# UroRads: Backend Specification

## Overview

The UroRads backend serves three functions: (1) Store and retrieve cases, (2) Store and retrieve chat messages, (3) Interface with OpenAI for explanation generation and chat responses. Hosted entirely on Replit using Replit's database and storage solutions.

## Architecture Overview

| Layer | Technology | Purpose |
|---|---|---|
| Frontend | React (PWA) | UI - served as static files |
| API | Node.js/Express or Python/Flask | REST endpoints (Replit decides) |
| Database | Replit DB / PostgreSQL | Case metadata, chat messages |
| File Storage | Replit Object Storage | CT images |
| LLM | OpenAI API | GPT-4o or GPT-4o-mini for vision + text |

## Data Flow

**Attending Upload Flow:**

1. Frontend captures image → sends to API
2. API stores image in Object Storage → gets imageUrl
3. API sends image + prompt to OpenAI → gets explanation
4. API returns explanation to frontend for review
5. On approve: API calls OpenAI for title + category
6. API saves Case record to database

**Learner View Flow:**

1. Frontend requests case (by ID or 'next')
2. API returns Case object + any existing chat messages
3. Frontend displays image + explanation + chat thread

**Learner Chat Flow:**

1. Frontend sends user message + caseId to API
2. API retrieves case (image, explanation) + chat history
3. API sends context + new question to OpenAI
4. API saves both user message and AI response to database
5. API returns AI response to frontend

## API Endpoints

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/cases | List all cases (for Archive). Returns: id, caseNumber, title, category |
| GET | /api/cases/:id | Get single case with full details + chat history |
| GET | /api/cases/next/:currentId | Get next case after currentId (for Next button) |
| POST | /api/cases/upload | Upload image, returns temporary imageUrl |
| POST | /api/cases/generate | Generate explanation from image + prompt |
| POST | /api/cases/publish | Approve and save case (triggers title + category generation) |
| POST | /api/chat | Send chat message, returns AI response |
| DELETE | /api/chat/:caseId | Clear chat history for a case (Clear Chat button) |

## Endpoint Details

### POST /api/cases/upload

Request: multipart/form-data with image file

```
// Request
Content-Type: multipart/form-data
Body: { image: <file> }

// Response
{
  "tempImageUrl": "/storage/temp/abc123.jpg",
  "tempImageId": "abc123"
}
```

### POST /api/cases/generate

Request: image reference + optional custom prompt

```
// Request
{
  "tempImageId": "abc123",
  "attendingPrompt": "Focus on the staghorn morphology"  // optional
}

// Response
{
  "explanation": "This axial CT image demonstrates...",
  "tempImageId": "abc123"
}
```

### POST /api/cases/publish

Request: final explanation (possibly edited) + image reference

```
// Request
{
  "tempImageId": "abc123",
  "explanation": "This axial CT image demonstrates...",
  "attendingPrompt": "Focus on the staghorn morphology"  // optional, for records
}

// Response
{
  "id": "case-uuid",
  "caseNumber": 15,
  "title": "Staghorn Calculus Left Kidney",
  "category": "Stones",
  "imageUrl": "/storage/cases/case-15.jpg",
  "explanation": "This axial CT image demonstrates...",
  "createdAt": "2025-01-15T10:30:00Z"
}
```

### POST /api/chat

Request: case ID + user message

```
// Request
```

```
{
  "caseId": "case-uuid",
  "message": "What would this look like on a KUB?"
}

// Response
{
  "id": "msg-uuid",
  "caseId": "case-uuid",
  "role": "ai",
  "content": "On a KUB (kidney-ureter-bladder) X-ray, this staghorn...",
  "createdAt": "2025-01-15T10:35:00Z"
}
```

## OpenAI Integration

Four distinct LLM calls using OpenAI's API. Use GPT-4o or GPT-4o-mini (vision-capable models).

| Call | Trigger | Input | Output |
|---|---|---|---|
| 1. Generate Explanation | Attending taps Generate | Image + prompt | Teaching explanation (2-3 paragraphs) |
| 2. Generate Title | Attending taps Approve | Explanation text | Short title (3-4 words) |
| 3. Generate Category | Attending taps Approve | Explanation text | Category label (1-2 words) |
| 4. Chat Response | Learner sends message | Image + explanation + chat history + question | Answer to question |

# LLM System Prompts

## 1. Generate Explanation

```
SYSTEM PROMPT:
You are a radiology teaching assistant for urology trainees.
Analyze this CT image and provide a teaching explanation.

Include:
1. What the image shows (anatomical orientation, structures visible)
2. Key finding identification (the pathology or abnormality)
3. Recognition features that help learners identify this in future
4. Relevant radiology first principles

Keep the explanation concise but educational (2-3 paragraphs).
Write for PGY-2 residents and new APPs learning uro-radiology.

{IF ATTENDING_PROMPT}
Additional guidance from the attending: {attendingPrompt}
{/IF}
```

## 2. Generate Title

```
SYSTEM PROMPT:
Based on this radiology case explanation, generate a short
descriptive title (3-4 words maximum).

Format: [Pathology] [Location/Qualifier]
Examples: "Staghorn Calculus Left Kidney", "Grade 3 Hydronephrosis",
"Renal Cell Carcinoma Upper Pole"

Return ONLY the title, no other text.
```

## 3. Generate Category

```
SYSTEM PROMPT:
Based on this radiology case explanation, assign ONE category
from this list:

- Stones
- Hydronephrosis
- Mass/Tumor
- Infection
- Trauma
- Congenital
- Vascular
- Bladder
- Prostate
- Other

Return ONLY the category name, no other text.
```

## 4. Chat Response

```
SYSTEM PROMPT:
You are a radiology teaching assistant. The learner is viewing
a uro-radiology case and has a follow-up question.

Context provided:
```

```
- The CT image they are viewing
- The teaching explanation for this case
- Previous chat messages (if any)

Answer their question in a helpful, educational manner.
Stay focused on the specific case and radiology concepts.
If they ask something unrelated to the case, gently redirect.
```

## Database Schema (Conceptual)

Let Replit choose the specific database. These are the required tables/collections:

### Table: cases

```
id              STRING PRIMARY KEY
case_number     INTEGER UNIQUE AUTO-INCREMENT
title           STRING
image_url       STRING
explanation     TEXT
category        STRING
attending_prompt  STRING NULLABLE
created_at      TIMESTAMP
```

### Table: chat_messages

```
id              STRING PRIMARY KEY
case_id         STRING FOREIGN KEY -> cases.id
role            STRING ('user' | 'ai')
content         TEXT
created_at      TIMESTAMP

INDEX on case_id for fast retrieval
```

## Environment Variables

| Variable | Description |
|---|---|
| OPENAI_API_KEY | OpenAI API key for GPT-4o access |
| OPENAI_MODEL | Model name (e.g., "gpt-4o" or "gpt-4o-mini") |

## Error Handling

| Error | HTTP Code | Response |
|---|---|---|
| Image upload fails | 400 | {"error": "Upload failed", "message": "..."} |
| OpenAI API error | 502 | {"error": "Generation failed", "message": "..."} |
| Case not found | 404 | {"error": "Not found", "message": "Case does not exist"} |
| Invalid request | 400 | {"error": "Bad request", "message": "..."} |

*Document 5 of N — UroRads Technical Documentation*