GyanMitra Website - Complete Development Guide

For: Frontend Developer (React)
Backend API: Already implemented and documented
Platform: Web (Desktop & Mobile Responsive)
Tech Stack: React 18+, React Router, Tailwind CSS (recommended)
Time Estimate: 60-80 hours

Table of Contents

1. Project Overview

What is GyanMitra?

GyanMitra is an AI-powered educational assistant that helps students (Grades 5-10) learn from NCERT textbooks through an intelligent chat interface.

Key Features:

- **AI Chat Interface** - Students ask questions, get answers with citations
- **Multi-subject Support** - Math, Science, Social Science, English, Hindi, Sanskrit
- **Conversation History** - Save and resume conversations
- **Source Citations** - Every answer includes NCERT textbook references
- **Multi-language Support** - English, Hindi, Marathi, Urdu
- **Email Verification** - Secure account system
- **Feedback System** - Rate AI responses

2. Technical Architecture

Frontend Structure:

```
gyanmitra-web/
├── public/
│   ├── index.html
│   ├── logo.svg
│   └── favicon.ico
├── src/
│   ├── components/        # Reusable UI components
│   ├── pages/            # Route pages
│   ├── services/         # API integration
│   ├── contexts/         # React Context (Auth, etc.)
│   ├── utils/            # Helper functions
│   ├── hooks/            # Custom React hooks
│   ├── styles/           # Global styles
│   ├── types/            # TypeScript interfaces
│   └── config/           # Configuration files
├── .env                  # Environment variables
└── package.json
```

Tech Stack Recommendations:

Layer	Technology	Purpose
Framework	React 18+	UI framework
Routing	React Router v6	Navigation
State	Context API	Global state
Styling	Tailwind CSS	Utility-first CSS
HTTP	Axios	API calls
Forms	React Hook Form	Form validation
Icons	Heroicons	Icon library
Toast	React Hot Toast	Notifications

3. Design System & Color Palette

Color Scheme:

```
/* Primary Colors (Purple Gradient) */
--primary-start: #667eea
--primary-end: #764ba2
--primary-solid: #6366f1

/* Success Colors (Green) */
--success-light: #E8F5E9
```

```
--success: #4CAF50
--success-dark: #2E7D32

/* Error Colors (Red) */
--error-light: #FFEBEE
--error: #F44336
--error-dark: #C62828

/* Neutral Colors */
--background: #FAFAFA
--surface: #FFFFFF
--card: #F5F5F5
--border: #E0E0E0

/* Text Colors */
--text-primary: #2C3E50
--text-secondary: #7F8C8D
--text-light: #95A5A6
--text-white: #FFFFFF

/* Shadow */
--shadow: 0px 2px 8px rgba(0, 0, 0, 0.1)
```

Typography:

- **Font Family:** Inter, SF Pro Display, -apple-system, sans-serif
- **Headings:** 700 weight
- **Body:** 400 weight
- **Small Text:** 11-13px
- **Body Text:** 14-16px
- **Headings:** 18-32px

Spacing System:

- Use 4px base unit (4, 8, 12, 16, 20, 24, 32, 40, 48)
- Consistent padding/margins across components

Border Radius:

- **Small:** 8px (buttons, inputs)
- **Medium:** 12px (cards)
- **Large:** 16px (modals)
- **Round:** 50% (avatars)

4. Module Breakdown

Module 1: Project Setup & Configuration (2 hours)

Purpose:

Set up the React project with all dependencies and folder structure.

Tasks:

1.1: Initialize React Project

- Create React app with TypeScript
- Install dependencies: React Router, Axios, Tailwind CSS
- Configure Tailwind CSS
- Set up absolute imports (@/ paths)

1.2: Environment Configuration

- Create `.env` file
- Add backend API URL
- Configure API base URL

Environment Variables:

```
REACT_APP_API_URL=https://api.gyanmitra.com/api
REACT_APP_WEBSITE_URL=https://gyanmitra.com
```

1.3: Folder Structure

Create the complete folder structure as shown in Technical Architecture section.

1.4: Global Styles

- Set up Tailwind CSS configuration
- Create global CSS file
- Define color variables
- Set up fonts

Deliverables:

- ✅ Working React development server
- ✅ Tailwind CSS configured
- ✅ Folder structure created

- ✅ Environment variables set

Module 2: Authentication System (8 hours)

Purpose:

Build complete user authentication with login, registration, and email verification.

Routes:

- `/login` - Login page
- `/register` - Registration page
- `/verify-email?token=xxx` - Email verification

Backend API Routes:

```
POST /api/auth/register
POST /api/auth/login
GET  /api/auth/verify-email?token=xxx
GET  /api/auth/me
```

Task 2.1: Auth Service (API Integration)

File: `src/services/authService.js`

Purpose: Handle all API calls related to authentication

Methods to Implement:

1. `register(name, email, password, grade, subjects)` → POST /auth/register
2. `login(email, password)` → POST /auth/login
3. `verifyEmail(token)` → GET /auth/verify-email
4. `getCurrentUser()` → GET /auth/me
5. `logout()` → Clear local storage

Storage:

- Store JWT token in localStorage
- Store user data in localStorage
- Include token in all API requests (Authorization header)

Task 2.2: Auth Context

File: `src/contexts/AuthContext.jsx`

Purpose: Manage global authentication state

State to Manage:

- `user` - Current user object
- `isAuthenticated` - Boolean
- `isLoading` - Loading state
- `login(email, password)` - Login function
- `register(...)` - Registration function
- `logout()` - Logout function

Features:

- Auto-load user on app start
- Redirect to login if not authenticated
- Persist token across page refreshes

Task 2.3: Protected Route Component

File: `src/components/ProtectedRoute.jsx`

Purpose: Restrict access to authenticated users only

Logic:

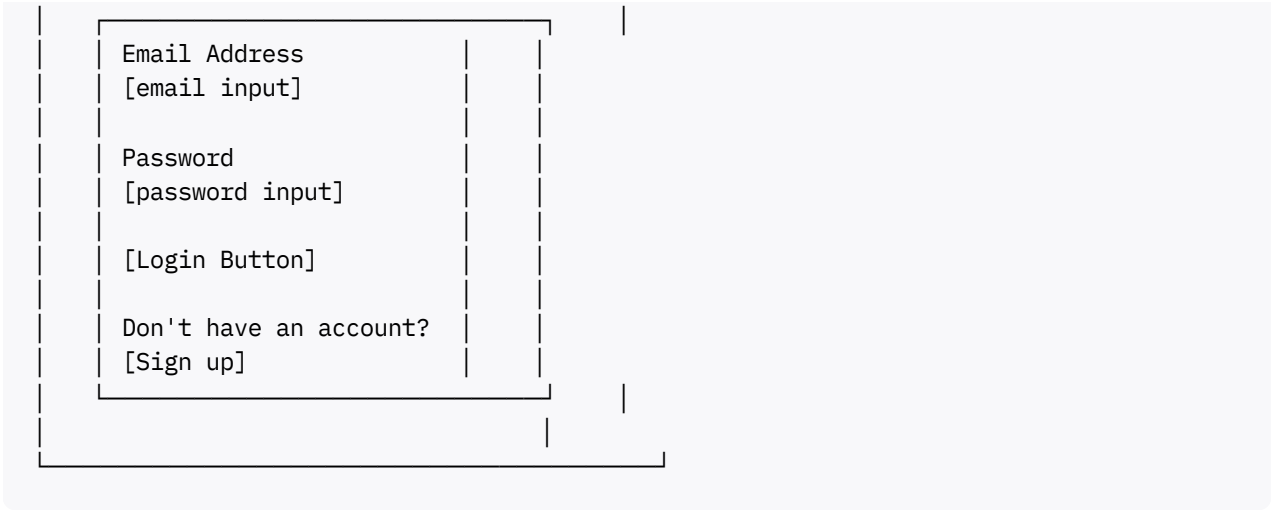
- Check if user is authenticated
- If yes → Render children
- If no → Redirect to /login

Task 2.4: Login Page

Route: `/login`
File: `src/pages/LoginPage.jsx`

Layout:

```
┌─────────────────────────────────────────┐
│                                         │
│          [GyanMitra Logo]               │
│                                         │
│       Welcome Back to GyanMitra         │
│      Learn with NCERT AI Assistant      │
│                                         │
```



Design Specs:

- **Container:** Max-width 400px, centered, padding 40px
- **Logo:** 60px height, centered, margin-bottom 32px
- **Title:** 28px font-size, 700 weight, primary color
- **Subtitle:** 14px, secondary text color
- **Form:** White card with shadow, 24px padding, 12px border-radius
- **Inputs:**
  - Height 48px
  - Border 1px solid #E0E0E0
  - Border-radius 8px
  - Font-size 14px
  - Padding 12px 16px
  - Focus: border color changes to primary
- **Button:**
  - Full width
  - Height 48px
  - Gradient background (primary-start → primary-end)
  - White text
  - Border-radius 8px
  - Font-weight 600
  - Hover: slight scale effect

Form Validation:

- Email: Required, valid email format
- Password: Required, min 6 characters
- Show error messages below inputs in red

- Disable button while submitting

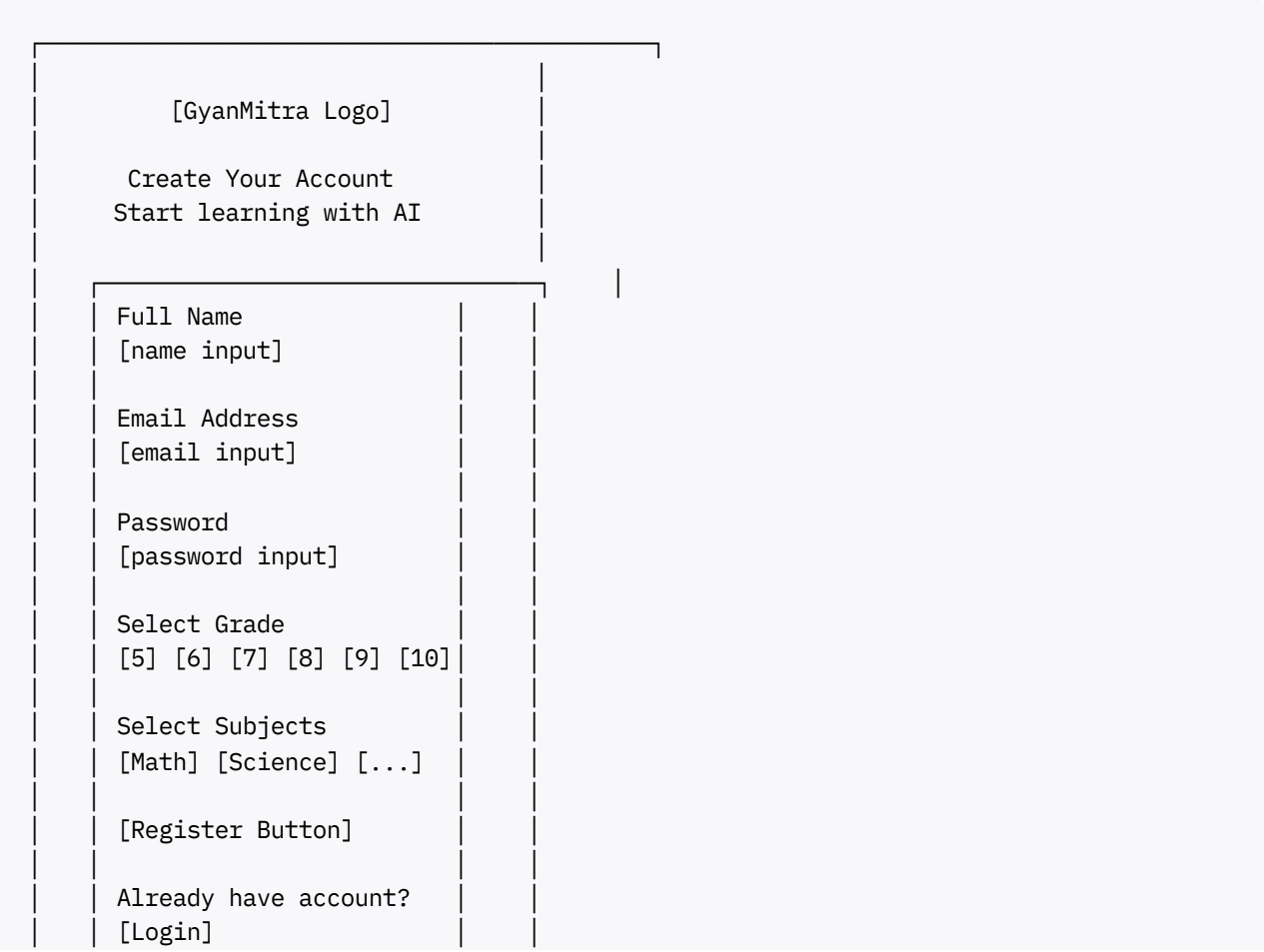
Success Flow:

1. User submits form
2. Show loading spinner on button
3. Call authService.login()
4. On success:
   - Store token in localStorage
   - Update AuthContext
   - Redirect to `/chat`
5. On error:
   - Show error toast notification
   - Enable form again

Task 2.5: Registration Page

Route: `/register`
File: `src/pages/RegisterPage.jsx`

Layout:



```
│    └──────────────────────────────┘      │
│                                          │
└──────────────────────────────────────────┘
```

Grade Selector:

- Display as pill buttons (5-10)
- Single selection
- Active grade has gradient background
- Inactive grades have border only

Subject Selector:

- Display as checkboxes with labels
- Multiple selection allowed
- Options: Math, Science, Social Science, English, Hindi, Sanskrit
- At least one subject required

Form Validation:

- Name: Required, min 3 characters
- Email: Required, valid email
- Password: Required, min 6 characters
- Grade: Required
- Subjects: At least one selected

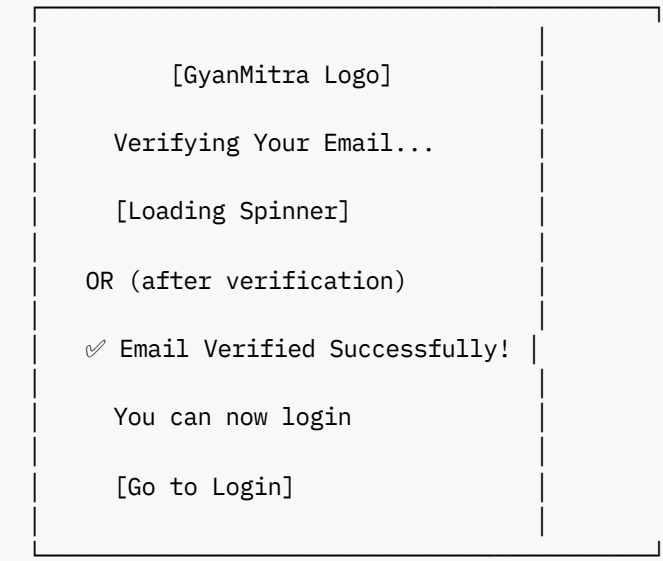
Success Flow:

1. User submits form
2. Call authService.register()
3. On success:
    - Show success message: "Registration successful! Please check your email to verify your account."
    - Redirect to `/login` after 3 seconds
4. On error:
    - Show error toast
    - Keep form data intact

Task 2.6: Email Verification Page

Route: `/verify-email?token=xxx`
File: `src/pages/VerifyEmailPage.jsx`

Layout:



Logic:

1. Extract token from URL query parameter
2. Call authService.verifyEmail(token)
3. Show loading spinner while verifying
4. On success:
    - Show success icon and message
    - Display "Go to Login" button
5. On error:
    - Show error message
    - Display "Go to Register" button

Design:

- Centered content
- Large icon (checkmark or error icon)
- 24px title
- Button below message

Module 2 Exit Criteria:

- ✅ User can register with email
- ✅ Email verification works
- ✅ User can login
- ✅ Protected routes work
- ✅ Auth persists across page refresh
- ✅ Logout works
- ✅ Form validation works
- ✅ Error handling implemented

Module 3: Main Chat Interface (10 hours)

Purpose:

Build the core AI chat interface where students interact with the assistant.

Route: `/chat`

Backend API:

```
POST /api/query
```

Task 3.1: Chat Service

File: `src/services/chatService.js`

Methods:

1. `sendQuery(query, grade, subject, language, conversationId)` → POST /query

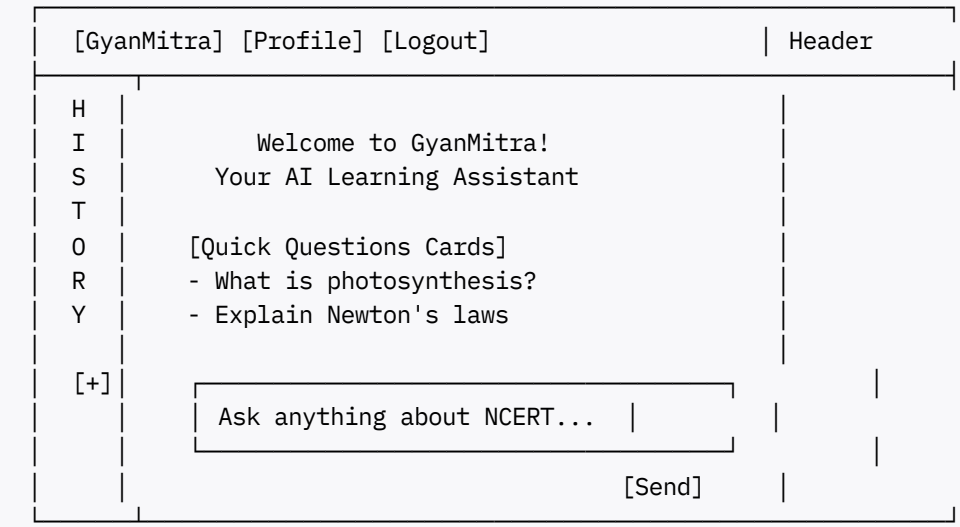
Response Handling:

- Parse response
- Extract answer, citations, conversationId
- Handle errors

Task 3.2: Chat Page Layout

File: `src/pages/ChatPage.jsx`

Layout (Desktop):



Layout Breakdown:

1. **Sidebar (Left):** 280px width
   - Conversation history list
   - "New Chat" button at top
   - Each conversation shows first message preview
   - Scroll if many conversations
2. **Main Area (Center):** Flex grow
   - Messages display area
   - Empty state (when no messages)
   - Message bubbles (user + AI)
   - Input bar fixed at bottom
3. **Header:** 60px height
   - Logo on left
   - User profile dropdown on right
   - Logout option

Task 3.3: Subject Selector

Position: Top of chat area
File: `src/components/chat/SubjectSelector.jsx`

Layout:

```
[Math ▼]  Grade 8
```

Design:

- Dropdown menu for subject selection
- Shows current grade (read-only)
- Options: Math, Science, Social Science, English, Hindi, Sanskrit
- Changes conversation context when switched

Task 3.4: Message Bubble Component

File: `src/components/chat/MessageBubble.jsx`

User Message Design:

```
                    ┌──────────────────────────────┐
                    │ What is photosynthesis?      │
                    │                       11:32  │
                    └──────────────────────────────┘
```

- Aligned right
- Gradient background (primary)
- White text
- Border-radius: 16px (8px on bottom-right)
- Max-width: 70%
- Padding: 12px 16px
- Timestamp below (11px, light text)

AI Message Design:

```
┌──────────────────────────────────────────────────┐
│ Photosynthesis is the process by which           │
│ green plants make food using sunlight.           │
│                                                  │
│ Sources:                                         │
│  ┌──────────────────────────────────────────┐    │
│  │ [1] NCERT Science Grade 8                │    │
│  │     Chapter 7: Nutrition in Plants       │    │
│  │     Page 92                              │    │
```

```
│  └──────────────────────────────────────────┘    │
│                                                  │
│ Was this helpful? 👍 👎                          │
│                                                  │
│                                    11:32 │       │
└──────────────────────────────────────────────────┘
```

- Aligned left
- White background
- Border: 1px solid #E0E0E0
- Border-radius: 16px (8px on bottom-left)
- Max-width: 70%
- Padding: 16px
- Timestamp below
- Citations section (if present)
- Feedback buttons (thumbs up/down)

Task 3.5: Citation Card Component

File: `src/components/chat/CitationCard.jsx`

Design:

```
┌──────────────────────────────────────────┐
│ [1] NCERT Science Grade 8          │     │
│     Chapter 7: Nutrition in Plants │     │
│     Page 92                        │     │
│                                    │     │
│     "Plants make their own food    │     │
│     through the process of..."     │     │
│                                    │     │
│     [View Details →]               │     │
└──────────────────────────────────────────┘
```

Specs:

- Background: #F5F5F5
- Border: 1px solid #E0E0E0
- Border-radius: 8px
- Padding: 12px
- Margin: 8px 0
- Number badge: Circular, primary color
- Title: 14px, bold

- Details: 13px, secondary color
- Excerpt: 12px, italic, gray
- Hover: slight shadow increase

Task 3.6: Input Bar Component

File: `src/components/chat/InputBar.jsx`

Design:

```
┌──────────────────────────────────────────────────────┐
│ Ask anything about NCERT textbooks...     │
│                                        [Send]│
└──────────────────────────────────────────────────────┘
```

Specs:

- Fixed at bottom of chat area
- Background: White
- Border-top: 1px solid #E0E0E0
- Padding: 16px
- Input:
  - Flex grow
  - Border: 1px solid #E0E0E0
  - Border-radius: 24px
  - Padding: 12px 20px
  - Font-size: 14px
  - Placeholder: secondary text
- Send button:
  - Circular (48px)
  - Gradient background
  - White arrow icon
  - Disabled when input empty
  - Hover: scale effect

Features:

- Enter key submits
- Disable input while loading
- Show loading spinner in send button

- Clear input after sending

Task 3.7: Empty State Component

File: `src/components/chat/EmptyState.jsx`

Design:

```
        [Large Icon]

    Welcome to GyanMitra!
    Your AI Learning Assistant

    Quick Questions:

    ┌──────────────────────────────────┐
    │  What is Pythagoras    │
    │    theorem?              │
    └──────────────────────────────────┘

    ┌──────────────────────────────────┐
    │  Explain photosynthesis│
    └──────────────────────────────────┘
```

Specs:

- Centered in chat area
- Icon: 96px, gradient color
- Title: 28px, bold
- Subtitle: 16px, secondary
- Quick question cards:
  - Width: 300px
  - Background: white
  - Border: 1px solid #E0E0E0
  - Border-radius: 12px
  - Padding: 16px
  - Hover: shadow + scale
  - Click: Send as query

Module 3 Exit Criteria:

- ✅ User can send messages
- ✅ AI responses display correctly
- ✅ Citations show properly
- ✅ Subject selector works
- ✅ Empty state shows
- ✅ Loading states work
- ✅ Messages scroll properly
- ✅ Input validation works

Module 4: Conversation History (6 hours)

Purpose:

Display list of past conversations in sidebar with ability to load and continue them.

Backend API:

```
GET /api/conversations?page=1&limit=10
GET /api/conversations/:id
DELETE /api/conversations/:id
```

Task 4.1: Conversation Service

File: `src/services/conversationService.js`

Methods:

1. `getConversations(page, limit)` → GET /conversations
2. `getConversationById(id)` → GET /conversations/:id
3. `deleteConversation(id)` → DELETE /conversations/:id

Task 4.2: Sidebar Component

File: `src/components/chat/Sidebar.jsx`

Layout:

```
┌──────────────────────────┐
│  [+ New Chat]            │
├──────────────────────────┤
│  Today                   │
```

Design:

- Background: #FAFAFA
- Border-right: 1px solid #E0E0E0
- "New Chat" button:
  - Full width
  - Primary gradient background
  - White text
  - 12px border-radius
  - Margin: 16px
  - Icon: Plus sign

Conversation Cards:

- Background: white (hover: #F5F5F5)
- Border-radius: 8px
- Padding: 12px
- Margin: 8px 16px
- Title: 14px, bold, truncate to 1 line
- Timestamp: 11px, light text
- Active conversation: primary border (2px)
- Hover: cursor pointer

Grouping:

- Group by date: "Today", "Yesterday", "Last 7 Days", "Older"
- Headers: 12px, uppercase, bold, secondary color
- Padding: 16px

Delete:

- Hover on card shows delete icon (X) on right

- Click delete → Confirm dialog → Remove from list

Task 4.3: Load Conversation Logic

Purpose: When user clicks a conversation in sidebar, load all messages

Flow:

1. User clicks conversation card
2. Call conversationService.getConversationById(id)
3. Display loading spinner in chat area
4. Transform messages to UI format
5. Display all messages in chat
6. Update current conversationId
7. Highlight active conversation in sidebar

Module 4 Exit Criteria:

- ✅ Conversations list in sidebar
- ✅ Grouped by date
- ✅ Clicking conversation loads it
- ✅ Delete conversation works
- ✅ Pagination (load more) works
- ✅ Active conversation highlighted
- ✅ New chat button creates new

Module 5: Feedback System (4 hours)

Purpose:

Allow users to rate AI responses with thumbs up/down.

Backend API:

```
POST /api/feedback
```

Task 5.1: Feedback Service

File: `src/services/feedbackService.js`

Methods:

1. `submitFeedback(conversationId, messageIndex, rating)` → POST /feedback

Task 5.2: Feedback Buttons Component

File: `src/components/chat/FeedbackButtons.jsx`

Design:

```
Was this helpful? 👍 👎
```

Specs:

- Position: Below AI message
- Font-size: 11px, secondary text
- Buttons:
  - Size: 32px × 32px
  - Border-radius: 50%
  - Background: transparent (hover: #F5F5F5)
  - Icons: 16px thumbs up/down
  - Gap: 8px between buttons
- After clicking:
  - Selected button: primary color
  - Non-selected: light gray
  - Both disabled
  - Fade out after 1.5 seconds

Logic:

1. User clicks 👍 or 👎
2. Call feedbackService.submitFeedback()
3. Highlight selected button
4. Store feedback in localStorage (to prevent re-showing)
5. Fade out buttons

Module 5 Exit Criteria:

- ✅ Feedback buttons show on AI messages
- ✅ Clicking sends feedback to backend
- ✅ Visual feedback on click
- ✅ Buttons disappear after feedback
- ✅ Don't show buttons if feedback already given

Module 6: User Profile & Settings (5 hours)

Purpose:

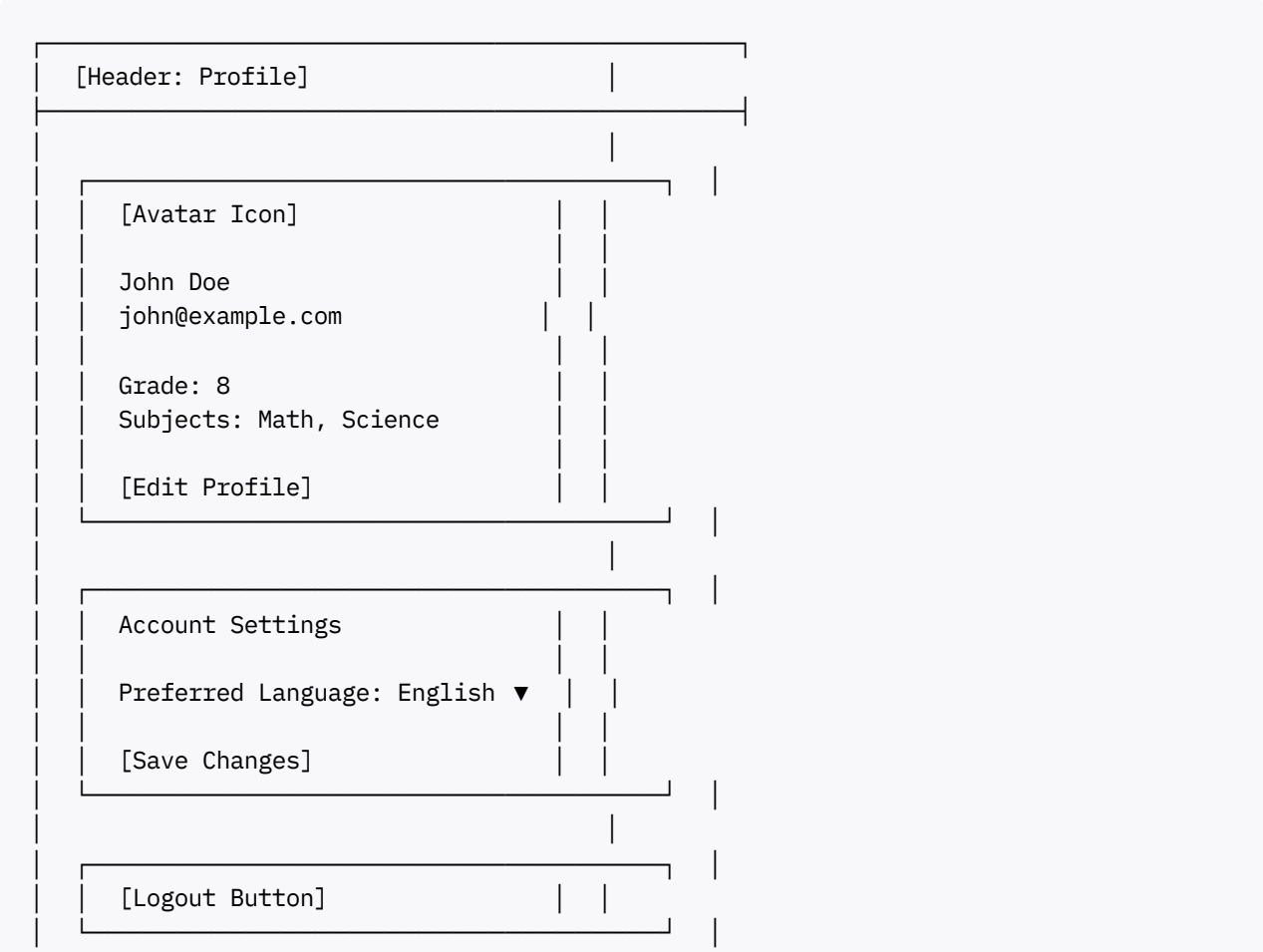
User profile page with account details and settings.

Route: `/profile`

Task 6.1: Profile Page Layout

File: `src/pages/ProfilePage.jsx`

Layout:



```
│                                          │
└──────────────────────────────────────────┘
```

Design:

- Max-width: 600px, centered
- Sections in white cards
- Avatar: 80px, circle, initials or icon
- Info: 16px name (bold), 14px email (secondary)
- Edit button: Secondary style
- Logout button: Red background

Module 6 Exit Criteria:

- ✅ Profile displays user info
- ✅ Edit profile works
- ✅ Change language works
- ✅ Logout button works

Module 7: Responsive Design (6 hours)

Purpose:

Make entire website responsive for mobile and tablet.

Breakpoints:

- Mobile: < 768px
- Tablet: 768px - 1024px
- Desktop: > 1024px

Mobile Layout Changes:

Chat Page:

```
┌──────────────────────────────┐
│ [☰] GyanMitra [🌐]  │ Header
├──────────────────────────────┤
│                              │
│  AI Message                  │
│                              │
│  User Message                │
│                              │
```

```
│                          │
│  [Input Bar]             │
└──────────────────────────┘
```

Changes:

- Hide sidebar by default
- Show hamburger menu (☰)
- Clicking hamburger opens sidebar as overlay
- Message bubbles: max-width 85%
- Input bar: full width
- Profile dropdown menu

Module 7 Exit Criteria:

- ✅ Mobile layout works
- ✅ Sidebar is toggleable on mobile
- ✅ Touch-friendly button sizes
- ✅ Text readable on mobile
- ✅ No horizontal scroll

Module 8: Error Handling & Loading States (4 hours)

Purpose:

Handle all error scenarios and show proper loading states.

Task 8.1: Loading States

Scenarios:

1. Initial page load → Skeleton loader
2. Sending message → Loading spinner in send button
3. Loading conversation → Skeleton in chat area
4. Loading history → Skeleton in sidebar

Task 8.2: Error States

Scenarios:

1. Network error → "Check your connection" banner
2. Server error → "Something went wrong" message
3. Auth error → Redirect to login
4. Invalid input → Form validation errors

Error Boundary:

- Catch React errors
- Show fallback UI
- Log to console/monitoring

Module 8 Exit Criteria:

- ✅ All loading states implemented
- ✅ All error scenarios handled
- ✅ Error boundary catches crashes
- ✅ User-friendly error messages

Module 9: Accessibility (3 hours)

Purpose:

Make website accessible to all users.

Checklist:

- ✅ Keyboard navigation works
- ✅ Focus indicators visible
- ✅ ARIA labels on buttons
- ✅ Alt text on images
- ✅ Color contrast meets WCAG AA
- ✅ Screen reader compatible
- ✅ Skip to main content link

Module 10: Performance Optimization (4 hours)

Purpose:

Optimize website performance.

Optimizations:

1. Code splitting (lazy load routes)
2. Memoize expensive components
3. Debounce input fields
4. Optimize images
5. Minimize bundle size
6. Cache API responses
7. Prefetch on hover

Module 10 Exit Criteria:

- ✅ Lighthouse score > 90
- ✅ First Contentful Paint < 1.5s
- ✅ Time to Interactive < 3.0s
- ✅ Bundle size < 300KB (gzipped)

Module 11: Testing & Deployment (6 hours)

Purpose:

Test thoroughly and deploy to production.

Task 11.1: Manual Testing

Test Cases:

1. Registration flow
2. Email verification
3. Login/Logout
4. Send message
5. View citations
6. Load conversations
7. Delete conversation

8. Give feedback
9. Profile update
10. Mobile responsiveness

Task 11.2: Deployment

Platforms:

- Vercel (recommended)
- Netlify
- AWS Amplify

Steps:

1. Build production bundle
2. Set environment variables
3. Deploy to platform
4. Configure custom domain
5. Set up HTTPS
6. Configure redirects

5. API Routes Reference

Base URL:

```
https://api.gyanmitra.com/api
```

Authentication:

All authenticated routes require:

```
Headers: {
  Authorization: "Bearer <jwt_token>"
}
```

Endpoints:

Method	Route	Purpose	Auth Required
POST	`/auth/register`	User registration	No
POST	`/auth/login`	User login	No
GET	`/auth/verify-email?token=xxx`	Verify email	No

Method	Route	Purpose	Auth Required
GET	`/auth/me`	Get current user	Yes
POST	`/query`	Submit question	Yes
GET	`/conversations`	List conversations	Yes
GET	`/conversations/:id`	Get conversation	Yes
DELETE	`/conversations/:id`	Delete conversation	Yes
POST	`/feedback`	Submit feedback	Yes
GET	`/health`	Health check	No

6. Deployment Guide

Environment Variables:

```
REACT_APP_API_URL=https://api.gyanmitra.com/api
REACT_APP_WEBSITE_URL=https://gyanmitra.com
```

Build Command:

```
npm run build
```

Deployment Platforms:

Recommended: Vercel

- Zero config deployment
- Automatic HTTPS
- Global CDN
- Instant rollbacks

Steps:

1. Push code to GitHub
2. Connect GitHub to Vercel
3. Set environment variables
4. Deploy

✅ Complete Checklist

```
Module 1: Project Setup ⏱ 2 hours
  ✓ React project initialized
  ✓ Dependencies installed
  ✓ Tailwind configured
  ✓ Folder structure created

Module 2: Authentication ⏱ 8 hours
  ✓ Auth service created
  ✓ Auth context setup
  ✓ Login page
  ✓ Register page
  ✓ Email verification
  ✓ Protected routes

Module 3: Chat Interface ⏱ 10 hours
  ✓ Chat service
  ✓ Chat page layout
  ✓ Message bubbles
  ✓ Citations
  ✓ Input bar
  ✓ Empty state

Module 4: Conversation History ⏱ 6 hours
  ✓ Sidebar component
  ✓ Load conversations
  ✓ Delete conversations
  ✓ Pagination

Module 5: Feedback System ⏱ 4 hours
  ✓ Feedback service
  ✓ Feedback buttons
  ✓ Feedback submission

Module 6: Profile & Settings ⏱ 5 hours
  ✓ Profile page
  ✓ Edit profile
  ✓ Settings

Module 7: Responsive Design ⏱ 6 hours
  ✓ Mobile layout
  ✓ Tablet layout
  ✓ Touch-friendly

Module 8: Error Handling ⏱ 4 hours
  ✓ Loading states
  ✓ Error states
  ✓ Error boundary

Module 9: Accessibility ⏱ 3 hours
  ✓ Keyboard navigation
  ✓ ARIA labels
  ✓ Screen reader

Module 10: Performance ⏱ 4 hours
```

```
  ✓ Code splitting
  ✓ Optimization
  ✓ Caching

Module 11: Testing & Deploy  6 hours
  ✓ Manual testing
  ✓ Deployment
  ✓ Production ready

_____________________________________________

Total: ~60 hours
```

Design Inspiration:

Reference these websites for design ideas:

- ChatGPT (chat interface)
- Notion (clean cards, spacing)
- Linear (color gradients, buttons)
- Vercel (landing page design)

Final Notes:

This guide provides a complete roadmap to build the GyanMitra website. Each module is designed to be completed sequentially, with clear deliverables and exit criteria.

Key Success Factors:

1. Follow the design system consistently
2. Test on multiple browsers
3. Prioritize user experience
4. Keep code clean and maintainable
5. Document as you build

Good luck building GyanMitra!