TrustMarket P2P Marketplace - Free Deployment Guide

Complete Step-by-Step Instructions for Publishing Your Application for Free

---

Table of Contents

---

1. Introduction

This comprehensive guide provides detailed, step-by-step instructions for deploying the TrustMarket P2P Marketplace application completely free using industry-leading platforms with generous free tiers. TrustMarket is a full-stack MERN (MongoDB, Express, React, Node.js) application that requires hosting for three main components: the React frontend, the Node.js backend API, and the MongoDB database.

The deployment strategy recommended in this guide utilizes a carefully selected combination of platforms that offer excellent free tiers suitable for development, testing, and even small-scale production deployments:

- **Vercel** for React frontend hosting (100GB bandwidth/month, global CDN)
- **Render** for backend API hosting (750 hours compute/month, automatic deployments)

- **MongoDB Atlas** for database hosting (512MB storage, replica set deployment)
- **Cloudinary** for media storage and optimization (25GB storage, 2GB bandwidth/month)

This configuration provides professional-grade infrastructure at zero cost while maintaining excellent performance, reliability, and scalability potential as your application grows.

---

2. Prerequisites

Before beginning the deployment process, ensure you have the following accounts and tools prepared:

2.1 Required Accounts

Create accounts on the following platforms before starting deployment. All platforms offer free tiers with no credit card required:

Platform	Purpose	Signup URL
GitHub	Source code hosting	github.com
Vercel	Frontend hosting	vercel.com
Render	Backend hosting	render.com
MongoDB Atlas	Database hosting	mongodb.com/atlas
Cloudinary	Media storage	cloudinary.com

2.2 Required Tools

Ensure the following tools are installed and configured on your local machine:

- **Node.js** (version 18 or higher) - Download from nodejs.org
- **Git** - Download from git-scm.com
- **npm** or **yarn** - Comes bundled with Node.js
- **GitHub CLI** (optional but recommended) - Install via `npm i -g gh`

2.3 Code Preparation

Before deployment, ensure your code is properly configured for the deployment environment:

1. Create environment variable files in both client and server directories:
   - `/client/.env.example` - Frontend environment template
   - `/server/.env.example` - Backend environment template

2. Ensure your package.json files have proper scripts:
   - Frontend: `"build": "vite build"` or `"build": "react-scripts build"`
   - Backend: `"start": "node server.js"` or `"start": "nodemon server.js"`
3. Verify your server.js or index.js exports the app properly for platform detection.

---

3. Uploading to GitHub

GitHub serves as the single source of truth for your codebase and enables automatic deployments across all platforms. Proper repository setup ensures smooth continuous deployment workflows.

3.1 Create GitHub Repository

1. Navigate to github.com and log into your account
2. Click the "+" icon in the top-right corner and select "New repository"
3. Configure your repository settings:
   - **Repository name**: `trustmarket` (or your preferred name)
   - **Description**: "India's Safest P2P Marketplace with Video Verification and Trust Scoring"
   - **Visibility**: Public (required for free Vercel/Render) or Private
   - **Initialize**: Do NOT initialize with README (we'll add one later)
4. Click "Create repository"

3.2 Prepare Local Repository

Open your terminal and navigate to your TrustMarket project directory:

```
# Navigate to project directory
cd /path/to/trustmarket

# Initialize git if not already initialized
git init

# Create .gitignore file if it doesn't exist
cat > .gitignore << 'EOF'
# Dependencies
node_modules/

# Build outputs
dist/
build/

# Environment files
.env
.env.local
.env.*.local

# OS files
.DS_Store
Thumbs.db

# IDE
.vscode/
.idea/

# Logs
logs/
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Coverage
coverage/

# Misc
*.pem
EOF
```

3.3 Commit and Push Code

Execute the following commands to upload your code to GitHub:

```
# Stage all files
git add .

# Create initial commit
git commit -m "Initial commit: TrustMarket P2P Marketplace

Features:
- Full-stack MERN application
- User authentication with JWT
- Listing management with media uploads
- Real-time messaging with Socket.io
- Trust scoring and verification system
- Progressive Web App (PWA) ready"

# Add GitHub repository as remote (replace YOUR_USERNAME with your
actual username)
git remote add origin https://github.com/YOUR_USERNAME/trustmarket.git

# Push to GitHub
git branch -M main
git push -u origin main
```

3.4 Verify Repository

1. Refresh your GitHub repository page
2. Verify all files are uploaded correctly
3. Check that .gitignore is working (node_modules should not appear)
4. Navigate into subdirectories to verify structure is correct

---

4. Frontend Deployment (Vercel)

Vercel provides exceptional free tier hosting specifically optimized for React applications, offering automatic deployments, global CDN distribution, and zero-configuration setup.

4.1 Connect Repository to Vercel

1. Navigate to vercel.com and sign up using your GitHub account

2. Grant Vercel access to your repositories when prompted
3. Click "Add New..." and select "Project"
4. Find and import your `trustmarket` repository from the list

4.2 Configure Build Settings

Vercel automatically detects React applications, but verify these settings:

Setting	Value
Framework Preset	Vite or Create React App (auto-detected)
Build Command	`npm run build`
Output Directory	`dist` (Vite) or `build` (CRA)
Install Command	`npm install`

4.3 Add Environment Variables

Click "Environment Variables" and add the following:

```
# Required for frontend
VITE_API_URL=https://your-backend-name.onrender.com
VITE_CLOUDINARY_CLOUD_NAME=your-cloud-name

# Optional - Analytics
VITE_GA_TRACKING_ID=
```

4.4 Deploy

1. Click "Deploy" to start the initial deployment
2. Watch the build logs for any errors
3. Upon completion, Vercel provides a live URL (e.g., `trustmarket.vercel.app`)

4.5 Verify Deployment

1. Visit your Vercel URL
2. Verify the homepage loads correctly
3. Test basic navigation
4. Open browser developer tools (F12) and check Console for errors

---

5. Backend Deployment (Render)

Render offers the most generous free tier for Node.js web services, providing 750 hours of monthly compute time with automatic deployments from GitHub.

5.1 Connect Repository to Render

1. Navigate to render.com and sign up using your GitHub account
2. Grant Render access to your repositories
3. Click "New+" and select "Web Service"
4. Find and select your `trustmarket` repository

5.2 Configure Build Settings

Configure the following settings in the Render dashboard:

Setting	Value
Name	trustmarket-api (or your preferred name)
Runtime	Node
Build Command	`npm install`
Start Command	`npm start` (or `node server.js`)
Root Directory	`server` (if backend is in separate directory)

5.3 Add Environment Variables

Add the following environment variables in the Render dashboard:

```
# Server Configuration
PORT=5000
NODE_ENV=production

# MongoDB Atlas Connection
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/
trustmarket

# JWT Secrets (use strong, unique values)
JWT_SECRET=your-super-secret-jwt-key-minimum-32-characters-long
REFRESH_TOKEN_SECRET=your-super-secret-refresh-token-minimum-32-
characters

# Cloudinary Configuration
CLOUDINARY_CLOUD_NAME=your-cloud-name
CLOUDINARY_API_KEY=your-api-key
CLOUDINARY_API_SECRET=your-api-secret

# Frontend URL (for CORS)
CLIENT_URL=https://your-frontend-name.vercel.app

# Optional: Email Configuration
SMTP_HOST=
SMTP_PORT=
SMTP_USER=
SMTP_PASS=
```

5.4 Deploy

1. Click "Create Web Service" to start deployment
2. Monitor build logs for any errors
3. Upon completion, Render provides a live URL (e.g., `trustmarket-api.onrender.com`)

5.5 Verify Backend

Test your backend API is working:

```
# Test health endpoint
curl https://your-backend-name.onrender.com/health

# Test API root
curl https://your-backend-name.onrender.com/api/v1
```

---

6. Database Setup (MongoDB Atlas)

MongoDB Atlas provides a robust free tier with replica set deployment, ensuring data durability and automatic failover.

6.1 Create MongoDB Atlas Account

1. Navigate to mongodb.com/atlas and sign up
2. Complete the account setup wizard
3. Verify your email address

6.2 Create Free Cluster

1. Click "Create" to start cluster creation
2. Select "Free" tier (M0)
3. Configure cluster settings:
   - **Cloud Provider**: AWS (recommended for broadest compatibility)
   - **Region**: Select region closest to your users (e.g., Mumbai for India)
   - **Cluster Name**: `trustmarket-cluster`
4. Click "Create Cluster"

6.3 Configure Network Access

By default, MongoDB Atlas blocks all external connections. For development:

1. Click "Network Access" in the left sidebar
2. Click "Add IP Address"
3. Select "Allow Access from Anywhere" (0.0.0.0/0)
4. Click "Confirm"

Security Note: For production, restrict to specific IP ranges or use VPC peering.

6.4 Create Database User

1. Click "Database Access" in the left sidebar

2. Click "Add New Database User"
3. Configure user credentials:
   - **Authentication Method**: Password
   - **Username**: trustmarket_user
   - **Password**: Generate a strong password
   - **Database User Privileges**: Read and Write to any database
4. Click "Add User"

6.5 Get Connection String

1. Click "Database" in the left sidebar
2. Click "Connect" on your cluster
3. Select "Connect your application"
4. Copy the connection string:
   `mongodb+srv://trustmarket_user:<password>@trustmarket-cluster.mongodb.net/trustmarket?retryWrites=true&w=majority`
5. Replace `<password>` with your actual database password
6. Use this string as your `MONGODB_URI` environment variable

6.6 Load Sample Data (Optional)

To test your application with sample data:

1. Click "Clusters" in the left sidebar
2. Click "Load Sample Dataset"
3. Follow the prompts to load sample data
4. This provides test databases for development

---

7. Media Storage (Cloudinary)

Cloudinary provides free image and video optimization, storage, and delivery with generous bandwidth allocations.

7.1 Create Cloudinary Account

1. Navigate to cloudinary.com and sign up
2. Select "Free" plan
3. Complete account setup with your name and email

7.2 Get API Credentials

1. After signup, you'll see your Cloudinary dashboard

2. Locate your "Cloud Name" at the top of the dashboard
3. Click the "Settings" icon (gear icon) in the left sidebar
4. Navigate to "Upload" tab
5. Find your "Upload preset" (or create one named "trustmarket")
6. Note your API credentials:
   - **Cloud Name**: Displayed prominently in dashboard
   - **API Key**: Available in Account Details
   - **API Secret**: Available in Account Details

7.3 Configure Upload Preset

1. Navigate to Settings > Upload
2. Click "Add upload preset"
3. Configure:
   - **Name**: trustmarket
   - **Signing Mode**: Unsigned (for frontend uploads)
   - **Folder**: trustmarket_uploads
   - **Resource Type**: Image and Video
4. Save the preset

7.4 Update Backend Configuration

Ensure your backend uses the Cloudinary credentials:

```
// In your server configuration
cloudinary.config({
  cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
  api_key: process.env.CLOUDINARY_API_KEY,
  api_secret: process.env.CLOUDINARY_API_SECRET
});
```

---

8. Environment Variables Configuration

Proper environment variable configuration ensures secure, configurable deployments across all platforms.

8.1 Backend Environment Variables

Create a `.env` file in your server directory:

```
# Server Configuration
PORT=5000
NODE_ENV=production

# MongoDB Atlas Connection String
# Format: mongodb+srv://username:password@cluster.mongodb.net/database
MONGODB_URI=mongodb+srv://trustmarket_user:YOUR_PASSWORD@trustmarket-
cluster.mongodb.net/trustmarket?retryWrites=true&w=majority

# JWT Configuration
# IMPORTANT: Use strong, unique secrets (minimum 32 characters)
JWT_SECRET=your-super-secret-jwt-key-at-least-32-characters-long
REFRESH_TOKEN_SECRET=your-super-secret-refresh-token-at-least-32-
characters

# Cloudinary Configuration
# Get these from your Cloudinary Dashboard
CLOUDINARY_CLOUD_NAME=your-cloud-name
CLOUDINARY_API_KEY=your-api-key
CLOUDINARY_API_SECRET=your-api-secret

# Frontend URL (for CORS configuration)
CLIENT_URL=https://your-frontend-name.vercel.app

# Email Configuration (Optional - for password reset, notifications)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password

# Socket.io Configuration
SOCKET_CORS_ORIGIN=https://your-frontend-name.vercel.app
```

8.2 Frontend Environment Variables

Create a `.env` file in your client directory:

```
# API Configuration
VITE_API_URL=https://your-backend-name.onrender.com

# Cloudinary Configuration
VITE_CLOUDINARY_CLOUD_NAME=your-cloud-name

# Optional: Analytics
VITE_GA_TRACKING_ID=

# Optional: Feature Flags
VITE_ENABLE_MOCK_DATA=false
```

8.3 Platform-Specific Configuration

Render Backend Configuration:
1. Navigate to your Render dashboard
2. Select your web service
3. Click "Environment Variables"
4. Add each variable from your backend .env file
5. Save and trigger redeploy

Vercel Frontend Configuration:
1. Navigate to your Vercel dashboard
2. Select your project
3. Click "Settings" > "Environment Variables"
4. Add each frontend variable
5. Changes trigger automatic redeploy

---

9. Custom Domain and SSL

Custom domains provide professional branding and improved SEO while all recommended platforms provide automatic SSL certificates.

9.1 Domain Registration

If you don't already own a domain:

1. Choose a domain registrar (Namecheap, GoDaddy, Google Domains)
2. Search for your preferred domain
3. Complete purchase (typically $10-15/year for .com domains)

9.2 Configure Vercel with Custom Domain

1. Navigate to Vercel dashboard > Project > Settings > Domains
2. Enter your domain name
3. Vercel provides DNS records to configure:
   - **CNAME** for subdomains (www, api)
   - **ALIAS/A** for root domain
4. Add records at your domain registrar

DNS Configuration at Registrar:

```
Type: CNAME
Name: www
Value: cname.vercel-dns.com
TTL: Auto

Type: ALIAS
Name: @
Value: your-project.vercel.app
TTL: Auto
```

9.3 Configure Render with Custom Domain

1. Navigate to Render dashboard > Web Service > Settings > Custom Domains
2. Enter your domain (e.g., api.yourdomain.com)
3. Render provides DNS records
4. Add CNAME record at your registrar:

```
Type: CNAME
Name: api
Value: your-backend.onrender.com
TTL: Auto
```

9.4 Verify SSL Certificates

All platforms automatically provision SSL certificates through Let's Encrypt:

1. Wait 24-48 hours for DNS propagation
2. Visit https://yourdomain.com
3. Verify browser shows lock icon
4. Click lock icon to view certificate details

10. Testing Your Deployment

Thorough testing ensures your deployed application functions correctly before announcing to users.

10.1 Frontend Testing

Visual Testing:
- [ ] Homepage loads correctly
- [ ] Images and icons display properly
- [ ] Layout responds to different screen sizes
- [ ] No console errors in browser DevTools

Functional Testing:
- [ ] Navigation links work correctly
- [ ] Search functionality returns results
- [ ] Listing details page loads for existing listings
- [ ] Authentication forms validate input

PWA Testing:
- [ ] Service worker registers successfully
- [ ] Application installs as PWA (Chrome)
- [ ] Offline functionality works (basic)
- [ ] App icon displays correctly

10.2 Backend Testing

API Endpoint Testing:

```
# Test health endpoint
curl https://your-backend.onrender.com/health

# Expected response: {"status": "ok", "timestamp": "..."}

# Test listings endpoint
curl https://your-backend.onrender.com/api/v1/listings

# Expected response: JSON array of listings
```

Functionality Testing:
- [ ] User registration works
- [ ] Login returns JWT tokens
- [ ] Creating listings succeeds

- [ ] Image upload works
- [ ] Real-time messaging connects

10.3 Integration Testing

1. Register a new user through the frontend
2. Verify confirmation email sends (if configured)
3. Login and create a listing with images
4. Check listing appears in search results
5. Test real-time messaging between users

---

11. CI/CD Setup

Continuous Integration and Deployment automates testing and deployment, ensuring code quality and rapid iteration.

11.1 GitHub Actions for Testing

Create `.github/workflows/ci.yml` in your repository:

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test-backend:
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: ./server
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run tests
        run: npm test

      - name: Run linter
        run: npm run lint

  test-frontend:
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: ./client
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
```

```
        with:
          node-version: '18'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run tests
        run: npm test

      - name: Build
        run: npm run build
        env:
          VITE_API_URL: https://your-backend.onrender.com
```

11.2 Automatic Deployments

Both Vercel and Render automatically deploy when you push to GitHub:

1. **Vercel**: Connects through GitHub integration
   - Push to `main` → Automatic production deployment
   - Push to other branches → Preview deployments
2. **Render**: Enable "Auto Deploy" in service settings
   - Push to `main` → Automatic deployment
   - Check "Deploy pull requests" for testing

11.3 Deployment Notifications

Configure Slack or Discord notifications for deployment status:

1. Create incoming webhook in Slack/Discord
2. Add webhook URL to GitHub repository settings
3. Configure workflow to send notifications on deployment status

---

12. Monitoring and Maintenance

Ongoing monitoring ensures your application remains healthy and performant.

12.1 Free Monitoring Tools

Uptime Monitoring (UptimeRobot):
1. Create free account atuptimerobot.com
2. Add monitoring for:

- Frontend URL (every 5 minutes)
- Backend health endpoint (every 5 minutes)
3. Configure email/Slack alerts

Error Tracking (Sentry):
1. Create free account atsentry.io
2. Add Sentry SDK to both frontend and backend
3. Configure error collection and alerting

Analytics (Vercel Analytics):
1. Enable in Vercel dashboard
2. Monitor page views, unique visitors
3. Track performance metrics

12.2 Regular Maintenance Tasks

Weekly:
- Review error logs in Render dashboard
- Check Sentry for new errors
- Monitor uptime reports

Monthly:
- Rotate API keys and secrets
- Review and optimize database indexes
- Check storage usage on Cloudinary

As Needed:
- Update dependencies for security patches
- Scale resources if approaching limits
- Add features based on user feedback

12.3 Performance Optimization

Frontend Optimization:
- Monitor Core Web Vitals in Vercel Analytics
- Optimize images before upload
- Implement lazy loading for images

Backend Optimization:
- Add response caching for frequently accessed data
- Optimize MongoDB queries with proper indexes
- Implement pagination for list endpoints

---

13. Troubleshooting

Common issues and their solutions for successful deployment.

13.1 Frontend Issues

Issue: "Cannot find module" errors during build

Solution: Ensure all dependencies are in package.json and package-lock.json is committed:

```
npm install
git add package.json package-lock.json
git commit -m "Add dependencies"
git push
```

Issue: API calls fail with CORS error

Solution: Add frontend URL to backend CORS configuration:

```
// In server/index.js or server/app.js
app.use(cors({
  origin: process.env.CLIENT_URL || 'http://localhost:5173',
  credentials: true
}));
```

Issue: Environment variables not loading

Solution: Ensure VITE_ prefix is used and redeploy after adding variables:

```
# Frontend requires VITE_ prefix
VITE_API_URL=https://your-backend.onrender.com
```

13.2 Backend Issues

Issue: "Connection refused" to MongoDB

Solution: Verify MongoDB Atlas IP whitelist includes 0.0.0.0/0 for development:
1. Go to MongoDB Atlas > Network Access
2. Add IP Address 0.0.0.0/0
3. Save and wait 60 seconds for propagation

Issue: JWT token errors

Solution: Ensure JWT_SECRET is set in Render environment variables and matches between deployments:

```
# Verify in Render dashboard
# Settings > Environment Variables
JWT_SECRET=your-32-character-secret
```

Issue: Build fails on Render

Solution: Check build logs and ensure root directory is correct:
1. Verify "Root Directory" points to server folder
2. Check package.json has required scripts
3. Ensure Node.js version is 18+

13.3 Database Issues

Issue: "Authentication failed" for MongoDB

Solution: Verify username and password in connection string:

```
# Connection string format
mongodb+srv://username:password@cluster.mongodb.net/database
# Ensure special characters in password are URL-encoded
```

Issue: Slow query performance

Solution: Add appropriate indexes:

```
// In your Listing model
listingSchema.index({ category: 1, createdAt: -1 });
listingSchema.index({ 'location.city': 1 });
listingSchema.index({ price: 1 });
```

13.4 Getting Help

If issues persist:

1. **Check platform status**: Visit status pages (status.vercel.com, status.render.com)
2. **Review logs**: Check Render build/execution logs and browser console
3. **Search documentation**: Review platform docs for updated guides
4. **Community support**: Stack Overflow, platform Discord servers

---

Quick Reference Card

Deployment URLs (Replace with your actual URLs)

Component	URL
Frontend	https://trustmarket.vercel.app
Backend API	https://trustmarket-api.onrender.com
Database	MongoDB Atlas Dashboard
Media	Cloudinary Dashboard

Environment Variables Summary

Backend (Render):
- MONGODB_URI
- JWT_SECRET
- REFRESH_TOKEN_SECRET
- CLOUDINARY_CLOUD_NAME
- CLOUDINARY_API_KEY
- CLOUDINARY_API_SECRET
- CLIENT_URL
- PORT

Frontend (Vercel):
- VITE_API_URL
- VITE_CLOUDINARY_CLOUD_NAME

Useful Commands

```
# Test backend health
curl https://your-api.onrender.com/health

# View logs on Render
render logs --service trustmarket-api

# Trigger manual deploy (push to GitHub)
git add .
git commit -m "Update"
git push
```

Conclusion

Congratulations! You have successfully deployed TrustMarket P2P Marketplace completely free using industry-leading platforms. Your application is now live with:

- ✓ React frontend hosted on Vercel's global CDN
- ✓ Node.js backend API on Render
- ✓ MongoDB database on Atlas with replica set
- ✓ Media storage and optimization on Cloudinary
- ✓ Automatic deployments from GitHub
- ✓ Custom domain with SSL certificate

Continue monitoring your application's performance and iterate based on user feedback. As your application grows, consider upgrading to paid tiers for additional resources and features.

Happy deploying! 🚀

Last Updated: December 2024
For TrustMarket P2P Marketplace v1.0