

# Version 1.0.0 - MVP Implementation Guide

## What We're Building (2 Weeks)

A simple system where admins can create AI-powered tools through conversation with Claude, deploy them as basic web pages, and monetize with ads.

## Day-by-Day Implementation Plan

### Day 1-2: Foundation Setup

#### Option A: Use the Complete Setup Script (RECOMMENDED - 30 minutes)

```
bash

# Download and run the all-in-one setup script
wget [URL to MVP Complete Setup Script]
# OR create it from artifact "DOWNLOAD - MVP Complete Setup Script (All-in-One)"
chmod +x mvp-complete-setup.sh
./mvp-complete-setup.sh

# This script handles EVERYTHING:
# - System dependencies
# - Project structure
# - Database creation
# - Schema setup
# - Basic API server
# - Nginx configuration
# - Frontend placeholder

# After running, just:
# 1. Add your Claude API key to .env
# 2. Update admin password in database (script shows you how)
# 3. Start coding!
```

#### Option B: Manual Setup (2 hours)

```
bash
```

*# Create project directory*

```
mkdir ~/prompt-machine && cd ~/prompt-machine
```

*# Run this setup script (creates all directories and basic config)*

```
cat > setup.sh << 'EOF'
```

```
#!/bin/bash
```

```
# MVP Setup Script - Minimal Version
```

```
# Install Node.js 18
```

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

```
sudo apt install -y nodejs nginx postgresql-client
```

```
# Create directories
```

```
mkdir -p api/src/{routes,services,middleware}
```

```
mkdir -p frontend
```

```
mkdir -p deployed-tools
```

```
# Create package.json
```

```
cd api
```

```
cat > package.json << 'PACKAGE'
```

```
{
```

```
  "name": "prompt-machine-mvp",
```

```
  "version": "1.0.0",
```

```
  "main": "src/index.js",
```

```
  "scripts": {
```

```
    "start": "node src/index.js",
```

```
    "dev": "nodemon src/index.js"
```

```
  },
```

```
  "dependencies": {
```

```
    "express": "^4.18.2",
```

```
    "pg": "^8.11.3",
```

```
    "bcrypt": "^5.1.1",
```

```
    "jsonwebtoken": "^9.0.2",
```

```
    "dotenv": "^16.3.1",
```

```
    "axios": "^1.5.0",
```

```
    "cors": "^2.8.5"
```

```
  },
```

```
  "devDependencies": {
```

```
    "nodemon": "^3.0.1"
```

```
  }
```

```
}
```

```
PACKAGE
```

```
npm install
cd ..

# Create .env file
cat > .env << 'ENV'
# Database
DB_HOST=sql.prompt-machine.com
DB_PORT=5432
DB_NAME=promptmachine_dbbeta
DB_USER=promptmachine_userbeta
DB_PASSWORD=94oE1q7K

# App
PORT=3001
JWT_SECRET=change_me_$(openssl rand -hex 32)

# Claude API (add your key)
CLAUDE_API_KEY=

# Domains
APP_URL=http://localhost:3001
ENV

echo "✅ MVP setup complete!"
EOF

chmod +x setup.sh && ./setup.sh
```

## Step 1.5: Database Setup (30 minutes)

```
bash

# First, create the database if it doesn't exist
PGPASSWORD=Uhr4ryPWey94oE1q7K psql -h sql.prompt-machine.com -U postgres << EOF
-- Check if database exists, if not create it
SELECT 'CREATE DATABASE promptmachine_dbbeta OWNER promptmachine_userbeta'
WHERE NOT EXISTS (SELECT FROM pg_database WHERE datname = 'promptmachine_dbbeta')\gexec

-- Grant permissions
GRANT ALL PRIVILEGES ON DATABASE promptmachine_dbbeta TO promptmachine_userbeta;
EOF

echo "✅ Database created/verified"
```

Step 2: Database Schema - MVP Version (30 minutes)

sql

-- mvp-schema.sql

-- Only the tables we need for MVP

CREATE EXTENSION IF NOT EXISTS "uuid-osp";

-- Users (simple version)

```
CREATE TABLE users (  
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),  
  email VARCHAR(255) UNIQUE NOT NULL,  
  password_hash VARCHAR(255) NOT NULL,  
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP  
);
```

-- Projects

```
CREATE TABLE projects (  
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),  
  user_id UUID REFERENCES users(id),  
  name VARCHAR(255) NOT NULL,  
  slug VARCHAR(255) UNIQUE NOT NULL,  
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP  
);
```

-- Prompts (simplified)

```
CREATE TABLE prompts (  
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),  
  project_id UUID REFERENCES projects(id),  
  system_prompt TEXT,  
  fields JSONB DEFAULT '[]',  
  is_active BOOLEAN DEFAULT true,  
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP  
);
```

-- Conversations (for prompt builder)

```
CREATE TABLE conversations (  
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),  
  project_id UUID REFERENCES projects(id),  
  messages JSONB DEFAULT '[]',  
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP  
);
```

-- Usage tracking (for billing)

```
CREATE TABLE usage_logs (  
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
```

```
project_id UUID REFERENCES projects(id),
tool_slug VARCHAR(255),
ip_address VARCHAR(45),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert default admin
INSERT INTO users (email, password_hash)
VALUES ('admin@prompt-machine.com', '$2b$12$TEMP_HASH');
```

## Run it:

```
bash

PGPASSWORD=94oE1q7K psql -h sql.prompt-machine.com -U promptmachine_userbeta -d promptmachine_dbbeta <
```

## Day 3-4: Core API

### Use Claude Code for Basic API:

```
bash

cd ~/prompt-machine/api

claude "Create a basic Express server at src/index.js with:
- Health check endpoint
- PostgreSQL connection using pg library
- Basic error handling
- CORS enabled
Use the .env file in the parent directory for configuration."

claude "Create src/routes/auth.js with:
- POST /login endpoint that checks email/password against users table
- Returns JWT token on success
- Use bcrypt for password comparison
- Simple implementation, no fancy features"

claude "Create src/middleware/auth.js with a simple JWT verification middleware"
```

## Day 5-6: Project Management

### Use Claude Code:

bash

claude "Create src/routes/projects.js with:

- GET /projects - list user's projects
- POST /projects - create new project
- GET /projects/:id - get single project

Use the auth middleware to protect all routes"

## Day 7-8: Claude Integration

### Create Claude Service:

bash

claude "Create src/services/claude.js that:

- Has a simple chat function that calls Claude API
- Uses CLAUDE\_API\_KEY from environment
- Returns Claude's response as plain text
- Uses claude-3-sonnet-20240229 model for cost savings"

### Create Prompt Builder Route:

bash

claude "Create src/routes/prompt-builder.js with:

- POST /prompt-builder/start - starts a new conversation
  - POST /prompt-builder/message - continues conversation
  - GET /prompt-builder/conversation/:projectId - gets conversation history
- Store messages in the conversations table"

## Day 9-10: Tool Generation & Deployment

### Simple Tool Generator:

bash

claude "Create src/services/toolGenerator.js that:

- Takes a prompt configuration
- Generates a simple HTML file with a form based on prompt.fields
- Includes inline JavaScript that calls our API
- Returns the HTML as a string"

### Deployment Service:

```
bash
```

```
claude "Create src/services/deploy.js that:
```

- Takes project ID
- Gets active prompt
- Generates HTML using toolGenerator
- Saves to deployed-tools/{project-slug}/index.html
- Returns the URL"

## Day 11-12: Basic Frontend

### Create Simple Admin UI:

```
bash
```

```
cd ~/prompt-machine/frontend
```

```
# Create a basic HTML/JS admin interface
```

```
claude "Create a single-page admin interface (index.html) with:
```

- Login form
- Project list
- Create project button
- Link to prompt builder

```
Use vanilla JavaScript and Tailwind CSS from CDN
```

```
Save JWT token to localStorage"
```

```
claude "Create prompt-builder.html that:
```

- Shows chat interface
- Sends messages to API
- Displays Claude responses
- Has a 'Deploy' button when ready"

## Day 13-14: Testing & Launch

### Nginx Configuration:

```
bash
```



*# Serve the frontend*

```
sudo nano /etc/nginx/sites-available/default
```

*# Add:*

```
location / {  
    root /home/ubuntu/prompt-machine/frontend;  
    try_files $uri $uri/ /index.html;  
}  
  
location /api {  
    proxy_pass http://localhost:3001;  
}  
  
location /tools {  
    alias /home/ubuntu/prompt-machine/deployed-tools;  
}
```

```
sudo nginx -t && sudo systemctl reload nginx
```

## Final Testing Checklist:

- ☐ Can create account
- ☐ Can login
- ☐ Can create project
- ☐ Can chat with Claude
- ☐ Can deploy tool
- ☐ Tool accepts input and returns Claude response
- ☐ Google AdSense shows on tools

## What We're NOT Building in MVP

- ✗ NO multi-LLM support (just Claude Sonnet)
- ✗ NO complex deployment (just save HTML files)
- ✗ NO email notifications
- ✗ NO analytics dashboard (just count rows)
- ✗ NO file exports
- ✗ NO two-factor auth
- ✗ NO Redis/caching
- ✗ NO job queues
- ✗ NO payment processing (just ads)

# Claude Code Usage Strategy

## When to use Claude Code:

1. **All basic CRUD operations** - Let Claude write the routes
2. **Database queries** - Claude knows PostgreSQL
3. **API integrations** - Claude can write the axios calls
4. **HTML generation** - Claude is great at templates
5. **Error handling** - Claude knows best practices

## When to write manually:

1. **Business logic** - You know your requirements
2. **Security decisions** - You make the calls
3. **Configuration** - You know your setup

## Success Criteria for v1.0.0

- ✓ Admin can login
- ✓ Admin can create project
- ✓ Admin can chat with Claude to build prompt
- ✓ Admin can deploy tool
- ✓ Users can use deployed tools
- ✓ Tools show Google ads
- ✓ System tracks usage

That's it! If these work, ship it!

## Next Steps After Launch

1. Add Google Analytics to see what users do
2. Add a feedback form
3. Watch server costs
4. Fix critical bugs only
5. Plan v1.1.0 based on user feedback

Remember: This is just v1.0.0. It's supposed to be minimal. The goal is to launch and learn, not to be perfect!