🎯 ULTRAA EVENTS - COMPLETE SETUP GUIDE

Step-by-Step Implementation for Beginners

📋 TABLE OF CONTENTS

🚀 PHASE 1: PREREQUISITES (Day 1-2)

What You Need:

- ✅ Computer with internet
- ✅ Gmail account
- ✅ Business documents (for WhatsApp verification)
- ✅ Bank account (for Razorpay)
- ✅ Phone number (for WhatsApp Business)

Required Accounts (ALL FREE to create):

1. GitHub Account (Code Storage)

- Go to: https://github.com
- Click "Sign Up"
- Choose FREE plan
- **Why?** Store your code safely

2. MongoDB Atlas (Database)

- Go to: https://www.mongodb.com/cloud/atlas

- Sign up → Choose FREE tier (M0)
- Create a cluster (takes 5 mins)
- Click "Connect" → Get connection string
- **Save this string** - looks like: `mongodb+srv://username:password@cluster.mongodb.net`

3. Railway.app or Render.com (Backend Hosting)

- **Option A - Railway** (Recommended): https://railway.app
  - Sign up with GitHub
  - FREE: $5 credit monthly (enough for testing)
- **Option B - Render**: https://render.com
  - Sign up
  - FREE tier available
- **Why?** Your backend code runs here 24/7

4. Gupshup (WhatsApp API)

- Go to: https://www.gupshup.io
- Sign up → Choose "WhatsApp Business API"
- Fill business details
- Submit documents (GST/Business registration)
- **Approval time:** 2-3 days
- **Cost:** Starting ₹1,500/month

5. Razorpay (Payments)

- Go to: https://razorpay.com
- Sign up → Complete KYC
- Get Test API Keys (start testing for FREE)
- Get Live API Keys (after KYC approval)
- **Cost:** 2% per transaction

🛠️ PHASE 2: BACKEND SETUP (Day 3-5)

Step 1: Install Development Tools

A. Install Node.js

1. Go to: https://nodejs.org
2. Download "LTS" version (Latest Stable)
3. Install (click Next, Next, Finish)
4. Verify: Open Terminal/Command Prompt

```bash
node --version
# Should show: v20.x.x or similar
```

B. Install VS Code (Code Editor)

1. Go to: https://code.visualstudio.com
2. Download & Install
3. Open VS Code

Step 2: Create Project

1. **Create folder on Desktop:**
   - Name it: `ultraa-events-backend`
2. **Open in VS Code:**
   - File → Open Folder → Select `ultraa-events-backend`
3. **Open Terminal in VS Code:**
   - View → Terminal (or press Ctrl + `)
4. **Initialize Project:**

```bash
npm init -y
```

5. **Install Required Packages:**

```bash
npm install express mongoose dotenv razorpay axios cors
```

(This takes 2-3 minutes)

Step 3: Create Files

Create these files in your project folder:

File 1: `.env` (Environment Variables)

```
MONGODB_URI=mongodb+srv://YOUR_USERNAME:YOUR_PASSWORD@cluster.mongodb.net/ultraa-events
RAZORPAY_KEY_ID=your_test_key_id
RAZORPAY_KEY_SECRET=your_test_secret
GUPSHUP_API_KEY=your_gupshup_key
GUPSHUP_APP_NAME=your_app_name
PORT=3000
```

Replace with your actual credentials!

File 2: `server.js`

Copy the complete backend code I provided in the artifact above.

File 3: `.gitignore`

```
node_modules/
.env
```

Step 4: Test Locally

1. **Run the server:**

```
bash

  node server.js
```

2. **You should see:**

```
✅ MongoDB Connected
🚀 Server running on port 3000
```

3. **Test API:**
   - Open browser
   - Go to: `http://localhost:3000/api/events`
   - Should see: `{"success": true, "events": []}`

✅ **Backend is working!**

PHASE 3: DEPLOY BACKEND (Day 6)

Option A: Deploy to Railway.app

1. **Push Code to GitHub:**

```bash
git init
git add .
git commit -m "Initial commit"
git branch -M main
```

- Create new repo on GitHub
- Follow GitHub's push instructions

2. **Deploy on Railway:**
   - Go to Railway.app dashboard
   - Click "New Project"
   - Select "Deploy from GitHub"
   - Choose your repository
   - Railway auto-detects Node.js
   - Click "Deploy"
3. **Add Environment Variables:**
   - Go to project → Variables
   - Add all variables from `.env` file
   - Click "Deploy"
4. **Get Your URL:**
   - Railway gives you a URL like: `https://your-app.railway.app`
   - **Save this URL!**

Option B: Deploy to Render.com

1. Push code to GitHub (same as above)
2. Go to Render.com → New → Web Service
3. Connect GitHub → Select repo
4. Build Command: `npm install`

5. Start Command: `node server.js`
6. Add environment variables
7. Click "Create Web Service"

✅ **Backend is now live 24/7!**

📱 PHASE 4: WHATSAPP INTEGRATION (Day 7-10)

Step 1: Gupshup Setup

1. **Login to Gupshup dashboard**
2. **Go to App Settings:**
   - Note your API Key
   - Note your App Name
3. **Configure Webhook:**
   - Go to "Webhooks" section
   - Add webhook URL: `https://your-app.railway.app/webhook/whatsapp`
   - Save
4. **Test WhatsApp:**
   - Use Gupshup's test environment
   - Send a message to your number
   - Check Railway logs to see if webhook received

Step 2: Create Message Templates

WhatsApp requires pre-approved templates for automated messages.

Template 1: Welcome Message

```
Hello {{1}}! Welcome to Ultraa Events 🎉

We're excited to help you book tickets for amazing events!

Reply with "EVENTS" to see upcoming events.
```

Template 2: Payment Confirmation

```
🎉 Payment Successful!

Your ticket for {{1}} is confirmed!

Order ID: {{2}}
Amount: ₹{{3}}

Show this QR code at venue: {{4}}

See you at the event!
```

Submit these for approval (takes 24 hours).

💳 PHASE 5: RAZORPAY INTEGRATION (Day 11-12)

Step 1: Setup Test Mode

1. **Login to Razorpay Dashboard**
2. **Go to Settings → API Keys**
3. **Generate Test Keys:**
   - Test Key ID: `rzp_test_xxxxx`
   - Test Secret: `xxxxxxxx`
4. **Add to your .env file**

Step 2: Test Payment Flow

1. Use Razorpay test cards:

```
Card: 4111 1111 1111 1111
CVV: Any 3 digits
Expiry: Any future date
```

2. Test the complete flow:
   - Create order
   - Make payment
   - Verify webhook

Step 3: Go Live

1. **Complete KYC** (takes 2-3 days)
2. **Generate Live Keys**

3. **Replace test keys with live keys**
4. **Test with real ₹1 transaction**

🖥️ PHASE 6: ADMIN PANEL (Day 13-15)

Deploy Admin Panel

Option 1: Use Vercel (Easiest - FREE)

1. **Prepare React App:**
   - Create new folder: `ultraa-admin`
   - Copy the admin panel code I provided
   - Initialize: `npx create-react-app .`
   - Replace `src/App.js` with admin panel code
2. **Install Vercel CLI:**

```bash
npm install -g vercel
```

3. **Deploy:**

```bash
cd ultraa-admin
vercel
```

- Login with GitHub
- Follow prompts
- Get URL: `https://your-admin.vercel.app`

Option 2: Use Netlify (Also FREE)

- Similar process to Vercel
- Drag & drop build folder

📱 PHASE 7: MOBILE QR SCANNER (Optional)

For entry scanning at venue, you need a mobile app.

Two Options:

Option 1: Progressive Web App (PWA)

- Build a web app that works on mobile
- Use device camera for QR scanning
- Cheaper & faster to build

Option 2: Native App

- Build Android/iOS app
- Better camera performance
- More expensive

Libraries for QR Scanning:

- `react-qr-reader` (web)
- React Native with `react-native-camera`

💰 COMPLETE COST BREAKDOWN

One-Time Costs:

- Domain (optional): ₹800/year
- **Total:** ₹800

Monthly Costs:

Testing Phase (First Month):

- Railway/Render: FREE (within limits)
- MongoDB: FREE (M0 tier)
- Razorpay: ₹0 (only transaction fee)
- Gupshup: ₹1,500/month
- **Total: ₹1,500/month**

Production (After Launch):

- Backend Hosting: ₹500/month
- Database: ₹0 (FREE tier sufficient for 100 events)
- WhatsApp API: ₹1,500-3,000/month (depends on messages)
- Razorpay: 2% per transaction

- **Total: ₹2,000-3,500/month**

Transaction Costs:

- Razorpay: 2% per ticket
- Example: ₹2,000 ticket = ₹40 fee
- You can add this to ticket price

✅ TESTING CHECKLIST

Before going live, test:

Backend:

- [ ] Can create user
- [ ] Can fetch events
- [ ] Can create order
- [ ] Payment verification works
- [ ] QR code generation works
- [ ] Webhook receives messages

WhatsApp:

- [ ] Welcome message sends
- [ ] User can see events
- [ ] Payment link received
- [ ] Confirmation message works

Payment:

- [ ] Test card payment works
- [ ] Order status updates
- [ ] Email/WhatsApp sent after payment

Admin Panel:

- [ ] Can create events
- [ ] Can view orders
- [ ] Dashboard shows correct stats

🐛 COMMON ISSUES & FIXES

1. "Cannot connect to MongoDB"

Fix: Check connection string in .env file

- Ensure IP whitelist includes `0.0.0.0/0` in MongoDB Atlas

2. "WhatsApp webhook not receiving"

Fix:

- Check webhook URL is correct
- Ensure Railway app is running
- Check Railway logs for errors

3. "Payment not completing"

Fix:

- Verify Razorpay keys are correct
- Check webhook signature verification
- Test with Razorpay test cards first

4. "QR code not working"

Fix:

- Ensure QR code contains correct order ID
- Check if order exists in database
- Verify scanner is hitting correct API endpoint

📞 SUPPORT RESOURCES

- **Railway Docs:** https://docs.railway.app
- **MongoDB Docs:** https://docs.mongodb.com
- **Razorpay Docs:** https://razorpay.com/docs
- **Gupshup Docs:** https://docs.gupshup.io
- **Node.js Docs:** https://nodejs.org/docs

🎯 TIMELINE SUMMARY

- **Week 1:** Account setup + Backend development
- **Week 2:** Deploy + WhatsApp integration
- **Week 3:** Payment integration + Testing
- **Week 4:** Admin panel + Final testing

Total Time: 3-4 weeks (working part-time)

🚀 READY TO LAUNCH?

Once everything is tested:

1. ✅ Switch Razorpay to Live mode
2. ✅ Get WhatsApp templates approved
3. ✅ Create your first event
4. ✅ Share WhatsApp link on Instagram
5. ✅ Start selling tickets!

Need help? Save this guide and follow step-by-step!