1. Summary Project Stack

Layer	Technology
Frontend	React.js (⚠️To discuss)
Backend	ASP.NET Core with C#
Microservices	REST APIs, gRPC, RabbitMQ for messaging
Database	PostgreSQL
Deployment	Docker, Kubernetes
DevOps	GitHub Actions, Prometheus
Communication	REST (external), RabbitMQ (internal) (⚠️To discuss)

To look into:

- https://dotnet.microsoft.com/en-us/apps/aspnet

Main Architecture:

MVC:

-

Secondary Architecture:

Microservices:

- LLM
- Anki cards
- Text extractor from pdf
- Backend

Process:

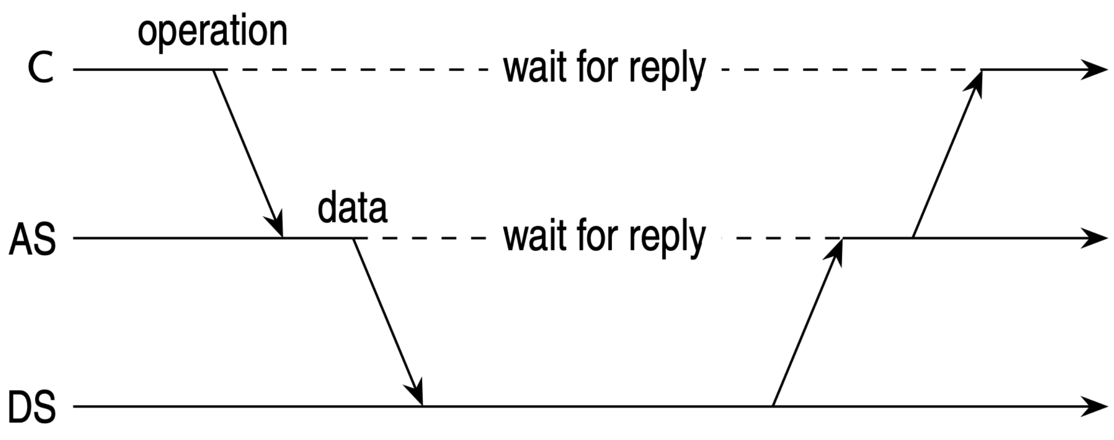
Workflow

1. User uploads a file to the **Controller Layer**.
2. **Controller Layer** forwards the file to the **Text Extraction Microservice**.
3. **Text Extraction Microservice** returns the extracted text to the Controller.
4. **Controller Layer** sends the text to the **LLM Microservice** for card generation.
5. Generated cards are sent to the **Anki Cards Microservice** for storage.

Backend:

- Spring boot

Layered Architecture:



Layer 1:

- View / Client

Layer 2:

- Controllers

Layer 3:

- Model
  - Microservices
  - Databases
  -

Internal service-to-service communication)

RESTful API:

Django:

Django will be used as a Gateway or API Layer to expose HTTP-based REST APIs or even GraphQL endpoints as part of the microservices ecosystem. It can serve as the front-facing layer that handles client requests, while it communicates with other microservices (implemented via gRPC) for business logic, processing, or data fetching.

Django could implement one or more microservices, handling certain aspects of the business logic, for example:

- **Authentication service**: A Django-based microservice could handle authentication, user management, and authorization using Django's built-in features.
- **User interface**: Django can serve HTML pages or templates as part of the web app while interacting with other microservices through gRPC or HTTP APIs.
- **Admin Panel**: Django provides a powerful admin interface, which can be useful for managing and monitoring services.

gRPC:

gRPC will be used for Internal Communication, instead of using HTTP between services, we can use gRPC to handle communication between Django and other microservices. For example:

  - **Django** can act as a client to call gRPC services for complex processing or interactions with external services.
  - Django's views or API endpoints can communicate with internal gRPC services using the gRPC client.

Django + gRPC Integration

Django doesn't have built-in support for gRPC, but we can integrate it by using a Python gRPC client within your Django views, models, or custom management commands.

Implementation:

gRPC

Cluster Management:

Use either Apache Spark or Hadoop (research still needed to be done)