THE BITCOIN CORPORATION

LLM bApp Development Instructions

AI-Assisted Application Development

Document Version: 1.0 | Date: November 2025 | The Bitcoin Corporation LTD

LLM Instructions for Bitcoin OS bApp Development

Comprehensive Guide for AI-Assisted bApp Creation & Completion

Version: 1.0.0

Target: Claude, GPT-4, and other Large Language Models

Last Updated: 2025-10-05

CRITICAL CONTEXT: You Are Building for Bitcoin OS

FUNDAMENTAL PRINCIPLE: Every bApp you create or complete must be a native component of the Bitcoin OS ecosystem. This is not a standalone application - it's part of a unified, blockchain-native operating system where every application follows identical architectural patterns and integrates seamlessly with the broader ecosystem.

---

1. MANDATORY TEMPLATE ADHERENCE

1.1 Bitcoin Writer as Master Template

RULE: Use Bitcoin Writer (`/Users/b0ase/Projects/bitcoin-writer`) as your ABSOLUTE template. Every bApp must follow this structure exactly:

```
{bApp-name}/
├── LICENSE-COMMUNITY (Copy from bitcoin-writer)
```

```
├── LICENSING.md (Copy from bitcoin-writer)
├── src/
│ ├── components/
│ │ ├── CleanTaskbar.tsx (MANDATORY - Copy and modify)
│ │ ├── UnifiedAuth.tsx (MANDATORY - Copy exact)
│ │ ├── {AppName}Editor.tsx (Core functionality)
│ │ ├── DevSidebar.tsx (MANDATORY - Copy exact)
│ │ └── Footer.tsx (MANDATORY - Copy exact)
│ ├── services/
│ │ ├── HandCashService.ts (MANDATORY - Copy exact)
│ │ ├── BlockchainDocumentService.ts (Adapt for your data type)
│ │ └── {AppName}Service.ts (App-specific logic)
│ ├── pages/ (All three sections MANDATORY)
│ │ ├── developers/ (GitHub contract integration)
│ │ ├── {usersauthorscreators}/ (Main user section)
│ │ └── publishers/ (Content distribution)
│ └── hooks/
│ ├── useHandCash.ts (MANDATORY - Copy exact)
│ └── useBlockchain.ts (MANDATORY - Copy exact)
├── github-issues/ (Token-based development contracts)
├── public/
│ ├── manifest.json (Update for your bApp)
│ └── {bapp}-icon.jpg (Your bApp icon)
└── package.json (Follow dependency patterns)
```

1.2 Non-Negotiable Components

These components MUST be copied exactly from bitcoin-writer:

1. **CleanTaskbar.tsx**

- macOS-style taskbar with bApps menu

- Multicolored "B" button for app switching

- Must include ALL 21+ bApps in menu

- Status area with HandCash integration

2. **UnifiedAuth.tsx**

- HandCash Connect primary authentication

- Google OAuth fallback

- Session management

- User profile integration

3. **DevSidebar.tsx**

- Development tools and statistics

- Desktop-only component

- Real-time development metrics

4. **Footer.tsx**

- Company information and links

- Consistent branding across all bApps

---

2. TECHNICAL REQUIREMENTS (MANDATORY)

2.1 Core Technology Stack

Frontend Framework (REQUIRED):

```
json

{

"react": "^18.2.0",

"typescript": "^4.9.5",

"tailwindcss": "latest",

"react-router-dom": "^6.16.0"

}
```

Blockchain Integration (REQUIRED):

```
json

{

"@handcash/handcash-connect": "^0.8.11",

"@bsv/sdk": "^1.4.24",

"crypto-js": "^4.1.1"

}
```

Development Configuration:

- Port assignment: Assign unique port (Writer: 2010, Wallet: 1050, etc.)
- Vercel deployment configuration
- ESLint and Prettier standards from bitcoin-writer

2.2 Authentication Pattern (EXACT IMPLEMENTATION)

MUST implement exactly as in bitcoin-writer:

```
typescript

// Copy this pattern exactly

const useHandCash = () => {

const [user, setUser] = useState(null);

const [isAuthenticated, setIsAuthenticated] = useState(false);

// Exact implementation from bitcoin-writer

const signIn = async () => {

// HandCash Connect OAuth2 flow

};

const signOut = () => {

// Clear session and redirect

};

return { user, isAuthenticated, signIn, signOut };

};
```

2.3 Blockchain Integration Pattern

Storage Service (Adapt from bitcoin-writer):

```
typescript

interface BlockchainService {

// Adapt these methods for your data type

save{DataType}ToBlockchain(data: {DataType}): Promise;

retrieve{DataType}FromBlockchain(txid: string): Promise<{DataType}>;

encrypt{DataType}(data: {DataType}, password: string): Promise;

decrypt{DataType}(encrypted: string, password: string): Promise<{DataType}>;

}
```

---

3. UI/UX STANDARDIZATION (MANDATORY)

3.1 Design System Requirements

Color Scheme (EXACT):

- Primary: Bitcoin Orange `#f7931a`
- App-specific accent: Choose complementary color
- Background: `bg-gradient-to-br from-gray-900 via-purple-900/20 to-gray-900`
- Text: White on dark theme

Typography:

```
css

font-family: -apple-system, BlinkMacSystemFont, 'SF Pro Text', system-ui, sans-serif;
```

Layout Structure (MANDATORY):

```
tsx
```

```
// This structure is required in every bApp
{/ MANDATORY /}
{/ App-specific content /}
{/ MANDATORY on desktop /}
{/ MANDATORY /}
```

3.2 Responsive Design Requirements

Breakpoints (Consistent across all bApps):

- Mobile: `max-width: 768px`
- Desktop: `min-width: 769px`

Mobile Navigation:

- Hamburger menu for CleanTaskbar
- Touch-optimized controls
- Collapsible sections

---

4. BUSINESS MODEL IMPLEMENTATION (MANDATORY)

4.1 Three-Section Architecture

Every bApp MUST have these exact sections:

1. **Developers Section (`/developers/`)**

```
/developers/
├── offer (Create development contracts)
├── offers (View active contracts)
├── grants (Grant applications)
└── tasks (Available GitHub issues)
```

2. **Main Users Section** (Name varies by app)

- Writers: `/authors/`

- Musicians: `/artists/`

- Painters: `/creators/`

- etc.

3. **Publishers Section (`/publishers/`)**

```
/publishers/
├── offer (Create publishing contracts)
├── offers (View publishing deals)
├── analytics (Performance metrics)
└── distribution (Content distribution tools)
```

4.2 Token Integration (MANDATORY)

Token Naming Convention:

- Format: `$B{APP}` (e.g., $BMUSIC, $BPAINT, $BDRIVE)
- Implement token display in UI
- Token reward system for contributions
- Exchange integration with Bitcoin Exchange

Required Token Features:

```
typescript
interface BAppToken {
symbol: string; // e.g., "BMUSIC"
name: string; // e.g., "Bitcoin Music Token"
balance: number;
transactions: TokenTransaction[];
rewards: TokenReward[];
}
```

4.3 GitHub Integration (MANDATORY)

Issue Template (Copy exact structure from bitcoin-writer):

```
markdown

# {Task Type}: {Description}

## 📋 Requirements

[ ] Requirement 1

[ ] Requirement 2

## 💰 Compensation

**Token Reward:** {amount} {$BTOKEN}

**Priority:** {High/Medium/Low}

**Category:** {Development/Content/Design}

## 📝 Deliverables

1. Deliverable 1

2. Deliverable 2

## 📧 How to Apply

Comment with HandCash handle and approach
```

---

5. DATA MODELING PATTERNS

5.1 Core Data Structure

Adapt this pattern for your bApp's primary data type:

```
typescript
```

```
// Bitcoin Writer uses documents, adapt for your domain

interface BAppDataModel {

id: string;

title: string;

content: any; // Your app's content type

encrypted: boolean;

txHash?: string; // Blockchain transaction

createdAt: Date;

updatedAt: Date;

author: string; // HandCash handle

version: number;

shares?: number; // Tokenized shares

price?: number; // BSV price

isPublished: boolean;

metadata: {

// App-specific metadata

};

}
```

5.2 Blockchain Storage Adaptation

Storage Methods (Copy from bitcoin-writer, adapt data):

1. **OP_RETURN**: Quick saves, small data

2. **OP_PUSHDATA4**: Secure storage, larger data

3. **Multisig P2SH**: Multi-party secured data

4. **NoteSV**: Advanced encryption

---

6. INTEGRATION REQUIREMENTS

6.1 Bitcoin OS Native Features

Window Management:

```
typescript

// Set proper window title for Bitcoin OS

useEffect(() => {

document.title = {bApp Name} - Bitcoin OS;

}, []);
```

bApps Menu Integration:

- Must include ALL other bApps in CleanTaskbar menu
- Current app indicator
- Consistent navigation patterns

6.2 Cross-App Communication

Data Sharing Standards:

```
typescript

interface BAppMessage {

from: string; // Source bApp

to: string; // Target bApp

type: 'import' 'export' 'share';

data: any;

format: string; // MIME type or custom format

}
```

6.3 Exchange Integration

Connect to Bitcoin Exchange ecosystem:

```
typescript

interface ExchangeIntegration {

getTokenPrice(symbol: string): Promise;

createBuyOrder(amount: number, price: number): Promise;
```

```
createSellOrder(amount: number, price: number): Promise;

getOrderBook(symbol: string): Promise;

}
```

---

7. DEVELOPMENT WORKFLOW

7.1 When Creating a New bApp

Step 1: Initial Setup

1. Copy entire bitcoin-writer directory structure

2. Rename all references from "writer" to your app name

3. Update package.json with new name and port

4. Replace Bitcoin Writer branding with your bApp branding

Step 2: Core Adaptation

1. Modify `{AppName}Editor.tsx` for your app's functionality

2. Adapt `BlockchainDocumentService.ts` for your data type

3. Update routes in CleanTaskbar for your app's sections

4. Create app-specific service classes

Step 3: Business Model Implementation

1. Set up developer, user, and publisher sections

2. Create GitHub issues with token rewards

3. Implement token integration

4. Set up exchange connectivity

7.2 When Completing a Half-Built bApp

Assessment Phase:

1. Compare existing structure to bitcoin-writer template

2. Identify missing mandatory components

3. Check for proper authentication integration

4. Verify blockchain service implementation

Completion Phase:

1. Add missing CleanTaskbar, UnifiedAuth, DevSidebar

2. Implement three-section business model

3. Add token integration and GitHub contracts

4. Ensure responsive design compliance

5. Connect to exchange ecosystem

7.3 Quality Assurance Checklist

Before considering any bApp complete:

- [ ] CleanTaskbar with bApps menu functional
- [ ] HandCash authentication working
- [ ] Three business sections implemented
- [ ] Token integration active
- [ ] Blockchain storage functional
- [ ] Mobile responsive design
- [ ] GitHub issues with token rewards
- [ ] Exchange connectivity established
- [ ] Proper licensing files included
- [ ] DevSidebar displaying correctly
- [ ] Footer with company information

---

8. COMMON PITFALLS TO AVOID

8.1 Architecture Violations

DO NOT:

- Create custom authentication systems (use UnifiedAuth exactly)
- Skip the three-section business model
- Implement different navigation patterns
- Use different color schemes or typography
- Create standalone apps (must integrate with Bitcoin OS)

8.2 Integration Failures

ALWAYS:

- Include ALL bApps in your CleanTaskbar menu
- Use HandCash for blockchain operations
- Store data on Bitcoin SV blockchain
- Implement token rewards for development
- Connect to the broader exchange ecosystem

8.3 Business Model Errors

REQUIRED:

- Every bApp must have its own token
- GitHub integration for development contracts
- Publisher section for content distribution
- Revenue sharing models implemented
- Exchange integration for token trading

---

9. TESTING AND VALIDATION

9.1 Integration Testing

Test these connections before completion:

1. HandCash authentication flow

2. Blockchain data storage and retrieval

3. bApps menu navigation to other apps

4. Token balance display and transactions

5. GitHub issue integration

6. Mobile responsive behavior

9.2 Business Model Testing

Verify these business flows:

1. Developer can claim GitHub issues

2. Users can earn/spend tokens

3. Publishers can distribute content

4. Cross-app data sharing works

5. Exchange integration functional

---

10. DEPLOYMENT REQUIREMENTS

10.1 Vercel Configuration

Required files:

```
json

// vercel.json

{

"functions": {

"src/pages/api//*.ts": {

"maxDuration": 30

}
```

```
},
"rewrites": [
{
"source": "/(.*)",
"destination": "/index.html"
}
]
}
```

10.2 Environment Variables

Standard environment variables:

```
bash

# HandCash (Required)

HANDCASH_APP_ID=your_app_id
HANDCASH_APP_SECRET=your_app_secret

# Google Auth (Fallback)

GOOGLE_CLIENT_ID=your_google_id
GOOGLE_CLIENT_SECRET=your_google_secret

# App-specific

BAPP_TOKEN_SYMBOL=BYOURTOKEN
EXCHANGE_API_URL=https://bitcoin-exchange.vercel.app/api
```

---

11. SUCCESS METRICS

11.1 Technical Completion

Your bApp is complete when:

- All mandatory components implemented exactly as specified
- Bitcoin Writer template structure followed precisely
- Three-section business model fully functional
- Token economy integrated and operational
- Exchange connectivity established
- Mobile and desktop responsive
- Authentication flows working correctly

11.2 Business Integration

Your bApp is business-ready when:

- GitHub issues with token rewards active
- Developer, user, and publisher workflows functional
- Token earning/spending mechanisms working
- Exchange integration allowing trading
- Cross-app data sharing implemented
- Revenue models operational

---

12. FINAL INSTRUCTIONS

12.1 Critical Success Factors

Remember: You are building a component of Bitcoin OS, not a standalone app.

1. **STANDARDIZATION**: Follow bitcoin-writer template exactly

2. **INTEGRATION**: Connect to all ecosystem components

3. **TOKENIZATION**: Implement full token economy

4. **BUSINESS MODEL**: Three-section architecture mandatory

5. **QUALITY**: Meet all technical and business requirements

12.2 When in Doubt

If you're unsure about any implementation detail:

1. Check bitcoin-writer for the exact pattern

2. Copy the implementation precisely

3. Adapt only the data model and app-specific logic

4. Maintain all UI/UX and business model standards

5. Test integration with broader ecosystem

12.3 Completion Verification

Before marking any bApp as complete:

- Run through complete quality assurance checklist
- Verify all integrations functional
- Test on both mobile and desktop
- Confirm token economy operational
- Validate exchange connectivity
- Ensure GitHub integration working

REMEMBER: The goal is NOT to build individual applications, but to build components of a unified Bitcoin OS ecosystem. Every decision should strengthen the ecosystem's coherence, usability, and commercial value.

© 2024 The Bitcoin Corporation LTD. All rights reserved.

For support: bitcoinappssuite@gmail.com

Website: https://thebitcoincorporation.website