⚡Quick Reference - Command Cheat Sheet

📁File Creation Commands

Create All Folders

bash

```bash
mkdir isteam-employee-hours
cd isteam-employee-hours
mkdir backend frontend docs
```

Backend Files to Create

```
backend/
├── app.py                  # Copy from artifact "Backend API - app.py"
├── requirements.txt        # Copy from artifact "requirements.txt"
├── .env                    # Copy from artifact ".env (Configuration)"
├── Procfile                # Copy from artifact "backend/Procfile"
├── render-build.sh         # Copy from artifact "backend/render-build.sh"
└── .gitignore              # Copy from artifact "backend/.gitignore"
```

Frontend Files to Create

bash

```bash
npx create-react-app frontend
cd frontend
npm install lucide-react
```

Then replace/update:

```
frontend/
├── public/
│   └── index.html          # Replace with artifact "frontend/public/index.html"
├── src/
│   ├── App.js              # Replace with artifact "iSteam Studio - Employee Hours System"
│   ├── index.js            # Replace with artifact "frontend/src/index.js"
│   └── index.css           # Replace with artifact "frontend/src/index.css"
└── package.json            # Add: "proxy": "http://localhost:5000"
```

🗄️PostgreSQL Commands

Start PostgreSQL

```bash
# Mac
brew services start postgresql@14

# Linux
sudo systemctl start postgresql

# Windows - Use Services app or:
net start postgresql-x64-14
```

Access PostgreSQL Terminal

```bash
psql postgres
```

Create Database

```sql
CREATE DATABASE isteam_hours;
\l                    -- List databases
\c isteam_hours       -- Connect to database
\dt                   -- List tables
\q                    -- Quit
```

🐍Backend Commands

Setup Virtual Environment

```bash
cd backend
python -m venv venv

# Activate
source venv/bin/activate       # Mac/Linux
venv\Scripts\activate          # Windows
```

Install Dependencies

```bash
pip install -r requirements.txt
```

Run Backend

```bash
python app.py
# Access at: http://localhost:5000
```

Initialize Database

Visit in browser: `http://localhost:5000/api/init-db`

Deactivate Virtual Environment

```bash
deactivate
```

⚛️Frontend Commands

Install Dependencies

```bash
cd frontend
npm install
npm install lucide-react
```

Run Development Server

```bash
npm start
# Opens automatically at: http://localhost:3000
```

Build for Production

```bash
npm run build
```

Test

```bash
npm test
```

🔧Git Commands

Initialize Repository

```bash
git init
git add .
git commit -m "Initial commit"
```

Connect to GitHub

```bash
git remote add origin https://github.com/YOUR_USERNAME/isteam-employee-hours.git
git branch -M main
git push -u origin main
```

Regular Updates

```bash
git add .
git commit -m "Your commit message"
git push
```

Check Status

```bash
git status
git log --oneline
```

🚀Deployment Commands

Render.com Deployment

No commands needed! Use web interface:

1. Sign up at render.com
2. Click "New +" → PostgreSQL (for database)
3. Click "New +" → Web Service (for backend)
4. Click "New +" → Static Site (for frontend)

Make render-build.sh Executable (Mac/Linux only)

```bash
chmod +x backend/render-build.sh
```

🧪Testing Commands

Test Backend API

```bash
# Using curl
curl http://localhost:5000/api/init-db

# Or visit in browser
open http://localhost:5000/api/init-db     # Mac
start http://localhost:5000/api/init-db    # Windows
```

Check if Ports are in Use

```bash
# Mac/Linux
lsof -i :5000
lsof -i :3000

# Windows
netstat -ano | findstr :5000
netstat -ano | findstr :3000
```

Kill Process on Port

```bash
# Mac/Linux
lsof -ti:5000 | xargs kill
lsof -ti:3000 | xargs kill

# Windows - Get PID from netstat, then:
taskkill /PID <PID_NUMBER> /F
```

📊Database Management

Backup Database

```bash
pg_dump -U postgres isteam_hours > backup.sql
```

Restore Database

```bash
psql -U postgres isteam_hours < backup.sql
```

Reset Database

```bash
psql -U postgres
DROP DATABASE isteam_hours;
CREATE DATABASE isteam_hours;
\q
```

Then visit: `http://localhost:5000/api/init-db`

🔍Debugging Commands

Check Python Version

```bash
python --version
# Should be 3.8 or higher
```

Check Node Version

```bash
node --version
npm --version
# Node should be 16 or higher
```

Check PostgreSQL Version

```bash
psql --version
# Should be 14 or higher
```

View Backend Logs (when running)

```bash
# Just watch the terminal where you ran 'python app.py'
```

View Render Logs

```
In Render dashboard:
1. Click on your service
2. Click "Logs" tab
3. View real-time logs
```

Check Environment Variables

```bash
# Backend
cat backend/.env          # Mac/Linux
type backend\.env         # Windows
```

🌐Useful URLs

Local Development

```
Backend API:    http://localhost:5000
Frontend:       http://localhost:3000
Init DB:        http://localhost:5000/api/init-db
Test Login:     http://localhost:3000
```

Production (Replace with your URLs)

```
Backend API:   https://isteam-hours-api.onrender.com
Frontend:      https://isteam-hours.onrender.com
Init DB:       https://isteam-hours-api.onrender.com/api/init-db
```

👤Test User Accounts

Employee Account

```
Email:    alex@isteam.com
Password: password
Role:     employee
```

Manager Account

```
Email:    manager@isteam.com
Password: password
Role:     manager
```

Admin Account

```
Email:    admin@isteam.com
Password: password
Role:     admin
```

📱Common Tasks Quick Reference

Start Everything Locally

```bash
# Terminal 1 - Backend
cd backend
source venv/bin/activate  # or venv\Scripts\activate on Windows
python app.py

# Terminal 2 - Frontend
cd frontend
npm start
```

Update After Code Changes

```bash
# If you changed backend code:
# Just save the file, Flask auto-reloads

# If you changed frontend code:
# React auto-reloads in browser

# If you changed dependencies:
pip install -r requirements.txt  # Backend
npm install                      # Frontend
```

Deploy Updates to Production

```bash
git add .
git commit -m "Description of changes"
git push

# Render automatically redeploys!
# Wait 5-10 minutes, then refresh your production URL
```

🐛Common Error Fixes

"ModuleNotFoundError: No module named 'flask'"

```bash
cd backend
pip install -r requirements.txt
```

"command not found: npm"

Install Node.js from https://nodejs.org

"psql: command not found"

Install PostgreSQL from https://www.postgresql.org/download/

"EADDRINUSE: address already in use"

```bash
# Kill process on that port (see "Check if Ports are in Use" above)
```

"Failed to fetch" in browser console

1. Make sure backend is running
2. Check API_BASE_URL in App.js
3. Verify CORS is enabled in Flask

Render deployment stuck

1. Check build logs in Render dashboard
2. Verify environment variables are set
3. Make sure files are in correct folders

💾Backup Before Deployment

```bash
# Backup entire project
cd ..
tar -czf isteam-employee-hours-backup.tar.gz isteam-employee-hours/

# Or just zip it
zip -r isteam-employee-hours-backup.zip isteam-employee-hours/
```

🎯Quick Deployment Checklist

```
□ Create all project files
□ Install PostgreSQL and create database
□ Test backend locally (http://localhost:5000)
□ Initialize local database (/api/init-db)
□ Test frontend locally (http://localhost:3000)
□ Test login and features
□ Push code to GitHub
□ Create Render account
□ Deploy PostgreSQL on Render
□ Deploy backend on Render
□ Initialize production database
□ Update frontend API URL
□ Deploy frontend on Render
□ Test production app
□ Celebrate! 🎉
```

📞Help Resources

Official Documentation

- Flask: https://flask.palletsprojects.com/
- React: https://react.dev/
- PostgreSQL: https://www.postgresql.org/docs/
- Render: https://render.com/docs

Community Help

- Stack Overflow: https://stackoverflow.com/
- Reddit r/flask: https://reddit.com/r/flask
- Reddit r/reactjs: https://reddit.com/r/reactjs

Learning Resources

- Python: https://docs.python.org/3/tutorial/
- JavaScript: https://javascript.info/
- SQL: https://www.postgresqltutorial.com/

🎉Success Metrics

Your deployment is successful when:

- ✅You can access your frontend URL
- ✅Login works with test credentials
- ✅Can clock in and out
- ✅Timer counts in real-time
- ✅Manager can approve entries
- ✅Dashboard shows data
- ✅No errors in browser console
- ✅Backend API responds correctly

Print this page and keep it handy while developing! 📄

Good luck with your deployment! 🚀