mode**: Cache data and send when online
3. **Multi-Language**: Support for multiple languages
4. **Auto-Update**: Automatic update mechanism
5. **Configuration File**: Support for pre-configured email/phone
6. **Silent Mode**: Command-line arguments for automation
7. **Logging**: Detailed logging for troubleshooting
8. **Retry Mechanism**: Automatic retry on network failures
9. **Progress Bar**: Visual progress indicator
10. **Email Verification**: Send verification code to email

### 14.13 Known Limitations

1. **HTTP Only**: Backend communication is not encrypted (should be HTTPS)
2. **No Offline Support**: Requires internet connection
3. **Limited Email Domains**: Only 5 domains whitelisted
4. **No Auto-Update**: Manual download required for updates
5. **Single Execution**: No persistent service or daemon
6. **No Configuration**: All settings hardcoded
7. **Limited Error Recovery**: No automatic retry mechanism
8. **No Logging**: No persistent logs for troubleshooting

### 14.14 Comparison with BPP Agent

| **Feature** | **Mini Agent** | **BPP Agent** |
| --- | --- | --- |
| **Purpose** | Device registration | License management |
| **Installation** | Portable executable | System service |
| **User Input** | Email + Phone | Minimal |
| **Service** | No | Yes (Windows Service / systemd) |
| **Extensions** | No | Yes (Chrome, Firefox, Edge) |
| **License Check** | No | Yes (periodic) |
| **API Server** | No | Yes (port 64321) |
| **GUI** | Windows only | Windows only |
| **Size** | ~8 MB | ~15 MB |
| **Complexity** | Simple | Complex |
| **Use Case** | Initial registration | Ongoing management |

### 14.15 Dependencies

#### 14.15.1 Go Dependencies

require (  
 golang.org/x/sys v0.0.0-20220715151400-c0bba94af5f8 // Windows only  
)

#### 14.15.2 System Dependencies

**Windows**: - Windows 10 or later - PowerShell 5.0 or later (for browser detection)

**Linux**: - bash - dmidecode (optional, for UUID and serial number) - Standard Linux utilities (cat, grep, etc.)

## Document Revision History

| **Version** | **Date** | **Changes** | **Author** |
| --- | --- | --- | --- |
| 1.0 | 2025-01-07 | Initial document creation | System Analysis |

## Approval

| **Role** | **Name** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Technical Lead |  |  |  |
| Backend Architect |  |  |  |
| Project Manager |  |  |  |

**End of Document**