

# SmartJar - Complete Project Folder Structure

```
smartjar/
├── public/
│   ├── index.html          # Main HTML entry point
│   ├── manifest.json       # PWA manifest
│   ├── sw.js               # Service worker for offline functionality
│   └── icons/
│       ├── icon-192.png    # PWA icons
│       ├── icon-512.png
│       └── favicon.ico
│   └── audio/
│       ├── jar-drop.mp3    # Sound effects for jar interactions
│       ├── success.mp3     # Achievement sounds
│       └── notification.mp3 # Alert sounds
├── src/
│   ├── js/
│   │   ├── main.js         # Main application entry point
│   │   ├── core/
│   │   │   ├── app.js      # Main app controller
│   │   │   ├── router.js   # Client-side routing
│   │   │   └── state-manager.js # Global state management
│   │   ├── services/
│   │   │   ├── database.js  # Supabase integration
│   │   │   ├── ai-agent.js  # Gemini AI integration
│   │   │   ├── ocr-service.js # Tesseract.js OCR processing
│   │   │   ├── voice-service.js # Web Speech API wrapper
│   │   │   ├── offline-storage.js # IndexedDB management
│   │   │   ├── sync-service.js # Offline-online sync
│   │   │   └── notification-service.js # Push notifications
│   │   ├── components/
│   │   │   ├── jar-system/
│   │   │   │   ├── jar-container.js # Main jar visualization
│   │   │   │   ├── jar-component.js # Individual jar component
│   │   │   │   ├── allocation-engine.js # Smart allocation logic
│   │   │   │   └── jar-animations.js # CSS animation helpers
│   │   │   ├── income-tracking/
│   │   │   │   ├── screenshot-upload.js # OCR screenshot handler
│   │   │   │   ├── manual-entry.js # Manual income entry form
│   │   │   │   └── income-parser.js # Parse income from different sources
```


- └─ ai-coach/
  - └─ chat-interface.js # AI chat UI
  - └─ voice-coach.js # Voice interaction handler
  - └─ behavioral-analyzer.js # Spending pattern analysis
  - └─ intervention-engine.js # Proactive financial advice


- └─ dashboard/
  - └─ dashboard.js # Main dashboard view
  - └─ progress-tracker.js # Goal tracking
  - └─ insights-panel.js # Financial insights
  - └─ history-view.js # Transaction history

- └─ onboarding/
  - └─ welcome-flow.js # User onboarding
  - └─ profile-setup.js # User profile creation
  - └─ jar-explanation.js # Explain jar concept
  - └─ demo-tutorial.js # Interactive tutorial









- └─ common/
  - └─ modal.js # Reusable modal component
  - └─ button.js # Button components
  - └─ form-elements.js # Form inputs
  - └─ loader.js # Loading states

- └─  utils/
  - └─ helpers.js # General utility functions
  - └─ currency.js # Currency formatting
  - └─ date-utils.js # Date manipulation
  - └─ validation.js # Input validation
  - └─ storage-utils.js # Storage helpers
  - └─ device-utils.js # Device detection

- └─  ai/
  - └─ agent-behaviors.js # Core AI agent logic
  - └─ financial-rules.js # Rule-based financial logic
  - └─ pattern-detection.js # Spending pattern algorithms
  - └─ prediction-engine.js # Financial outcome predictions
  - └─ context-builder.js # Build context for AI calls

- └─  config/
  - └─ constants.js # App constants
  - └─ supabase-config.js # Database configuration
  - └─ ai-config.js # AI service configuration
  - └─ app-config.js # General app settings

- └─  styles/

└─ main.css	# Main stylesheet
└─  components/	
└─ jars.css	# Jar visualization styles
└─ dashboard.css	# Dashboard styles
└─ forms.css	# Form styling
└─ modals.css	# Modal styles
└─ buttons.css	# Button styles
└─ animations.css	# CSS animations
└─  responsive/	
└─ mobile.css	# Mobile-specific styles
└─ tablet.css	# Tablet styles
└─ desktop.css	# Desktop styles
└─  themes/	
└─ light.css	# Light theme
└─ dark.css	# Dark theme
└─ variables.css	# CSS custom properties
└─  assets/	
└─  images/	
└─ jar-empty.svg	# Empty jar illustration
└─ jar-filled.svg	# Filled jar illustration
└─ money-animation.gif	# Money dropping animation
└─ coach-avatar.png	# AI coach character
└─ onboarding/	# Onboarding illustrations
└─ step1.svg	
└─ step2.svg	
└─ step3.svg	
└─ icons/	# App icons
└─ salary-jar.svg	
└─ emergency-jar.svg	
└─ future-jar.svg	
└─  fonts/	
└─ custom-fonts.woff2	# Any custom fonts (if needed)
└─  workers/	
└─ ocr-worker.js	# Tesseract.js worker
└─ sync-worker.js	# Background sync worker
└─ ai-worker.js	# AI processing worker
└─  database/	
└─ schema.sql	# Supabase database schema
└─ migrations/	# Database migrations
└─ 001_initial_setup.sql	
└─ 002_add_ai_features.sql	

```
| | └─ 003_add_offline_support.sql
| |
| └─ seed-data.js          # Sample data for testing
|   └─ backup/            # Database backup scripts
|     └─ backup-schema.sql
|
| └─ tests/
|   └─ unit/
|     └─ allocation-logic.test.js # Test allocation algorithms
|     └─ ocr-service.test.js      # Test OCR functionality
|     └─ ai-agent.test.js         # Test AI agent behaviors
|
|   └─ integration/
|     └─ offline-sync.test.js     # Test offline functionality
|     └─ voice-integration.test.js # Test voice features
|
|   └─ e2e/
|     └─ user-journey.test.js     # Complete user flow tests
|     └─ demo-scenario.test.js    # Hackathon demo scenarios
|
| └─ docs/
|   └─ README.md                # Project documentation
|   └─ API.md                   # API documentation
|   └─ DEPLOYMENT.md            # Deployment guide
|   └─ DEMO.md                  # Demo script for judges
|   └─ USER-GUIDE.md           # User manual
|   └─ architecture/
|     └─ system-design.md        # System architecture
|     └─ ai-agent-design.md      # AI agent architecture
|     └─ offline-strategy.md     # Offline-first approach
|
| └─ deployment/
|   └─ vercel.json              # Vercel deployment config
|   └─ scripts/
|     └─ build.js               # Build script
|     └─ deploy.js              # Deployment script
|     └─ setup-env.js           # Environment setup
|
|   └─ environment/
|     └─ .env.example            # Environment variables template
|     └─ dev.env                 # Development environment
|     └─ prod.env                # Production environment
|
| └─ hackathon/
|   └─ pitch-deck.md            # Presentation slides
|   └─ demo-script.md           # Demo walkthrough
|   └─ user-personas.md         # Target user profiles
```

```

├── market-research.md      # Market analysis
├── business-model.md       # Revenue strategy
├── ── submission/
│   ├── round1-submission.md # Round 1 materials
│   ├── round2-demo.md      # Round 2 demo plan
│   └── technical-architecture.md # Technical deep-dive
├── ── tools/
│   ├── data-generator.js    # Generate test data
│   ├── ocr-tester.js       # Test OCR accuracy
│   ├── ai-prompt-tester.js  # Test AI responses
│   └── performance-monitor.js # Performance testing
├── ── examples/
│   ├── sample-screenshots/  # Test OCR images
│   │   ├── paytm-payment.jpg
│   │   ├── gpay-received.jpg
│   │   ├── swiggy-earnings.png
│   │   └── uber-payout.jpg
│   └── demo-data/
│       ├── sample-users.json    # Demo user profiles
│       ├── sample-transactions.json # Demo transaction data
│       └── sample-conversations.json # AI conversation examples
├── package.json             # Node.js dependencies (for tools only)
├── .gitignore               # Git ignore rules
├── LICENSE                  # MIT License
├── README.md                # Main project README
└── CONTRIBUTING.md          # Contribution guidelines

```

## Implementation Priority Order

### Week 1: Foundation (Core Jar System)

1. `src/components/jar-system/` - Build jar visualization
2. `src/services/database.js` - Set up Supabase connection
3. `src/components/income-tracking/manual-entry.js` - Basic income input
4. `styles/components/jars.css` - Jar animations

### Week 2: Smart Features

1. `src/services/ocr-service.js` - Tesseract.js integration
2. `workers/ocr-worker.js` - Background OCR processing
3. `src/ai/financial-rules.js` - Rule-based advice system

4. `src/services/offline-storage.js` - IndexedDB setup

## Week 3: AI & Polish

1. `src/services/ai-agent.js` - Gemini API integration
2. `src/services/voice-service.js` - Web Speech API
3. `src/components/onboarding/` - User onboarding flow
4. `styles/responsive/mobile.css` - Mobile optimization



## Key Features by Folder

### Offline-First Architecture:

- `workers/sync-worker.js` - Background sync
- `src/services/offline-storage.js` - Local data management
- `public/sw.js` - Service worker for PWA

### AI Agent System:

- `src/ai/agent-behaviors.js` - Autonomous AI behaviors
- `src/ai/pattern-detection.js` - Spending pattern analysis
- `src/components/ai-coach/` - AI interaction components

### Screenshot OCR:

- `src/services/ocr-service.js` - Tesseract.js wrapper
- `examples/sample-screenshots/` - Test images for development
- `workers/ocr-worker.js` - Process images in background

### Hackathon Ready:

- `hackathon/` folder contains all presentation materials
- `examples/` folder has demo data ready to go
- `tools/` folder helps with quick testing and setup

This structure supports the complete free tech stack (Vanilla JS + Supabase + Gemini + Tesseract.js + Web Speech API + Vercel) and is optimized for the 48-hour hackathon timeline while maintaining scalability for post-hackathon development.