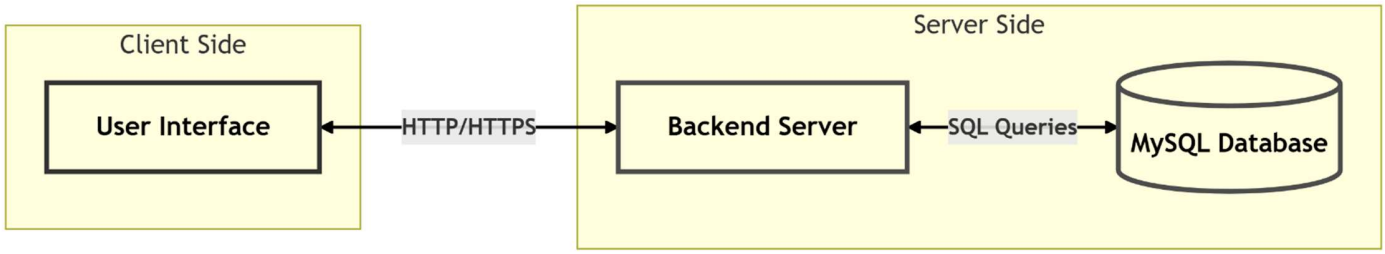
Design Document – Firmware Installer

1. System Architecture Overview

TekMedika Firmware Updater is a comprehensive tool designed to simplify firmware installation for Arduino Mega and NodeMCU boards. The system is divided into three main components:

1. **Installer**: A Python-based Desktop GUI application that provides an intuitive interface for users to install firmware on master and slave boards.
2. **Backend Server**: A Node.js/Express-based server that records firmware installation details and provides APIs for interaction.
3. **Database**: A MySQL database that stores firmware installation records, including system UID, firmware version, and timestamps.

High-Level Architecture



---

2. Components

2.1 Installer

- **Technology**: Python (Tkinter for GUI)
- **Purpose**: Provides a user-friendly interface for firmware installation, version selection, and board management.
- **Key Features**:
  - Detects connected boards (Master/Slave) via USB.
  - Downloads firmware files dynamically from a GitHub repository.
  - Installs firmware on the boards using arduino-cli and esptool.
  - Displays progress and logs for each step.
  - Registers firmware installation details with the backend server.

2.2 Backend Server

- **Technology**: Node.js with Express.js
- **Purpose**: Handles API requests from the installer and interacts with the database.
- **Key Features**:
  - Records firmware installation details (system UID, firmware version, timestamp).

- Provides endpoints for retrieving installation history.
- Secures API interactions using a hashed secret key.

2.3 Database

- **Technology**: MySQL
- **Purpose**: Stores firmware installation records.
- **Schema**:
  - **Table**: firmware_installations

Field	Type	Description
id	INT (PK, AUTO_INCREMENT)	Unique record ID
system_uid	VARCHAR	Unique identifier for the system
firmware_version	VARCHAR	Installed firmware version
installation_timestamp	TIMESTAMP	Timestamp of the installation
verification_status	BOOLEAN	Whether the installation was verified

---

3. Firmware Installation Flow

3.1 Login Process

1. The user is prompted to log in with one of the following credentials:
   - **Admin Login**:
     - Email: admin
     - Password: admin
     - **Features**:
       - Install any version of the firmware.
       - Erase the flash memory of both Master (Arduino Mega) and Slave (NodeMCU) boards.
     - Access to advanced options.
   - **User Login**:
     - Email: user
     - Password: user
     - **Features**:
       - Install only the latest firmware version.
       - No access to flash erase functionality.

3.2 Firmware Selection

1. After logging in, the user selects the firmware version to install from a dropdown menu.
   - **Admin**: Can select any available firmware version.
   - **User**: Restricted to the latest firmware version.
2. The selected firmware files are downloaded dynamically from the GitHub repository.

3.3 Master Board (Arduino Mega) Installation

1. Detect the connected Arduino Mega board using serial.tools.list_ports.
2. Use arduino-cli to:
   - Compile the firmware file (.hex).
   - Upload the firmware to the board.
3. Verify the installation by:
   - Extracting the system UID and firmware version via serial communication.
   - Displaying the verification status to the user.

3.4 Slave Board (NodeMCU) Installation

1. Detect the connected NodeMCU board using serial.tools.list_ports.
2. Use arduino-cli to:
   - Flash the firmware file (.bin) to the board.
3. Verify the installation by:
   - Extracting the system UID and firmware version via serial communication.
   - Displaying the verification status to the user.

3.5 Flash Erase (Admin Only)

1. Admin users have the option to erase the flash memory of both Master and Slave boards.
2. The process involves:
   - Sending specific commands to the boards to clear their memory.
   - Confirming the success of the erase operation.

---

4. UID & Version Extraction Logic

- **Purpose**: To uniquely identify the system and verify the installed firmware version.
- **Process**:
  1. Establish a serial connection with the board.
  2. Send specific commands:
     - H: Handshake to ensure communication.
     - U: Retrieve the system UID.
     - V: Retrieve the firmware version.
  3. Parse the responses and return the extracted values.
- **Implementation**: The logic is implemented in the get_device_info function in api.py.

---

5. API Flow and Database Interaction

5.1 API Endpoints

1. **POST /api/firmware-installation**
   - **Purpose**: Record firmware installation details.
   - **Request Body**:

```
{
  "system_uid": "unique-system-id",
  "firmware_version": "x.x.x",
  "verification_status": true,
}
```

- **Response**: Success or error message.

2. **GET /api/firmware-installation/:system_uid**
   - **Purpose**: Retrieve firmware installation history for a specific system UID.
   - **Response**: List of installation records.

5.2 Database Interaction

- **Insert Operation**: When a firmware installation is completed, the installer sends a POST request to the backend, which inserts the details into the firmware_installations table.
- **Query Operation**: The backend retrieves installation history for a given system UID using a SELECT query.

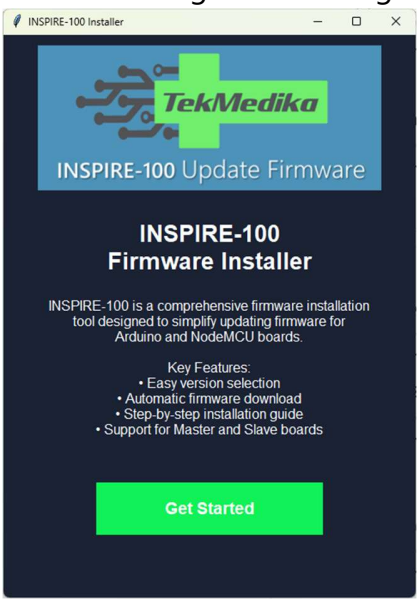
---

6. Project Structure

```
TekMedika-Firmware-Updater/
├── main.py              # Main entry point of the application
├── src/
│   ├── arduino/         # Arduino-specific scripts
│   │   ├── arduino.py
│   ├── nodemcu/         # NodeMCU-specific scripts
│   │   ├── nodemcu.py
│   ├── ui/              # UI components
│   │   ├── home_page.py
│   │   ├── login_page.py
│   │   ├── version_selection_page.py
│   │   ├── download_screen.py
│   │   ├── run_installation.py
│   │   ├── erasing_pages.py
│   │   ├── utils.py
│   │   ├── assets/      # UI assets
│   ├── backend/         # Backend API logic
│   │   ├── api.py
│   ├── config/          # Configuration files
│   │   ├── config.py
│   │   ├── color.py
│   ├── utils/           # Utility functions
│   │   ├── utils.py
├── backend/             # Node.js backend server
│   ├── app.js
│   ├── package.json
├── requirements.txt     # Python dependencies
├── setup_script.iss     # Setup script for installer
```

7. UI Flow

7.1 Home Page

- **Purpose**: Welcome screen with a brief description of the tool.
- **Actions**: Navigate to the login page.
- 

7.2 Login Page

- **Purpose**: Authenticate users (Admin/User).
- **Actions**: Validate credentials and navigate to the version selection page.
- 

7.3 Version Selection Page

- **Purpose**: Allow users to select the firmware version to install.
- **Actions**:
  - Admins can access additional erase options.

- Users can proceed to download the selected firmware.

7.4 Download Screen

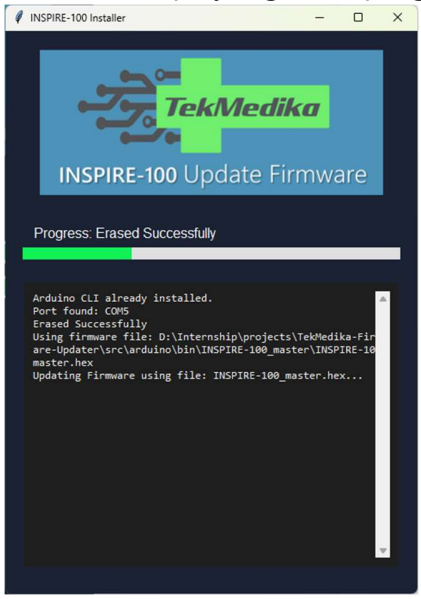
- **Purpose**: Display download progress for firmware files.
- **Actions**: Proceed to connection instructions after download completion.

7.5 Connection Instructions

- **Purpose**: Guide users to connect the appropriate board (Master/Slave).
- **Actions**: Start the installation process.

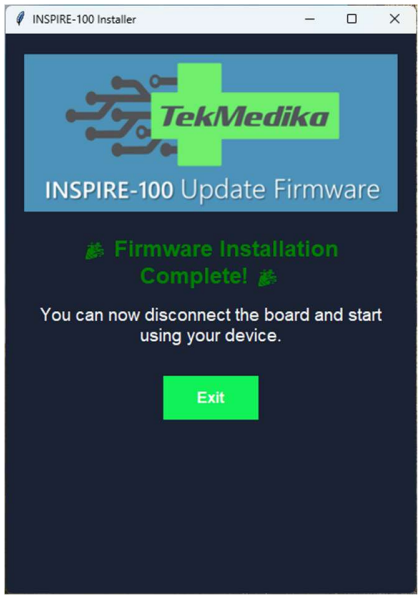
- 

7.6 Installation Progress

- **Purpose**: Show progress of firmware installation.
- **Actions**: Display logs and progress bar.
- 

7.7 Completion Screen

- **Purpose**: Confirm successful installation.
- **Actions**: Exit the application or restart the process.

- 

---

8. Python Dependencies (with Versions)

altgraph==0.17.4
certifi==2025.1.31
charset-normalizer==3.4.1
future==1.0.0
idna==3.10
iso8601==2.1.0
packaging==24.2
pefile==2023.2.7
pillow==11.1.0
psutil==7.0.0
pyinstaller==6.12.0
pyinstaller-hooks-contrib==2025.2
pyserial==3.5
pywin32-ctypes==0.2.3
PyYAML==6.0.2
requests==2.32.3
setuptools==78.1.0
urllib3==2.3.0