IPS Tech Community Forum

Core Version (MVP) – Detailed Documentation

---

1. Project Overview

1.1 Project Title

IPS Tech Community Forum

1.2 Objective

To build a structured, searchable, and organized web-based community platform that replaces unstructured WhatsApp discussions with a knowledge-centric forum system for IPS Tech Community members.

1.3 Problem Statement

The current WhatsApp group: - Lacks structured discussion - Is not searchable effectively - Causes repeated questions - Does not preserve knowledge long-term - Makes categorization difficult

1.4 Proposed Solution

A web-based forum that: - Organizes discussions by categories - Enables structured Q&A - Supports search functionality - Tracks user contributions - Preserves knowledge permanently

---

2. Scope of the Project

In Scope (MVP)

- User authentication
- Role-based access
- Categories & subcategories
- Post creation
- Commenting & replying
- Upvote system
- Search functionality
- Tagging system
- Basic admin moderation

Out of Scope (Future Phases)

- AI-based search
- Mentor booking
- Internship board
- Resume analyzer
- Payment integration

---

3. System Architecture

3.1 Architecture Type

Client-Server Architecture

3.2 Technology Stack

Frontend

- React.js
- Tailwind CSS / Bootstrap
- Axios (API calls)

Backend

- Node.js
- Express.js

Database

- MongoDB (NoSQL)

Authentication

- JWT (JSON Web Token)
- bcrypt (Password hashing)

Hosting (Optional)

- Frontend: Vercel / Netlify
- Backend: Render / Railway
- Database: MongoDB Atlas

---

4. User Roles & Permissions

4.1 Student

- Register & login

- Create posts
- Comment & reply
- Upvote posts
- Edit/delete own posts

4.2 Admin

- All student permissions
- Delete inappropriate posts
- Manage categories
- Block users
- View reports

---

5. Functional Requirements

5.1 User Authentication Module

Features

- Register (Name, Email, Password, Role)
- Login
- Logout
- Forgot password (optional for MVP)
- JWT-based session management

Validation

- Email uniqueness
- Password minimum 6 characters
- Proper email format

5.2 Category Management

Default Categories

- Programming
- AI/ML
- Web Development
- Hackathons
- Internships
- Placements
- Projects

Features

- View categories
- Admin can add/delete categories

5.3 Post Management

Create Post

Fields: - Title - Description - Category - Tags (multiple) - Author

View Post

- Title
- Content
- Author
- Timestamp
- Upvotes count
- Comments section

Edit/Delete

- Only by author or admin

5.4 Comment System

- Add comment
- Reply to comment
- Edit/delete own comment
- Nested replies (optional in MVP)

5.5 Upvote System

- Users can upvote once per post
- Prevent duplicate voting
- Store user ID in upvote list

5.6 Search Functionality

Search based on: - Title - Tags - Category - Keywords in content

Use MongoDB text indexing.

5.7 Tagging System

- Users can add tags during post creation
- Tags stored as array
- Searchable

- Displayed under post title

Example:

MERN #Python #Hackathon

---

6. Non-Functional Requirements

6.1 Performance

- Page load < 3 seconds
- Efficient pagination for posts

6.2 Security

- Password hashing (bcrypt)
- JWT authentication
- Input validation
- XSS protection
- CORS configuration

6.3 Scalability

- Modular backend structure
- Separate controllers
- Clean API routing

6.4 Usability

- Clean UI
- Mobile responsive
- Easy navigation

---

7. Database Design (MongoDB Schema)

7.1 User Schema

```
{
  _id,
  name: String,
  email: String,
  password: String (hashed),
  role: "student" | "admin",
  createdAt: Date
}
```

7.2 Category Schema

```
{
  _id,
  name: String,
  description: String,
  createdAt: Date
}
```

7.3 Post Schema

```
{
  _id,
  title: String,
  content: String,
  author: ObjectId (User),
  category: ObjectId (Category),
  tags: [String],
  upvotes: [ObjectId],
  comments: [ObjectId],
  createdAt: Date,
  updatedAt: Date
}
```

7.4 Comment Schema

```
{
  _id,
  post: ObjectId (Post),
  author: ObjectId (User),
  content: String,
  parentComment: ObjectId (optional),
  createdAt: Date
}
```

8. API Endpoints Structure

Auth Routes

- POST /api/auth/register

- POST /api/auth/login

Category Routes

- GET /api/categories
- POST /api/categories (Admin)

Post Routes

- GET /api/posts
- GET /api/posts/:id
- POST /api/posts
- PUT /api/posts/:id
- DELETE /api/posts/:id

Comment Routes

- POST /api/comments
- DELETE /api/comments/:id

Upvote Route

- POST /api/posts/:id/upvote

---

9. UI Structure

9.1 Homepage

- Navbar
- Categories sidebar
- Trending posts
- Recent posts

9.2 Post Page

- Title
- Tags
- Author
- Content
- Upvote button
- Comments section

9.3 Dashboard

- My Posts
- Create Post
- Profile

---

10. Folder Structure (Backend)

```
/backend
  /controllers
  /models
  /routes
  /middlewares
  /config
  server.js
```

---

11. Development Roadmap

Week 1

- Setup backend
- Setup MongoDB
- Create user auth

Week 2

- Categories
- Post CRUD
- Comment system

Week 3

- Upvote
- Search
- UI polishing

Week 4

- Testing
- Deployment

---

12. Future Enhancements

- AI Smart Search
- Duplicate question detection
- Leaderboard
- Internship board
- Resume analyzer
- WhatsApp sync
- Real-time chat

13. Success Metrics

- Number of registered users
- Number of posts created
- Daily active users
- Reduction in repeated questions
- Search usage frequency

14. Conclusion

The IPS Tech Community Forum MVP provides: - Structured knowledge management - Organized discussion channels - Searchable and permanent storage - Scalable foundation for future AI integration

It transforms a casual WhatsApp group into a professional, centralized technical knowledge platform.