guide
MTS

User’s & Developer’s Documentation

Developed By

AlgoDevOpss

By Yash Balotiya

www.algodevopss.in

www.linkedin.com/in/balotiyash

www.github.com/balotiyash

1. INTRODUCTION

1.1 Overview

The **Guide Motor Training School Management System (Guide MTS)** is a comprehensive desktop application developed to modernize and streamline the operational workflow of motor training schools. Built on the Electron framework (version 37.10.3), this cross-platform application combines the power of web technologies with native desktop capabilities to deliver a robust, secure and user-friendly management solution. The system addresses the critical needs of driving schools by providing an integrated platform for managing students, instructors, vehicles, finances and regulatory documentation.

In the contemporary educational landscape, motor training schools face numerous administrative challenges including student enrolment management, fee tracking, vehicle maintenance records, fuel consumption monitoring and compliance with regulatory requirements such as Form 14 documentation. Traditional paper-based or fragmented digital systems often lead to inefficiencies, data inconsistencies and difficulty in generating timely reports. Guide MTS eliminates these pain points by offering a centralized, database-driven solution that ensures data integrity, operational transparency and regulatory compliance.

1.2 Application Architecture

Guide MTS follows a modern desktop application architecture leveraging:

- **Frontend Technologies**: HTML5, CSS3 (with Grid/Flexbox layouts) and ES6+ JavaScript for a responsive and intuitive user interface.
- **Backend Framework**: Node.js with Electron, enabling seamless integration between the presentation layer and data management layer.
- **Database System**: SQLite3 with Better-SQLite3 driver, providing a lightweight yet powerful relational database solution that requires no separate server installation.
- **Build System**: Electron Builder for packaging and distribution across Windows, macOS and Linux platforms.

- **UI Enhancement Libraries**: Chart.js for analytics visualization and Flatpickr for date selection components.
- **Security Architecture**: Context isolation and preload scripts for secure IPC (Inter-Process Communication) between main and renderer processes.

The application employs a modular architecture with clearly separated concerns: the main process handles database operations and system-level tasks, while renderer processes manage the user interface. This separation ensures better security, maintainability, and scalability.

1.3 Core Functionalities

1.3.1 Student Lifecycle Management

The system provides end-to-end student management capabilities, tracking students from initial registration through license acquisition. It maintains comprehensive records including:

- Personal and contact information
- Learning license details
- Permanent license details
- Fee payment history
- Training progress and milestones

1.3.2 Financial Management

A robust payment tracking system enables:

- Fee collection and receipt generation
- Account management
- Balance reporting with customizable date ranges
- Collection reports for financial auditing
- Invoice generation with professional templates

1.3.3 Vehicle and Resource Management

The application efficiently manages the school's vehicular assets through:

- Master car registry with complete vehicle specifications
- Fuel consumption tracking
- Maintenance scheduling and reminders
- Vehicle assignment to instructors and students

1.3.4 Instructor Management

Dedicated instructor module for:

- Instructor profiles and credentials
- Assignment tracking
- Performance monitoring

1.3.5 Job and Work Management

Comprehensive work order system supporting multiple job types:

- Job master data
- Detailed work descriptions for different job categories
- Task assignment and progress tracking

1.3.6 Dashboard and Analytics

Real-time business intelligence through:

- Interactive charts powered by Chart.js
- Key performance indicators (KPIs) display
- Monthly and custom date range analytics

- Reminder system for upcoming tasks and deadlines
- Visual representation of school operations

1.3.7 Document Generation

Automated creation of official documents:

- Form 14 generation for RTO submissions
- Professional invoices and receipts
- Custom report generation

1.3.8 Search and Retrieval

Advanced search functionality enabling quick access to:

- Student records
- Payment history
- Vehicle information
- Instructor details

1.4 Security Features

Guide MTS implements multiple layers of security:

- **Role-based authentication** with secure login system
- **Context isolation** preventing unauthorized access to Node.js APIs from renderer processes
- **Content Security Policy (CSP)** restricting resource loading to trusted sources
- **Data validation** at both client and server sides
- **Database backup and restore capabilities** ensuring data protection

1.5 Technical Specifications

- **Application Version**: 2.0.0
- **Platform Support**: Windows 10/11, macOS 10.15+, Ubuntu 18.04+
- **Minimum System Requirements**: Node.js 16.0+, 4GB RAM, 500MB disk space
- **Database**: SQLite3 (guide-mts-database.sqlite3)
- **Deployment**: NSIS installer for Windows, DMG for macOS, AppImage for Linux
- **Auto-update Support**: Integrated with electron-updater for seamless updates

1.6 Development Approach

The application follows modern software engineering best practices including:

- Modular code organization with clear separation of concerns
- IPC handler registration for maintainable communication patterns
- Comprehensive logging using electron-log for debugging and monitoring
- Route-based architecture for scalable feature addition
- Utility modules for reusable functionality

1.7 Target Users and Use Cases

Guide MTS is designed for:

- **Motor Training School Administrators**: Managing daily operations, financial tracking and reporting
- **Front Desk Staff**: Student registration, fee collection and appointment scheduling
- **Instructors**: Accessing student information and training schedules
- **Management**: Reviewing analytics, generating reports and making data-driven decisions

1.8 Project Significance

This application represents a significant advancement in motor training school administration by:

- Reducing manual paperwork and administrative overhead
- Minimizing data entry errors through validation
- Providing instant access to critical information
- Ensuring regulatory compliance with automated document generation
- Enabling data-driven decision making through comprehensive analytics
- Offering a cost-effective alternative to expensive cloud-based solutions with one-time installation

The Guide MTS system demonstrates the effective application of modern desktop application development techniques to solve real-world business challenges in the education and training sector.

2. Setup and Installation Instructions

2.1 System Requirements

Before installing the Guide Motor Training School Management System, ensure your computer meets the following minimum requirements:

2.1.1 Hardware Requirements

- **Processor**: Intel Core i3 or equivalent (minimum), Intel Core i5 or higher (recommended)
- **RAM**: 4GB (minimum), 8GB or higher (recommended)
- **Storage**: 500MB free disk space for application installation
- **Additional Storage**: 1GB for database and backups
- **Display**: 1366 x 768 resolution (minimum), 1920 x 1080 (recommended)

2.1.2 Software Requirements

- **Operating System:**
    - Windows 10 or Windows 11 (64-bit)
    - macOS 10.15 (Catalina) or later
    - Ubuntu 18.04 LTS or later (Linux)
- **Node.js**: Version 24.6.0 or higher (for development setup)
- **npm**: Version 11.6.4 or higher (included with Node.js)
- **Electron.js:** Version 37.10.3
- **SQLite3**: Integrated with the application (no separate installation required)

2.1.3 Network Requirements

- Internet connection for initial download and automatic updates
- Local network configuration (optional) for multi-user access scenarios

2.2 Installation Methods

Guide MTS supports two installation approaches based on user requirements:

2.2.1 End-User Installation (Production Environment)

This method is recommended for motor training schools deploying the application for daily operations.

Step 1: Download the Installer

- Visit the official releases page:

  https://github.com/balotiyash/guide-mts-electron/releases
- Download the appropriate installer for your operating system:
    - **Windows:** Guide-MTS-Setup-2.0.0.exe (NSIS Installer) or latest

- **macOS**: Guide-MTS-2.0.0.dmg (Disk Image)
- **Linux**: Guide-MTS-2.0.0.AppImage (AppImage Package)

Step 2: Run the Installation Wizard

For Windows:

1. Double-click the downloaded Guide-MTS-Setup-2.0.0.exe file
2. If prompted by Windows SmartScreen, click "More info" and then "Run anyway"
3. The installation wizard will launch with the following options:
    - Select installation directory (default: C:\Program Files\Guide MTS)
    - Choose whether to create a desktop shortcut
    - Choose whether to create a Start Menu shortcut
4. Click "Install" to begin the installation process
5. Wait for the installation to complete (approximately 1-2 minutes)
6. Click "Finish" to complete the setup

For macOS:

1. Double-click the downloaded Guide-MTS-2.0.0.dmg file
2. Drag the Guide MTS application icon to the Applications folder
3. Eject the DMG file
4. Navigate to Applications and launch Guide MTS
5. If prompted with a security warning, go to System Preferences → Security & Privacy and click "Open Anyway"

For Linux:

1. Make the AppImage executable:
    - chmod +x Guide-MTS-2.0.0.AppImage

2. Run the application:
   - ./Guide-MTS-2.0.0.AppImage
3. Optionally, move the AppImage to /usr/local/bin for system-wide access

Step 3: Initial Launch

1. Launch Guide MTS from:
   - **Windows**: Desktop shortcut or Start Menu
   - **macOS**: Applications folder
   - **Linux**: Command line or application menu
2. The login screen will appear on first launch
3. Use default credentials (if provided by administrator) or contact system administrator for access

Step 4: Database Configuration

- On first launch, the application automatically prompts user to select database file (.sqlite3/.db)
- Database location:
  - **Windows**: C:\Users\[Username]\AppData\Roaming\guide-mts\database\
  - **macOS**: ~/Library/Application Support/guide-mts/database/
  - **Linux**: ~/.config/guide-mts/database/

2.2.2 Developer Installation (Development Environment)

This method is intended for developers who wish to modify, extend, or contribute to the application.

Step 1: Install Prerequisites

1. Download and install Node.js (v24.6.0 or higher) from https://nodejs.org/
2. Verify installation by running:
    - node --version
    - npm --version
3. Install Git for version control (optional but recommended):
    - # Windows: Download from https://git-scm.com/
    - # macOS: brew install git
    - # Linux: sudo apt-get install git

Step 2: Clone the Repository

- git clone https://github.com/balotiyash/guide-mts-electron.git
- cd guide-mts-electron

Step 3: Navigate to Application Directory

- cd main-application

Step 4: Install Dependencies

- npm install

Below are the required dependencies for the project with versions. JSON data is on last page.

- **Production Dependencies:**
    - better-sqlite3 (v12.5.0): High-performance SQLite3 driver
    - electron-log (v5.4.3): Logging framework
    - electron-updater (v6.6.2): Auto-update functionality
    - electron-store (v10.1.0): Persistent data storage

  - express (v5.2.1): API server framework
  - cors (v2.8.5): Cross-origin resource sharing
  - bonjour (v3.5.0): Network service discovery
- **Development Dependencies:**
  - electron (v37.3.0): Desktop application framework
  - electron-builder (v26.0.12): Application packaging tool
  - @electron/rebuild (v4.0.1): Native module rebuilder
  - electron-rebuild (v3.2.9): Alternative rebuild tool

Step 5: Database Setup for Development

1. Copy the database schema from guide-db-migration/Schema Structure/
2. Place the database file in appropriate location or configure path in application settings
3. (Optional) Run Jupyter notebooks from guide-db-migration to populate sample data:
   - # Ensure Python and Jupyter are installed
   - jupyter notebook insert_student.ipynb
   - jupyter notebook insert_instructor.ipynb
   - jupyter notebook insert_car.ipynb

Step 6: Start Development Server

- npm start

This launches the application in development mode with:

- Hot reload capabilities
- Developer console enabled
- Debugging tools accessible (Press Ctrl+Shift+I or Cmd+Option+I)

Step 7: Build for Production (Optional)

To create distribution packages:

For current platform:

- npm run build

For Windows distribution with auto-publish:

- npm run dist

Output locations:

- **Windows**: dist/Guide MTS Setup 2.0.0.exe
- **macOS**: dist/Guide MTS-2.0.0.dmg
- **Linux**: dist/Guide MTS-2.0.0.AppImage

2.3 Post-Installation Configuration

2.3.1 Database Initialization

- The application automatically creates necessary database tables on first launch
- Database schema includes 12 core tables:
  - customers: Customer's Data
  - daily_fuel_entries: Daily Fuel Data
  - instructors: Instructor's Data
  - payments: Payment Data
  - reminders: Reminders Data, Note: This table is not in use
  - sqlite_sequence: Sequence Data for Auto Increment IDs
  - vehicle_kilometers: Kilometer Ran Data
  - vehicles: Vehicle's Data
  - work_description: Customer's Work Description Data

2.3.2 Data Migration (If Applicable)

- If migrating from legacy systems, use the provided Jupyter notebooks in guide-db-migration
- Import CSV data files from 29092025_Latest_DB/csv/ directory
- Verify data integrity after migration

2.4 Verification and Testing

After installation, verify the system is functioning correctly:

1. **Login Test**: Ensure you can successfully log in
2. **Dashboard Loading**: Verify dashboard displays without errors
3. **Database Connectivity**: Check that data loads properly
4. **Module Access**: Test access to all modules (Registration, Payment, Search, etc.)
5. **Report Generation**: Generate a sample report to verify PDF creation
6. **Backup Functionality**: Perform a test backup and verify file creation

2.5 Troubleshooting Common Installation Issues

Issue 1: Application Won't Launch on Windows

- **Solution**: Install Microsoft Visual C++ Redistributable packages
- Download from: https://support.microsoft.com/en-us/help/2977003/

Issue 2: Database Permission Errors

- **Solution**: Ensure the application has write permissions to the database directory
- Run application as administrator (Windows) or grant proper file permissions (Linux/macOS)

Issue 3: npm install Fails During Development Setup

- **Solution**: Clear npm cache and retry
    - npm cache clean --force
    - npm install

Issue 4: Native Module Build Errors

- **Solution**: Rebuild native modules
    - npm install @electron/rebuild
    - npx electron-rebuild

Issue 5: Application Crashes on Startup

- **Solution**: Check electron-log files for error details
- Location: %APPDATA%/guide-mts/logs/ (Windows) or ~/Library/Logs/guide-mts/ (macOS)

2.6 Update and Maintenance

The application includes automatic update functionality:

- **Automatic Updates**: Application checks for updates on startup
- **Update Notification**: Users are prompted when new versions are available
- **Update Process**: Downloads and installs updates in the background
- **Rollback**: Previous version is preserved in case rollback is needed

For manual updates, download the latest installer from the releases page and run it. The existing database and configuration files will be preserved.

2.7 Directory Structure and Organization

The Guide MTS application follows a modular architecture with clear separation of concerns, organized as follows:

2.7.1 Root Level Overview

```
guide-mts/
├── database/            # SQLite database files
├── guide-db-migration/  # Data migration tools (Jupyter notebooks & CSV files)
├── main-application/    # Core Electron application
├── output/              # Build artifacts
└── README.md            # Project documentation
```

2.7.2 Main Application Structure

```
main-application/
├── src/
│   ├── assets/          # Icons, images, GIFs, and SVGs
│   ├── scripts/         # All JavaScript code (detailed below)
│   ├── styles/          # CSS files (page-specific and shared)
│   ├── views/           # HTML templates (14 pages)
│   └── temp/            # Temporary runtime files
├── package.json         # Dependencies and build configuration
└── license.txt          # End-user license agreement
```

2.7.3 Scripts Organization (src/scripts/)

The application logic is divided into specialized directories:

Core Files:

- main.js - Electron main process (app lifecycle, window management)
- preload.js - Security layer (context bridge for IPC)
- menu.js - Application menu configuration
- server.js - Express API server (port 3000)

Subdirectories:

1. **main/ipc/** - 13 IPC handlers for communication between processes
   - Examples: dashboardHandler.js, dataEntryHandler.js, paymentHandler.js
2. **main/services/** - 15 service modules containing business logic
   - Examples: dbService.js, dashboardService.js, invoiceService.js
3. **renderer/** - 16 client-side scripts, one per HTML view
   - Examples: login.js, dashboard.js, payment_entry.js
4. **routes/** - 6 API route definition files
   - Examples: dashboard.routes.js, dataEntry.routes.js
5. **utilities/** - Helper functions organized by feature domain
   - 11 subdirectories: dashboard, dataEntry, form14, fuelEntry, masterEntry, paymentEntry, reminders, reports, searchPage, sms, vehicleEntry
6. **imports/** - Self-hosted libraries (Chart.js, Flatpickr, InputMask)

2.7.4 Architectural Benefits

- **Separation of Concerns**: IPC handlers manage communication, services handle business logic
- **Modularity**: Each feature has dedicated handler, service, renderer, and utility files
- **Security**: Context isolation between main and renderer processes
- **Scalability**: Easy addition of new modules without affecting existing code
- **Maintainability**: Clear naming conventions and organized structure

2.7.5 File Naming Conventions

- Handlers: *Handler.js (IPC communication)
- Services: *Service.js (business logic & database operations)
- Routes: *.routes.js (API endpoints)
- Views: snake_case.html
- Styles: snake_case.css (matching view names)
- Renderers: snake_case.js (matching view names)

package.json

```
{
  "name": "guide-mts",
  "version": "2.0.0",
  "description": "Guide Motor Training School
Management System",
  "author": "Yash Balotiya",
  "main": "src/scripts/main.js",
  "type": "module",
  "scripts": {
    "start": "electron .",
    "dist": "electron-builder --win --x64 --publish
always",
    "build": "electron-builder"
  },
  "devDependencies": {
    "@electron/rebuild": "^4.0.1",
    "electron": "^37.3.0",
    "electron-builder": "^26.0.12",
    "electron-rebuild": "^3.2.9"
  },
  "build": {
    "appId": "com.balotiyash.app",
    "productName": "Guide MTS",
        "copyright": "Copyright © 2025
AlgoDevOpss by Yash Balotiya",
    "icon": "src/assets/icons/icon",
    "files": [
      "**/*",
      "!node_modules/fsevents/**",
      "!guide-db-migration/**",
      "!out/**",
      "!dist/**",
      "!src/database/**"
    ],
    "asarUnpack": [
      "src/views/**",
      "src/scripts/preload.js",
      "src/scripts/renderer/**",
      "src/styles/**",
      "src/assets/**"
    ],
    "mac": {
      "target": "dmg"
    },
    "win": {
      "target": "nsis"
    },
    "linux": {
      "target": "AppImage"
    },
    "nsis": {
      "oneClick": false,
       "allowToChangeInstallationDirectory":
true,
      "license": "license.txt",
      "perMachine": false,
      "createDesktopShortcut": true,
      "createStartMenuShortcut": true,
      "shortcutName": "Guide MTS"
    },
    "publish": [
      {
        "provider": "github",
        "owner": "balotiyash",
        "repo": "guide-mts-electron"
      }
    ]
  },
  "dependencies": {
    "better-sqlite3": "^12.5.0",
    "bonjour": "^3.5.0",
    "cors": "^2.8.5",
    "electron-log": "^5.4.3",
    "electron-store": "^10.1.0",
    "electron-updater": "^6.6.2",
    "express": "^5.2.1"
  },
  "nodeVersion": "24.6.0"
}
```