Something Unique

- Design Document

Table of Contents

Overview

This platform aims to create a **dedicated space for college students** to interact, share, and collaborate within their own universities. Students can sign up using their campus email IDs, select their universities, and explore forums, lost & found posts, and anonymous confessions specific to their college.

The platform emphasizes **user anonymity**, ensuring that no personal details are shared in public. **Key features** include:

- **Forum pages** for each university.
- Lost & found sections to locate or report missing items.
- Anonymous confession posting.
- Search functionality for posts, users, and lost items.
- Profile and account settings management.

This platform is **tailored to the younger generation** with a humorous and catchy design theme, and it adopts modern technologies like **Next.js**, **Express.js**, **Tailwind CSS**, and **MongoDB**.

Functional Requirements

- User registration/login via student email and OTP (no password).
- Forums specific to each university.
- Lost & Found section with keyword search capability.
- Anonymous confessions with username visibility only.
- Profile management with anonymity features.
- Admin roles: Super Admin, College Admins, Club Admins, and Students.
- AI-based post moderation (future scope).
- Posts include text and attachments (images, videos).

Non-Functional Requirements

- **Scalability**: Support additional colleges as needed.
- **Security**: Maintain user anonymity and data privacy.
- **Usability**: Fun and engaging interface for Gen Z users.

Project Goals

- Offer a centralized hub for college-specific discussions.
- Ensure user anonymity while enabling profile and post visibility.
- Provide intuitive search and posting functionalities.

Technology Stack

- **Frontend**: Next.js, Tailwind CSS, shadcn.
- **Backend**: Express.js.
- **Database**: MongoDB.
- **Hosting**: Vercel .

Roles and Permissions

- **Super Admin**: Full platform control, including adding colleges. -
- **College Admins**: Approve and manage posts/clubs.
- **Club Admins**: Post updates about their college events.
- **Students**: Create and interact with posts anonymously.

Website Theme

- **Color Scheme**: Black and white with optional themes (zinc, purple, blue, orange).
- **Typography**: Humorous, catchy, and cool to appeal to students.

Architecture

High-Level Architecture

The architecture consists of four main layers:

- **Client Layer**: Handles the user interface in the web browser.
- **Frontend Layer**: Built using **React** (Next.js), managing routing, state, and components.
- **Backend Layer**: Includes the **Express.js server**, which handles authentication, API routing, controllers, and middleware.
- **Database Layer**: Uses **MongoDB** for data storage, along with Mongoose for handling database calls.

Frontend Design

Folder Structure

```
public
src
├────────app
│    ├────────app
│    │    └────────feed
│    ├────────fonts
│    ├────────login
│    ├────────profile
│    │    └────────[username]
│    ├────────register
│    ├────────settings
│    │    └────────updateusername
│    └────────store
├────────components
│    ├────────auth
│    │    └────────register
│    ├────────feed
│    ├────────navigation
│    ├────────settings
     │    ├────────sidebar
│    └────────ui
├────────constants
├────────lib
└────────types
```

- *The **app directory** represents routes in the Next.js project.*
- ***Components**: Modular UI components like navigation, feed, and settings.*
- ***Constants**: Holds shared constants used throughout the app.*
- ***Lib**: Shared utility functions and services.*

Backend Design

Folder Structure

```
├────────middleware
├────────models
├────────public
│    └────────images
├────────routes
└────────utils
```

- **Middleware**: Contains utilities like request fetchers.
- **Models**: Mongoose models representing application entities.
- **Routes**: API route handlers for authentication, posts, users, and comments.
- **Utils**: Helper functions for tasks like email notifications.

API Routes

/api/feed

Endpoint	Request Type	Description
/get-by-category	GET	Fetch posts by category.
/get-post	GET	Retrieve a single post.
/delete-post	DELETE	Remove a post.
/vote-post	POST	Upvote or downvote a post.

/api/comments

Endpoint	**Request Type**	**Description**
/add	POST	Add a comment to a post.
/delete	DELETE	Remove a comment.
/get-comments-for-post	GET	Retrieve all comments for a specific post.

/api/auth

Endpoint	**Request Type**	**Description**
/create-user	POST	Register a new user.
/get-user	GET	Fetch user details.
/check-username-availability	GET	Check if a username is available.
/send-otp	POST	Send OTP to the registered email.
/get-colleges	GET	Retrieve a list of available colleges.
/verify-otp	POST	Verify the OTP for authentication.
/delete-user	DELETE	Remove a user account.

/api/user

Endpoint	Request Type	Description
/get-suggestions	GET	Fetch user suggestions across universities.
/update-profile-picture	POST	Update the user’s profile picture.
/update-username	POST	Update the username.
/update-bio	POST	Update the bio information.
/get-profile	GET	Retrieve the profile of a user.
/get-user	GET	Fetch user details.
/create-post	POST	Create a new post.

Security Measures

User Authentication

JWT Tokens

- Tokens are securely generated and signed with a secret key to ensure their integrity.
- Tokens are stored in the browser's localStorage, allowing persistent user sessions.

Password Protection

- Passwords are securely hashed with bcrypt before being saved in the MongoDB database.
- No plain-text passwords are ever stored or logged, ensuring user privacy.

User Authorization

Secured Routes

- Backend APIs require valid JWT tokens for any sensitive route access, ensuring only authenticated users can interact with specific resources.

Input Safety

- Validation checks are applied both on the client-side (via Next.js components) and server-side (via Express.js middleware) to ensure data integrity and security.

Basic Models

User Model

The **User** schema represents a user within the platform, storing essential information like their username, email, bio, and college affiliation. Here's a breakdown of its fields:

- **username**: A required, unique string that serves as the user's display name.
- **email**: A required, unique string representing the user’s email address, used for login and communication.
- **bio**: An optional string where the user can describe themselves.
- **college_id**: A reference to the **College** model, linking the user to their college. This is a required field, indicating the student's current college.
- **avatar**: A string representing the URL of the user's profile picture. If not provided, it defaults to an empty string.
- **timestamps**: Automatically generated fields that track when the user was created and last updated (i.e., createdAt and updatedAt).

Post Model

The **Post** schema represents a post created by a user on the platform. It includes various categories such as "lostAndFound," "confession," "forum," and "event." Here's an explanation of the fields:

- **user_id**: A required reference to the **User** model, identifying the user who created the post.
- **college_id**: A required reference to the **College** model, indicating the college where the post is related. This helps filter posts by college.
- **category**: A required string that categorizes the post. It is restricted to one of the following options: lostAndFound, confession, forum, or event.
- **caption**: A required string providing the content or description of the post.
- **attachments**: An array of objects that can include either images or videos. Each attachment includes a type (either 'image' or 'video') and a url pointing to the media file. This field defaults to an empty array.

About the Developers

The **Something unique** platform is being developed by a dedicated team of students and developers working together to create a user-friendly, engaging, and secure platform for college students. This platform aims to provide a space for students to share experiences, stay connected, and engage in various activities specific to their universities.

- **Developer 1**: Sahil Poonia
  - *Role*: Full-stack Developer
  - *Responsibilities*: Designing and developing the front-end using Next.js, building out backend API routes in Express.js, and ensuring seamless integration between the two. Focused on making the platform interactive and user-friendly, with special attention to security and user anonymity.
- **Developer 2**: Utsav Singh
  - *Role*: Backend Developer
  - *Responsibilities*: Handling backend logic, database models, user authentication, and implementing features like post moderation and administrative roles. Ensuring that the backend is scalable, secure, and optimized for performance.