# Business Requirements Document (BRD)

## Student Data Viewer Application

### 1. Project Overview

#### 1.1 Purpose

The Student Data Viewer application is designed to provide a user-friendly interface for managing student information. It allows users to view, add, edit, and delete student records through a web-based interface.

#### 1.2 Scope

The application will handle basic student information including names, dates of birth, and addresses. It provides essential CRUD (Create, Read, Update, Delete) operations for student records.

### 2. Functional Requirements

#### 2.1 Student Record Management

* **View Students:** Display a list of all students with their basic information
* **Add Student:** Allow adding new student records with the following details:
  + Name (required)
  + Date of Birth (required)
  + Address (required)
* **Edit Student:** Enable modification of existing student records
* **Delete Student:** Allow removal of student records with confirmation
* **Search:** Provide ability to search students by name or address

#### 2.2 User Interface Requirements

* Clean and intuitive web interface
* Responsive design for various screen sizes
* Form validation for required fields
* Confirmation dialogs for critical actions (e.g., deletion)
* Success/error notifications for user actions
* Search functionality with real-time filtering

### 3. Technical Requirements

#### 3.1 Frontend

* HTML5, CSS3, and JavaScript
* Responsive design using modern CSS
* Client-side form validation
* Dynamic content updates without page reload
* Error handling and user feedback

#### 3.2 Backend

* Node.js server with Express.js framework
* RESTful API endpoints for CRUD operations
* JSON file storage for student data
* Error handling and validation
* Proper HTTP status codes and responses

#### 3.3 Data Structure

Student record format:

{  
 "Name": "string",  
 "DateOfBirth": "YYYY-MM-DD",  
 "Address": "string"  
}

### 4. API Endpoints

#### 4.1 GET /api/students

* Purpose: Retrieve all student records
* Response: Array of student objects

#### 4.2 POST /api/students

* Purpose: Create new student record
* Request Body: Student object
* Response: Success/error message

#### 4.3 PUT /api/students/:index

* Purpose: Update existing student record
* Request Body: Updated student object
* Response: Success/error message

#### 4.4 DELETE /api/students/:index

* Purpose: Delete student record
* Response: Success/error message

### 5. Non-Functional Requirements

#### 5.1 Performance

* Page load time under 2 seconds
* Smooth real-time search filtering
* Immediate feedback for user actions

#### 5.2 Security

* Input validation and sanitization
* Protection against common web vulnerabilities
* Secure file system operations

#### 5.3 Usability

* Intuitive navigation
* Clear error messages
* Consistent UI/UX design
* Responsive feedback for all user actions

### 6. Future Enhancements

* User authentication and authorization
* Advanced search filters
* Data export functionality
* Student photo upload
* Pagination for large datasets
* Database integration
* Audit logging

### 7. Success Criteria

* Successfully perform all CRUD operations
* Smooth and responsive user interface
* Proper error handling and user feedback
* Efficient search functionality
* Data persistence across sessions