API Documentation

Overview

Clarity Razor provides a RESTful API for tile generation, file management, and user operations. All API routes require authentication unless otherwise specified.

Base URL: `http://localhost:3000` (development) or your deployed URL

Authentication: Session-based via NextAuth.js (cookies)

Table of Contents

Authentication

Sign Up

Create a new user account.

Endpoint: `POST /api/signup`

Request Body:

```json
{
  "name": "John Doe",
  "email": "john@doe.com",
  "password": "SecurePassword123!"
}
```

Response (201 Created):

```json
{
  "message": "User created successfully"
}
```

Errors:
- `400` : Invalid input or email already exists
- `500` : Server error

Sign In

Authenticate and create a session.

Endpoint: `POST /api/auth/signin`

Note: Uses NextAuth.js - call `signIn('credentials', { email, password })` from client

Response: Sets session cookie, redirects to `/dashboard`

Sign Out

Endpoint: `POST /api/auth/signout`

Note: Uses NextAuth.js - call `signOut()` from client

Response: Clears session cookie

Get Session

Endpoint: `GET /api/auth/session`

Response (200 OK):

```
{
  "user": {
    "id": "user_abc123",
    "email": "john@doe.com",
    "name": "John Doe"
  },
  "expires": "2026-01-30T00:00:00.000Z"
}
```

Tile Operations

Generate Tile

Generate a Clarity Tile from user input and optional files.

Endpoint: `POST /api/generate-tile`

Authentication: Required

Request Body:

```json
{
  "objective": "Launch new product by Q2 2026",
  "constraints": ["Budget: $50k", "Team: 3 people", "Timeline: 4 months"],
  "contextDump": "We're a startup with limited resources...",
  "mode": "razor",
  "fileIds": ["file_xyz789", "file_abc123"],
  "tags": ["product", "launch"],
  "encryptionPassword": "optional-password-for-encrypted-files"
}
```

Parameters:
- `objective` (string, required): Main goal or objective
- `constraints` (array, optional): List of constraints
- `contextDump` (string, optional): Additional context
- `mode` (enum, required): One of: `razor`, `backcast`, `drill`, `connection_hunt`, `deepagent_task_spec`
- `fileIds` (array, optional): IDs of uploaded files to process
- `tags` (array, optional): Tags for organization
- `encryptionPassword` (string, optional): Password for encrypted files

Response (200 OK):

```
{
  "id": "tile_xyz789",
  "objective": "Launch new product by Q2 2026",
  "constraints": ["Budget: $50k", "Team: 3 people"],
  "context": "Parsed and structured context...",
  "keyFindings": [
    "Market research shows demand",
    "Competitors are slow to adapt"
  ],
  "assumptions": [
    "Budget will be approved",
    "No major technical blockers"
  ],
  "risks": [
    {
      "description": "Timeline slippage",
      "severity": "high",
      "mitigation": "Weekly sprints with buffer time"
    }
  ],
  "dependencies": [
    "Design team availability",
    "Marketing campaign ready"
  ],
  "successMetrics": [
    "1000 users in first month",
    "Positive reviews >4.5 stars"
  ],
  "tasks": [
    {
      "description": "Finalize product requirements",
      "priority": "high",
      "estimatedEffort": "2 weeks"
    },
    {
      "description": "Build MVP",
      "priority": "high",
      "estimatedEffort": "6 weeks"
    }
  ],
  "nextAction": "Schedule kickoff meeting with team",
  "tags": ["product", "launch"],
  "createdAt": "2025-12-31T23:30:00Z",
  "updatedAt": "2025-12-31T23:30:00Z"
}
```

Errors:
- `400` : Invalid input (missing objective, invalid mode)
- `401` : Unauthorized (not logged in)
- `404` : File not found or doesn’t belong to user
- `500` : Server error or AI processing error

List Tiles

Get all tiles for the authenticated user.

Endpoint: `GET /api/tiles`

Authentication: Required

Query Parameters:
- `search` (string, optional): Search in objective, context, or tags
- `tags` (string, optional): Comma-separated tag filter (e.g., `tags=product,launch`)
- `limit` (number, optional): Max results (default: 50)
- `offset` (number, optional): Pagination offset (default: 0)

Example: `GET /api/tiles?search=product&tags=launch&limit=10`

Response (200 OK):

```
{
  "tiles": [
    {
      "id": "tile_xyz789",
      "objective": "Launch new product",
      "tags": ["product", "launch"],
      "createdAt": "2025-12-31T23:30:00Z",
      "updatedAt": "2025-12-31T23:30:00Z"
    }
  ],
  "total": 42,
  "limit": 10,
  "offset": 0
}
```

Errors:
- `401` : Unauthorized
- `500` : Server error

Get Single Tile

Retrieve a specific tile by ID.

Endpoint: `GET /api/tiles/[id]`

Authentication: Required

Response (200 OK):

```
{
  "id": "tile_xyz789",
  "objective": "Launch new product",
  "constraints": [...],
  "context": "...",
  // ... full tile data
}
```

Errors:
- `401` : Unauthorized
- `403` : Forbidden (tile doesn’t belong to user)
- `404` : Tile not found
- `500` : Server error

Delete Tile

Delete a tile and optionally its associated files.

Endpoint: `DELETE /api/tiles/[id]`

Authentication: Required

Query Parameters:
- `deleteFiles` (boolean, optional): Also delete associated files (default: false)

Example: `DELETE /api/tiles/tile_xyz789?deleteFiles=true`

Response (200 OK):

```
{
  "message": "Tile deleted successfully",
  "filesDeleted": 3
}
```

Errors:
- `401` : Unauthorized
- `403` : Forbidden
- `404` : Tile not found
- `500` : Server error

File Upload

Get Presigned Upload URL

Request a presigned URL for direct upload to S3.

Endpoint: `POST /api/upload/presigned`

Authentication: Required

Request Body:

```
{
  "fileName": "requirements.pdf",
  "fileSize": 1048576,
  "contentType": "application/pdf",
  "isPublic": false
}
```

Parameters:
- `fileName` (string, required): Original filename
- `fileSize` (number, required): File size in bytes
- `contentType` (string, required): MIME type
- `isPublic` (boolean, optional): Public access (default: false)

Response (200 OK):

```json
{
  "uploadUrl": "https://bucket.s3.amazonaws.com/path?signature=...",
  "cloud_storage_path": "clarity-razor/uploads/1735689000000-requirements.pdf",
  "expiresIn": 3600
}
```

Usage Flow:
1. Get presigned URL from this endpoint
2. Upload file directly to S3 using the URL (PUT request)
3. Call `/api/upload/complete` with metadata

Client-Side Upload Example:

```javascript
const response = await fetch('/api/upload/presigned', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    fileName: file.name,
    fileSize: file.size,
    contentType: file.type,
    isPublic: false,
  }),
});

const { uploadUrl, cloud_storage_path } = await response.json();

// Upload directly to S3
await fetch(uploadUrl, {
  method: 'PUT',
  headers: { 'Content-Type': file.type },
  body: file,
});

// Complete the upload
await fetch('/api/upload/complete', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    cloud_storage_path,
    fileName: file.name,
    fileSize: file.size,
    mimeType: file.type,
    isPublic: false,
    encrypted: false,
    deleteAfterUse: false,
    retentionPeriod: '7_days',
  }),
});
```

Errors:
- `400` : Invalid input (missing fields, invalid file type)
- `401` : Unauthorized
- `500` : Server error

Complete Upload

Register file metadata after successful S3 upload.

Endpoint: `POST /api/upload/complete`

Authentication: Required

Request Body:

```json
{
  "cloud_storage_path": "clarity-razor/uploads/1735689000000-requirements.pdf",
  "fileName": "requirements.pdf",
  "originalName": "Project Requirements.pdf",
  "fileSize": 1048576,
  "mimeType": "application/pdf",
  "isPublic": false,
  "encrypted": true,
  "encryptionKey": "base64-encoded-metadata",
  "fileHash": "sha256-hash",
  "deleteAfterUse": false,
  "retentionPeriod": "7_days"
}
```

Parameters:
- `cloud_storage_path` (string, required): S3 key from presigned URL response
- `fileName` (string, required): Sanitized filename
- `originalName` (string, optional): Original filename
- `fileSize` (number, required): File size in bytes
- `mimeType` (string, required): MIME type
- `isPublic` (boolean, optional): Public access flag
- `encrypted` (boolean, optional): Whether file is encrypted
- `encryptionKey` (string, optional): Encryption metadata (salt + IV)
- `fileHash` (string, optional): SHA-256 hash for integrity
- `deleteAfterUse` (boolean, optional): Ephemeral mode
- `retentionPeriod` (enum, optional): `1_hour`, `24_hours`, `7_days`, `never`

Response (201 Created):

```json
{
  "id": "file_xyz789",
  "fileName": "requirements.pdf",
  "cloud_storage_path": "clarity-razor/uploads/1735689000000-requirements.pdf",
  "fileSize": 1048576,
  "encrypted": true,
  "deleteAfterUse": false,
  "expiresAt": "2026-01-07T23:30:00Z"
}
```

Errors:
- `400`: Invalid input or file not found in S3
- `401`: Unauthorized
- `500`: Server error

User Management

Export User Data (GDPR Article 15)

Export all user data in JSON format.

Endpoint: `GET /api/user/data-export`

Authentication: Required

Response (200 OK):

```json
{
  "exportedAt": "2025-12-31T23:30:00Z",
  "user": {
    "id": "user_abc123",
    "email": "john@doe.com",
    "name": "John Doe",
    "createdAt": "2025-01-01T00:00:00Z"
  },
  "tiles": [
    {
      "id": "tile_xyz789",
      "objective": "Launch product",
      "tags": ["product"],
      "createdAt": "2025-12-15T10:30:00Z"
    }
  ],
  "files": [
    {
      "id": "file_def456",
      "fileName": "requirements.pdf",
      "uploadedAt": "2025-12-15T10:25:00Z",
      "encrypted": true,
      "expiresAt": "2025-12-22T10:25:00Z"
    }
  ],
  "auditLogs": [
    {
      "action": "UPLOAD",
      "timestamp": "2025-12-15T10:25:00Z",
      "fileId": "file_def456",
      "metadata": {}
    }
  ]
}
```

Headers:
- `Content-Type: application/json`
- `Content-Disposition: attachment; filename="clarity-data-export-[timestamp].json"`

Errors:
- `401` : Unauthorized
- `500` : Server error

Bulk Delete User Data (GDPR Article 17)

Delete user files, tiles, and optionally the account.

Endpoint: `POST /api/user/bulk-delete`

Authentication: Required

Request Body:

```json
{
  "deleteFiles": true,
  "deleteTiles": true,
  "deleteAccount": false,
  "confirmation": "DELETE_MY_DATA"
}
```

Parameters:
- `deleteFiles` (boolean, required): Delete all user files
- `deleteTiles` (boolean, required): Delete all user tiles
- `deleteAccount` (boolean, optional): Delete user account (default: false)
- `confirmation` (string, required): Must be exact string `"DELETE_MY_DATA"`

Response (200 OK):

```json
{
  "success": true,
  "deletedFiles": 15,
  "deletedTiles": 23,
  "accountDeleted": false,
  "message": "Data deletion completed successfully"
}
```

Errors:
- `400` : Invalid confirmation string
- `401` : Unauthorized
- `500` : Server error

System Operations

Cleanup Expired Files

Run the automated cleanup job to delete expired files.

Endpoint: `POST /api/cleanup`

Authentication: API Key required

Headers:

```
Authorization: Bearer YOUR_CLEANUP_API_KEY
```

Response (200 OK):

```
{
  "success": true,
  "stats": {
    "filesProcessed": 42,
    "filesDeleted": 38,
    "filesFailed": 4,
    "errors": [
      {
        "fileId": "file_xyz",
        "error": "S3 deletion failed"
      }
    ]
  },
  "timestamp": "2025-12-31T02:00:00Z"
}
```

Errors:
- `401` : Missing or invalid API key
- `500` : Server error

Scheduling:
Run via cron job (recommended daily at 2 AM):

```
0 2 * * * curl -X POST https://your-domain.com/api/cleanup \
  -H "Authorization: Bearer ${CLEANUP_API_KEY}"
```

Get Cleanup Statistics

Get statistics about file retention and cleanup.

Endpoint: `GET /api/cleanup`

Authentication: API Key required

Headers:

```
Authorization: Bearer YOUR_CLEANUP_API_KEY
```

Response (200 OK):

```
{
  "totalFiles": 1523,
  "expiredFiles": 42,
  "ephemeralFiles": 15,
  "encryptedFiles": 378,
  "retentionBreakdown": {
    "1_hour": 5,
    "24_hours": 28,
    "7_days": 1234,
    "never": 256
  },
  "lastCleanup": "2025-12-31T02:00:00Z"
}
```

Errors:
- `401` : Missing or invalid API key
- `500` : Server error

Error Handling

Standard Error Response

All errors follow this format:

```
{
  "error": "Human-readable error message",
  "code": "ERROR_CODE",
  "details": {
    "field": "Additional context"
  }
}
```

HTTP Status Codes

Code	Meaning	Common Causes
200	OK	Successful request
201	Created	Resource created successfully
400	Bad Request	Invalid input, validation error
401	Unauthorized	Not logged in, invalid session
403	Forbidden	Insufficient permissions
404	Not Found	Resource doesn’t exist
409	Conflict	Resource already exists
413	Payload Too Large	File size exceeds limit
429	Too Many Requests	Rate limit exceeded
500	Internal Server Error	Server error, AI processing error
503	Service Unavailable	Temporary service disruption

Common Error Codes

- `INVALID_INPUT` : Request validation failed
- `UNAUTHORIZED` : Authentication required
- `FORBIDDEN` : Insufficient permissions

- `NOT_FOUND` : Resource not found
- `ALREADY_EXISTS` : Resource already exists
- `FILE_TOO_LARGE` : File exceeds size limit
- `ENCRYPTION_ERROR` : Client-side encryption failed
- `AI_PROCESSING_ERROR` : LLM generation failed
- `S3_ERROR` : S3 operation failed
- `DATABASE_ERROR` : Database operation failed

Rate Limits

⚠️ **Current Status**: Rate limiting not implemented

Recommended Limits (for production):

Endpoint	Limit	Window
`POST /api/generate-tile`	10 requests	1 minute
`POST /api/upload/*`	100 requests	1 minute
`GET /api/tiles`	1000 requests	1 minute
`POST /api/signup`	5 requests	1 hour
`POST /api/cleanup`	1 request	1 hour

Rate Limit Headers (to be implemented):

```
X-RateLimit-Limit: 10
X-RateLimit-Remaining: 7
X-RateLimit-Reset: 1735689600
```

Webhooks

⚠️ **Not currently implemented**

Planned Events:
- `tile.generated` - New tile created
- `file.uploaded` - File upload completed
- `file.expired` - File automatically deleted
- `user.deleted` - User account deleted

SDK / Client Libraries

⚠️ **Not currently available**

The API follows REST conventions and can be consumed with any HTTP client:

JavaScript/TypeScript:

```javascript
const response = await fetch('/api/generate-tile', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ objective: '...', mode: 'razor' }),
});

const tile = await response.json();
```

cURL:

```bash
curl -X POST http://localhost:3000/api/generate-tile \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=..." \
  -d '{"objective":"Launch product","mode":"razor"}'
```

Changelog

Version 1.0 (December 2025)

- Initial release
- Tile generation with 5 modes
- File upload with encryption
- GDPR compliance features
- Audit logging
- Automated cleanup

Last Updated: December 31, 2025
API Version: 1.0
Maintained By: Clarity Razor Team