unmute

Because silence isn't strength.

unmute

Planning

The Problem:

unmute

Loneliness among men is a documented and growing social issue, often linked to stigma around emotional expression, a lack of supportive friendships, and minimal help-seeking behaviour. Many men don’t have safe, non-performative spaces to connect and share experiences without fear of judgment.

Solution:

The app was developed to tackle these issues by providing men with a safe, judgement-free space to connect and express themselves. It recognises that many men find it difficult to open up about their emotions, so it introduces anonymous posting to make seeking support easier and more comfortable.

Alongside community interaction, users can privately reflect through a secure journal and mood tracker, helping them monitor their mental well-being over time and a curated resources page offers trusted information and professional support options. Every feature is designed with privacy and confidentiality at its core.

Considered Limitations:

unmute

Moderation Needs

Anonymous platforms may attract trolling or toxic behaviour; moderation tools and role-based access will need to be robust.

Trust Building

Gaining user trust will require clear UX writing, privacy transparency, and ethical design.

Scalability

Features like the discussion boards require performance testing and optimisation to ensure smooth interaction and reliable data handling under load.

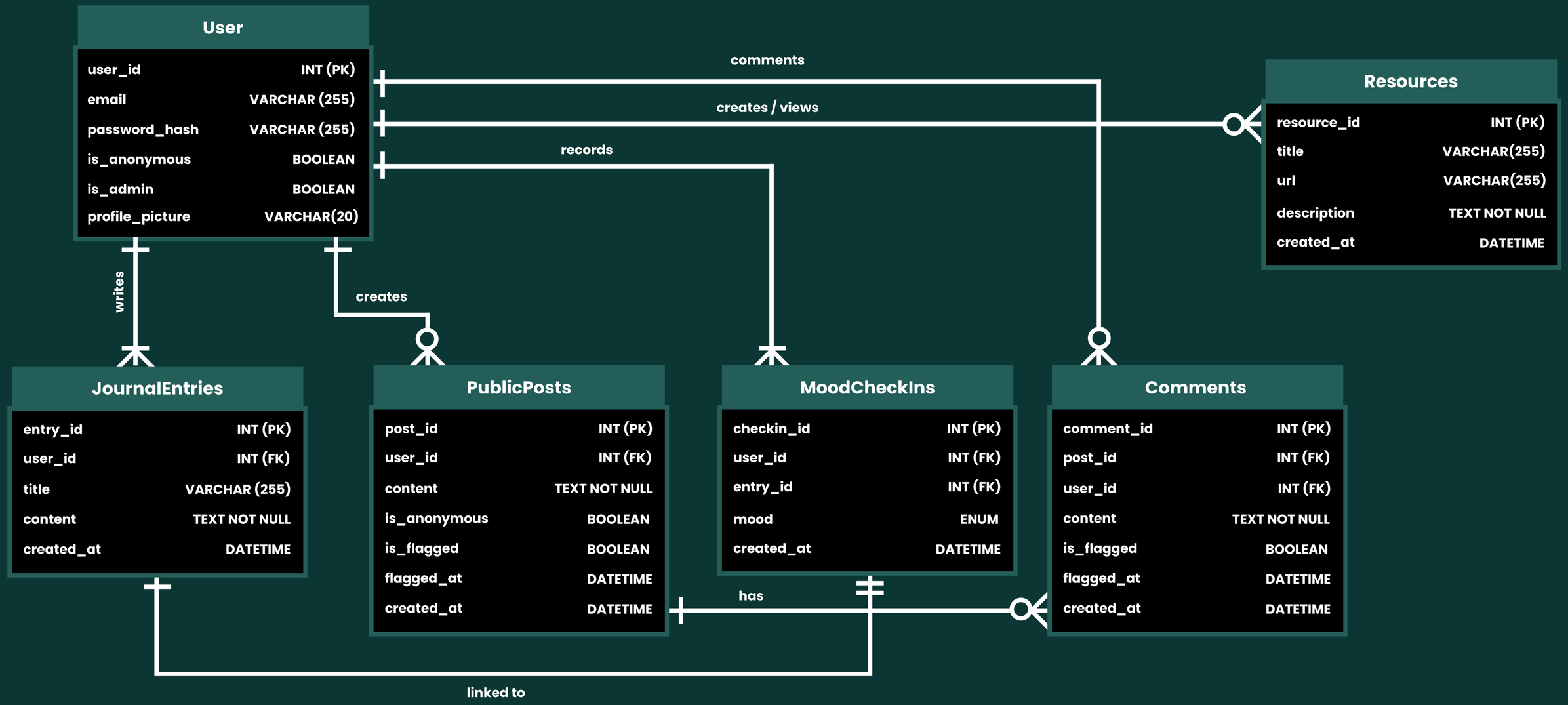
Technologies & Frameworks used:

unmute

Layer	Technology	Purpose
Frontend	VITE React.js	SPA for fast, responsive user experience
Backend	Node.js + Express	Non-blocking, lightweight, well-suited for RESTful APIs
Relational DB	MySQL	Structured relational data (Users, Threads, Admin data)
Auth	Bcrypt + JWT	Secure user login with role access (e.g., admin vs regular user)
Hosting	Heroku + Railway	React app, including the backend, was hosted on Heroku, while the MySQL database was hosted on Railway.

ER Diagram

unmute



unmute

UI & Brand Identity

UI & Brand Identity

unmute

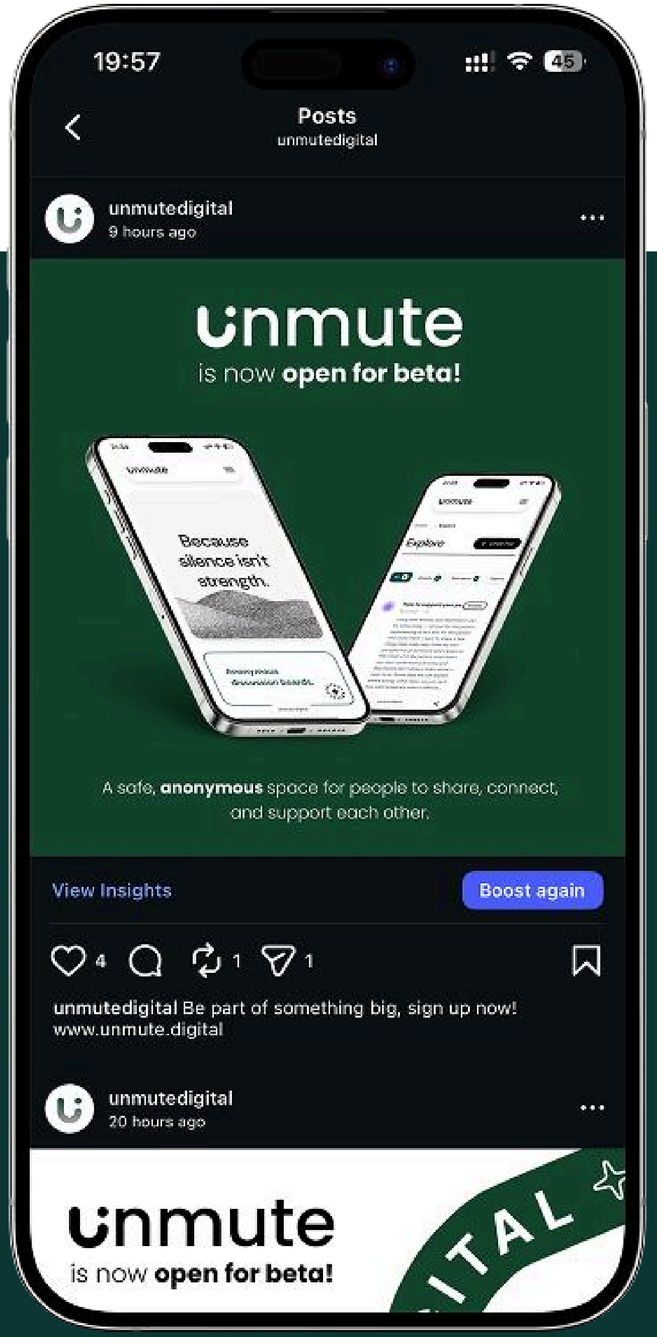
Modern & Minimal:
Designed for a calm, uncluttered user experience

Colour Palette:
Muted tones with green accents for key elements.

Responsive Design:
Optimised for phones and tablets.

Subtle Animations:
Adds personality without cluttering the interface

Brand Presence: Created an Instagram account to explore potential online identity



unmute

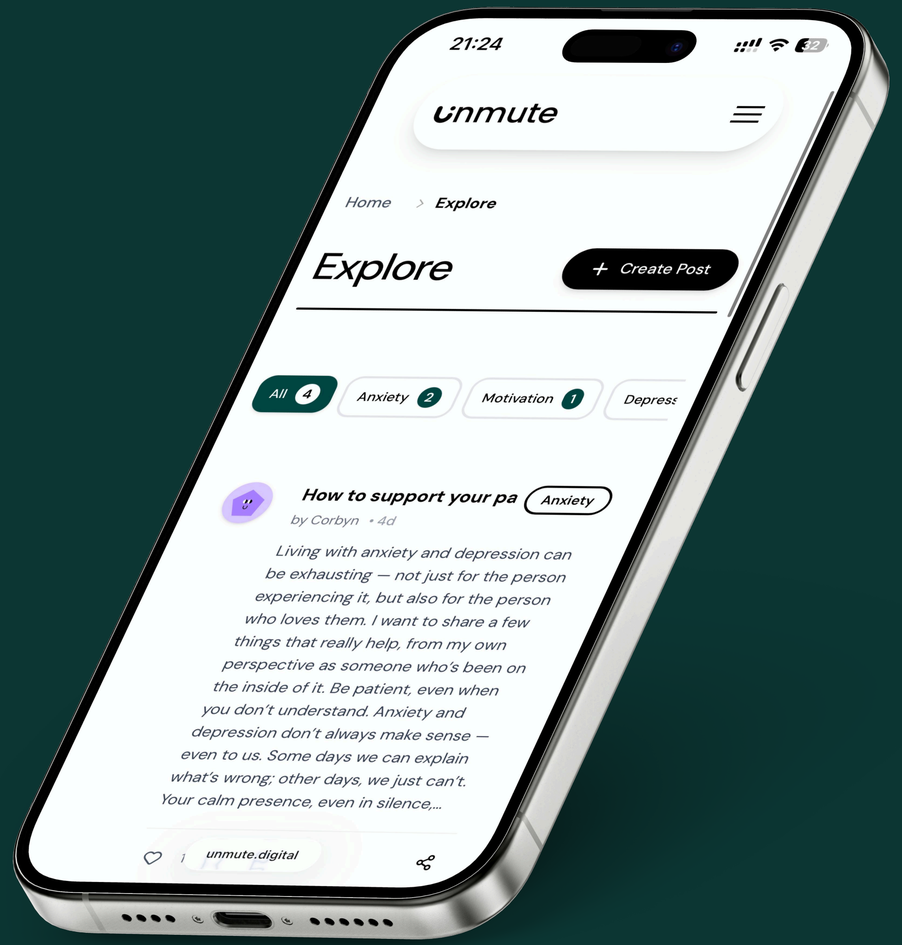
Core Features

Explore

unmute

The Explore Page is the heart of community interaction. Here, users can browse posts shared by others, filter them by topic, and join discussions that matter most to them. Each post can be liked or commented on, creating space for encouragement and meaningful conversation.

By allowing users to explore topics relevant to their experiences, this feature fosters genuine connection and reminds them that they're not alone.

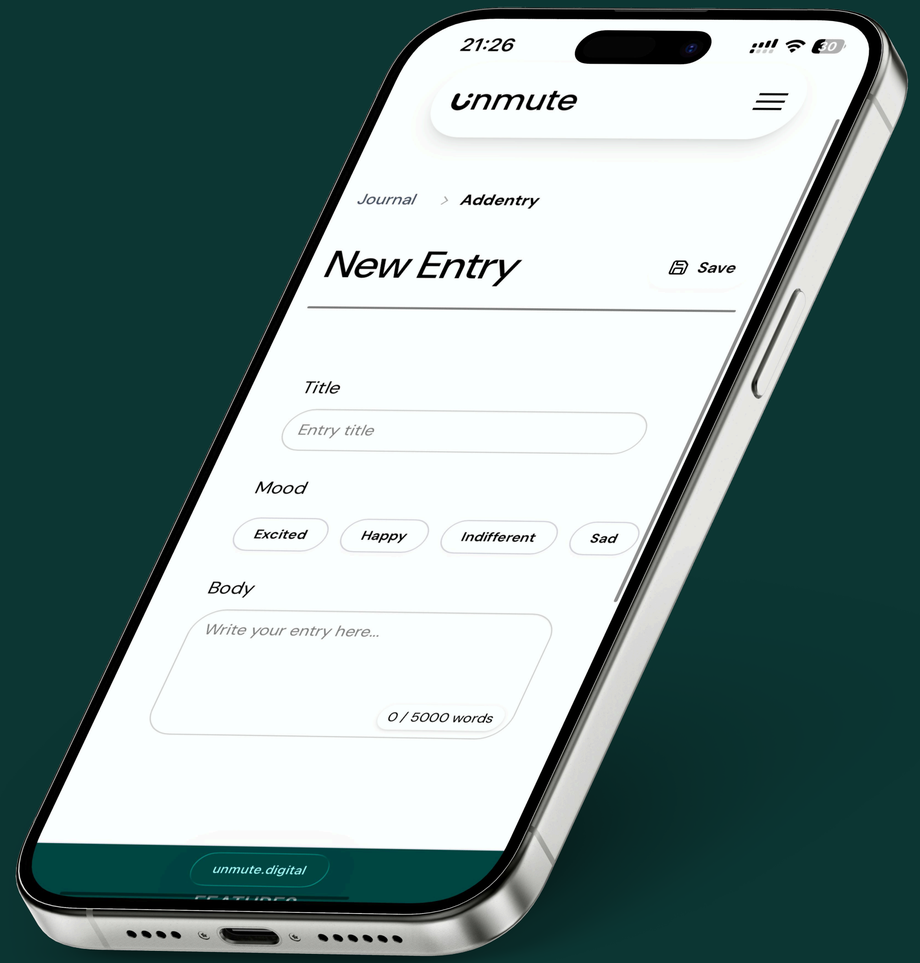


Journal

unmute

The Journal Page provides a private and secure space for users to reflect on their thoughts and emotions. Each user has their own personal journal, where they can create, edit, and delete entries freely.

Unlike the community features, journal entries are completely private and only visible to the user, encouraging honest self-reflection without judgement. This feature helps users track their emotional growth and build awareness of their mental well-being over time.

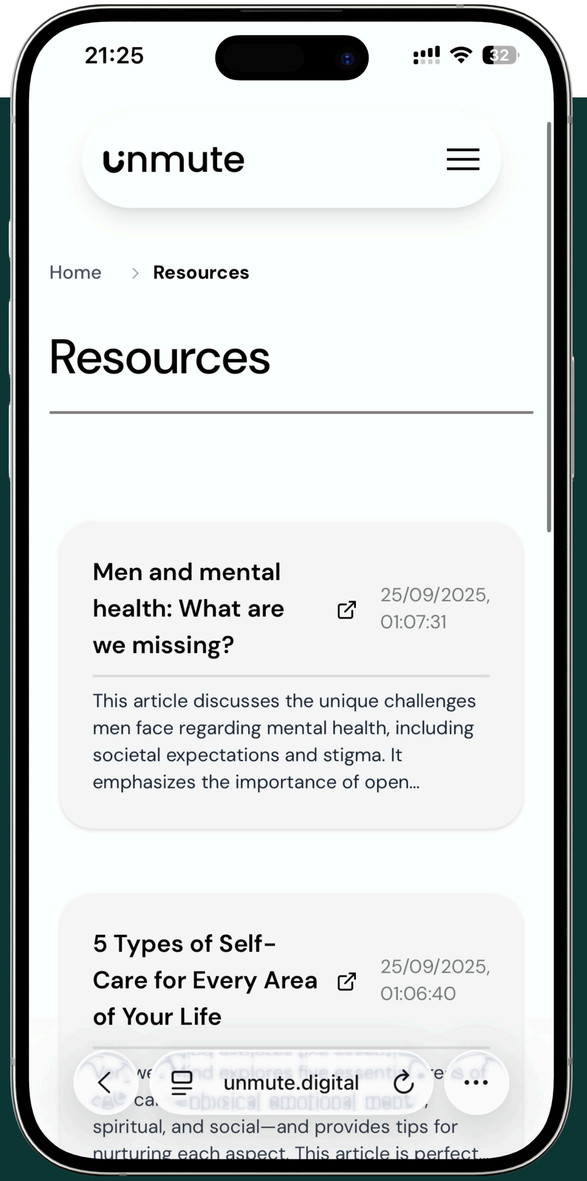


Resources

unmute

The Resources Page provides users with a curated collection of trusted mental health support materials. This includes articles, guides, and professional contacts designed to offer guidance and further assistance.

Users can easily browse and access resources relevant to their needs, making it a valuable tool for learning, self-help, and connecting with professional support when necessary.

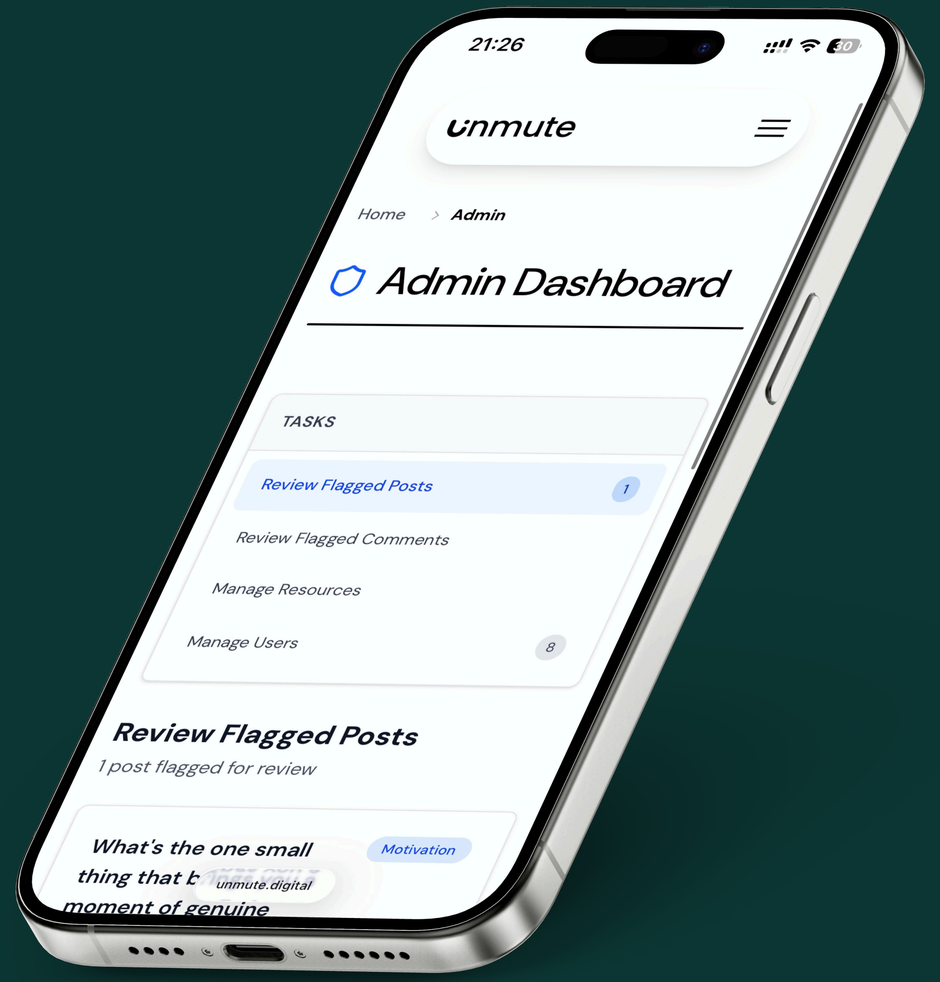


Admin Dashboard

unmute

The Admin Dashboard is the central hub for managing the app and its community. Admins can review flagged posts and comments, ensuring the platform remains a safe and supportive environment.

Admins also have tools for managing users and updating the resources page, including adding or removing entries. This feature ensures that both the community and the content stay well-moderated, accurate, and trustworthy.



unmute

CODEBASE

CRUD Functionality

unmute

Now that you understand what the app is, let's dive in to how it actually works.

Explore Page

Posts – Users can Create, read, flag, and delete posts
Comments – Users can Add, view, flag, and delete

Journal Page

Journal Entries – Users can add and delete entries

Admin Dashboard

Resources – Admins can add or delete trusted resources
Users – Admin can edit (like making an account an admin) or delete users

Example: Creating a post (Create)

```
1  // POST route: allows users to create a new post
2  router.post('/addpost', authenticateToken, async
   (req, res) => {
3    // Extract post details from the request body
4    const { title, topic, content, isAnonymous } =
   req.body;
5
6    // Insert the new post into the PublicPosts table
7    // '?' placeholders prevent SQL injection by
   using prepared statements
8    await pool.query(
9      'INSERT INTO PublicPosts (user_id, title,
   topic, content, is_anonymous) VALUES (?, ?, ?, ?,
   ?)',
10     [req.user.id, title, topic, content,
   isAnonymous]
11   );
12
13   // Send a success response back to the frontend
14   res.status(201).json({ message: 'Post created
   successfully' });
15 });
```

extracted and simplified from routes/posts.js

CRUD Functionality

unmute

Example: Making someone an admin (Update)

```
// POST route: toggle a user's admin status (admin access required)
router.post('/users/:id/toggle-admin', requireAdmin, async (req, res) => {
  try {
    // Get the user ID from the URL parameter
    const userId = req.params.id;

    // Fetch the user's current admin status from the database
    const [rows] = await pool.query(
      'SELECT user_id, is_admin FROM users WHERE user_id = ? LIMIT 1',
      [userId]
    );

    // If no user is found, send a 404 error
    if (rows.length === 0) {
      return res.status(404).json({ status: 'error', error: 'User not found' });
    }


```

```
  // Flip the current admin value: 1 → 0 or 0 → 1
    const newAdminStatus = rows[0].is_admin ? 0 : 1;

    // Update the user's admin status in the database
    await pool.query(
      'UPDATE users SET is_admin = ? WHERE user_id = ?',
      [newAdminStatus, userId]
    );

    // Respond with a success message and the new status
    res.json({
      status: 'ok',
      message: `User ${newAdminStatus ? 'granted' : 'revoked'} admin
status`,
      is_admin: !!newAdminStatus
    });
  } catch (err) {
    // Log and return any unexpected errors
    console.error('POST /admin/users/:id/toggle-admin error:', err);
    res.status(500).json({ status: 'error', error: err.message });
  }
});

```

extracted from routes/admin.js

CRUD Functionality

unmute

Example: Deleting a comment (Delete)

```
// DELETE route: remove a comment (allowed for the owner or an admin)
router.delete('/:postId/comments/:commentId', async (req, res) => {
  try {
    // Extract the post and comment IDs from the URL
    const { postId, commentId } = req.params;

    // Decode the JWT token to identify the user and their role
    const payload = getPayloadFromReq(req);
    const userId = payload && (payload.id || payload.user_id || payload.userId)
      ? (payload.id || payload.user_id || payload.userId)
      : null;
    const isAdmin = payload && (payload.is_admin === 1 || payload.is_admin === true);

    // Block unauthorised users (not logged in or no admin rights)
    if (!userId && !isAdmin)
      return res.status(401).json({ status: 'error', error: 'Unauthorized' });

    // Fetch the comment to confirm it exists and who owns it
    const [rows] = await pool.query(
      'SELECT comment_id, user_id FROM comments WHERE comment_id = ? AND post_id = ? LIMIT 1',
      [commentId, postId]
    );
    if (rows.length === 0)
      return res.status(404).json({ status: 'error', error: 'Comment not found' });
```

```
    // Check permission: only the author or an admin can delete it
    const comment = rows[0];
    if (!isAdmin && comment.user_id !== Number(userId)) {
      return res.status(403).json({ status: 'error', error: 'Forbidden' });
    }

    // Delete the comment from the database
    await pool.query(
      'DELETE FROM comments WHERE comment_id = ? AND post_id = ?',
      [commentId, postId]
    );

    // Send success response
    res.json({ status: 'ok', message: 'Comment deleted' });
  } catch (err) {
    // Handle unexpected errors
    console.error('DELETE /posts/:postId/comments/:commentId error:', err);
    res.status(500).json({ status: 'error', error: err.message });
  }
});
```

extracted from routes/posts.js

Secure Authentication & Role Management

unmute

User Authentication:

Users log in via email or username + password

- **Password Security:** Passwords hashed with bcrypt
- **Token-based Auth:** JWT tokens include user ID and is_admin flag
- **Role Management:** Only admins can access certain routes (like /toggle-admin)
- **Access Control:** Anonymous posting vs admin moderation

```
// Middleware to protect routes and verify JWT
function authenticateToken(req, res, next) {
  // Extract token from Authorization header
  const token = req.headers.authorization?.split(' ')[1];

  // Block if no token provided
  if (!token) return res.sendStatus(401);

  // Verify the token using secret
  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403); // Forbidden if invalid
    req.user = user; // Attach decoded payload to request
    next(); // Continue to protected route
  });
}

// Example usage in a route
router.get('/protected', authenticateToken, (req, res) => {
  res.json({ message: `Hello user ${req.user.id}, admin: ${req.user.is_admin}` });
});
```

extracted and simplified from routes/auth.js

Backend Integration & API Architecture

unmute

Built using Node.js and Express.js

- Connected to MySQL database using a shared connection pool
- Implements RESTful API endpoints for posts, comments, journals, and resources etc.
- Uses prepared statements to prevent SQL injection
- JWT authentication middleware protects private routes
- Integrates seamlessly with React via Axios

unmute

SEO Optimisation

SEO Overview

unmute

Goal:
Improve visibility, discoverability, and user engagement.

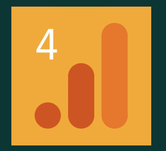
Focus Areas:
- Technical SEO (robots.txt, sitemap.xml)
- On-page SEO (meta tags, titles, descriptions)
- Social Sharing (Open Graph, Twitter Cards)
- Structured Data (JSON-LD schema)
- Performance Optimisations (fast loading, Core Web Vitals)
- Analytics & Accessibility (GA4, semantic HTML)

Technical

On Page

Social

Structured Data



Analytics

Technical SEO Foundation

unmute

robots.txt

- Allows search engines to crawl public pages
- Blocks private/admin pages
- References sitemap location

Impact:

Guides search engines on what to index

sitemap.xml

- Lists all public pages with priority & update frequency
- Example priorities: Home (1.0), Explore (0.9), Resources (0.8), Signup (0.7), Login (0.6)

Impact:

Helps search engines discover and index pages efficiently

```
# robots.txt for unMute
# Allow all search engines to crawl the site

User-agent: *
Allow: /

# Disallow admin and private pages
Disallow: /admin
Disallow: /account
Disallow: /addpost
Disallow: /addentry
Disallow: /addresource

# Sitemap location
Sitemap: https://www.unmute.digital/sitemap.xml
```

robots.txt

On-Page SEO & Social Sharing

unmute

Primary Meta Tags Includes:

- Optimised title: “unMute - Breaking the Silence on Men’s Mental Health”
- Compelling description (160 characters)
- Targeted keywords for men’s mental health
- Language declaration etc.

Open Graph Tags (Facebook, LinkedIn):

- og:type, og:url, og:title, og:description, og:image, og:site_name, og:locale
- **Impact:** Professional and appealing link previews

Twitter Card Tags:

- Large image card format with optimised title, description, and image
- Impact: Improved appearance when shared on Twitter/X

```
1   <meta property="og:type" content="website" />
2       <meta property="og:url" content="https://www.unmute.digital/" />
3       <meta property="og:title" content="unMute - Breaking the Silence on Men's Mental Health" />
4       <meta property="og:description" content="A safe, judgment-free platform where men can share their stories, connect anonymously, and access mental health resources." />
```

index.html

Structured Data

unmute

WebSite Schema:

- Defines the site for search engines
- Includes SearchAction markup for potential Google search box
- Impact: May enable sitelinks searchbox in Google results

Organisation Schema:

- Establishes unMute as an organisation
- Includes logo and description for Knowledge Graph
- Impact: Helps Google understand the brand and display richer search results

```
<!-- Structured Data (JSON-LD) -->

    <script type="application/ld+json">
    {
      "@context": "https://schema.org",
      "@type": "Organization",
      "name": "unMute",
      "url": "https://www.unmute.digital",
      "logo": "https://www.unmute.digital/unmute.png",
      "description": "Breaking the silence on men's mental health through anonymous support and community connection.",
      "sameAs": []
    }
    </script>
```

index.html

SEO Highlights & Impact

unmute

The unMute app features a professional-grade SEO setup covering all essential areas. The technical foundation, including robots.txt, sitemap.xml, and canonical URLs, guides search engines on what to index. Each page is optimised with meta tags, titles, descriptions, and targeted keywords, while Open Graph and Twitter Cards ensure polished social previews. Structured data using WebSite and Organization JSON-LD enhances brand recognition and may enable rich search results.

Performance optimisations, such as preconnect, DNS prefetch, and async script loading, improve Core Web Vitals, while semantic HTML, alt text, and ARIA labels boost accessibility and crawlability. A dynamic SEO component updates metadata for each page, and GA4 analytics provides data-driven insights to refine content strategy. Together, these measures improve search rankings, enable faster indexing, increase user engagement, and deliver a high-quality user experience recognised by search engines.

unmute

Hosting

Hosting

unmute

Frontend + Backend

Hosted on Heroku since I found it the easiest for a student project compared to other hosting providers.

Database

MySQL hosted on Railway, since my first option **Google Cloud SQL** didn't have as generous free usage limits.

SEO Optimisation

Optimised with Meta tags, sitemap, structured data, GA4 (as mentioned earlier)

Performance Tweaks

I used Async scripts along with DNS Prefetch and Preconnect to speed up loading for external resources like Google Fonts, Cloudinary, and analytics scripts improving my Core Web Vitals and SEO ranking signals.

unmute

Live Demo

unmute

Analytics

Analytics Overview

unmute

Active users	New users	Average engagement time p...	Event count
55	58	3m 20s	1.6K

Top pages/screens

PAGE TITLE AND SCREEN CLASS	VIEWS	ACTIVE USERS	EVENT COUNT	BOUNCE RATE
unMute	965	37	1.4K	40.7%
unMute - Breaking the Silence on Men'...	73	21	170	57.1%
Explore Stories & Community | unMute	48	9	51	30.0%
About Us - Our Mission | unMute	13	3	13	33.3%

Analytics Overview

unmute

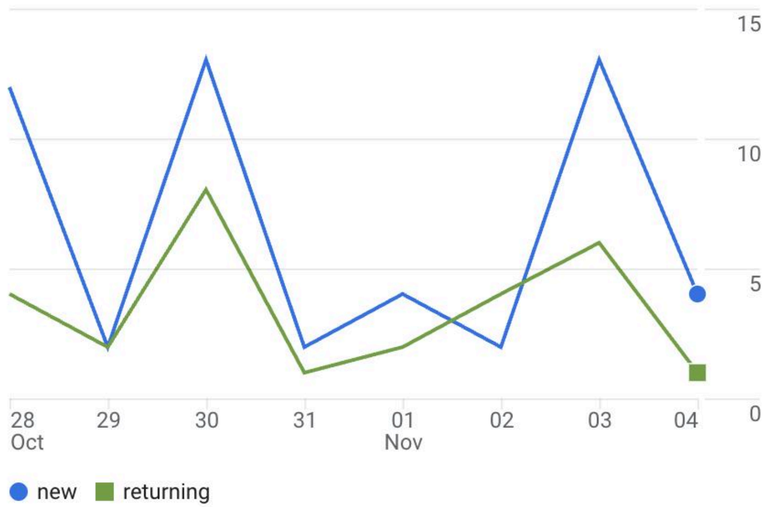
Active users by First user source / medium

FIRST USER SOURCE / ME…	ACTIVE USERS
(direct) / (none)	33
google / organic	6
facebook.com / referral	4
l.instagram.com / referral	4
m.facebook.com / referral	4

Sessions by Session source/medium

SESSION SOURCE/MEDIUM	SESSIONS
(direct) / (none)	65
google / organic	31
l.instagram.com / referral	7
ig / paid	5
facebook.com / referral	4
m.facebook.com / referral	4

New vs. Returning users



unmute

Reflection

Reflection & Conclusion

unmute

One of the challenges I had was hosting my MySQL database on Google Cloud. I started there because of the free $300 credit, but the credits ran out quickly and wouldn't even last a month at the pace I was using it. Because of that, I decided to migrate my database to Railway, which turned out to be a much better option for my project. It's lighter, faster to deploy, and the setup is far simpler compared to Google Cloud's more complex configuration process. Railway also integrates seamlessly with Heroku, which made connecting my backend and database much smoother.

Overall, I really enjoyed working with MySQL throughout this project. Once it was set up properly, it was stable, reliable, and easy to scale as my app grew. It gave me a solid understanding of how relational databases power real applications and how data connects across users, posts, and journals.

In conclusion, this project taught me how to bring together every layer of a full-stack application, from frontend design and backend logic to deployment, database management, and optimisation. unMute was more than just a technical project for me, it allowed me to create a space that represents the importance of open conversations around men's mental health in a fun and engaging way.