**QR-Gen Project Documentation**

**Overview**

QR-Gen is a fully offline web application built with Node.js, EJS, and Bootstrap for inventory management with QR code generation. The application allows users to add inventory items and generate QR codes that contain JSON data, which can be scanned to retrieve item information.

**Features**

- **\*\*Multi-Table Support\*\***: Manage inventory across 3 tables (IT, ATG, LPSS)

- **\*\*QR Code Generation\*\***: Generate QR codes that encode JSON data directly

- **\*\*Admin Panel\*\***: Password-protected admin interface to view all items

- **\*\*Offline Operation\*\***: Works completely offline without internet access

- **\*\*Data Export\*\***: Download QR codes as high-quality JPG files

- **\*\*Responsive Design\*\***: Modern UI built with Bootstrap

**Technology Stack**

- **\*\*Backend\*\***: Node.js with Express.js

- **\*\*Template Engine\*\***: EJS

- **\*\*Database\*\***: MySQL (via XAMPP)

- **\*\*Frontend\*\***: Bootstrap 5.3.8

- **\*\*QR Generation\*\***: qrcode library

- **\*\*Authentication\*\***: bcryptjs with express-session

**Prerequisites**

- XAMPP (Apache + MySQL)

- Node.js 18+ (currently using v22.17.1)

- npm (currently using v9.6.6)

**Installation & Setup**

1. Database Setup

1. Start XAMPP and ensure MySQL is running

2. Create database named `qr`

**3. Import the provided SQL dump to create tables:**

   - `it` - IT inventory items

   - `atg` - ATG inventory items

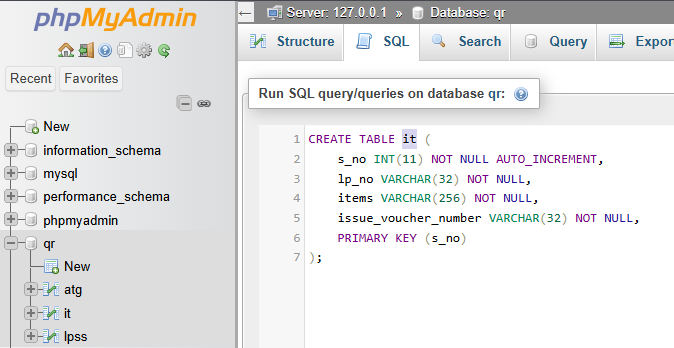
   - `lpss` - LPSS inventory items

**To add new tables-**

**Go to my phpMyAdmin :** <http://localhost/phpmyadmin>

After that we will select the database we have created which is ‘qr’ and then go the SQL section.

**Write this sql query in the SQL section**



CREATE TABLE table\_name (

s\_no INT(11) NOT NULL AUTO\_INCREMENT,

lp\_no VARCHAR(32) NOT NULL,

items VARCHAR(256) NOT NULL,

issue\_voucher\_number VARCHAR(32) NOT NULL,

PRIMARY KEY (s\_no)

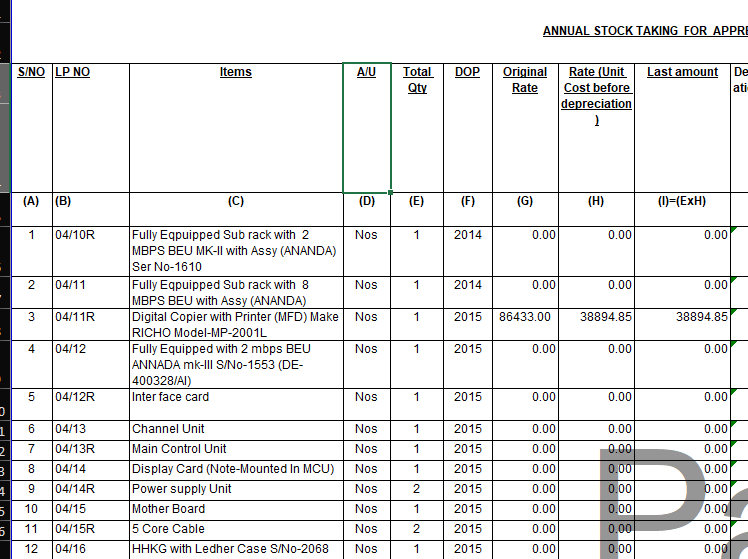
);

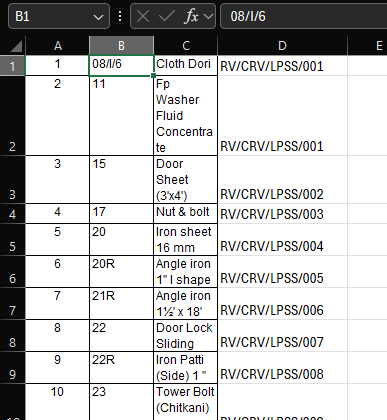
Change name with the new name of table you want to make.

**This will create a basic structure of table**

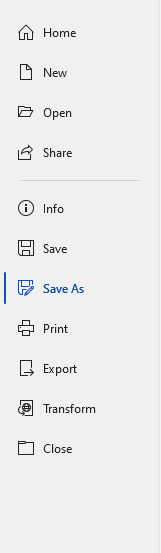
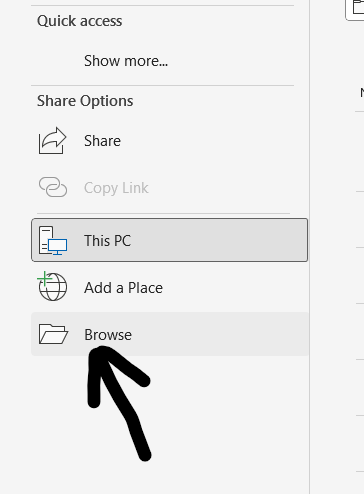
After that we will need to import data in this table –

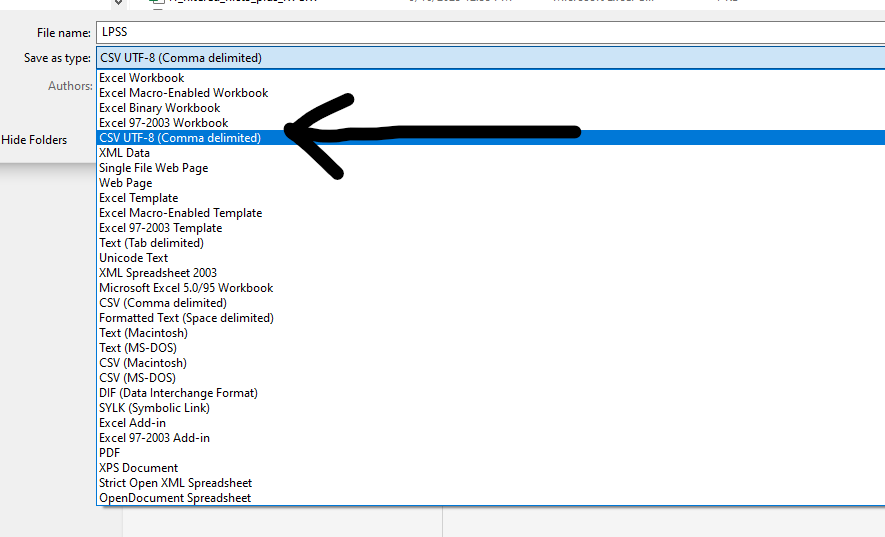
**Steps to import data in the table –**

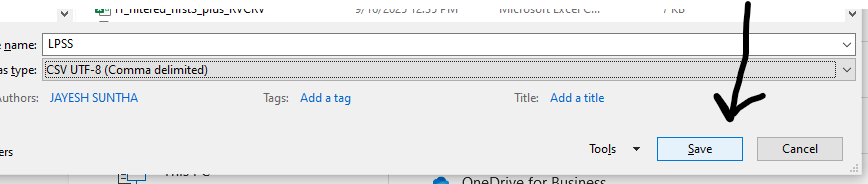
# From this excel sheet we will need to filter out the 3 first columns and make a new excel sheet.



# In this format as shown in the picture. For the Issue Voucher number we will need to find it manually from the ledger file and put it here in this format.

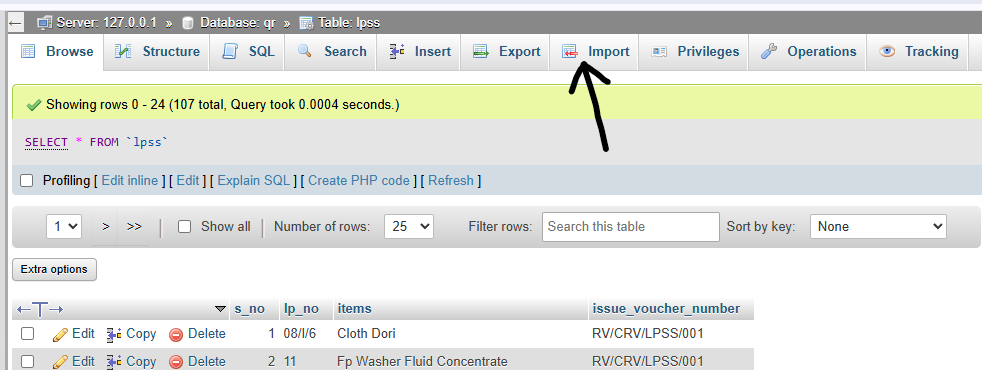
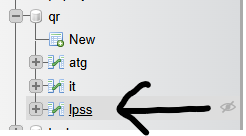
 



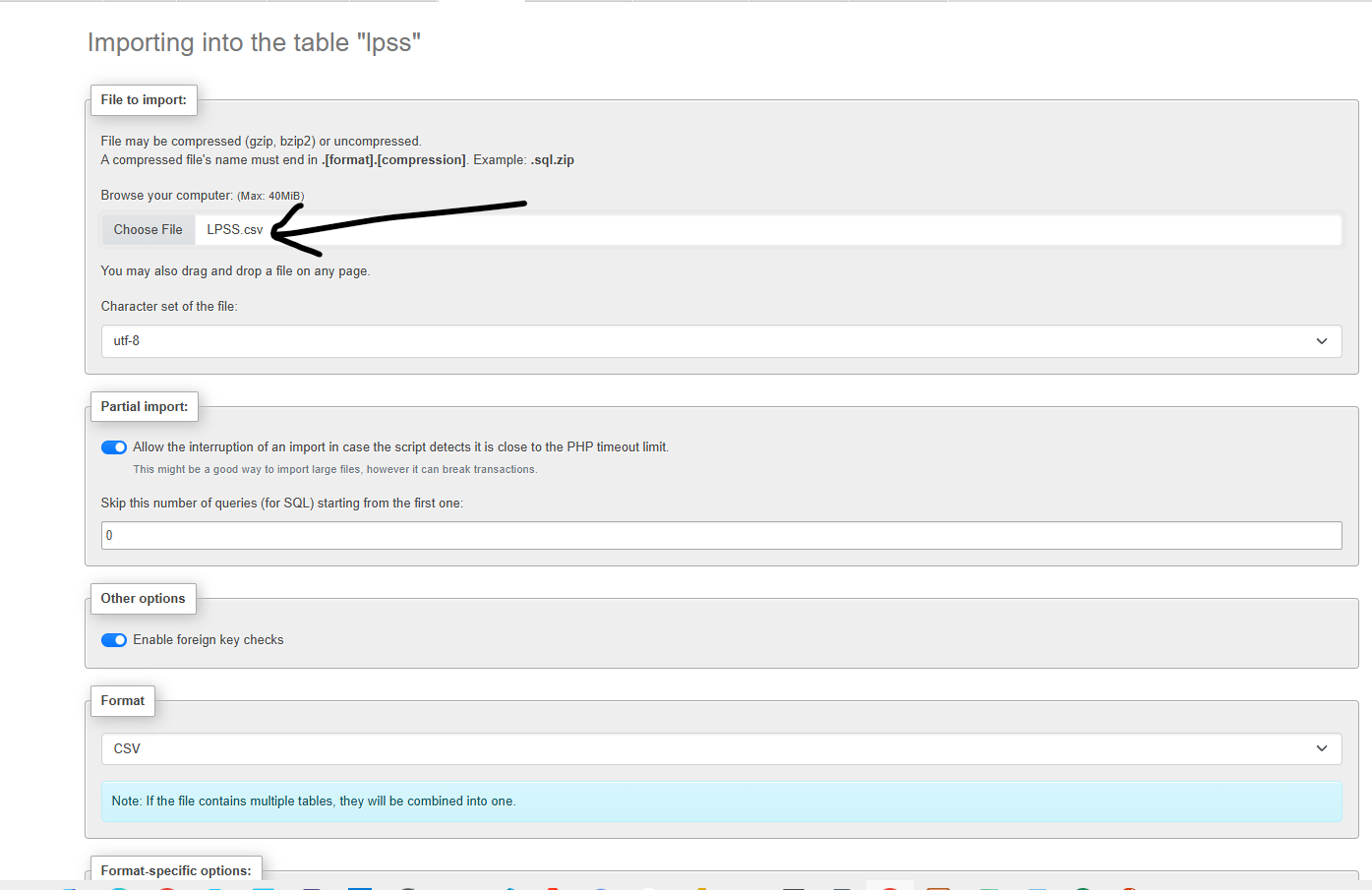


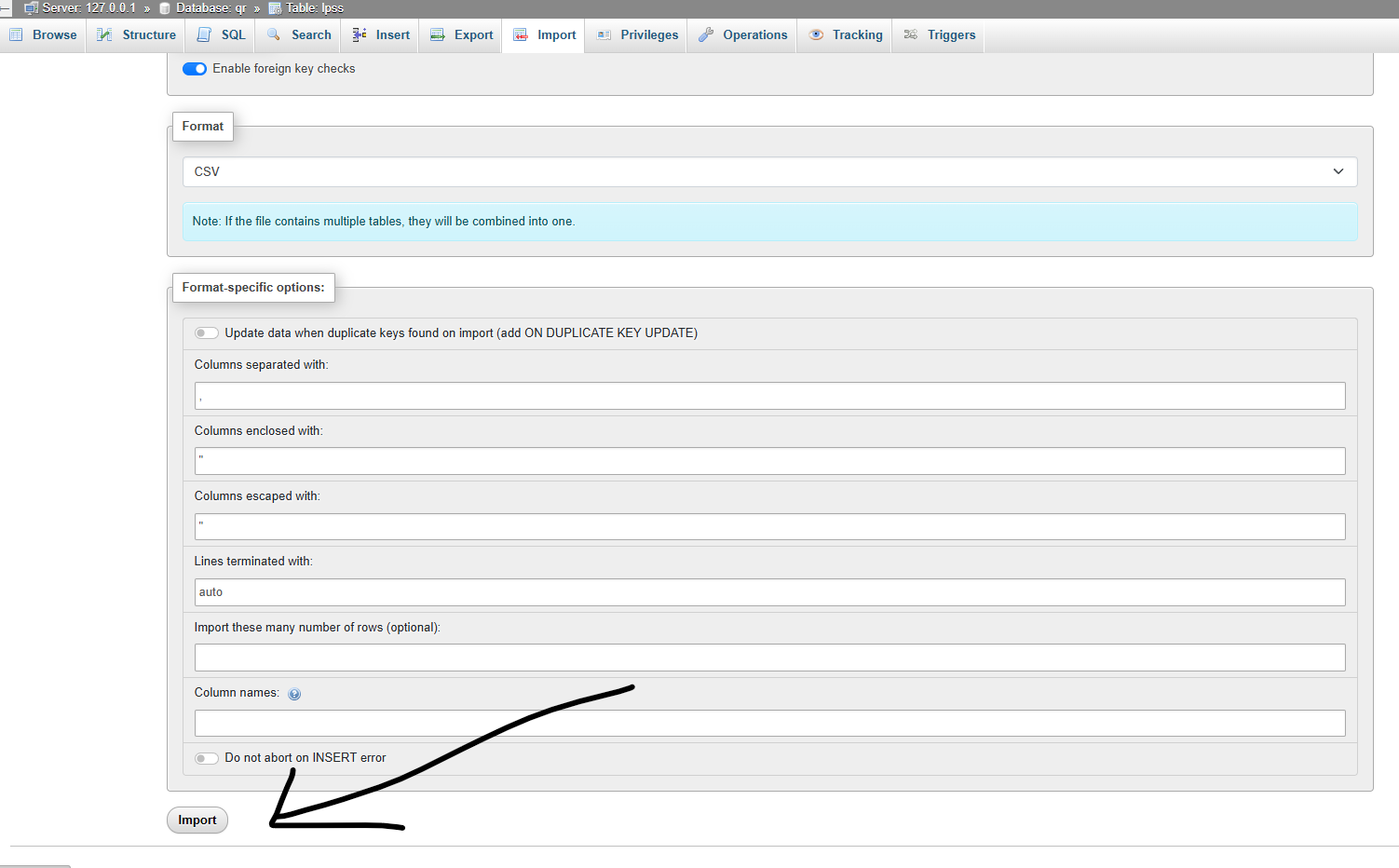
Afte the excel file is made we will need to change this file in the .csv (comma separated values) format. We will go to save changes and then click on browse after that we will find and click on CSV UTF-8 format and save the file.

After saving the file we will go to myPhpAdmin and import the this file.



We will click on the table on which we want to import data on. After that we will go to the import. And on the import page we will choose file we just saved in the .csv format. Scroll down and click on the import button.





2. Application Setup

```bash

*# Navigate to project directory*

cd C:\xampp\htdocs\QR-Gen

*# Install dependencies*

npm install

*# Create environment file (optional)*

*# Copy .env.example to .env and modify as needed*

```

### 3. Configuration

**Create a `.env` file with the following variables:**

```env

**APP\_PORT=3000**

**DB\_HOST=127.0.0.1**

**DB\_USER=root**

**DB\_PASSWORD=**

**DB\_NAME=qr**

**SESSION\_SECRET=please\_change\_me\_offline**

**ADMIN\_USERNAME=itadmin**

**ADMIN\_PASSWORD=it@12345**

```

### 4. Start Application

```bash

*# Development mode with auto-restart*

npm run dev

*# Production mode*

npm start

```

## Application Structure

```

**QR-Gen/**

├── server.js              # Main application file

├── package.json           # Dependencies and scripts

├── .env                   # Environment configuration

├── views/                 # EJS templates

│   ├── form.ejs          # Add item form

│   ├── success.ejs       # Success page with QR

│   ├── admin\_login.ejs   # Admin login page

│   └── admin\_panel.ejs   # Admin dashboard

├── public/               # Static assets

│   ├── css/

│   │   ├── bootstrap.min.css

│   │   └── app.css

│   └── js/

│       └── bootstrap.bundle.min.js

└── node\_modules/         # Dependencies

```

## Usage Guide

### Adding Items

1. Navigate to `http://localhost:3000/form`

2. Select the target table (IT, ATG, or LPSS)

**3. Fill in the required fields:**

   - LP No: Item identifier

   - Items: Item description

   - Issue Voucher Number: Voucher reference

4. Click "Generate QR"

5. View the generated QR code and JSON data

### Admin Panel

1. Navigate to `http://localhost:3000/admin`

**2. Login with credentials:**

   - Username: `itadmin`

   - Password: `it@12345`

3. Use the table navigation to switch between IT, ATG, and LPSS

4. View all items with QR codes

5. Download QR codes as JPG files

6. Copy JSON data to clipboard

### QR Code Features

**- \*\*Data Format\*\*: QR codes contain JSON data in the format:**

  ```json

  {

    "s\_no": 1,

    "lp\_no": "03/08",

    "items": "Key board and Mouse",

    "issue\_voucher\_number": "RV/CRV/IT/001"

  }

  ```

- **\*\*Quality\*\***: High-resolution QR codes (1024px) for download

- **\*\*Compatibility\*\***: Works with any QR scanner app

## API Endpoints

### Public Routes

- `GET /` - Redirects to form

- `GET /form` - Display add item form

- `POST /form` - Process form submission

- `GET /success/:id` - Display success page with QR

- `GET /api/item/:id` - JSON API for item data

### Admin Routes

- `GET /admin/login` - Admin login page

- `POST /admin/login` - Process admin login

- `GET /admin` - Admin dashboard (requires auth)

- `POST /admin/logout` - Admin logout

## Database Schema

### Tables Structure

**All tables follow the same schema:**

```sql

**CREATE TABLE table\_name (**

  s\_no INT(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  lp\_no VARCHAR(32) NOT NULL,

  items VARCHAR(256) NOT NULL,

  issue\_voucher\_number VARCHAR(32) NOT NULL

);

```

### Auto-increment Management

- The application automatically manages `s\_no` auto-increment

- New items continue from the last `s\_no` in each table

- Removes any invalid `s\_no = 0` entries on startup

## Security Features

- **\*\*Session-based Authentication\*\***: Secure admin login

- **\*\*Password Hashing\*\***: bcryptjs for password security

- **\*\*Input Validation\*\***: Server-side validation for all inputs

- **\*\*SQL Injection Protection\*\***: Parameterized queries

## Customization

### Adding New Tables

1. Create table in MySQL with the standard schema

2. Add table option to form dropdown in `views/form.ejs`

3. Add table link to admin navbar in `views/admin\_panel.ejs`

### Changing Admin Credentials

**1. Update `.env` file:**

   ```env

**ADMIN\_USERNAME=your\_username**

**ADMIN\_PASSWORD=your\_password**

   ```

2. Restart the application

### Styling Modifications

- Edit `public/css/app.css` for custom styles

- Modify Bootstrap classes in EJS templates

- Update color scheme in CSS variables

## Troubleshooting

### Common Issues

1. **\*\*"Table doesn't exist" Error\*\***

   - Ensure MySQL is running in XAMPP

   - Check database name in `.env` file

   - Verify table names match exactly

2. **\*\*Login Issues\*\***

   - Check admin credentials in `.env`

   - Clear browser cache/cookies

   - Try incognito/private window

3. **\*\*QR Code Not Scanning\*\***

   - Ensure QR contains JSON data (not URL)

   - Check QR code quality/contrast

   - Try different QR scanner apps

4. **\*\*Database Connection Issues\*\***

   - Verify XAMPP MySQL is running

   - Check database credentials

   - Ensure database `qr` exists

### Debug Mode

- Check console output for available tables

- Look for error messages in terminal

- Verify `.env` file configuration

## Performance Considerations

- **\*\*Database Pooling\*\***: Uses connection pooling for efficiency

- **\*\*QR Generation\*\***: Cached QR codes for admin panel

- **\*\*Static Assets\*\***: Local Bootstrap files for offline use

- **\*\*Session Management\*\***: Efficient session handling

## Future Enhancements

- Bulk QR code generation

- CSV export functionality

- Search and filter capabilities

- Item editing/deletion features

- Print-friendly QR layouts

- Multi-language support

## Support

**For issues or questions:**

1. Check this documentation

2. Review console error messages

3. Verify database connectivity

4. Check file permissions

## License

This project is for internal use only. All rights reserved.

**Goal**

**Add an already-created table (example: newtable) to the app so you can:**

* Insert records into it from the form.
* View its records in the admin panel.
* No backend code changes are needed if the schema matches (s\_no, lp\_no, items, issue\_voucher\_number).

**1) Add the table to the Form dropdown**

File: views/form.ejs

* **Find the table select in the form (near these lines):**

form.ejs

<select class="form-select" name="table" required>

<option value="">Choose a table...</option>

<option value="it">IT</option>

<option value="atg">ATG</option>

<option value="lpss">LPSS</option>

</select>

**Insert this new option inside the same <select> (order doesn’t matter):**

<option value="newtable">NEWTABLE</option>

**Result sample:**

<select class="form-select" name="table" required>

<option value="">Choose a table...</option>

<option value="it">IT</option>

<option value="atg">ATG</option>

<option value="lpss">LPSS</option>

<option value="newtable">NEWTABLE</option>

</select>

**What this does:**

* Lets users choose NEWTABLE so the form posts to that table.
* The server already reads table from the form and inserts into it.

**2) Add the table to the Admin navbar**

File: views/admin\_panel.ejs

* **Find the navbar with table links after line 30 (near these lines):**

Admin\_panel.ejs

<div class="navbar-nav">

<a class="nav-link <%= currentTable === 'it' ? 'active fw-bold' : '' %>" href="/admin?table=it">IT</a>

<a class="nav-link <%= currentTable === 'atg' ? 'active fw-bold' : '' %>" href="/admin?table=atg">ATG</a>

<a class="nav-link <%= currentTable === 'lpss' ? 'active fw-bold' : '' %>" href="/admin?table=lpss">LPSS</a>

</div>

**Add this link inside the same <div class="navbar-nav">:**

<a class="nav-link <%= currentTable === 'newtable' ? 'active fw-bold' : '' %>" href="/admin?table=newtable">NEWTABLE</a>

**Result sample:**

<div class="navbar-nav">

<a class="nav-link <%= currentTable === 'it' ? 'active fw-bold' : '' %>" href="/admin?table=it">IT</a>

<a class="nav-link <%= currentTable === 'atg' ? 'active fw-bold' : '' %>" href="/admin?table=atg">ATG</a>

<a class="nav-link <%= currentTable === 'lpss' ? 'active fw-bold' : '' %>" href="/admin?table=lpss">LPSS</a>

<a class="nav-link <%= currentTable === 'newtable' ? 'active fw-bold' : '' %>" href="/admin?table=newtable">NEWTABLE</a>

</div>

**What this does:**

* Adds a tab in the admin to view newtable.
* Highlights the tab when you’re on /admin?table=newtable.

**3) Restart the server**

* **In the nodemon terminal, type: rs**
* **Watch the console. You should see your new table listed:**
* **Available tables: …, 'newtable'**

**4) Verify**

* **Insert:**
* **Open http://localhost:3000/form**
* **Choose NEWTABLE, fill fields, click Generate QR**
* You’ll land on /success/:id?table=newtable with the QR and JSON
* **View:**
* Open http://localhost:3000/admin?table=newtable
* You’ll see the list with QR and Download JPG buttons

**Notes (must match)**

* **Your new table in MySQL must have exactly these columns:**
* s\_no (INT, AUTO\_INCREMENT, PRIMARY KEY)
* lp\_no (VARCHAR(32))
* items (VARCHAR(256))
* issue\_voucher\_number (VARCHAR(32))
* No backend changes are required if the schema matches. The app already inserts into and reads from whatever table you select.