Oregon State University
Ecampus

CS361: Assignment 9: Sprint 3 Plan (for Milestone #3)

Overview

Plan Microservices B, C, and D you'll make for your main program to call.

Instructions

Complete each item below by replacing the highlighted text (**Usability note**: double-click the text to select it).

1) What is your **Sprint Goal**? (e.g., fully implement the spell-checker, grammar checker, email address validation microservices) The Sprint Goal must clearly communicate what each of the B, C, and D microservices will do.

Fully implement Microservice B (Audio Streaming), Microservice C (File Deletion), and Microservice D (File Preview) so that users can play uploaded audio in browser, remove unwanted files, and preview PDFs and images without downloading them.

2) Define **at least two user stories** for each microservice (B, C, and D). Provide your user stories and their functional and non-functional acceptance criteria (and associated quality attributes).

   Requirements for *each* microservice:

   - You must implement at least two user stories for each microservice.
   - Each user story must have a name.
   - Each user story must use the "As a… I want to... so that…" format.
   - Each user story must have at least one functional acceptance criterion.
   - All functional acceptance criteria must use the "Given… when… then…" format.

   Requirements for the *set of* microservices:

   - At least three different quality attributes must appear at least once on a user story's "back of index card".
   - Each quality attribute must be converted to a non-functional requirement.
   - All non-functional requirements must be testable.

Microservice B:

First user story

(Front of index card)

Stream Audio Playback

As a user, **I want to** play my uploaded audio files directly in the browser **so that** I don't have to download them first.

(Back of index card)

Acceptance criteria

Functional requirements

- Given a valid audio file ID, when the user navigates to /api/stream/{fileId}, then the service streams the audio with correct Content-Type: audio/* headers.

Quality attributes & Non-functional requirements

- **Performance:** 95% of initial audio stream requests must start sending data within 100 ms.
- **Reliability:** The stream must not drop or stall for more than 1 second during playback (measured under normal network conditions).

Second user story

(Front of index card)

Support Byte-Range Requests

As a user, **I want to** seek within an audio track **so that** I can jump to any point without reloading.

(Back of index card)

Acceptance criteria

Functional requirements

- Given a Range: bytes=start- header in the request to /api/stream/{fileId}, when the service receives it, then it responds with 206 Partial Content and the requested byte range.

Quality attributes & Non-functional requirements

- Scalability: The service must handle 200 concurrent range-request streams without error.
- Maintainability: Range-handling logic must be implemented in a single module under ten lines and covered by unit tests.

Microservice C:

First user story

(Front of index card)

Delete File

As a user, **I want to** delete a file I previously uploaded **so that** I can manage storage and privacy.

(Back of index card)

Acceptance criteria

Functional requirements

- Given a valid file ID and authenticated user, when they send DELETE /api/files/{fileId}, then the service removes the file from disk and its metadata from the database, returning HTTP 204.

Quality attributes & Non-functional requirements

- **Reliability:** 99.9% of delete requests must succeed under normal load.
- **Consistency:** After deletion, subsequent GET /api/files/{fileId} must return HTTP 404 (verifiable via integration tests).

Second user story

(Front of index card)

Handle Non-existent File Deletion

As a user, **I want to** get an error if I try to delete a file that doesn't exist **so that** I know whether my request succeeded.

(Back of index card)

Acceptance criteria

Functional requirements

- Given an invalid or already-deleted file ID, when DELETE /api/files/{fileId} is called, then the service returns HTTP 404 with JSON { error: "Not found" }.

Quality attributes & Non-functional requirements

- **Security:** The service must not reveal any internal paths or stack traces in its error response (verified by code review).
- **Performance:** 95% of 404 delete responses must be sent within 50 ms.

Microservice D:

First user story

(Front of index card)

Preview Document Snippet

As a user, **I want to** preview the first page of an uploaded PDF **so that** I can decide if it's the right document.

(Back of index card)

Acceptance criteria

Functional requirements

- Given a PDF file ID, when the user requests GET /api/preview/pdf/{fileId}, then the service returns a PNG image of page 1 at 800×600 resolution.

Quality attributes & Non-functional requirements

- **Performance:** 90% of PDF previews must be generated within 300 ms.
- **Scalability:** The service must handle 50 concurrent preview generations without failures.

Second user story

(Front of index card)

Preview Image Thumbnail

As a user, **I want to** see a full-sized preview of an uploaded image **so that** I can inspect details before downloading.

(Back of index card)

Acceptance criteria

Functional requirements

- *Given an image file ID, when the user requests GET /api/preview/image/{fileId}, then the service returns the original image scaled to a maximum width of 1200 px (preserving aspect ratio) with Content-Type matching the image type.*

Quality attributes & Non-functional requirements

- **Reliability:** 99% of image previews must match the original aspect ratio and format.
- **Maintainability:** Scaling logic must reside in a single, well-documented utility function under ten lines.

3) What kind of **communication pipe** will each microservice use? (e.g., text files, REST API)
Note: You can use the same type of communication pipe for all three microservices or different types.

All three of the microservices will use RESTful HTTP endpoints that accept and return JSON or binary content.

This would be a good time to make a new repository for each of your microservices.

Submission

Upload a document in PDF or Word format via Canvas.

Grading

You are responsible for satisfying all criteria listed in the Canvas rubric for this assignment.

Questions?

Please ask via Ed so that others can benefit from the answer.