Knowledge Goal Demonstration Document

Helektron Study Assistant - Capstone Project

[KG1] Endpoint Definitions

File Reference: main.py:584-597

Code Fragment:

```python
@app.get("/", response_class=HTMLResponse, tags=["UI"])
def index(request: Request):
    """Main page (KG1 - Endpoint Definition, KG8 - UI Endpoint)."""
    return templates.TemplateResponse("index.html", {"request": request})

@app.post("/upload", response_class=HTMLResponse, tags=["UI"])
async def upload_material(
    request: Request,
    file: UploadFile = File(...),
    session_id: Optional[str] = Form(None),
    session_repo: dict = Depends(get_session_repo),
    file_service: dict = Depends(get_file_service),
    rag_service: dict = Depends(get_rag_service),
):
    ...

@app.delete("/session/{session_id}/file/{file_index}",response_class=HTMLRespons
e, tags=["UI"])
@app.get("/api/transcript/{session_id}",response_model=TranscriptResponse,
tags=["API"])
@app.put("/api/session/{session_id}", tags=["API"])
@app.delete("/api/session/{session_id}", response_model=DeleteResponse,
tags=["API"])
```

Justification:
The application defines 14 unique endpoint URL paths using FastAPI route decorators including `@app.get("/")`, `@app.post("/upload")`,

`@app.delete("/session/{session_id}/file/{file_index}")`,
and other `/api/*` endpoints. Each endpoint has a unique URL path that maps to a specific handler function.

[KG2] HTTP Methods & Status Codes

File Reference: main.py:872-882

Code Fragment:

```python
@app.delete("/api/session/{session_id}", response_model=DeleteResponse,
tags=["API"])
def delete_session_api(session_id: str):
    """Delete session (KG1, KG2-DELETE, KG6-Delete, KG7-JSON)."""
    if not validate_session_id(session_id):
        raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST,
                            detail="Invalid session ID")

    if not delete_session(session_id):
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail="Session not found")

    clear_vector_store(session_id)
    return DeleteResponse(success=True, message="Session deleted",
                          deleted_item=session_id)
```

Justification:
This endpoint shows HTTP method usage with `@app.delete` for resource deletion. It returns appropriate status codes:

- **400 Bad Request** for invalid input
- **404 Not Found** when the session doesn't exist
- **200 OK** on success

The application uses GET, POST, PUT, and DELETE throughout with explicit status code handling.

[KG3] Endpoint Validation

File Reference: main.py:62-74

Code Fragment:

```python
def validate_file_extension(filename: str) -> str:
    """Validate file extension (KG3)."""
    if not filename or '.' not in filename:
        raise ValueError("Invalid filename")
    ext = filename.rsplit('.', 1)[-1].lower()
    if ext not in ALLOWED_EXTENSIONS:
        raise ValueError(f"File type .{ext} not allowed. Allowed:
{ALLOWED_EXTENSIONS}")
    return ext

def validate_session_id(session_id: str) -> bool:
    """Validate UUID format (KG3)."""
    pattern = re.compile(
        r'^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$',
re.I)
    return bool(pattern.match(session_id))
```

Justification:
The app includes clear input validation steps to make sure it only processes safe and expected data. The `validate_file_extension()` function checks that users upload approved file types, while `validate_session_id()` uses a regular expression to confirm that session IDs follow the proper UUID format. These checks happen before any data is handled further, allowing the system to catch and reject invalid inputs early with helpful error messages.

[KG4] Dependency Injection

File Reference: main.py:538-548, 589-597

Code Fragment:

```python
def get_session_repo():
    """Dependency that provides session repository functions (KG4)."""
    return {
```

```
        "load": load_sessions,
        "save": save_sessions,
        "get": get_session,
        "create": create_session,
        "update": update_session,
        "delete": delete_session,
        "get_or_create": get_or_create_session,
    }

@app.post("/upload", response_class=HTMLResponse, tags=["UI"])
async def upload_material(
    request: Request,
    file: UploadFile = File(...),
    session_id: Optional[str] = Form(None),
    session_repo: dict = Depends(get_session_repo),
    file_service: dict = Depends(get_file_service),
    rag_service: dict = Depends(get_rag_service),
):
```

Justification:
This part of the code uses FastAPI's `Depends()` feature to pull in the session repository, file-handling tools, and RAG functions whenever an endpoint needs them. Because these pieces are injected instead of being created inside the route, the route logic stays clean and doesn't depend on any specific implementation. It also makes the app much easier to test and swap out components later without rewriting everything.

[KG5] Data Model

File Reference: main.py:26-43

Code Fragment:

Python
```
class FileEntry(BaseModel):
    """Schema for an uploaded file entry."""
    name: str = Field(..., min_length=1, max_length=255)
    type: str
    text: str = ""
```

```
    added_at: str = Field(default_factory=lambda:
datetime.utcnow().isoformat())

class Session(BaseModel):
    """Schema for a study session."""
    id: str = Field(default_factory=lambda: str(uuid.uuid4()))
    name: Optional[str] = None
    created_at: str = Field(default_factory=lambda:
datetime.utcnow().isoformat())
    updated_at: str = Field(default_factory=lambda:
datetime.utcnow().isoformat())
    files: List[FileEntry] = Field(default_factory=list)

class SessionUpdateRequest(BaseModel):
    """Request schema for updating a session (KG5)."""
    name: Optional[str] = Field(None, min_length=1, max_length=100)
```

Justification:
The data structures for the app are defined using Pydantic models. Classes like `Session`, `FileEntry`, and `TranscriptResponse` describe exactly what fields exist, what types they should be, and how they're formatted. This gives the rest of the application a consistent, predictable way to work with stored files, transcripts, and sessions.

[KG6] CRUD Operations & Persistent Data

File Reference: main.py:105-151

Code Fragment:

Python
```
def load_sessions() -> Dict[str, Any]:
    """Load sessions from JSON file (KG6 - Read)."""
    if not os.path.exists(SESSIONS_PATH):
        return {}
    with open(SESSIONS_PATH, "r") as f:
        try:
            return json.load(f)
        except json.JSONDecodeError:
            return {}
```

```python
def create_session() -> Session:
    """Create a new session (KG6 - Create)."""
    session = Session()
    sessions = load_sessions()
    sessions[session.id] = session.model_dump()
    save_sessions(sessions)
    return session

def update_session(session: Session) -> Session:
    """Update a session (KG6 - Update)."""
    sessions = load_sessions()
    session.updated_at = datetime.utcnow().isoformat()
    sessions[session.id] = session.model_dump()
    save_sessions(sessions)
    return session

def delete_session(session_id: str) -> bool:
    """Delete a session (KG6 - Delete)."""
    sessions = load_sessions()
    if session_id not in sessions:
        return False
    del sessions[session_id]
    save_sessions(sessions)
    return True
```

Justification:
The app handles creating, reading, updating, and deleting sessions and uploaded files through a small set of focused helper functions. All session data is saved to a JSON file, so everything persists even if the server restarts. This setup covers the full CRUD cycle: new sessions are created, existing ones are loaded and updated, and old ones can be deleted along with their data.

[KG7] API Endpoints & JSON

File Reference: main.py:817–850

Code Fragment:

Python

```python
@app.get("/api/session/{session_id}", tags=["API"])
def get_session_api(session_id: str):
    """Get session details as JSON (KG1, KG2-GET, KG7-JSON)."""
    if not validate_session_id(session_id):
        raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST,
                            detail="Invalid session ID")

    session = get_session(session_id)
    if not session:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail="Session not found")

    return JSONResponse(status_code=status.HTTP_200_OK, content={
        "id": session.id,
        "name": session.name,
        "created_at": session.created_at,
        "updated_at": session.updated_at,
        "file_count": len(session.files)
    })
```

Justification:
The project exposes several JSON-based API routes under `/api/…`. These endpoints return structured JSON responses that other applications can use. They also use HTTP response codes to signal whether requests succeeded, failed validation, or couldn't find the requested resource.

[KG8] UI Endpoints & HTMX

File Reference: templates/fragments/upload_status.html:14–23

Code Fragment:

HTML

```html
<!-- Delete Button (KG8 - HTMX, KG9 - User Delete Interaction) -->
<button
    class="delete-btn"
    hx-delete="/session/{{ session_id }}/file/{{ loop.index0 }}"
    hx-target="#materials-panel"
    hx-swap="outerHTML"
```

```
    hx-confirm="Delete '{{ f.name }}'?"
    title="Delete file">
    *tarsh can emoji*
</button>
```

Justification:
This demonstrates how HTMX is used to send a `DELETE` request and instantly update the interface. When the button is clicked, it makes the server call and then refreshes just the affected part of the page. This makes the interaction feel smoother and more responsive.

[KG9] User Interaction (CRUD)

File Reference: templates/index.html:19–30

Code Fragment:

```HTML
<form
    hx-post="/upload"
    hx-target="#materials-panel"
    hx-swap="outerHTML"
    hx-indicator="#upload-loading"
    enctype="multipart/form-data">
    <input type="file" name="file" required
           accept=".txt,.pdf,.pptx,.mp4,.m4a,.wav,.webm">
    <input type="hidden" name="session_id" id="session_id_input">
    <br><br>
    <button class="primary" type="submit">Upload</button>
</form>
```

Justification:
Users interact with the app's full CRUD flow through the interface. Uploading a file adds it to the session, the materials panel shows everything that's been saved, and HTMX delete buttons let users remove items instantly.

[KG10] Separation of Concerns

File Reference: main.py:38–41, 93–95, 118–121, 206–208, 347–349, 467–469, 552–555, 599–601, 832–834

Code Fragment:

```
# ============================================================================
# SECTION 1: DATA MODELS (KG5 - Data Model)
# ============================================================================

# ============================================================================
# SECTION 2: CONFIGURATION
# ============================================================================

#============================================================================
# SECTION 3: DATA ACCESS LAYER (KG6 - CRUD Operations)
# ============================================================================

# ============================================================================
# SECTION 4: RAG SERVICE (Vector Store for Retrieval-Augmented
Generation)
# ============================================================================

# ============================================================================
# SECTION 5: FILE EXTRACTION SERVICE (KG10 - Separation of Concerns)
# ============================================================================

# ============================================================================
# SECTION 6: AI SERVICE - OLLAMA (KG10 - Separation of Concerns)
# ============================================================================

# ============================================================================
# SECTION 7: DEPENDENCY INJECTION (KG4)
# ============================================================================

# ============================================================================
# SECTION 8: UI ENDPOINTS (KG1, KG8 - UI Endpoints & HTMX)
# ============================================================================

# ============================================================================
```

```
# SECTION 9: API ENDPOINTS (KG1, KG7 - API Endpoints & JSON)
# ================================================================
```

Justification:
The application is divided into clear sections: data modeling, configuration, CRUD layer, RAG logic, file extraction, AI service, dependency injection, UI routes, and API routes. This allows for strict separation of concerns and makes the system maintainable (and more readable).