

Rabt | Organization Contact Directory

Developed by Kraitto

Version 1.0.0

December 1, 2024

Welcome

Thank you for purchasing Rabt, our Next.js-based contact directory app. This comprehensive documentation covers the system overview, features, and detailed installation procedures.

Support Channels

For any inquiries beyond the scope of this documentation, we're here to help:

- Technical Support: **support@kraitto.com**
- Website: **www.kraitto.com**
- Contact Form: Available on our website
- Social Media: Follow us for updates and support
- Tutorials: <https://github.com/kraittoofficial/rabt-previews/>

Package Contents

Core Package (rabt-v1.0.0.zip)

- .env.example - Environment configuration template
- Dockerfile - Container configuration file
- docker-compose.yml - Docker orchestration configuration
- Ecosystem.config.js - PM2 process management settings
- LICENSE.txt - Software license agreement

Documentation

- Rabt v1.0.0 Documentation.pdf - Comprehensive user and deployment guide
- Preview Images - Visual reference materials

Additional Support

For general inquiries and discussions:

- Visit our Envato Market "Item Discussion" section
- Review our documentation
- Contact our support team

We're committed to providing excellent support for your Rabt implementation. While we can't guarantee immediate solutions to all issues, we strive to assist you promptly and effectively.

Mohammad Saiful Islam

Founder & Partner

Kraitto

Table of Contents

# Rabt Introduction.....	4
# Core Value Proposition.....	4
# Business Applications.....	4
1. Corporate Contacts Directory.....	4
2. Management, Accounts, Sales & Customer Service.....	4
3. Professional Networking.....	4
# Feature Architecture.....	4
1. Contact Directory Management.....	4
Contact Information Structure.....	4
Interactive Features.....	4
Search and Discovery.....	5
2. Security and Access Management.....	5
Authentication System.....	5
Role-based Access Control (RBAC).....	5
3. Advanced Features.....	5
Contact Operations.....	5
Organizational Structure.....	6
Data Management.....	6
4. User Experience.....	6
# Technical Implementation.....	6
Technology Stack.....	6
# System Requirements.....	7
Development Environment.....	7
Docker Environment.....	7
Production Environment.....	7
# Getting Started.....	7
# Support.....	7
# License.....	7
# Initial Setup.....	8
1. System Requirements Verification.....	8
2. MongoDB Setup.....	8
3. Email Server Configuration.....	8
4. OAuth Configuration.....	8
# Customization.....	9
1. Customizing Your Installation.....	9
Logo Customization.....	9
Splash Screen Customization.....	9
# Deployment Methods.....	10
Method 1: Vercel Deployment.....	10
Pre-deployment Setup.....	10
Deployment Steps.....	10
Method 2: Docker Deployment.....	12
Pre-deployment Setup.....	12
Configuration Files.....	12
Deployment Steps.....	14
Method 3: PM2 Deployment.....	16
Installation & Setup.....	16
Configuration.....	17
Deployment Steps.....	18
# Super Admin Setup.....	19
Method 1: Using MongoDB Compass.....	19
Method 2: Development Server with OAuth.....	19
Method 3: Using MongoDB Shell.....	19
Security Notes.....	20
# Troubleshooting Guide.....	20
Database Issues.....	20
Build Failures.....	20

Runtime Errors..... 20

Performance Issues..... 21

Security Checklist..... 21

Rabt Overview and Features

Rabt Introduction

Rabt is a contact management system designed for enterprise-scale organizations. It provides a secure, efficient, user-friendly platform for managing professional contacts, organizational directories, and personal networks. The solution emphasizes role-based access control, advanced contact management features, and seamless integration capabilities.

Core Value Proposition

- Centralized contact management with access controlling
- Role-based access control for organizational hierarchy
- Seamless contact sharing and synchronization
- Advanced search and filtering capabilities
- Multi-platform accessibility and responsive design

Business Applications

1. Corporate Contacts Directory
 - Employee contact management
 - Department organization
 - New hire onboarding
2. Management, Accounts, Sales & Customer Service
 - Client database management
 - Quick contact access
 - Customer relationship management
3. Professional Networking
 - Personal contact organization
 - Business card digitization
 - Network relationship mapping

Feature Architecture

1. Contact Directory Management

Contact Information Structure

- Personal Details
 - Name and designation
 - Departmental affiliation
 - Organizational mapping
 - Multiple contact points (2× phone, WhatsApp, email)
 - Office location and address

Interactive Features

- Smart Action Buttons
 - Direct call integration through default dialer

- WhatsApp messaging through redirecting to WhatsApp
- Email client integration through default mail app
- Contact sharing suite
 - Digital business card (vCard)
 - QR code generation
 - Clipboard integration

Search and Discovery

- Advanced Search Engine
 - Full-text search across all fields
 - Real-time results
 - Smart filtering system
- Multiple Filtering
 - Organization-based filtering
 - Department categorization
 - Designation hierarchy
 - Quick reset functionality

2. Security and Access Management

Authentication System

- Multi-channel Authentication
 - Email with secure passcode
 - Google OAuth integration
 - Facebook OAuth integration

Role-based Access Control (RBAC)

- Hierarchical User Roles
 - User (Base access)
 - Staff (Extended access)
 - Admin (Management access)
 - Super Admin (Complete control)
- Permission Matrix

Feature	User	Staff	Admin	Super Admin
Contact Directory	-	✓	✓	✓
Profile Management	✓	✓	✓	✓
Contact Management	-	-	✓	✓
User Management	-	-	✓	✓
System Logs	-	-	-	✓

3. Advanced Features

Contact Operations

- Bulk Management
 - Mass import capability
 - Batch export functionality
 - Multi-select delete operations
- Data Tracking

- Contact history logging
- Change tracking
- Audit trails

Organizational Structure

- Hierarchy Management
 - Group structure definition
 - Department mapping
 - Relationship visualization

Data Management

- Import/Export Suite
 - Excel compatibility
 - vCard standard support
 - Backup system
- Notification System
 - Action-based alerts
 - Email notifications
 - Role change communications

4. User Experience

- Responsive Design
 - Mobile-first approach
 - Cross-device compatibility
- Accessibility
 - Dark/Light mode support
 - Screen reader compatibility

Technical Implementation

Technology Stack

- Front-end & Back-end:
 - Next.js 15 (App Router)
 - React 19
 - TypeScript 5
- Styling:
 - Tailwind CSS
 - Shadcn/ui
- Database:
 - MongoDB Atlas
- Authentication:
 - NextAuth.js 4

System Requirements

Development Environment

- Node.js ($\geq 20.x$)
- NPM ($\geq 10.x$)
- Git

Docker Environment

- Docker Engine ($\geq 24.0.0$)
- Docker Compose ($\geq 2.20.0$)
- Minimum 4GB RAM (recommended 8GB)
- 10GB free disk space

Production Environment

- Minimum 4GB RAM
- Minimum 20GB storage
- Modern web browser support
- SSL/TLS certificate
- Network Requirements:
 - Outbound access for:
 - SMTP server (email notifications)
 - MongoDB connection
 - OAuth providers (Google, Facebook)

Getting Started

For detailed deployment instructions, please refer to our [Installation Guide](#).

For tutorials, previews, and additional information, visit our [GitHub Preview Repository](#).

Support

For technical support, inquiries, or customization requests, please contact us at support@kraitto.com.

License

This software is released under a Commercial License. For licensing details, please refer to the [LICENSE.txt](#) file in the root directory.

For licensing inquiries, customization requests, or support, please contact us directly.

Rabt Deployment Guide

Initial Setup

1. System Requirements Verification

1. Check Node.js version:
node --version # *Must be v20 or later*
2. Verify package manager installation:
npm --version # *or*
pnpm --version

2. MongoDB Setup

1. Choose deployment option:
 - Option A: Local MongoDB
 1. Install MongoDB Community Edition
 2. Start MongoDB service
 3. Create database and user
 4. Get connection string in format:
mongodb://username:password@localhost:27017/database
 - Option B: MongoDB Atlas
 1. Create account at mongodb.com
 2. Create new cluster (M2 recommended for <100 users)
 3. Set up database access (username/password)
 4. Whitelist IP addresses
 5. Get connection string from Atlas dashboard

3. Email Server Configuration

1. Gmail Setup (Recommended for starting):
 1. Enable 2-Factor Authentication
 2. Generate App Password
 3. Use following settings:
 - SMTP_HOST=smtp.gmail.com
 - SMTP_PORT=587
 - SMTP_USER=your-gmail@gmail.com
 - SMTP_PASSWORD=your-app-password

4. OAuth Configuration

1. Google OAuth:
 1. Go to Google Cloud Console
 2. Create new project
 3. Enable OAuth 2.0
 4. Configure OAuth consent screen
 5. Create credentials
 6. Save Client ID and Secret

2. Facebook OAuth:
 1. Go to Facebook Developers
 2. Create new app
 3. Add Facebook Login
 4. Configure OAuth settings
 5. Save App ID and Secret

Customization

1. Customizing Your Installation

Logo Customization

1. Logo Requirements:
 - Container size: 128px × 32px (w-32 h-8)
 - Aspect ratio: 4:1 (recommended)
 - Minimum resolution: 256px × 64px (2x for retina displays)
 - Optimal resolution: 512px × 128px (4x)
 - Format: Provide both light and dark versions for theme support
2. Logo Placement:
 - Place your logo files in the public/logo directory
 - Recommended filenames:
 - logo-light.png # for light theme
 - logo-dark.png # for dark theme
3. Update Header Component:
 - Navigate to components/theme/Header.tsx
 - Update the Image src prop with your logo paths
 - The Next.js Image component with fill and object-contain will automatically handle scaling

Splash Screen Customization

1. To change the loading screen title:
 - Open components/theme/SplashScreen.tsx
 - Locate the h1 element with “Rabt Contact Directory”
 - Replace with your desired application name

Deployment Methods

Method 1: Vercel Deployment

Pre-deployment Setup

1. Install Vercel CLI:
`npm i -g vercel`
2. Create .env file with all required variables
3. Verify project structure follows Vercel requirements

```

1 PS K:\kraitto\rabt-v1.0.0> npm install -g vercel
2
3 added 243 packages in 29s
4
5 18 packages are looking for funding
6   run `npm fund` for details
7 PS K:\kraitto\rabt-v1.0.0>

```

Deployment Steps

1. Login to Vercel:
`vercel login`

```

1 PS K:\kraitto\rabt-v1.0.0> vercel login
2 Vercel CLI 39.1.2
3 > NOTE: The Vercel CLI now collects telemetry regarding usage of the CLI.
4 > This information is used to shape the CLI roadmap and prioritize features.
5 > You can learn more, including how to opt-out if you'd not like to participate in this program, by visiting the following URL:
6 > https://vercel.com/docs/cli/about-telemetry
7 ? Log in to Vercel Continue with Email
8 ? Enter your email address: rabt@kraitto.com
9 We sent an email to rabt@kraitto.com. Please follow the steps provided inside it and make sure the security code matches .... ....
10 > Success! Email authentication complete for rabt@kraitto.com
11 Congratulations! You are now logged in. In order to deploy something, run `vercel`.
12 ? Connect your Git Repositories to deploy every branch push automatically (https://vercel.link/git).
13 PS K:\kraitto\rabt-v1.0.0>

```

2. Initial deployment:
`vercel`

```

1 PS K:\kraitto\rabt-v1.0.0> vercel
2 Vercel CLI 39.1.2
3 ? Set up and deploy "K:\kraitto\rabt-v1.0.0"? yes
4 ? Which scope should contain your project? Kraitto
5 ? Link to existing project? no
6 ? What's your project's name? rabt-v1-0-0
7 ? In which directory is your code located? ./
8 Local settings detected in vercel.json:
9 Auto-detected Project Settings (Next.js):
10 - Build Command: next build
11 - Development Command: next dev --port $PORT
12 - Install Command: `yarn install`, `pnpm install`, `npm install`, or `bun install`
13 - Output Directory: Next.js default
14 ? Want to modify these settings? no
15 ? Linked to kraitto/rabt-v1-0-0 (created .vercel and added it to .gitignore)
16 ? Inspect: https://vercel.com/kraitto/rabt-v1-0-0/... [7s]
17 ? Production: https://rabt-v1-0-0-kraitto.vercel.app [7s]
18 ? Deployed to production. Run `vercel --prod` to overwrite later (https://vercel.link/2F).
19 ? To change the domain or build command, go to https://vercel.com/kraitto/rabt-v1-0-0/settings
20 PS K:\kraitto\rabt-v1.0.0>

```

3. Configure environment variables:

1. Open Vercel dashboard
 2. Go to Project Settings
 3. Navigate to Environment Variables
 4. Import .env file or add manually
 5. Verify all required variables are set
4. Production deployment:
- `vercel --prod`



```
1 PS K:\kraitto\rabt-v1.0.0> vercel --prod
2 Vercel CLI 39.1.2
3 🔗 Inspect: https://vercel.com/kraitto/rabt-v1-0-0/... [4s]
4 ✅ Production: https://rabt-v1-0-0-kraitto.vercel.app [4s]
5 PS K:\kraitto\rabt-v1.0.0>
```

Method 2: Docker Deployment

Pre-deployment Setup

1. Install Docker:

Ubuntu

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

```
sudo sh get-docker.sh
```

Verify installation

```
docker --version
```

Configuration Files

1. Create docker-compose.yml:

```
version: "3.8"
```

```
services:
```

```
  app:
```

```
    image: ghcr.io/krait0/rabt:latest
```

```
    build:
```

```
      context: .
```

```
      dockerfile: Dockerfile
```

```
    ports:
```

```
      - "3000:3000"
```

```
    environment:
```

```
      # Database
```

```
      MONGODB_URI: "mongodb://username:password@localhost:27017/database"
```

```
      DB_ADMIN_EMAIL: "admin@example.com"
```

```
      # Email Configuration
```

```
      SMTP_HOST: "smtp.example.com"
```

```
      SMTP_PORT: "587"
```

```
      SMTP_USER: "your-smtp-user"
```

```
      SMTP_PASSWORD: "your-smtp-password"
```

```
      SMTP_FROM: "noreply@example.com"
```

```
      # NextAuth Configuration
```

```
      NEXTAUTH_URL: "http://localhost:3000"
```

```
      NEXTAUTH_SECRET: "your-nextauth-secret"
```

```
      # Google OAuth
```

```
      GOOGLE_CLIENT_ID: ""
```

```
      GOOGLE_CLIENT_SECRET: ""
```

```
      # Facebook OAuth
```

```
      FACEBOOK_CLIENT_ID: ""
```

```
FACEBOOK_CLIENT_SECRET: ""
```

```
# Application
```

```
NODE_ENV: "production"
```

```
PORT: "3000"
```

```
# Security
```

```
CORS_ORIGINS: "https://yourdomain.com,https://api.yourdomain.com"
```

```
restart: unless-stopped
```

```
networks:
```

```
- rabt-network
```

```
volumes:
```

```
- rabt-uploads:/app/public/uploads
```

```
- rabt-data:/app/data
```

```
- rabt-cache:/app/.next/cache
```

```
networks:
```

```
  rabt-network:
```

```
    driver: bridge
```

```
volumes:
```

```
  rabt-uploads:
```

```
  rabt-data:
```

```
  rabt-cache:
```

```
2. Create Dockerfile:
```

```
# Stage 1: Dependencies
```

```
FROM node:20-alpine AS deps
```

```
LABEL org.opencontainers.image.source="https://github.com/kraitio/rabt"
```

```
LABEL org.opencontainers.image.description="Rabt Organization Contact Directory  
Developed by Kraitio"
```

```
LABEL org.opencontainers.image.licenses="Commercial License"
```

```
RUN apk add --no-cache libc6-compat
```

```
WORKDIR /app
```

```
COPY package.json package-lock.json* ./
```

```
COPY .env .env.local* ./
```

```
RUN npm ci --force
```

```
# Stage 2: Builder
```

```
FROM node:20-alpine AS builder
```

```
WORKDIR /app
```

ENV NEXT_TELEMETRY_DISABLED=1

ENV NODE_ENV=production

ENV SKIP_ENV_VALIDATION=1

ENV SKIP_SMTP_VERIFY=true

COPY --from=deps /app/node_modules ./node_modules

COPY --from=deps /app/.env* ./

COPY . .

RUN node --max_old_space_size=4096 node_modules/.bin/next build

Stage 3: Runner

FROM node:20-alpine **AS** runner

WORKDIR /app

ENV NODE_ENV=production

ENV NEXT_TELEMETRY_DISABLED=1

RUN addgroup --system --gid 1001 nodejs

RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public

COPY --from=builder /app/.next/standalone ./

COPY --from=builder /app/.next/static ./next/static

COPY --from=builder /app/.env* ./

RUN mkdir -p .next/cache

RUN chown -R nextjs:nodejs .next

RUN chmod -R 755 .next

USER nextjs

EXPOSE 3000

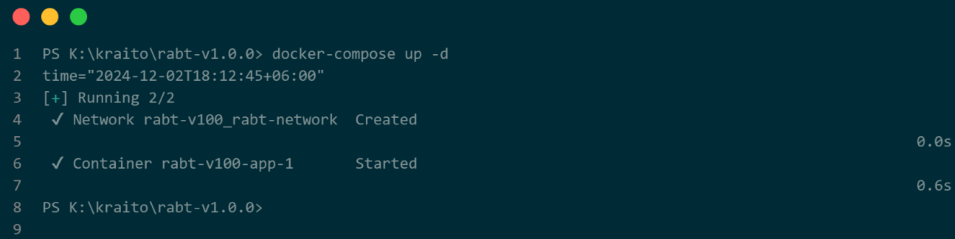
ENV PORT=3000

ENV HOSTNAME="0.0.0.0"

CMD ["node", "server.js"]

Deployment Steps

1. Start the containers:
docker-compose up -d



```
1 PS K:\kraito\rabt-v1.0.0> docker-compose up -d
2 time="2024-12-02T18:12:45+06:00"
3 [+] Running 2/2
4   ✓ Network rabt-v100_rabt-network    Created
5                                     0.0s
6   ✓ Container rabt-v100-app-1        Started
7                                     0.6s
8 PS K:\kraito\rabt-v1.0.0>
```

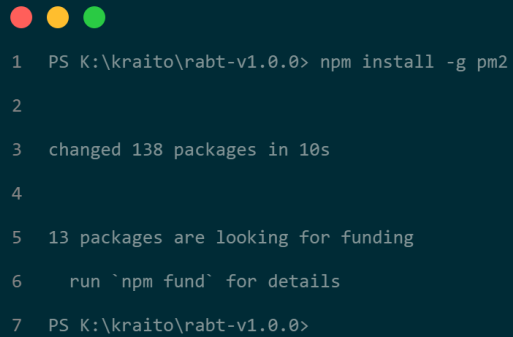
2. Verify deployment:
docker-compose ps
docker-compose logs

Method 3: PM2 Deployment

Installation & Setup

1. Install PM2:

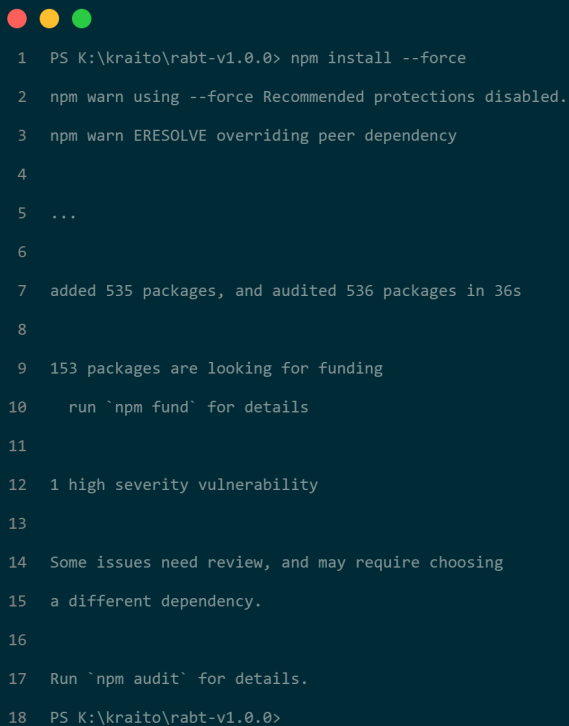
`npm install -g pm2`



```
1 PS K:\krait0\rabt-v1.0.0> npm install -g pm2
2
3 changed 138 packages in 10s
4
5 13 packages are looking for funding
6   run `npm fund` for details
7 PS K:\krait0\rabt-v1.0.0>
```

2. Install dependencies:

`npm install --force`



```
1 PS K:\krait0\rabt-v1.0.0> npm install --force
2 npm warn using --force Recommended protections disabled.
3 npm warn ERESOLVE overriding peer dependency
4
5 ...
6
7 added 535 packages, and audited 536 packages in 36s
8
9 153 packages are looking for funding
10   run `npm fund` for details
11
12 1 high severity vulnerability
13
14 Some issues need review, and may require choosing
15 a different dependency.
16
17 Run `npm audit` for details.
18 PS K:\krait0\rabt-v1.0.0>
```

3. Build application:

`npm run build`


```

1 PS K:\kraitto\rabt-v1.0.0> npm run build
2
3 > rabt@1.0.0 build
4 > next build
5
6   ▲ Next.js 15.0.3
7   - Environments: .env
8
9   Creating an optimized production build ...
10  ✓ Compiled successfully
11  ✓ Linting and checking validity of types
12  ✓ Collecting page data
13  ✓ Generating static pages (23/23)
14  ✓ Collecting build traces
15  ✓ Finalizing page optimization
16
17  ...
18
19  o (Static)   prerendered as static content
20  f (Dynamic) server-rendered on demand
21
22 PS K:\kraitto\rabt-v1.0.0>

```

Configuration

Create ecosystem.config.js:

```

module.exports = {
  apps: [
    {
      name: "rabt",
      script: "./node_modules/next/dist/bin/next",
      args: "start",
      //instances: 'max',
      exec_mode: "cluster",
      autorestart: true,
      watch: false,
      max_memory_restart: "1G",
      env_production: {
        NODE_ENV: "production",
        PORT: 3000,
      },
      env_development: {
        NODE_ENV: "development",
        PORT: 3000,
      },
    },
  ],
};

```

Deployment Steps

1. Start application:

`pm2 start ecosystem.config.js --env production`

```

1 PS K:\kraitto\rabt-v1.0.0> pm2 start ecosystem.config.js --env production
2 [PM2][WARN] Applications rabt not running, starting...
3 [PM2] App [rabt] launched (1 instances)
4
5  id | name | namespace | version | mode | pid | uptime | 0 | status | cpu | mem | user | watching |
6  ---|---|---|---|---|---|---|---|---|---|---|---|---|
7  0 | rabt | default | 15.0.3 | cluster | 4440 | 0s | 0 | online | 0% | 42.6mb | nasim | disabled |
8
9
10 PS K:\kraitto\rabt-v1.0.0>

```

2. Setup auto-restart:

`pm2 startup`

`pm2 save`

```

1 PS K:\kraitto\rabt-v1.0.0> pm2 save
2 [PM2] Saving current process list...
3 [PM2] Successfully saved in C:\Users\inasi\.pm2\dump.pm2
4 PS K:\kraitto\rabt-v1.0.0>

```

3. Monitor application:

`pm2 monit`

`pm2 logs rabt`

Super Admin Setup

Method 1: Using MongoDB Compass

1. Open MongoDB Compass and connect to your database
2. Navigate to the users collection
3. Find the user you want to make Super Admin
4. Click Edit Document
5. Change the role field value from "User" to "Super Admin"
6. Click Save

Example document modification:

```
{
  "_id": ObjectId("..."),
  "name": "Admin User",
  "email": "admin@example.com",
  "role": "Super Admin", // Change this field
  "isVerified": true,
  // ... other fields
}
```

Method 2: Development Server with OAuth

1. Stop the production server if running
2. Modify the OAuth sign-in code in route.ts temporarily:

```
const newUser = await User.createWithoutValidation({
  name: user.name || profile?.name || "User",
  email: user.email,
  role: "Super Admin", // Change this from "User" to "Super Admin"
  isVerified: true,
  oauth: {
    [account.provider]: {
      providerAccountId: account.providerAccountId,
      accessToken: account.access_token,
      refreshToken: account.refresh_token,
    },
  },
});
```

3. Start the development server:

npm run dev

4. Sign in using Google or Facebook OAuth
5. Verify in MongoDB that the user was created with Super Admin role
6. Revert the code change back to "User" role
7. Restart the production server

Method 3: Using MongoDB Shell

1. Connect to your MongoDB instance:

```
mongosh "your-connection-string"
```

2. Run the following command:

```
db.users.updateOne(
  { email: "admin@example.com" },
  { $set: { role: "Super Admin" } }
);
```

Security Notes

- Create only one Super Admin account initially
- Document the Super Admin email for organizational records
- Change the role back to "User" in the code after creating the Super Admin
- Consider implementing an audit log for role changes
- Regularly review user roles and permissions

Troubleshooting Guide

Database Issues

1. Connection failures:

```
# Test MongoDB connection
```

```
mongosh "your-connection-string"
```

- Check IP whitelist
- Verify credentials
- Test network connectivity

Build Failures

1. Clear build cache:

```
rm -rf .next
```

```
npm run build
```

2. Check dependency conflicts:

```
npm ls
```

Runtime Errors

1. Check logs based on deployment method:

```
# Vercel
```

```
vercel logs
```

```
# Docker
```

```
docker logs rabt
```

```
# PM2
```

```
pm2 logs rabt
```

2. Common fixes:

- Restart application
- Verify environment variables
- Check system resources
- Monitor memory usage

Performance Issues

1. Monitor metrics:

PM2

pm2 monit

Docker

docker stats rabt

2. Optimization steps:
 - Scale instances
 - Adjust memory limits
 - Enable caching
 - Optimize database queries

Security Checklist

- ☐ Environment variables properly set
- ☐ CORS origins configured
- ☐ OAuth credentials secured
- ☐ Database access restricted
- ☐ SMTP credentials protected
- ☐ SSL/TLS enabled
- ☐ Regular security updates
- ☐ Proper error handling