

# **NIPH Data Sharing Platform System**

## **(NIPH DSP, Part III)**

### **IV. System Implementation**

- 1. Technologies Using**
- 2. Core Features & Module Breakdown**
- 3. Key Considerations & Implementation Details**
- 4. Project Structure**
- 5. MySQL Database Schema (database/schema.sql)**
- 6. Project Setup (Database, Backend, Frontend)**
- 7. Backend (Node.js with Express)**
- 8. Frontend (Vue 3, Vite, with Tailwind CSS)**
- 9. Testing and Review**

## 1. Technologies

- **Frontend:** VueJS 3, Vite, Tailwind CSS 4
- **Backend:** Node.js for API
- **Database:** MySQL
- **R Integration:** R-serve for executing R scripts

## 2. Core Features & Module Breakdown

### A. Public Facing Website (Frontend)

#### 1. Header (Sticky technical)

- \* **Logo:** Displays the NIPH logo.
- \* **Search Bar:** Allows users to search for data source.
- \* **Main Navigation:**
  - **Home:** Links to the main landing page.
  - **Classifications:** Displays data source categories (dsps\_tbl\_dspscategory).
  - **Announcements:** Lists announcements (dsps\_tbl\_announcement).
  - **About Us:** Displays information from dsps\_tbl\_dspsabout (Goal, Vision, Mission).
  - **Contact Us:** Provides contact information, google map and feedback form.
- \* **Utility Navigation:**
  - **FAQ:** Links to the FAQ page (dsps\_tbl\_dspsfaq).
  - **Register:** Navigates to the user registration form.
  - **Login:** Navigates to the user login form.
  - **Social Media Links:** Icons linking to FB, YouTube, Telegram (dsps\_tbl\_social).

#### 2. Footer

- \* **Introduction to NIPH:** Short description.
- \* **Contact Us:** Key contact details.
- \* **Copyright:** Copyright information.

#### 3. Home Page

- \* **Slideshow:** Displays slides from dsps\_tbl\_dspslide.
- \* **Featured Data Sources:** Showcase popular/ newly added data sources.
- \* **Announcements:** Latest announcements.
- \* **Quick Links:** To classifications, FAQs, etc.

#### 4. Classifications Page

- \* Lists and filter feature all data source categories from dsps\_tbl\_dspscategory.
- \* Clicking on a category displays data sources belonging to that category.

#### 5. Announcements Page

- \* Displays a list of all announcements from dsps\_tbl\_announcement.
- \* Each announcement shows title, description, and possibly an image.

#### 6. About Us Page

- \* Displays content from dsps\_tbl\_dspsabout (Goal, Vision, Mission, etc.).

#### 7. Contact Us & Feedback Page

- \* Contact information.
  - \* Google map
  - \* Feedback form for users to submit inquiries (dsps\_tbl\_feedback).
- Stores: dspsfb\_client\_ip, dspsfb\_name, dspsfb\_email, dspsfb\_body\_text.

## 8. FAQ Page

- \* Displays questions and answers from dsps\_tbl\_dspsfaq.

## 9. Data Source Details Page

- \* Login and Accessed when a user clicking on a data source.
- \* Displays dspsds\_title\_en, dspsds\_description, dspsds\_public\_date.
- \* **Anonymous Tracking:** On viewing, log dsps\_tbl\_anonymous (fkdspsds\_id, dspsano\_client\_ip, dspsano\_datetime, dspsano\_action='view\_intro').
- \* **Download/Access Button:**
  - If public: allows direct download/access. \* If restricted: prompts login/registration or request access.

## 10. User Authentication (Registration & Login)

- \* **Registration:**
  - Captures ist\_tbl\_people information (id card, name, dob, etc.).
  - Creates an entry in ist\_tbl\_users with isu\_status (e.g., 'pending' for admin approval).
  - Password hashing.
- \* **Login:**
  - Authenticates users against ist\_tbl\_users.
  - Uses JWT or session-based authentication for secure access.

## B. Backend (API & Database Interaction)

### 1. API Endpoints:

#### \* Public Endpoints:

- /api/slides: Get all slides.
- /api/announcements: Get all announcements.
- /api/about: Get about us content.
- /api/faq: Get all FAQs.
- /api/social-links: Get social media links.
- /api/categories: Get all data source categories.
- /api/data-sources: Get public data sources (with optional search/filter).
- /api/data-sources/:id: Get details of a specific data source.
- /api/feedback: Submit feedback.
- /api/register: Register new user.
- /api/login: User login.
- /api/anonymous-view: Log anonymous data source views.

#### \* Authenticated Endpoints (for CPanel):

- User management (CRUD for ist\_tbl\_people, ist\_tbl\_users).
- Content management (CRUD for dsps\_tbl\_dspsabout, dsps\_tbl\_dspsslide, dsps\_tbl\_dspsfaq, dsps\_tbl\_announcement, dsps\_tbl\_social, dsps\_tbl\_dspscategory).
- Data source management (CRUD for dsps\_tbl\_datasource, dsps\_tbl\_typedatasource).
- Data source permission management (dsps\_tbl\_datasource\_permission).
- Data source usage tracking (dsps\_tbl\_datasource\_used).
- Feedback management (dsps\_tbl\_feedback).

#### \* R Script Execution:

- /api/run-r-script: Endpoint for running R scripts (requires proper authentication and authorization).

## 2. Database Interaction:

- \* ORM (Sequelize for Node.js) with MySQL.
- \* Secure password hashing (e.g., bcrypt).
- \* Input validation.

## C. Control Panels (CPanels - Authenticated Access)

Each CPanel will have a distinct set of permissions based on the user's role(isu\_status).

### 1. Content Management CPanel (Admin)

- \* **Manage About Us:** CRUD for dsps\_tbl\_dspsabout.
- \* **Manage Slides:** CRUD for dsps\_tbl\_dspsslide (upload dspsslide\_photoname).
- \* **Manage FAQ:** CRUD for dsps\_tbl\_dspsfaq.
- \* **Manage Announcements:** CRUD for dsps\_tbl\_announcement (upload dspsann\_photopath).
- \* **Manage Social Links:** CRUD for dsps\_tbl\_social.
- \* **Manage Data Categories:** CRUD for dsps\_tbl\_dspscategory.
- \* **Manage Feedback:** View, respond to (dspsfb\_respond\_text), and change status of feedback entries from dsps\_tbl\_feedback.

### 2. DAC Staff CPanel

- \* **User Management:**
  - \* View and approve/reject new user registrations (ist\_tbl\_users.isu\_status).
  - \* Manage existing users (ist\_tbl\_people, ist\_tbl\_users - update roles, status).
- \* **Data Source Oversight:**
  - \* View all data sources, including pending ones.
  - \* Monitor data source permissions and requests.
- \* **Reports/Analytics:** \* View statistics on data source views (dsps\_tbl\_anonymous).
- \* View statistics on data source usage (dsps\_tbl\_datasource\_used).
- \* Generate reports on user activity.
- \* **System Configuration:** (Potentially, if needed for overall system settings)
- \* **R Script Execution:** Ability to run pre-defined or custom R scripts for data analysis/reporting.

### 3. Data Owner CPanel

- \* **My Data Sources:**
  - \* CRUD operations for dsps\_tbl\_datasource (upload dspsds\_filename, link to fkdspsstds\_id, fkdspscate\_id).
  - \* Set dspsds\_status (public/private/pending).
- \* **Manage Data Source Permissions:**
  - \* View and manage access requests for their data sources from dsps\_tbl\_datasource\_permission.
  - \* Grant/revoke permissions (dspsdsp\_permission, dspsdsp\_startus).
- \* **View Data Source Usage:**
  - \* See who used their data sources (dsps\_tbl\_datasource\_used).
- \* **Run R:** Ability to upload and run R scripts on their data sources (if applicable and authorized).

#### 4. Data User CPanel

- \* **My Profile:** View and edit personal information (ist\_tbl\_people).
- \* **My Data Access Requests:** View status of their requests for data source access.
- \* **Download/Access Data Sources:** Access data sources they have permission for.
- \* **View My Usage History:** See which data sources they have used (dsps\_tbl\_datasource\_used).
- \* **Run R:** Ability to run R scripts on data they have access to.

### 3. Key Considerations & Implementation Details

#### 1. User Roles & Permissions:

- \* Implement robust role-based access control (RBAC) on the backend to ensure users only access authorized resources and CPanels.
- \* ist\_tbl\_users.isu\_status will be crucial for determining access levels.

#### 2. Data Security:

- \* **Authentication:** JWT or session-based authentication.
- \* **Authorization:** Implement middleware on the backend to check user roles and permissions for every API request.
- \* **Password Hashing:** Use strong hashing algorithms (e.g., bcrypt) for isu\_password.
- \* **Data Encryption:** Consider HTTPS for all communication.
- \* **Input Validation:** Sanitize and validate all user inputs to prevent SQL injection and XSS attacks.

#### 3. File Uploads:

- \* For dspsslide\_photoname, dspsann\_photopath, dspsds\_filename, implement secure file upload mechanisms. Store files in a designated directory and save only the path/name in the database.
- \* Consider cloud storage (e.g., S3) for scalability if needed.

#### 4. R Integration:

- \* **Backend Component:** A dedicated service or module on the backend to handle R script execution.
- \* **Rserve:** A common choice for integrating R with other applications. The backend would send R code to Rserve, and Rserve would return the results.
- \* **Security:** Running arbitrary R scripts can be a security risk. Implement strict sandboxing and validation of R scripts, especially for Data Users. DAC staff might have more freedom.
- \* **Output:** Determine how R script outputs (e.g., data frames, plots) will be returned to the frontend.

#### 5. Frontend State Management (Vuex or Pinia):

- \* For larger applications, a state management library (Vuex or Pinia) will be beneficial for managing user authentication state, global data, and CPanel specific data.

#### 6. Responsive Design:

- \* Tailwind CSS will greatly assist in building a responsive UI that works well on various devices.

## 7. Error Handling:

\* Implement comprehensive error handling on both frontend (user-friendly messages) and backend (logging, proper HTTP status codes).

## 8. Logging:

\* Log significant events, errors, and user actions (e.g., data source views and usage) for auditing and debugging.

## 4. Project Structure

```
dsp/
├── frontend/
│   ├── public/
│   │   └── favicon.ico
│   ├── src/
│   │   ├── assets/
│   │   │   └── logo.png
│   │   ├── components/
│   │   │   ├── public/
│   │   │   │   ├── Header.vue
│   │   │   │   ├── Footer.vue
│   │   │   │   ├── SlideShow.vue
│   │   │   │   ├── DataSourceCard.vue
│   │   │   │   └── FeedbackForm.vue
│   │   │   ├── auth/
│   │   │   │   ├── LoginForm.vue
│   │   │   │   └── RegisterForm.vue
│   │   │   └── cpanels/
│   │   │       ├── DACStaffDashboard.vue
│   │   │       ├── DataOwnerDashboard.vue
│   │   │       ├── DataUserDashboard.vue
│   │   │       └── ContentManagement.vue
│   │   ├── views/
│   │   │   ├── HomeView.vue
│   │   │   ├── ClassificationsView.vue
│   │   │   ├── AnnouncementsView.vue
│   │   │   ├── AboutUsView.vue
│   │   │   ├── ContactUsView.vue
│   │   │   ├── FAQView.vue
│   │   │   └── DataSourceDetailView.vue
│   │   ├── router/
│   │   │   └── index.js
│   │   ├── stores/
│   │   │   └── auth.js
│   │   ├── services/
│   │   │   └── api.js
│   │   ├── main.js
│   │   ├── App.vue
│   │   └── index.css (Tailwind CSS)
│   ├── tailwind.config.js
│   └── vite.config.js
```

```

| — package.json
— backend/
  | — config/
  |   | — db.js (Database connection)
  | — controllers/
  |   | — authController.js
  |   | — publicController.js
  |   | — userController.js
  |   | — contentController.js
  |   | — dataSourceController.js
  |   | — feedbackController.js
  | — middleware/
  |   | — authMiddleware.js (JWT verification, RBAC)
  | — models/
  |   | — userModel.js
  | — routes/
  |   | — authRoutes.js
  |   | — publicRoutes.js
  |   | — apiRoutes.js
  |   | — adminRoutes.js
  | — r_scripts/
  |   | — r_script.R
  | — services/
  |   | — rService.js (Rserve integration)
  | — app.js
  | — package.json
— database/
  | — schema.sql (MySQL Schema)
— README.md

```

## 5. MySQL Database Schema (database/schema.sql)

This script will create your tables.  
SQL

```

-- Disable foreign key checks for easier setup (enable after creating tables)
SET FOREIGN_KEY_CHECKS = 0;

-- Table 1: ist_tbl_people
CREATE TABLE IF NOT EXISTS ist_tbl_people (
  pkisp_id INT AUTO_INCREMENT PRIMARY KEY,
  isp_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
  isp_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
  isp_regby_id INT, -- Self-referencing or FK to users if an admin registered them
  isp_idcard VARCHAR(50) UNIQUE NOT NULL,
  isp_firstname_en VARCHAR(100) NOT NULL,
  isp_lastname_en VARCHAR(100) NOT NULL,
  isp_sex ENUM('Male', 'Female', 'Other') NOT NULL,
  isp_dob DATE,
  isp_pob VARCHAR(255), -- Place of birth
  isp_nationality VARCHAR(100),

```

```

    isp_marital_status ENUM('Single', 'Married', 'Divorced', 'Widowed'),
    isp_phone_number VARCHAR(20),
    isp_email VARCHAR(255) UNIQUE NOT NULL,
    isp_telegram VARCHAR(100),
    isp_note TEXT
);

-- Table 2: ist_tbl_users
CREATE TABLE IF NOT EXISTS ist_tbl_users (
    pkisu_id INT AUTO_INCREMENT PRIMARY KEY,
    isu_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    isu_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    isu_regby_id INT, -- ID of the user who registered this user (e.g., an admin)
    fkisp_id_of INT UNIQUE NOT NULL, -- Foreign key to ist_tbl_people
    isu_name VARCHAR(100) UNIQUE NOT NULL, -- Username
    isu_password VARCHAR(255) NOT NULL, -- Hashed password
    isu_status ENUM('Pending', 'Active', 'Inactive', 'DAC Staff', 'Data Owner', 'Data User')
    DEFAULT 'Pending',
    FOREIGN KEY (fkisp_id_of) REFERENCES ist_tbl_people(pkisp_id)
);

-- Table 3: dsps_tbl_dspsabout
CREATE TABLE IF NOT EXISTS dsps_tbl_dspsabout (
    pkdspsabout_id INT AUTO_INCREMENT PRIMARY KEY,
    dspsabout_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspsabout_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspsabout_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspsabout_title_en VARCHAR(255) NOT NULL,
    dspsabout_description TEXT,
    FOREIGN KEY (dspsabout_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

-- Table 4: dsps_tbl_dspsslide
CREATE TABLE IF NOT EXISTS dsps_tbl_dspsslide (
    pkdspsslide_id INT AUTO_INCREMENT PRIMARY KEY,
    dspsslide_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspsslide_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspsslide_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspsslide_title_en VARCHAR(255) NOT NULL,
    dspsslide_description TEXT,
    dspsslide_photoname VARCHAR(255), -- Path or filename of the photo
    FOREIGN KEY (dspsslide_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

-- Table 5: dsps_tbl_dspsfaq
CREATE TABLE IF NOT EXISTS dsps_tbl_dspsfaq (
    pkdspsfaq_id INT AUTO_INCREMENT PRIMARY KEY,
    dspsfaq_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspsfaq_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspsfaq_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspsfaq_title_en VARCHAR(255) NOT NULL, -- Question
    dspsfaq_description TEXT, -- Answer

```



```

    FOREIGN KEY (dspsfaq_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

```

-- Table 6: dsps\_tbl\_announcement

```

CREATE TABLE IF NOT EXISTS dsps_tbl_announcement (
    pkdspsann_id INT AUTO_INCREMENT PRIMARY KEY,
    dspsann_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspsann_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspsann_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspsann_title VARCHAR(255) NOT NULL,
    dspsann_description TEXT,
    dspsann_photopath VARCHAR(255),
    dspsann_status ENUM('Active', 'Inactive') DEFAULT 'Active',
    FOREIGN KEY (dspsann_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

```

-- Table 7: dsps\_tbl\_social

```

CREATE TABLE IF NOT EXISTS dsps_tbl_social (
    pkdspssocial_id INT AUTO_INCREMENT PRIMARY KEY,
    dspssocial_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspssocial_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspssocial_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspssocial_name VARCHAR(100) NOT NULL, -- e.g., 'Facebook', 'YouTube',
    'Telegram'
    dspssocial_link VARCHAR(255) NOT NULL,
    FOREIGN KEY (dspssocial_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

```

-- Table 13: dsps\_tbl\_dspscategory

```

CREATE TABLE IF NOT EXISTS dsps_tbl_dspscategory (
    pkdspscate_id INT AUTO_INCREMENT PRIMARY KEY,
    dspscate_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspscate_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspscate_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspscate_title_en VARCHAR(255) NOT NULL,
    dspscate_details TEXT,
    FOREIGN KEY (dspscate_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

```

-- Table 8: dsps\_tbl\_typedatasource

```

CREATE TABLE IF NOT EXISTS dsps_tbl_typedatasource (
    pkdspstds_id INT AUTO_INCREMENT PRIMARY KEY,
    dspstds_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
    dspstds_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
    dspstds_reg_by INT, -- FK to ist_tbl_users (DAC Staff)
    dspstds_name_en VARCHAR(100) NOT NULL, -- e.g., 'CSV', 'JSON', 'API', 'PDF'
    dspstds_name_kh VARCHAR(100),
    FOREIGN KEY (dspstds_reg_by) REFERENCES ist_tbl_users(pkisu_id)
);

```

-- Table 9: dsps\_tbl\_datasource

```
CREATE TABLE IF NOT EXISTS dsps_tbl_datasource (  
    pkdspds_id INT AUTO_INCREMENT PRIMARY KEY,  
    dspds_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,  
    dspds_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,  
    dspds_reg_by INT, -- FK to ist_tbl_users (Data Owner)  
    fkdspstds_id INT NOT NULL, -- FK to dsps_tbl_typedatasource  
    fkdspscate_id INT NOT NULL, -- FK to dsps_tbl_dspscategory  
    fkisp_id_of INT NOT NULL, -- The person (Data Owner) who owns this data source  
    dspds_filename VARCHAR(255), -- Path to the actual data file or API endpoint  
    dspds_title_en VARCHAR(255) NOT NULL,  
    dspds_title_kh VARCHAR(255),  
    dspds_description TEXT,  
    dspds_public_date DATE,  
    dspds_status ENUM('Public', 'Private', 'Pending Approval') DEFAULT 'Pending  
Approval',  
    FOREIGN KEY (fkdspstds_id) REFERENCES  
    dsps_tbl_typedatasource(pkdspstds_id),  
    FOREIGN KEY (fkdspscate_id) REFERENCES  
    dsps_tbl_dspscategory(pkdspscate_id),  
    FOREIGN KEY (fkisp_id_of) REFERENCES ist_tbl_people(pkisp_id)  
);
```

-- Table 10: dsps\_tbl\_datasource\_permission

```
CREATE TABLE IF NOT EXISTS dsps_tbl_datasource_permission (  
    pkdspdsp_id INT AUTO_INCREMENT PRIMARY KEY,  
    dspdsp_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,  
    dspdsp_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,  
    dspdsp_reg_by INT, -- Who initiated the permission change (Data Owner or DAC  
Staff)  
    fkdspds_id INT NOT NULL, -- Foreign key to dsps_tbl_datasource  
    fkisp_id_of INT NOT NULL, -- The person (Data User) requesting/granted permission  
    dspdsp_datetime DATETIME DEFAULT CURRENT_TIMESTAMP, -- When the  
permission was granted/requested  
    dspdsp_permission ENUM('Granted', 'Requested', 'Denied') NOT NULL,  
    dspdsp_notes TEXT,  
    dspdsp_status ENUM('Active', 'Inactive') DEFAULT 'Active',  
    FOREIGN KEY (fkdspds_id) REFERENCES dsps_tbl_datasource(pkdspds_id),  
    FOREIGN KEY (fkisp_id_of) REFERENCES ist_tbl_people(pkisp_id)  
);
```

-- Table 11: dsps\_tbl\_anonymous

```
CREATE TABLE IF NOT EXISTS dsps_tbl_anonymous (  
    pkdsp sano_id INT AUTO_INCREMENT PRIMARY KEY,  
    dsp sano_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,  
    fkdspds_id INT NOT NULL, -- Foreign key to dsps_tbl_datasource  
    dsp sano_client_ip VARCHAR(45), -- IPv4 or IPv6  
    dsp sano_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,  
    dsp sano_action VARCHAR(50), -- e.g., 'view_intro', 'download_attempt'  
    FOREIGN KEY (fkdspds_id) REFERENCES dsps_tbl_datasource(pkdspds_id)  
);
```

```
-- Table 12: dsps_tbl_datasource_used
CREATE TABLE IF NOT EXISTS dsps_tbl_datasource_used (
  pkdspsdspused_id INT AUTO_INCREMENT PRIMARY KEY,
  dspsdspused_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
  dspsdspused_mod_datetime DATETIME ON UPDATE CURRENT_TIMESTAMP,
  dspsdspused_reg_by INT, -- Who initiated the action (User ID)
  fkdspsdsused_id INT NOT NULL, -- Foreign key to dsps_tbl_datasource
  fkisp_id_of INT NOT NULL, -- The person (Data User or Data Owner) who used it
  dspsdspused_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (fkdspsdsused_id) REFERENCES
  dsps_tbl_datasource(pkdspsds_id),
  FOREIGN KEY (fkisp_id_of) REFERENCES ist_tbl_people(pkisp_id)
);

-- Table 14: dsps_tbl_feedback
CREATE TABLE IF NOT EXISTS dsps_tbl_feedback (
  pkdspsfb_id INT AUTO_INCREMENT PRIMARY KEY,
  dspsfb_reg_datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
  dspsfb_res_datetime DATETIME,
  dspsfb_res_by INT, -- FK to ist_tbl_users (DAC Staff who responded)
  dspsfb_client_ip VARCHAR(45),
  dspsfb_name VARCHAR(255),
  dspsfb_email VARCHAR(255),
  dspsfb_body_text TEXT NOT NULL,
  dspsfb_respond_text TEXT,
  dspsfb_status ENUM('New', 'In Progress', 'Responded', 'Closed') DEFAULT 'New',
  FOREIGN KEY (dspsfb_res_by) REFERENCES ist_tbl_users(pkisp_id)
);

-- Enable foreign key checks after all tables are created
SET FOREIGN_KEY_CHECKS = 1;
```

## 6. Project Setup (Database, Backend, Frontend)

### 1. Database Setup:

- \* Ensure MySQL is running.
- \* Connect to your MySQL server (e.g., using MySQL Workbench, DBeaver).
- \* Create a new database: `CREATE DATABASE db_dsp;`
- \* Use the database: `USE db_dsp;`
- \* Run the contents of `database/schema.sql` to create all tables.

### 2. Backend Setup:

- \* Navigate to the `backend/` directory in your terminal.
- \* Install dependencies: `npm install`
- \* Create a `.env` file in the `backend/` directory and fill in your MySQL credentials.
- \* Start the backend server: `npm start` (or `node app.js`)
- \* The backend should be running on <https://api.dsp.niph.org.kh/>.

### 3. Frontend Setup:

- \* Navigate to the `frontend/` directory in a **separate terminal**.
- \* Install dependencies: `npm install`
- \* Ensure Tailwind CSS is initialized: `npm install tailwindcss @tailwindcss/vite`.

- \* Create a .env or .env.local file in the frontend/ directory and set VITE\_API\_BASE\_URL.
- \* Start the development server: npm run dev
- \* The frontend should open in your browser, typically on <https://dsp.niph.org.kh/>.

## **7. Backend (Node.js with Express)**

## **8. Frontend (Vue 3, Vite, with Tailwind CSS)**

## **9. Testing and Review**