Stake Engine Game Builder - Complete Build Guide

🎯 Overview

This guide will help you build the Stake Engine Game Builder for:

- ✅ Windows (.exe installer)
- ✅ macOS (.dmg or .app)
- ✅ Linux (.AppImage, .deb, .rpm)
- ✅ iOS (.ipa)
- ✅ Android (.apk)

📋 Prerequisites

For Desktop Builds (Windows, macOS, Linux)

bash

```
# Install Node.js (v18 or higher)
# Download from: https://nodejs.org/

# Verify installation
node --version
npm --version
```

For Mobile Builds (iOS, Android)

bash

```
# Install React Native CLI
npm install -g react-native-cli

# For iOS: Install Xcode (macOS only)
# Download from Mac App Store

# For Android: Install Android Studio
# Download from: https://developer.android.com/studio
```

🚀 STEP 1: Project Setup

Create Project Structure

bash

```bash
# Create main directory
mkdir stake-engine-builder
cd stake-engine-builder

# Initialize project
npm init -y
```

Install Dependencies

bash

```bash
# Core dependencies
npm install react react-dom lucide-react

# Desktop build tools (Electron)
npm install --save-dev electron electron-builder

# For development
npm install --save-dev @vitejs/plugin-react vite
```

📂 STEP 2: Project File Structure

Create this directory structure:

```
stake-engine-builder/
├── public/
│   ├── icon.png (512x512)
│   └── index.html
├── src/
│   ├── App.jsx (your React component)
│   └── main.jsx
├── electron/
│   ├── main.js
│   └── preload.js
├── package.json
├── vite.config.js
└── electron-builder.yml
```

📝 STEP 3: Configuration Files

package.json

json

```json
{
  "name": "stake-engine-builder",
  "version": "1.0.0",
  "description": "Professional Casino Game Builder for Stake Engine",
  "main": "electron/main.js",
  "author": "Your Name",
  "license": "MIT",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "electron:dev": "concurrently \"npm run dev\" \"wait-on http://localhost:5173 && electron .\"",
    "electron:build": "npm run build && electron-builder",
    "build:win": "npm run build && electron-builder --win",
    "build:mac": "npm run build && electron-builder --mac",
    "build:linux": "npm run build && electron-builder --linux"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "lucide-react": "^0.263.1"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.0.0",
    "electron": "^28.0.0",
    "electron-builder": "^24.9.1",
    "vite": "^5.0.0",
    "concurrently": "^8.2.0",
    "wait-on": "^7.2.0"
  },
  "build": {
    "appId": "com.stake.engine.builder",
    "productName": "Stake Engine Builder",
    "directories": {
      "output": "dist-electron"
    },
    "files": [
      "dist/**/*",
      "electron/**/*",
      "package.json"
    ],
    "win": {
      "target": ["nsis"],
      "icon": "public/icon.png"
    },
    "mac": {
      "target": ["dmg", "zip"],
      "icon": "public/icon.png",
      "category": "public.app-category.developer-tools"
    },
    "linux": {
      "target": ["AppImage", "deb", "rpm"],
      "icon": "public/icon.png",
      "category": "Development"
    }
  }
}
```

vite.config.js

javascript

```javascript
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import path from 'path'

export default defineConfig({
  plugins: [react()],
  base: './',
  build: {
    outDir: 'dist',
    emptyOutDir: true
  },
  server: {
    port: 5173
  }
})
```

electron/main.js

📋 ✓

javascript

```
const { app, BrowserWindow } = require('electron');
const path = require('path');
const isDev = process.env.NODE_ENV === 'development';

function createWindow() {
  const mainWindow = new BrowserWindow({
    width: 1400,
    height: 900,
    minWidth: 1200,
    minHeight: 800,
    backgroundColor: '#0f212e',
    webPreferences: {
      preload: path.join(__dirname, 'preload.js'),
      nodeIntegration: false,
      contextIsolation: true
    },
    icon: path.join(__dirname, '../public/icon.png'),
    autoHideMenuBar: true,
    title: 'Stake Engine Game Builder'
  });

  if (isDev) {
    mainWindow.loadURL('http://localhost:5173');
    mainWindow.webContents.openDevTools();
  } else {
    mainWindow.loadFile(path.join(__dirname, '../dist/index.html'));
  }
}

app.whenReady().then(createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});
```

electron/preload.js

javascript

```javascript
const { contextBridge } = require('electron');

contextBridge.exposeInMainWorld('electronAPI', {
  platform: process.platform,
  version: process.versions.electron
});
```

💻 STEP 4: Building Desktop Applications

Windows Build

bash

```bash
# On Windows or any OS with wine installed
npm install
npm run build:win

# Output: dist-electron/Stake Engine Builder Setup 1.0.0.exe
```

Windows Installer Features:

- Auto-update support
- Start menu shortcuts
- Desktop icon
- Uninstaller included

macOS Build

bash

```
# Must be run on macOS
npm install
npm run build:mac

# Output:
# - dist-electron/Stake Engine Builder-1.0.0.dmg
# - dist-electron/Stake Engine Builder-1.0.0-mac.zip
```

macOS App Features:

- Code-signed (with certificate)
- Notarized (for distribution)
- Native menu bar
- Touch Bar support

Linux Build

bash

```
# On Linux or using Docker
npm install
npm run build:linux

# Output:
# - dist-electron/Stake Engine Builder-1.0.0.AppImage
# - dist-electron/stake-engine-builder_1.0.0_amd64.deb
# - dist-electron/stake-engine-builder-1.0.0.x86_64.rpm
```

📱 STEP 5: Building Mobile Applications

iOS Build (macOS only)

Setup React Native Project

bash

```
# Create new React Native project
npx react-native init StakeEngineBuilder
cd StakeEngineBuilder

# Install dependencies
npm install lucide-react-native
```

Build iOS

bash

```
# Install CocoaPods dependencies
cd ios
pod install
cd ..

# Run on simulator
npx react-native run-ios

# Build for device (requires Apple Developer Account)
# 1. Open ios/StakeEngineBuilder.xcworkspace in Xcode
# 2. Select your team in Signing & Capabilities
# 3. Select "Any iOS Device" as target
# 4. Product > Archive
# 5. Distribute App > Ad Hoc or App Store
```

iOS Build Requirements:

- Mac computer with macOS
- Xcode 14+
- Apple Developer Account ($99/year)
- Valid provisioning profile

Android Build

Build Android APK

bash

```
# Generate release APK
cd android
./gradlew assembleRelease

# Output: android/app/build/outputs/apk/release/app-release.apk
```

Build Android App Bundle (for Play Store)

bash

```
cd android
./gradlew bundleRelease

# Output: android/app/build/outputs/bundle/release/app-release.aab
```

Sign APK (Required for distribution)

bash

```
# Generate keystore
keytool -genkeypair -v -storetype PKCS12 -keystore stake-builder.keystore -alias stake-key -keyalg RSA -keysize 2048 -v

# Add to android/app/build.gradle:
signingConfigs {
    release {
        storeFile file('stake-builder.keystore')
        storePassword 'your-password'
        keyAlias 'stake-key'
        keyPassword 'your-password'
    }
}
```

🎨 STEP 6: Custom Icons & Branding

Create App Icons

Desktop Icon (512x512)

Create `public/icon.png` at 512x512 pixels

iOS Icons

Required sizes:

- 20x20, 29x29, 40x40, 58x58, 60x60, 76x76, 80x80, 87x87, 120x120, 152x152, 167x167, 180x180, 1024x1024

Use a tool like:

bash

```bash
npm install -g app-icon
app-icon generate -i icon.png --platforms=ios
```

Android Icons

Required: 48x48, 72x72, 96x96, 144x144, 192x192, 512x512

bash

```bash
app-icon generate -i icon.png --platforms=android
```

📦 STEP 7: Build All Platforms at Once

Universal Build Script

Create `build-all.sh`:

bash

```bash
#!/bin/bash

echo "Building Stake Engine Builder for all platforms..."

# Build desktop versions
echo "Building Windows..."
npm run build:win

echo "Building macOS..."
npm run build:mac

echo "Building Linux..."
npm run build:linux

echo "All desktop builds complete!"
echo "Check dist-electron/ folder for installers"
```

Make executable:

bash

```bash
chmod +x build-all.sh
./build-all.sh
```

🔧 STEP 8: Advanced Configuration

Auto-Update Setup (Desktop)

bash

```
npm install electron-updater

# Add to electron/main.js:
const { autoUpdater } = require('electron-updater');

app.whenReady().then(() => {
  autoUpdater.checkForUpdatesAndNotify();
});
```

Database Integration

bash

```
npm install better-sqlite3
# For local project storage
```

File System Access

bash

```
npm install electron-store
# For user preferences and settings
```

📊 STEP 9: Testing

Test Desktop Build

bash

```
# Development mode
npm run electron:dev

# Production build test
npm run build
npm start
```

Test Mobile Builds

bash

```
# iOS Simulator
npx react-native run-ios

# Android Emulator
npx react-native run-android

# Physical Device
npx react-native run-android --deviceId=<device-id>
```

🚀 STEP 10: Distribution

Windows

- Upload to Microsoft Store
- Host on your website
- Use auto-updater for updates

macOS

- Upload to Mac App Store
- Notarize for Gatekeeper
- Distribute via DMG

Linux

- Publish to Snap Store
- Publish to Flathub
- Host AppImage on GitHub

iOS

- Submit to Apple App Store
- Requires App Store Connect account
- Review process: 1-3 days

Android

- Publish to Google Play Store
- Upload signed AAB file
- Review process: Few hours

🛠️ Complete Build Commands Reference

bash

```bash
# Desktop Development
npm run dev                # Start Vite dev server
npm run electron:dev       # Start Electron in dev mode

# Desktop Production Builds
npm run build:win          # Windows installer
npm run build:mac          # macOS DMG/ZIP
npm run build:linux        # Linux AppImage/DEB/RPM

# Mobile Development
npx react-native start     # Start Metro bundler
npx react-native run-ios   # Run iOS simulator
npx react-native run-android  # Run Android emulator

# Mobile Production Builds
cd android && ./gradlew assembleRelease  # Android APK
cd ios && xcodebuild archive             # iOS Archive
```

📝 Important Notes

Code Signing (Required for Distribution)

Windows:

bash

```bash
# Sign with SignTool
signtool sign /f certificate.pfx /p password /t http://timestamp.server.com app.exe
```

macOS:

bash

```bash
# Sign with Apple Developer Certificate
codesign --deep --force --verify --verbose --sign "Developer ID Application: Your Name" app.app
```

iOS/Android:

- Automatic with Xcode/Android Studio
- Requires valid certificates

System Requirements

Development Machine:

- 16GB RAM minimum
- 50GB free disk space
- SSD recommended
- Multi-core processor (i5/Ryzen 5 or better)

For macOS builds:

- Must use macOS (or macOS VM)
- Xcode Command Line Tools
- Valid Apple Developer account

For iOS builds:

- Mac with latest macOS
- Xcode 14+
- iOS 13+ target

🎯 Quick Start Checklist

- □ Install Node.js v18+
- □ Clone/create project directory
- □ Run `npm install`
- □ Add your app icon (512x512)
- □ Update package.json with your info

- □ Test in dev mode: `npm run electron:dev`
- □ Build for your platform
- □ Test the installer
- □ Sign the application (for distribution)
- □ Distribute or publish

💡 Pro Tips

1. **Use Environment Variables:**
   - Create `.env` for API keys
   - Never commit secrets to Git
2. **Optimize Bundle Size:**
   - Use code splitting
   - Lazy load heavy components
   - Minimize dependencies
3. **Error Handling:**
   - Implement crash reporting (Sentry)
   - Log errors to file
   - Auto-update error fixes
4. **Performance:**
   - Use production builds
   - Enable compression
   - Optimize images
5. **Security:**
   - Keep dependencies updated
   - Use contextIsolation in Electron
   - Validate all user inputs

🆘 Troubleshooting

Build Fails on Windows

bash

```bash
# Try running as Administrator
# Install Windows Build Tools
npm install --global windows-build-tools
```

macOS Code Signing Issues

bash

```
# List certificates
security find-identity -v -p codesigning

# Import certificate
security import certificate.p12 -k ~/Library/Keychains/login.keychain
```

Linux Missing Dependencies

bash

```bash
# Ubuntu/Debian
sudo apt-get install build-essential

# Fedora
sudo dnf install gcc-c++ make
```

iOS Pod Install Fails

bash

```bash
cd ios
pod deintegrate
pod install --repo-update
```

Android Gradle Build Fails

bash

```bash
cd android
./gradlew clean
./gradlew assembleRelease --info
```

📚 Additional Resources

- Electron Documentation
- Electron Builder
- React Native Docs
- Vite Documentation
- Code Signing Guide

🎉 You're Ready!

Follow these steps and you'll have your Stake Engine Game Builder running on all platforms. The entire build process (for all platforms) takes about 15-30 minutes once everything is set up.

Need Help? Check the troubleshooting section or consult the official documentation for each tool.