cPanel Deployment Guide for TFG Gaming Club

This guide will help you deploy the TFG Gaming Club application to cPanel hosting with MySQL database.

Prerequisites

- cPanel hosting account with Node.js support (minimum Node.js 18.x)
- MySQL database access via cPanel
- SSH access (recommended) or FTP access
- Domain or subdomain configured in cPanel

Step 1: Set Up MySQL Database

1.1 Create MySQL Database

1. Log in to your cPanel account
2. Navigate to **MySQL® Databases**
3. Create a new database:
   - Database Name: `tfg_gaming` (or your preferred name)
   - Click **Create Database**

1.2 Create MySQL User

1. In the same **MySQL Databases** section
2. Create a new user:
   - Username: `tfg_user` (or your preferred username)
   - Password: Generate a strong password
   - Click **Create User**

1.3 Add User to Database

1. In the **Add User To Database** section
2. Select your user and database
3. Grant **ALL PRIVILEGES**
4. Click **Make Changes**

1.4 Note Your Database Details

You'll need these for the connection string:
- **Database Name**: `cpanel_username_tfg_gaming` (cPanel adds prefix automatically)
- **Database User**: `cpanel_username_tfg_user`
- **Database Host**: Usually `localhost` (check cPanel for exact hostname)
- **Database Password**: The password you set
- **Port**: Usually `3306`

Step 2: Prepare Application Files

2.1 Build the Application Locally

Before uploading, build the application:

```
cd /home/ubuntu/tfg_gaming_club/nextjs_space

# Install dependencies
yarn install

# Generate Prisma client for MySQL
yarn prisma generate

# Build the application
yarn build
```

2.2 Files to Upload

You need to upload these files/folders to your cPanel:

```
tfg_gaming_club/nextjs_space/
├── .next/                # Built application (after running 'yarn build')
├── app/                  # Application source
├── components/           # UI components
├── lib/                  # Utility libraries
├── prisma/               # Database schema
├── public/               # Static assets
├── scripts/              # Utility scripts
├── node_modules/         # Dependencies (or run yarn install on server)
├── package.json          # Dependencies list
├── yarn.lock             # Lock file
├── next.config.js        # Next.js configuration
├── tsconfig.json         # TypeScript configuration
├── tailwind.config.ts    # Tailwind CSS config
└── postcss.config.js     # PostCSS configuration
```

Step 3: Upload Files to cPanel

Option A: Using SSH (Recommended)

1. Connect to your server via SSH
2. Navigate to your public_html or application directory
3. Use SCP/RSYNC to upload files:

```
# From your local machine
rsync -avz --exclude 'node_modules' /home/ubuntu/tfg_gaming_club/nextjs_space/ user-
name@your-domain.com:~/public_html/
```

Option B: Using File Manager/FTP

1. Compress the application folder (exclude node_modules to save space)
2. Upload via cPanel File Manager or FTP client

3. Extract on the server

Step 4: Configure Environment Variables

4.1 Create .env File

In your application root on the server, create a `.env` file:

```
# MySQL Database Connection
# Replace these with your actual cPanel MySQL details
DATABASE_URL="mysql://cpanel_username_tfg_user:your_password@localhost:3306/cpan-
el_username_tfg_gaming"

# NextAuth Secret (generate a new one)
# Run: openssl rand -base64 32
NEXTAUTH_SECRET="your-nextauth-secret-here"

# NextAuth URL (your domain)
NEXTAUTH_URL="https://your-domain.com"

# Discord OAuth (if using Discord login)
DISCORD_CLIENT_ID="your-discord-client-id"
DISCORD_CLIENT_SECRET="your-discord-client-secret"

# Discord Webhook (for notifications)
DISCORD_WEBHOOK_URL="your-discord-webhook-url"
```

4.2 Secure the .env File

Make sure `.env` is not publicly accessible:

```
chmod 600 .env
```

Add to `.htaccess` to block access:

```
<Files ".env">
    Order allow,deny
    Deny from all
</Files>
```

Step 5: Install Dependencies and Set Up Database

5.1 Install Node Modules

SSH into your server and run:

```
cd ~/public_html  # or your application directory
yarn install
```

5.2 Generate Prisma Client

```
yarn prisma generate
```

5.3 Push Database Schema

Create the database tables:

```
yarn prisma db push
```

This will:
- Create all necessary tables in your MySQL database
- Set up the schema as defined in `prisma/schema.prisma`

5.4 Seed Initial Data (Optional)

Populate the database with initial data:

```
yarn prisma db seed
```

This creates:
- Default admin user
- Initial games
- Default settings

Step 6: Configure Node.js Application in cPanel

6.1 Set Up Node.js App

1. In cPanel, navigate to **Setup Node.js App**
2. Click **Create Application**
3. Configure:
   - **Node.js Version**: 18.x or higher
   - **Application Mode**: Production
   - **Application Root**: Path to your application (e.g., `public_html`)
   - **Application URL**: Your domain or subdomain
   - **Application Startup File**: `server.js` (we'll create this next)

6.2 Create Server File

Create `server.js` in your application root:

```
const { createServer } = require('http');
const { parse } = require('url');
const next = require('next');

const dev = process.env.NODE_ENV !== 'production';
const hostname = 'localhost';
const port = process.env.PORT || 3000;

const app = next({ dev, hostname, port });
const handle = app.getRequestHandler();

app.prepare().then(() => {
  createServer(async (req, res) => {
    try {
      const parsedUrl = parse(req.url, true);
      await handle(req, res, parsedUrl);
    } catch (err) {
      console.error('Error occurred handling', req.url, err);
      res.statusCode = 500;
      res.end('internal server error');
    }
  })
    .once('error', (err) => {
      console.error(err);
      process.exit(1);
    })
    .listen(port, () => {
      console.log(`> Ready on http://${hostname}:${port}`);
    });
});
```

6.3 Update package.json

Ensure your `package.json` has the correct start script:

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "node server.js",
    "lint": "next lint"
  }
}
```

6.4 Start the Application

In cPanel Node.js App Manager:
1. Click **Start App** or **Restart App**
2. Check the application status

Step 7: Configure Web Server (Apache/Nginx)

7.1 Set Up Reverse Proxy (.htaccess for Apache)

cPanel usually auto-generates this, but verify your `.htaccess` in the application root:

```
RewriteEngine On
RewriteRule ^$ http://127.0.0.1:3000/ [P,L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ http://127.0.0.1:3000/$1 [P,L]
```

Replace `3000` with your actual Node.js port if different.

Step 8: Set Up Admin User

8.1 Create First Admin User

After deployment, register a user via the website, then SSH into your server:

```
cd ~/public_html

# Update the username in the script
node -e "const { PrismaClient } = require('@prisma/client'); const prisma = new Pris-
maClient(); prisma.user.update({ where: { username: 'your_username' }, data: { isAd-
min: true } }).then(() => console.log('Admin set')).finally(() => prisma.$disconnect()
);"
```

Or use the pre-made script:

```
# Edit scripts/set-sneakyvale-admin.ts to match your username
yarn tsx scripts/set-sneakyvale-admin.ts
```

Step 9: Set Up Scheduled Tasks (Cron Jobs)

For the Monday booking summary notification:

1. In cPanel, navigate to **Cron Jobs**
2. Add a new cron job:
   - **Minute**: 0
   - **Hour**: 17 (5:00 PM)
   - **Day**: *
   - **Month**: *
   - **Weekday**: 1 (Monday)
   - **Command**:
   `bash`
   `cd /home/cpanel_username/public_html && /usr/bin/node -r dotenv/config /usr/bin/yarn tsx scripts/scheduled/monday-booking-summary.ts`

Troubleshooting

Application Won’t Start

1. **Check Node.js version**: Ensure it’s 18.x or higher
   ```bash
      node --version
   ```
2. **Check application logs** in cPanel Node.js App Manager

3. **Verify database connection**:
   ```bash
      yarn prisma db pull
   ```

Database Connection Errors

1. **Verify credentials** in `.env`
2. **Check MySQL hostname** - might not be `localhost`
3. **Ensure user has privileges**
4. **Test connection**:
   ```bash
      mysql -h localhost -u cpanel_username_tfg_user -p cpanel_username_tfg_gaming
   ```

Static Files Not Loading

1. Ensure `.next` folder is uploaded
2. Check file permissions:
   ```bash
      chmod -R 755 .next public
   ```

Port Already in Use

1. Change port in `server.js` or environment variable
2. Update `.htaccess` reverse proxy configuration

Maintenance

Updating the Application

1. Pull/upload latest code
2. Run migrations if database changed:
   ```bash
      yarn prisma db push
   ```
3. Rebuild:
   ```bash
      yarn build
   ```
4. Restart app in cPanel Node.js App Manager

Database Backups

1. Use cPanel **Backup Wizard** for automatic backups

2. Or manually via command line:
   ```bash
   mysqldump -u username -p database_name > backup.sql
   ```

Monitoring

1. Check application logs regularly in cPanel
2. Monitor database size in cPanel MySQL section
3. Set up uptime monitoring (e.g., UptimeRobot)

Security Checklist

- [ ] `.env` file is secured (chmod 600)
- [ ] Database user has minimal necessary privileges
- [ ] Strong passwords for database and NextAuth
- [ ] HTTPS/SSL certificate installed
- [ ] Discord webhook URL is secret
- [ ] Regular backups configured
- [ ] Application error logs don't expose sensitive data
- [ ] Firewall rules configured (if VPS)

Support

For issues specific to:
- **Next.js**: Next.js Documentation (https://nextjs.org/docs)
- **Prisma**: Prisma Documentation (https://www.prisma.io/docs)
- **cPanel Node.js**: Contact your hosting provider
- **Application bugs**: Check the application repository

Quick Reference

Important Files

- `.env` - Environment variables
- `server.js` - Node.js server entry point
- `prisma/schema.prisma` - Database schema
- `.htaccess` - Apache configuration
- `package.json` - Dependencies

Common Commands

```
# Install dependencies
yarn install

# Generate Prisma client
yarn prisma generate

# Push database schema
yarn prisma db push

# Seed database
yarn prisma db seed

# Build application
yarn build

# Start application (production)
yarn start

# View database in browser
yarn prisma studio
```

Database Connection String Format

```
mysql://USERNAME:PASSWORD@HOST:PORT/DATABASE_NAME

Example:
mysql://cpanel_user_tfg:SecurePass123@localhost:3306/cpanel_user_gaming
```

Good luck with your deployment! 🎮