

# Project Plan

---

## School Service Request & Voting Platform

---

### 1. Project Overview

This project is a web-based platform that allows users to submit school-related service requests, view them in a public feed, and interact with them through upvotes and downvotes. The system is designed as a **small-scale, unauthenticated application** intended to demonstrate core web development concepts including UI design, client-server interaction, database operations, and state management.

The platform uses a **centralized online database** and a **public feed-style user interface**, similar to social media platforms, while deliberately avoiding full authentication mechanisms due to project scope.

---

### 2. Objectives

- Provide a simple way for students to submit service requests
  - Allow community-driven prioritization through voting
  - Demonstrate database-backed CRUD operations
  - Show understanding of client-side and server-side responsibilities
  - Build a clean, modern, and responsive UI
- 

### 3. Project Scope

#### In Scope

- Public request submission
- Request feed display
- Online database storage
- Voting (upvote / downvote / toggle)
- Sorting by time and votes
- Client-side interaction tracking via localStorage

- Delete or update request only by creator (client key match)

## Out of Scope

- User accounts or authentication
  - Security hardening
  - DDoS protection
  - Real-time updates (e.g., WebSockets)
  - Role-based permissions
- 

## 4. System Architecture (High Level)

### Frontend

- HTML5
- CSS3 + Bootstrap
- JavaScript (Vanilla)

### Backend

- Simple REST API
- Handles:
  - Insert request
  - Fetch requests
  - Vote actions
  - Update / delete requests

### Database

- Centralized online database
  - Stores:
    - Requests
    - Votes
    - Client keys
-

## 5. User Interface Design

### 5.1 Main Feed Page

Top Section:

- Input box: "Write a request..."
- Category selector
- Submit button

Middle Section:

- Sorting controls:
  - Most recent
  - Most upvoted

Feed Section:

- Request cards:
  - Content
  - Category
  - Timestamp
  - Vote count
  - Upvote / Downvote buttons
  - Delete button (visible only if client key matches)

Empty State:

- Message encouraging first request submission
- 

## 6. Core Features

### 6.1 Request Submission

- Users can submit a request without authentication
  - A client-generated identifier is stored in localStorage
  - Identifier is saved with the request in the database
- 

### 6.2 Request Feed

- All requests are displayed in a scrollable feed
  - Requests are loaded from the database
  - Feed updates after submission or voting
- 

## 6.3 Voting System

- Upvote and downvote supported
  - Each browser can vote once per request
  - Votes can be toggled
  - Vote records are stored in the database
- 

## 6.4 Sorting & Preferences

- Users can sort requests by:
    - Most recent
    - Most upvoted
  - Selected preference is remembered locally
- 

## 6.5 Update & Delete Control

- Requests can be deleted or updated only if:
    - The client key in localStorage matches the request's stored key
  - No authentication is used
- 

# 7. Data Design

## Requests Table

id  
content  
category  
created\_at  
client\_key

## Votes Table

```
id
request_id
client_key
vote_type
```

---

## 8. Non-Functional Considerations

- Responsive design using Bootstrap
  - Compatible with modern browsers
  - No authentication or security hardening
  - Designed as a prototype
  - Potential vulnerabilities acknowledged and documented
- 

## 9. Limitations & Assumptions

- Client-side identifiers can be cleared or manipulated
  - The system is vulnerable to spam and abuse
  - Suitable only for small-scale or academic use
  - Security is intentionally minimal due to scope
- 

## 10. Development Timeline (3 Weeks)

### Week 1 – Design & Setup

- Finalize UI layout
- Set up database schema
- Build static frontend structure

### Week 2 – Core Functionality

- Request submission logic
- Feed rendering
- Voting system
- Sorting logic

## Week 3 – Polishing & Testing

- UI refinement
  - Bug fixing
  - Edge case handling
  - Presentation preparation
- 

## 11. Work Distribution (Final)

### You – Lead Developer / Architect

- Overall system design
  - Database schema creation
  - Backend API logic
  - Voting logic and toggle rules
  - Client key generation and validation
  - Sorting algorithms
  - Final integration
- 

### Team Member 2 – Frontend Developer

- HTML structure
  - Bootstrap layout
  - Feed UI implementation
  - Forms and input components
  - Sorting controls UI
- 

### Team Member 3 – UI/UX & Styling

- Custom CSS
  - Responsive behavior
  - Visual feedback (vote states, buttons)
  - Empty states and error messages
  - UI polish and consistency
- 

## 12. Evaluation Readiness

This project demonstrates:

- Practical web development skills
- Database interaction
- Client-server communication
- Thoughtful trade-offs
- Real-world inspired UI design

It is intentionally scoped to balance **functionality**, **clarity**, and **academic requirements**.

---

## 13. Final Note (Important)

Your project is **not simple**, but it is **controlled**.

As long as you **explain why decisions were made**, this will be seen as a **strong submission**, not an overambitious one.

---