Bluff Game Deployment Guide

This guide will help you deploy your bluff game application with a MongoDB database.

Prerequisites

1. **MongoDB Atlas Account** (Free tier available)
2. **Heroku Account** (Free tier available) or **Railway** or **Render**
3. **GitHub Account** (for code hosting)

Step 1: Set Up MongoDB Atlas

1. Go to MongoDB Atlas and create a free account
2. Create a new cluster (M0 Free tier is sufficient)
3. Set up database access:
    - Create a database user with username and password
    - Add your IP address to the IP whitelist (or use 0.0.0.0/0 for all IPs)
4. Get your connection string:
    - Click "Connect" on your cluster
    - Choose "Connect your application"
    - Copy the connection string

Step 2: Update Environment Variables

Update `backend/config.env` with your MongoDB connection string:

```
MONGODB_URI=mongodb+srv://<username>:
<password>@<cluster>.mongodb.net/bluff-game?retryWrites=true&w=majority
PORT=4000
NODE_ENV=production
```

Step 3: Deploy Backend

Option A: Deploy to Heroku

1. Install Heroku CLI and login:

    ```
    npm install -g heroku
    heroku login
    ```

2. Create a new Heroku app:

    ```
    
    ```

```
cd backend
heroku create your-bluff-game-backend
```

3. Set environment variables:

```
heroku config:set MONGODB_URI="your-mongodb-connection-string"
heroku config:set NODE_ENV="production"
```

4. Deploy:

```
git add .
git commit -m "Add database integration"
git push heroku main
```

Option B: Deploy to Railway

1. Go to Railway and connect your GitHub repo
2. Create a new project from your repository
3. Set environment variables in the Railway dashboard
4. Deploy automatically

Option C: Deploy to Render

1. Go to Render and create an account
2. Create a new Web Service from your GitHub repo
3. Set environment variables in the Render dashboard
4. Deploy

Step 4: Update Frontend

Update the API URL in your frontend to point to your deployed backend:

```
// In frontend/src/App.jsx, change:
const API_URL = "https://your-backend-url.herokuapp.com";
```

Step 5: Deploy Frontend

Option A: Deploy to Vercel

1. Install Vercel CLI:

```
npm install -g vercel
```

2. Deploy from frontend directory:

```
cd frontend
vercel
```

Option B: Deploy to Netlify

1. Go to Netlify and connect your GitHub repo
2. Set build command: `npm run build`
3. Set publish directory: `dist`
4. Deploy

Option C: Deploy to GitHub Pages

1. Add to `frontend/package.json`:

```
{
  "homepage": "https://yourusername.github.io/your-repo-name",
  "scripts": {
    "predeploy": "npm run build",
    "deploy": "gh-pages -d dist"
  }
}
```

2. Install gh-pages:

```
npm install --save-dev gh-pages
```

3. Deploy:

```
npm run deploy
```

Step 6: Test Your Deployment

1. Test creating a room
2. Test joining a room
3. Test the game functionality
4. Verify database persistence

Environment Variables Reference

Backend (.env)

- `MONGODB_URI`: Your MongoDB connection string
- `PORT`: Server port (usually set by hosting platform)
- `NODE_ENV`: Environment (development/production)

Frontend

- Update `API_URL` in `App.jsx` to point to your deployed backend

Troubleshooting

Common Issues

1. **CORS Errors**: Make sure your backend CORS settings include your frontend domain
2. **Database Connection**: Verify your MongoDB connection string and network access
3. **Environment Variables**: Ensure all environment variables are set in your hosting platform
4. **Build Errors**: Check that all dependencies are properly installed

Debugging

1. Check your hosting platform's logs
2. Use MongoDB Atlas dashboard to verify database connections
3. Test API endpoints using tools like Postman

Security Considerations

1. **Environment Variables**: Never commit sensitive data to your repository
2. **CORS**: Configure CORS to only allow your frontend domain
3. **Database**: Use strong passwords and limit IP access
4. **HTTPS**: Ensure your deployment uses HTTPS

Monitoring

1. Set up logging for your application
2. Monitor database performance
3. Set up alerts for errors
4. Track user activity and game statistics

Scaling Considerations

1. **Database**: Consider upgrading MongoDB Atlas plan for more storage/performance
2. **Backend**: Scale your backend service based on traffic
3. **CDN**: Use a CDN for static assets
4. **Load Balancing**: Consider load balancing for high traffic

Cost Optimization

1. **MongoDB Atlas**: Free tier includes 512MB storage

2. **Heroku**: Free tier available (with limitations)
3. **Vercel/Netlify**: Free tiers available
4. **Monitor Usage**: Keep track of your usage to avoid unexpected charges