Mind Map Mini - Complete Documentation

Table of Contents

- 1. Overview
- 2. Installation Guide
- 3. Architecture
- 4. User Guide
- 5. API Reference
- 6. File Structure
- 7. Comparison with Mind Map Master
- 8. <u>Troubleshooting</u>

Overview

Mind Map Mini is a lightweight, portable mind mapping application that requires no database or complex setup. All data is stored as JSON files locally, making it perfect for:

- Students who need a simple tool without cloud dependencies
- professionals working in restricted environments
- û Users who prioritize data privacy
- Quick prototyping and brainstorming sessions
- III Offline-first workflows

Key Features

Feature	Description	
No Database	All maps stored as JSON files	
Dual Methods	GRINDE (learning-optimized) and Buzan (classic)	
Auto-save	Never lose work with automatic saving	
Multiple Exports	JSON, Markdown, HTML, Text	
Templates	Pre-built templates for common use cases	
Portable	Entire app in just 2 files	
Privacy-First	All data stays on your machine	
4	•	

Installation Guide

Quick Start (30 seconds)

```
bash
# 1. Create project directory
mkdir mindmap-mini
cd mindmap-mini
# 2. Save the files
# - Save app.py from the Flask artifact
# - Create templates/index.html from the HTML artifact

# 3. Install Flask
pip install Flask flask-cors

# 4. Run the application
python app.py

# 5. Open browser
# Navigate to http://localhost:5000
```

Detailed Installation

Option 1: Using Setup Script

1. Create a new directory:

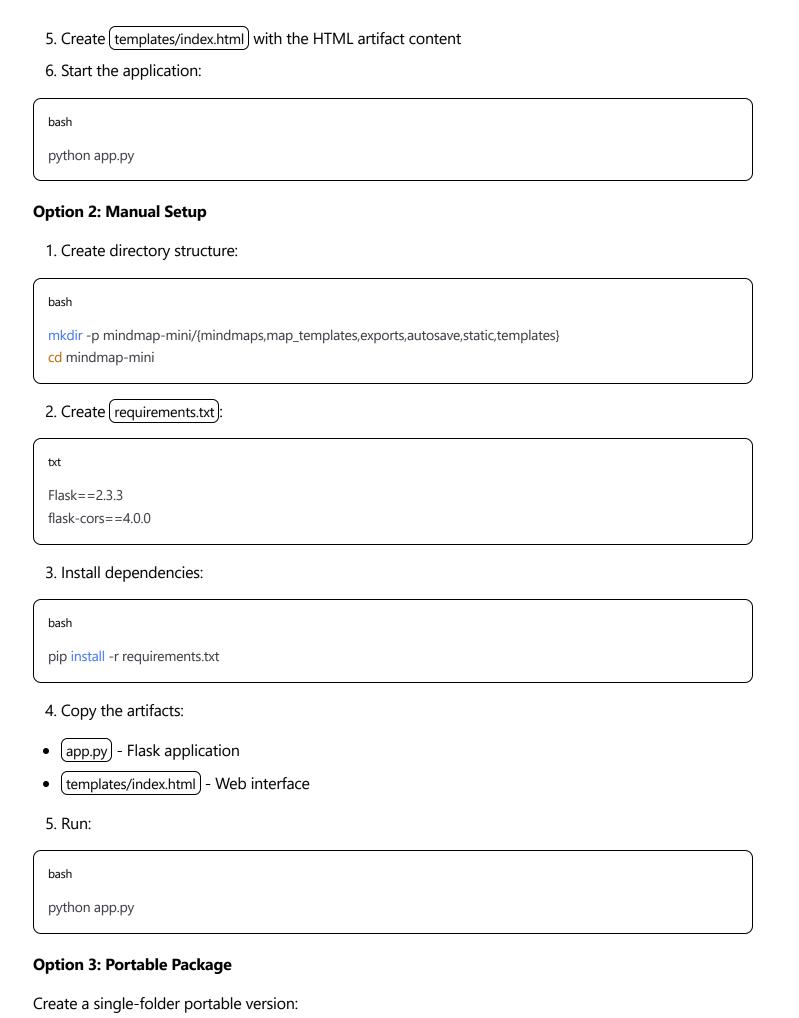
bash

mkdir mindmap-mini

cd mindmap-mini

- 2. Create (app.py) with the Flask code from the artifact
- 3. Create (setup.py) with the setup script code
- 4. Run setup:

bash
python setup.py



```
bash

# Create virtual environment

python -m venv venv

source venv/bin/activate # Windows: venv\Scripts\activate

# Install dependencies

pip install Flask flask-cors

# Create run script

echo "venv/bin/python app.py" > run.sh

chmod +x run.sh
```

Now the entire folder is portable - just copy and run!

Architecture

System Design

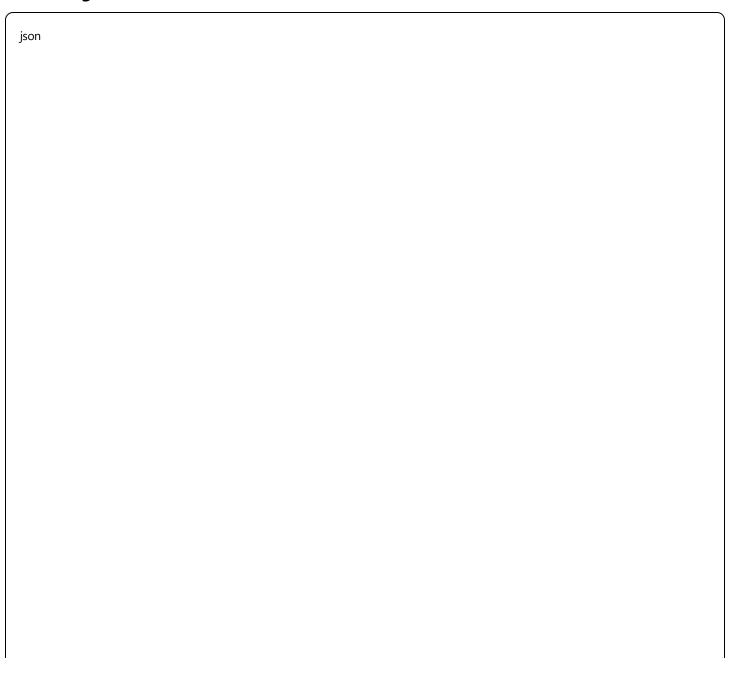
```
Web Browser
HTML5 Canvas Interface
- GRINDE/Buzan Modes
- Real-time Rendering
   HTTP/JSON
 Flask Server
RESTful API
- Map CRUD Operations
- Template Management
- Export Functions
   File I/O
Local File System
JSON Files
```



Data Flow

- 1. **Create/Edit**: User actions → Canvas → API → JSON file
- 2. **Load**: JSON file \rightarrow API \rightarrow Canvas \rightarrow Visual render
- 3. **Export**: JSON data → Converter → Output format
- 4. **Auto-save**: Canvas state \rightarrow Background save \rightarrow Temp file

File Storage Structure



```
// Example: mindmaps/abc123.json
 "id": "abc123",
 "title": "Project Planning",
 "mode": "grinde",
 "created": "2024-01-15T10:30:00",
 "modified": "2024-01-15T11:45:00",
 "nodes": [
   "id": "node_1",
   "x": 400,
   "y": 300,
   "text": "Main Goal",
   "type": "central",
   "color": "#6366f1",
   "size": 30
 ],
 "connections": [
   "source": "node_1",
   "target": "node_2",
   "type": "arrow",
   "color": "#6366f1"
 ]
```

User Guide

Creating Your First Mind Map

1. Start with a Central Idea

- Open the application
- Double-click the center of canvas
- Enter your main topic

2. Add Branches

- Click "Group" button for main categories
- Click "Concept" for sub-ideas

• Click "Detail" for specific points

3. Connect Ideas

- Click "Connect" tool
- Click source node, then target node
- Use "Arrow" for directional flow

4. Organize Visually

- Drag nodes to reposition
- Use colors to categorize
- Adjust sizes for hierarchy

GRINDE Method Guide

G - **Grouped** (Regroupé)

- What: Organize related concepts together
- How: Use the Group node type for categories
- Why: Improves memory through spatial association
- Example: Group all "Resources" nodes in one area

R - Reflective (Réflexif)

- What: Transform information into your own words
- How: Never copy text directly, always paraphrase
- Why: Forces deep understanding
- Example: "Photosynthesis" → "Plants eat sunlight"

I - Interconnected (Interconnecté)

- What: Create multiple connections between ideas
- How: Use the Connect tool liberally
- **Why**: Strengthens neural pathways
- Example: Link "Budget" to both "Timeline" and "Resources"

N - Non-verbal (Non-verbal)

- What: Use visual elements
- **How**: Add emojis, colors, vary sizes

• Why: Engages visual memory

D - Directional (Directionnel)

• What: Show cause-and-effect relationships

How: Use arrow connections

Why: Clarifies logical flow

• **Example**: "Research" → "Design" → "Implementation"

E - Emphasized (Accentué)

• What: Highlight importance visually

• **How**: Use size, color, position

Why: Guides attention to key concepts

Example: Make critical path nodes larger and brighter

Keyboard Shortcuts

Shortcut	Action	
Double-click	Create new node	
Delete	Delete selected	
Ctrl+S	Save map	
Ctrl+Z	Undo	
Ctrl+Y	Redo	
Shift+Drag	Pan canvas	
Ctrl+Scroll	Zoom	
Escape	Deselect	
Tab	Next node	
Enter	Edit selected	
4	•	

Tips & Tricks

1. Start Simple: Begin with 5-7 main branches maximum

2. **Use Templates**: Don't reinvent the wheel

3. **Color Code**: Assign meaning to colors consistently

4. Regular Saves: Use Ctrl+S frequently

- 5. Review Mode: Zoom out to see the big picture
- 6. Export Often: Keep backups in different formats

API Reference

Endpoints

Maps Management

GET /api/maps

- Returns list of all saved maps
- Response: ({ success: true, maps: [...] })

GET /api/map/{id}

- Get specific map by ID
- Response: ({ success: true, data: {...} })

POST /api/map

- Save new or update existing map
- Body: Complete map JSON
- Response: {{ success: true, id: "map_id" }

DELETE /api/map/{id}

- Delete map (moves to trash)
- Response: ({ success: true })

Templates

GET /api/templates

- Get available templates
- Response: { success: true, templates: [...] }

GET /api/template/{id}

- Get specific template
- Response: ({ success: true, data: {...} })

Export

GET /api/export/{map_id}/{format}

- Export map in specified format
- Formats: json, markdown, html, text
- Returns: File download

Utilities

POST /api/autosave

- Save temporary version
- Body: Map data
- Response: ({ success: true })

GET /api/stats

- Get usage statistics
- Response: ({ success: true, stats: {...} })

JavaScript API (Frontend)

```
javascript

// Create a node

const node = new Node(x, y, text, type);

// Add to map

currentMap.nodes.push(node);

// Create connection

const conn = new Connection(sourceNode, targetNode, 'arrow');

// Save map

await saveMap();

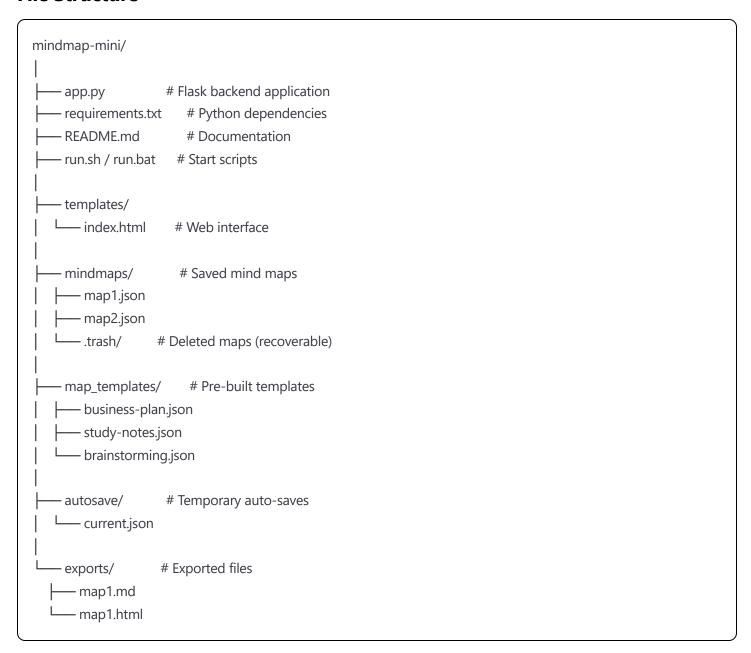
// Load map

await loadMap(mapId);

// Export

await exportMap('markdown');
```

File Structure



Comparison

Mind Map Mini vs Mind Map Master

Feature	Mini	Master
Storage	JSON files	Database
Setup Time	30 seconds	10+ minutes
Dependencies	Flask only	Flask, Redis, PostgreSQL, Docker
Collaboration	No	Yes (real-time)
User Management	No	Yes
	•	'

Feature	Mini	Master
File Size	~50KB	~5MB
Offline Work	Yes	Limited
Portability	High	Low
Enterprise Features	No	Yes
Best For	Personal use	Teams
4	1	•

When to Use Mini

Choose Mini when you need:

- Quick setup without configuration
- Complete data privacy
- Offline-first workflow
- Portable solution
- Simple personal tool

When to Use Master

Choose Master when you need:

- Team collaboration
- User authentication
- Cloud storage
- Advanced analytics
- Enterprise integration

Troubleshooting

Common Issues

1. Port Already in Use

```
bash
# Change port in app.py
app.run(debug=True, port=5001) # Use different port
```

2. Templates Not Loading

bash

Check file structure

Is templates/ # Should show index.html

3. Can't Save Maps

bash

Check permissions

chmod 755 mindmaps/

4. Lost Work

bash

Check autosave folder

Is autosave/ # Recovery files here

5. Export Not Working

bash

Create exports folder

mkdir exports

Performance Tips

1. Large Maps: Keep under 500 nodes for best performance

2. **Browser**: Use Chrome/Firefox for best Canvas support

3. **Memory**: Close unused tabs to free memory

4. Cleanup: Periodically clear autosave folder

Data Recovery

If you lose data:

- 1. Check (autosave/) folder for recent saves
- 2. Check (mindmaps/.trash/) for deleted maps
- 3. Use browser's localStorage (if available)
- 4. Restore from exports

Advanced Usage

Custom Templates

Create your own template:

```
json

// map_templates/custom.json
{

"title": "My Template",

"mode": "grinde",

"nodes": [
{

"id": "1",

"text": "Start Here",

"type": "central",

"y": 400,

"y": 300
}

],

"connections": []
}
```

Batch Operations

Process multiple maps:

```
python
import os
import json

# Convert all maps to Markdown
for filename in os.listdir('mindmaps'):
   if filename.endswith('.json'):
     with open(f'mindmaps/{filename}', 'r') as f:
     data = json.load(f)
     # Process data...
```

Integration Examples

With Obsidian

Export as Markdown and import to Obsidian vault

With Notion

Export as Markdown, copy-paste to Notion page

With GitHub

Store JSON files in Git for version control

Support & Resources

Getting Help

1. **Documentation**: This guide

2. **GitHub Issues**: Report bugs

3. Community: Discord/Forum

4. Email: support@example.com

Contributing

Mind Map Mini is open source! Contribute by:

- Reporting bugs
- Suggesting features
- Submitting pull requests
- Creating templates
- Writing tutorials

License

MIT License - Free for personal and commercial use

Quick Reference Card



Mind Map Mini - Quick Reference

CREATING NODES:

- Double-click canvas → New node
- Toolbar buttons → Specific types

- Quick Add (+) → Fast creation

CONNECTIONS:

- Connect tool → Click two nodes
- Arrow tool → Directional link
- Auto-connect → New nodes link to selected

NAVIGATION:

- Drag nodes → Reposition
- Shift+Drag → Pan view
- Mouse wheel → Zoom
- Reset button → Center view

SAVING:

- Ctrl+S → Manual save
- Auto-save → Every 2 seconds
- Export \rightarrow Multiple formats

MODES:

- GRINDE \rightarrow Learning-optimized
- Buzan → Traditional radial

Remember: All data saved locally as JSON!

Mind Map Mini v1.0 - Simple, Private, Powerful

