Project Plan

OpenVidStreamer

by Radoslav Radev

Advanced Software Engineering Semester 6

26/02/2024

ver. 1

Document Versioning

Date	**26/02/2024**
Version	V1
State	Defined (Initial Draft)
Author	Radoslav Radev

Contents

1. Introduction

1.1 General Information

This is the Project Plan for my individual project in Semester 6

The project aims to create an open video streaming platform with it`s own economy.

The final product will include a React frontend and a scalable backend system consisting of multiple microservices.

This Project will fulfill:

- *Building a scalable backend system*
- *making scalable design choices, with the appropriate technologies*
- *Showing professional skills and manners*
- *Deploy the project in a cloud environment*

*More about **Learning Outcomes**:*
https://fhict.instructure.com/courses/13669/outcomes

1.2 About the “Client”

For this project, I will be both the developer and the client, since I will be putting my business idea into realization

2. Project Statement

2.1 Stakeholders

Formal Client:

Radoslav Radev - 491245@student.fontys.nl

Tutor (technical teachers):

Maja Pesic – m.pesic@fontys.nl

Xuemei Pu - x.pu@fontys.nl

Developer:

Radoslav Radev – 491245@student.fontys.nl

2.2 Problem Description

OpenVidStreamer is a video sharing platform operating with its own unique economy, offering a monthly subscription model for users to access content. To incentivize content creation, the platform rewards users who upload videos by monetarily compensating them based on the watch time their uploads accumulate from other viewers. This system encourages active participation and content generation, enhancing the platform's value and diversity.

3. Project Objectives

3.1 Project Approach

The entire project (paperwork and software) will be completed using Agile methodology.

The semester's nature allows for the approach to be divided into a number of sub-goals:

- *Backend Architecture*
- *Development of the Backend system*
- *design and development of the frontend SPA*
- *Paperwork*
- *scalability integration*
- *Testing (functionality back end, front end)*
- *Quality Assurance*
- *Performance testing*
- *Cloud Deployment*

Agile methodology:

"a software development and project management method that is iterative and aids teams in delivering value to clients more quickly and with fewer road bumps"

Project Goal

The goal of this project is to develop a scalable video sharing platform that can handle enormous demand (half a million users)

3.3 Deliverables & Non-Deliverables

Deliverables:

- *Full stack web application*
- *Project Plan*
- *Design Document (Technical Document)*

Non-Deliverables:

-

3.4 Project Constrains

Constraint	Description
Time Constraint	The deadline for the final versioning of the project is 23/06/2024 23:59
Technology	The technology stack: .NET Aspire, RabbitMQ, Ocelot, Consul, Kubernetes, React+Vite, MySQL, Redis

3.5 Project Risks

Risk	Probability	Impact	Migration
Technical Difficulties	High	Medium	Research and analyze for a technical stack. Apply best practices and refer to the documentation.
Misunderstanding the requirements and deliverables	Medium	High	Document the requirements. Define clear deliverables.
Scope creep	Medium	Medium	Clear project schedule. Verify the scope with the stakeholders.
Poor time management	Low	High	Prioritize important tasks.

4. Requirements Breakdown

4.1 Application Functional Requirements

The program requirements are the requirements regarding the functions and workflow of the project.

The application MUST:

- *Allow users (customers) to watch videos uploaded to the platform*
- *Allow users (customers) to upload videos to the platform*

- *Provide users (customers) with recommendations on videos to watch*
- *Provide users (customers) with a payment portal for their monthly subscription*

4.2 Non Functional Requirements

- **User experience:**
  - The platform must achieve a score of at least 85/100 on the System Usability Scale (SUS), ensuring it is considered intuitive by the majority of its users.
  - Page load times should not exceed 2 seconds on 80% of occasions, as measured across a representative user base and set of actions.
- **Security:**
  - The platform must comply with the latest version of the OWASP Top 10 Application Security Risks, with no known vulnerabilities of medium severity or higher.
  - Implement Token based Authentication for all user accounts and secure all data transmissions with TLS 1.3.
- **scalability:**
  - The architecture must support horizontal scaling, allowing for the addition of resources (e.g., servers) to handle at least 300,000 concurrent users without degradation of performance beyond 10%.
  - Implement auto-scaling policies that automatically adjust resources based on real-time demand, ensuring response times remain below 3 seconds for 95% of requests, even during peak usage.

5. Research Questions

Main Research Question:

What are the optimal design choices for implementing MicroServices in a .NET-based open video sharing platform, similar to YouTube ?

Methodology (Design Phase): The study employs the DOT framework, focusing on the Design aspect to evaluate architectural components and design decisions crucial for developing efficient MicroServices in .NET.

Sub-Questions:

- **API Gateway Selection**
    - **Question:** Which API Gateway is most suited for .NET Core applications?
    - *Methodology (Design Phase):* Performance, flexibility, and integration capabilities of Ocelot and YARP are analyzed.
- **Message Broker Usage**
    - **Question:** How do RabbitMQ and Apache Kafka compare for MicroServices communication?
    - *Methodology (Design Phase):* Flexibility, integration, community support, and use case appropriateness are compared.
- **Communication Protocols**
    - **Question:** What are the advantages of choosing between non-brokered and brokered communication for MicroServices?
    - *Methodology (Design Phase):* The study explores gRPC, SignalR, and RabbitMQ, focusing on their integration within a .NET environment.
- **Service Discovery and Management**
    - **Question:** Which offers better support for MicroServices discovery and management in .NET applications, Consul or Eureka?
    - *Methodology (Design Phase):* Integration features and support for both Consul and Eureka are examined.

6. Project Phasing

5.1 Work Division

Sprints are the smaller iterations that make up the semester, as was already explained. Three weeks are spent on each sprint. New features will be added to the project after each sprint, and project deliverables will be shown to the stakeholders.

Sprint I (Initial phase):

Activities:

- *All around **research** and planning of the project*
- *Initial draft of the **Project Plan***
- *Initial draft of the **Technical Document***
- *Design and architecture of the backend system*
-

Deliverables:

- *Initial draft of the Project Plan/Technical Document*
- *Design diagrams*

7. User story Gantt chart

A chart regarding time estimation of features

Glossary

Project Plan – *"A project plan, according to the Project Management Body of Knowledge is a formal, approved document used to guide both project execution and project control"*

Stakeholder – *"A person with interest or concern in something, especially business"*

Agile methodology – "*An iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches****"***

Full-Stack* – *"In software development, it refers to both client-side and server-side parts of the application"

***Technical Document* – "**A document that contains architectural information of the application as well as information such as features or user stories"

***Git* – "**Free open-source version control system used by developers for work management**"**

API* – "Application programming interface, way for two programs to communicate with each other***"**

***REST API* – "**Representational state transfer, an application programing interface (**API**)"

***JavaScript* –** "*Programing language mainly used for building front end web applications*"

***React* –** *"JavaScript framework"*

***Database* –** *"Organized collection of data stored and accessed electronically"*

***ORM* –** *"Object relational mapper, used to make database connection easier"*

Integration testing – *“A phase in software testing in which individual software modules are combined and tested as a whole”*

Unit testing – *“A phase in software testing in which individual modules are tested separately”*