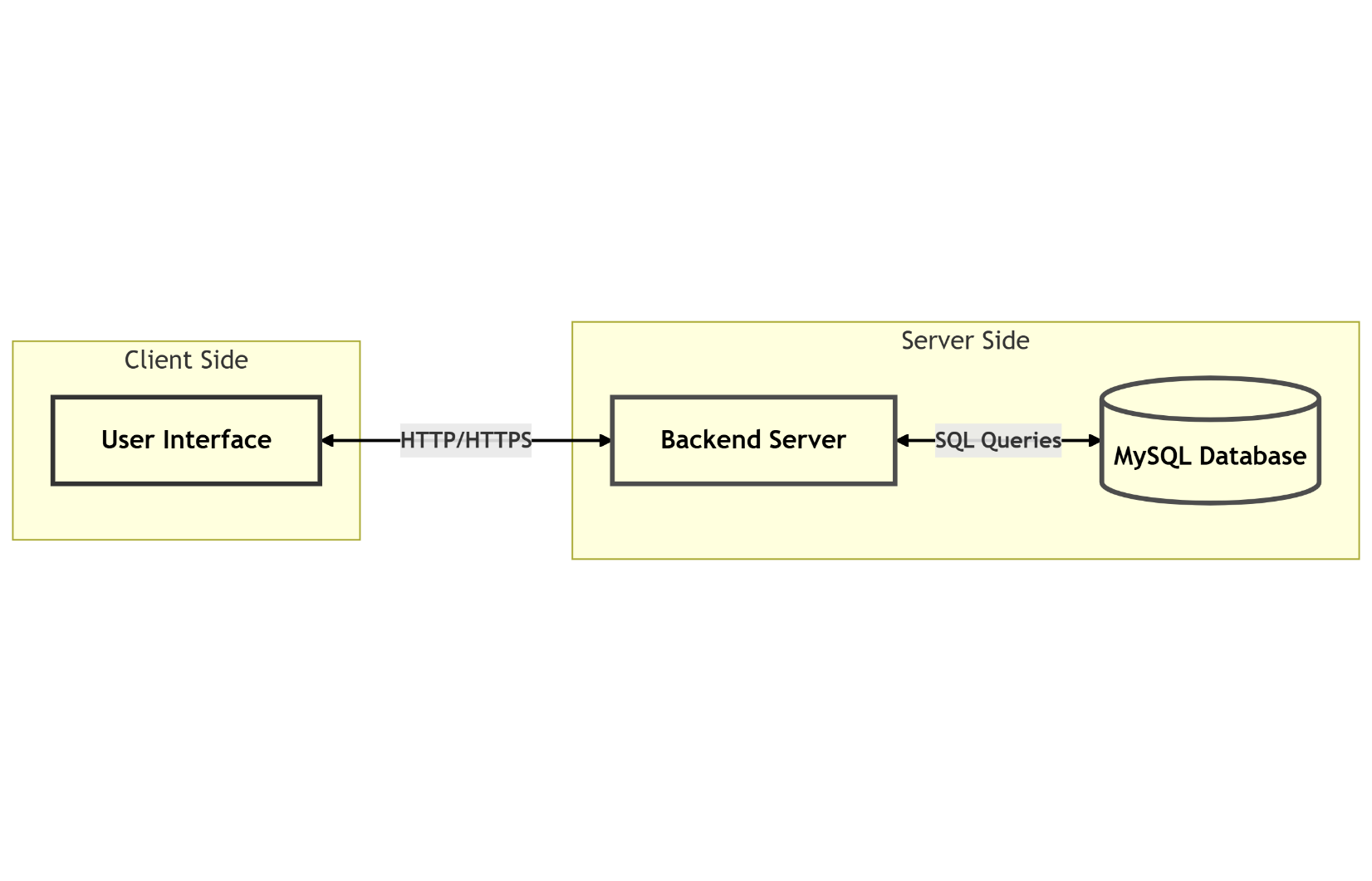
# Design Document – Firmware Installer

## 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 as well as update or install UID into the system.
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 Home Page – Update Check

1. When the application launches, users are directed to the **Home Page**.
2. The app automatically checks for the latest version using the firmware/appReleases API.
3. If a newer version is available:
   * A notification is shown to the user with a prompt to update.
4. After the update check, users are directed to the **Firmware Selection Page**.

### 3.2 Admin Login

1. On the **Firmware Selection Page**, an **"Login"** button is available.
2. Clicking the button opens a login prompt with the following credentials:
   * **Email:** admin
   * **Password:** admin
3. On successful login, the user enters **Admin Mode** with access to additional features:
   * Ability to select and install **any** firmware version.
   * Access to **Flash Erase** functionality for both boards.
   * Advanced options become visible in the UI.

### 3.3 Firmware Selection

1. On the **Firmware Selection Page**, users select the firmware version from a dropdown menu.
   * **Default User Mode:** Can only install the **latest** firmware version.
   * **Admin Mode:** Can select **any available** firmware version.
2. The selected firmware files are dynamically downloaded from the GitHub repository.

### 3.4 Master Board (Arduino Mega) Installation

1. The application detects the connected **Arduino Mega** board using serial.tools.list\_ports.
2. Using arduino-cli, the following actions are performed:
   * **Upload** the compiled firmware to the board.
3. After installation, the app verifies the operation by:
   * Extracting the **System UID** and **Firmware Version** via serial communication.
   * Displaying the verification status to the user.

### 3.5 Slave Board (NodeMCU) Installation

1. The application detects the connected **NodeMCU** board using serial.tools.list\_ports.
2. Using arduino-cli, the following actions are performed:
   * **Flash** the selected firmware file (.bin) to the board.
3. After installation, the app verifies the operation by:
   * Extracting the **System UID** and **Firmware Version** via serial communication.
   * Displaying the verification status to the user.

### 3.6 Flash Erase (Admin Only)

1. **Admin Mode** users can access the **Flash Erase** feature for both boards.
2. The process includes:
   * Uploading a Empty file to the **Master** and **Slave** boards to clear their flash memory.
   * Receiving confirmation of the **successful erase** operation from each board.

**3.7 UID Installation Flow**

1. **UID Installation Feature:**
   * Accessed from the Version Selection Page.
   * Requires Admin login.
2. **UID Download Process:**
   * System downloads UID installation firmware from GitHub.
   * Progress is displayed.
3. **Connection Instructions:**
   * Users connect Master board (Arduino Mega) to computer.
   * Safety warnings regarding power are displayed.
4. **UID Installation:**
   * Firmware uploads to Arduino Mega.
   * System automatically generates a unique UID during installation.
5. **Barcode Generation:**
   * Upon success, a Code128 barcode is generated for the UID.
   * Barcode can be saved for future reference.
6. **API Integration:**
   * If a previous UID existed, it's deprecated in the database (verification\_status set to false).
   * The new UID is registered in the database.
7. **Completion:**
   * Success screen confirms UID installation and displays new UID.
   * Users can return to the main firmware selection screen.

## 4. UID & Version Extraction Logic

* **Purpose**: To uniquely identify the system and verify the installed firmware version.
* **Process**:
  + Establish a serial connection with the board.
  + Send specific commands:
    - H: Handshake to ensure communication.
    - U: Retrieve the system UID.
    - V: Retrieve the firmware version.
  + 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.

1. **GET /api/firmware-installation/:system\_uid**
   * **Purpose**: Retrieve firmware installation history for a specific system UID.
   * **Response**: List of installation records.
2. **POST /api/deprecate-uid**
   * **Purpose:** Deprecate previous UID records (sets verification\_status to false).
   * **Request Body:**

{

"old\_system\_uid": "unique-system-id-to-deprecate",

"secret": "hashed-secret-key"

}

* + **Response:** Count of deprecated records

1. **GET /download-excel**
   * **Purpose:** Downloads all firmware installation records in **Excel format**.
   * **Response:** File: firmware\_installations.xlsx
   * **Excel Columns:**

| ID |
| --- |
| System UID |
| Firmware Versio |
| Installation Timestamp |
| Verification Status (shown as **Verified** or **Not Verified**) |

### 5.2 Backend GUI

#### /view-installations

* **Purpose:** Renders an HTML table view of all firmware installation records.
* **Displayed Columns:**
  + ID
  + System UID
  + Firmware Version
  + Installation Timestamp
  + Verification Status

### 5.3 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.
* **Update Operation**: When a new firmware is installed on an existing system, the backend updates the existing record with the new firmware version.
* **Deprecate Operation**: When a new UID is installed, all previous records for the old UID are marked as deprecated by setting verification\_status to false.
* **Query Operation**: The backend retrieves installation history for a given system UID using a SELECT query.

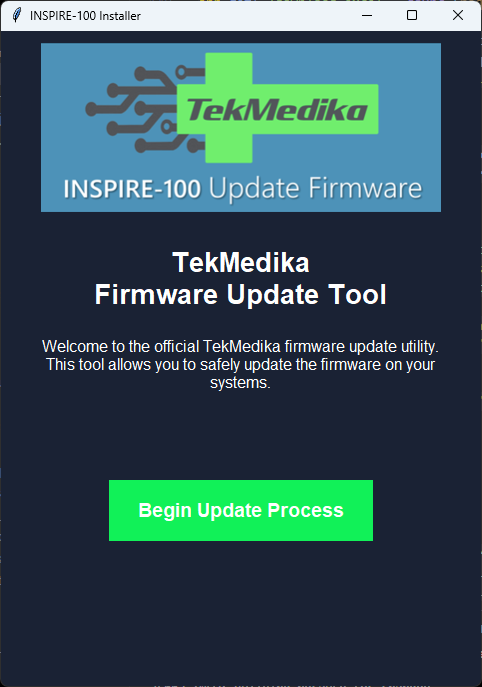
## 6. Project Structure

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  │ ├── uid\_installation/ # UID installation components  │ │ ├── barcode\_screen.py  │ │ ├── completion\_screen.py  │ │ ├── download\_screen.py  │ │ ├── uid\_connection\_instructions.py  │ │ ├── uid\_installation\_screen.py │ ├── backend/ # Backend API logic │ │ ├── api.py │ ├── config/ # Configuration files │ │ ├── config.py │ │ ├── color.py │ ├── utils/ # Utility functions │ │ ├── [utils.py](http://utils.py)  │ │ ├── erasing.py  │ ├── bin/ # Firmware binary files  │ │ ├── InstallSystemUid.ino.mega.hex  │ │ ├── empty/  │ │ ├── empty.ino │── backend/ # Node.js backend server │ ├── app.js │ ├── package.json  │ ├── views/  │ │ ├── home.ejs  │ │ ├── table.ejs │── requirements.txt # Python dependencies │── setup\_script.iss # Setup script for installer |
| --- |

## 7. UI Flow

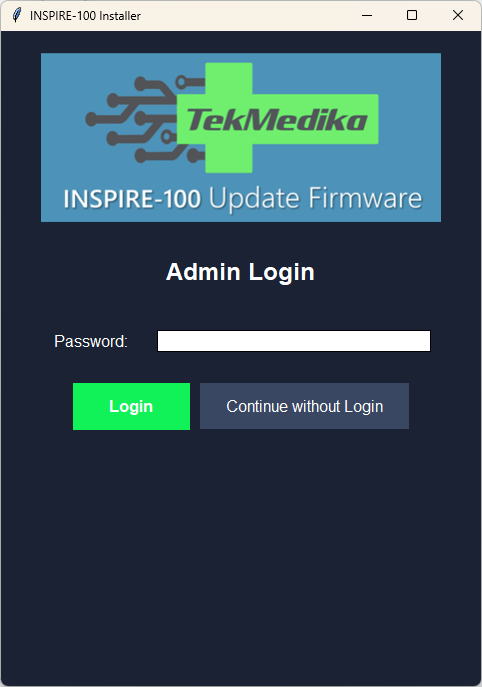
### 7.1 Home Page

* **Purpose**: Welcome screen with a brief description of the tool.
* **Actions**: Navigate to the login page.
* * Home Page when the app is Latest*
* * Home Page when the app is not Latest*

### 7.2 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.
* * Default Version Selection Page*
*  *Version Selection Page after admin login*

### 7.3 Login Page

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

### 7.4 Download Screen

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

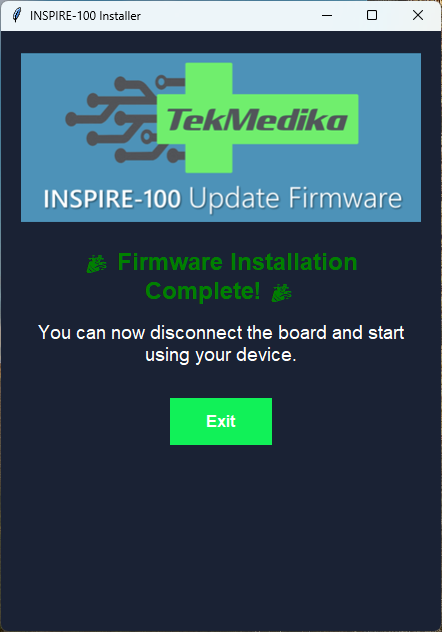
### 7.5 Connection Instructions

* **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.
* 

**7.8 UID Installation Screens**

**7.8.1 UID Selection from Version Page**

• **Purpose**: Provide access to the UID installation feature.

• **Actions**: Admin users can select the "Install System UID" button and trigger the UID installation flow.

**7.8.2 UID Download Screen**

• **Purpose**: Download the required UID installation firmware.

• **Actions**: Display progress of the UID firmware download and automatically proceed to the connection instructions upon completion.

* 

**7.8.3 UID Connection Instructions**

• **Purpose**: Guide users to connect the Master board properly.

• **Actions**: Provide clear instructions with safety warnings, And Allow users to proceed to installation or go back.

* 

**7.8.4 UID Installation Screen**

• **Purpose**: Display the UID installation process.

• **Actions**: Show real-time logs of the installation steps and display progress information.

**7.8.5 UID Completion Screen**

• **Purpose**: Confirm successful UID installation.

• **Actions**: Display confirmation message with the installed UID and allow users to return to the main version selection screen.

## 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