Software Requirements Specification

for

PDF-Editor

Version 0.7 approved, production

Prepared by Dipankar Das

26-07-2022 (LAST UPDATED)

https://pdf-web-editor.azurewebsites.net/

Table of Contents

Changelog

Name	Date	Remarks	Version
Init build	01/4/22	● Ability to Merge 2 PDFs ● can also download the Merged PDF for docker image: `dipugodocker/pdf-editor:0.1v`	0.1v
Simple web app with frontend and backend	26/4/22	● Test cases, simple frontend provided ● Added K8s, Terraform, ArgoCD, Helm, Kustomize, Docker-compose	0.5v
Microservice arch. achieved	7/6/22	● Divided frontend and backend into separate docker container ● Added monitoring tools	0.6v
New UI/UX for Frontend	21/7/22	● Express js with bootstrap for Frontend ● Unexposed the backend	0.7v

Introduction

Purpose

PDF-Editor is meant to Edit PDFs* in a fast and easy way possible, bypassing any unnecessary steps that may delay the user's experience.

* Currently only the Merge feature is provided

Project Scope

To provide a web-based application having day-to-day pdf-tools like merging, rotation, and more. It eliminates the need for installing applications on your computer, thus it works on any device be it mobile or laptop.

This project also helps in understanding microservice applications and how they communicate with each other in Docker and Kubernetes.

Github Repo link

Requirements

Functional

- Inputs: PDF files
- Output: Merged PDF file
- User has to provide 2 pdfs one after the other the order of **upload does matter**
- Then clicking Download will download the merged PDF to the client's side

Non-Functional

- Each PDF file size should be **below 10MB**
- PDF is stored in the **frontend** then using *HTTP POST* to upload it to **backend**
- All the PDFs are stored in the local file storage
- Once the second file is stored in the backend it automatically triggers merge and is available for download
- Thus **clear** becomes necessary

Life Cycle Model used

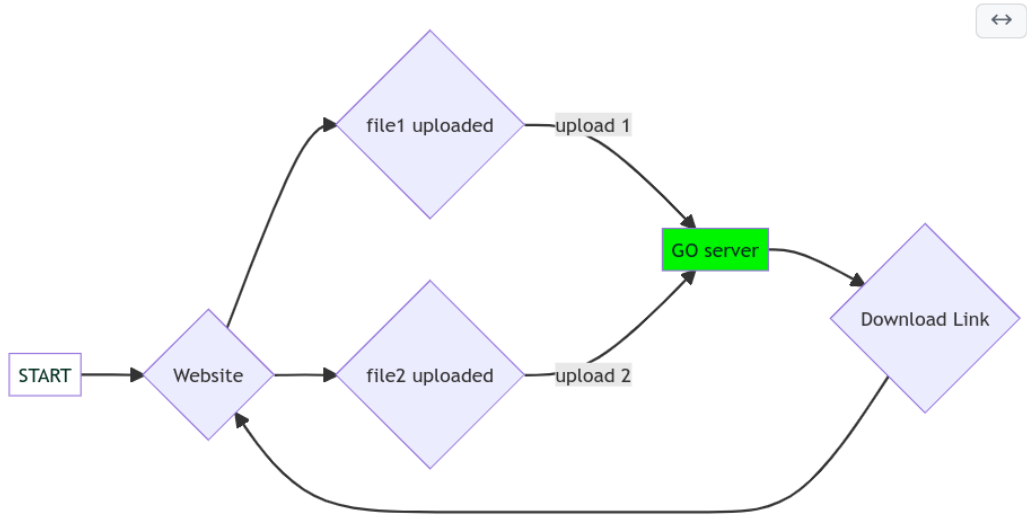
- Continuous Integration with Continuous Staging(current beta) and manual Delivery to the production
- Agile Approach is used every code changes is tested and pushed to DockerHub

Intended Audience and reading suggestions

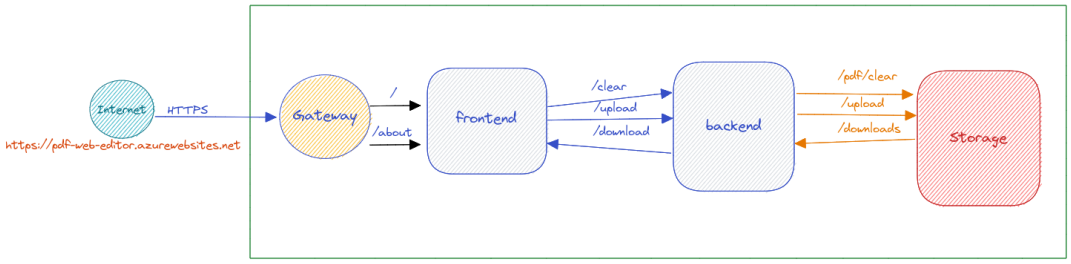
It is useful for the analyst(s) and programmer(s) that will write and subsequently implement the requirements stated on this file, as well as to people that may happen to join the team midway development.

Overall Description

Control Flow diagram of the web server



Detailed Network traffic *



* subjected to change

Product Functions

- homePage(..)
  Home page or the index page for the frontend

- uploadFile1(*)
  Upload file 1 then file 2 to the backend and in turn saved to the storage

- download(*)
  Download the automatically merged PDF

- clearUploads(*)
  Delete the PDF Uploaded and also the merged one as well

Operating System

Developer perspective

- Any system that has Docker installed

User perspective

- Any System which is connected to the internet

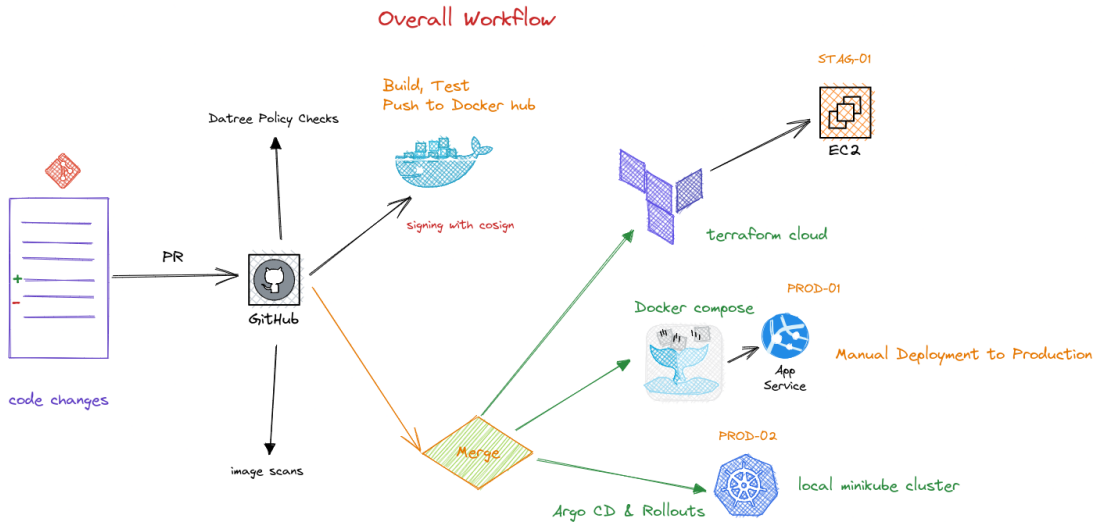
Design and implementation constraints

Browser apps can burn through hardware capabilities fast. The app needs to be lightweight.

Tech Stack

- **Go** for the backend
- **Javascript** for the frontend
- **Docker** for container management
- **Kubernetes** for container orchestration
- **AWS EC2** for the latest application (***main*** branch)
- **Azure Web app** for the deployment of production
- **ArgoCD & Rollouts** for Continuous Delivery
- **Prometheus, Grafana** for monitoring
- **Terraform** for the EC2 IAC
- **Tracing** using Jaeger

Overall Workflows (DevOps)

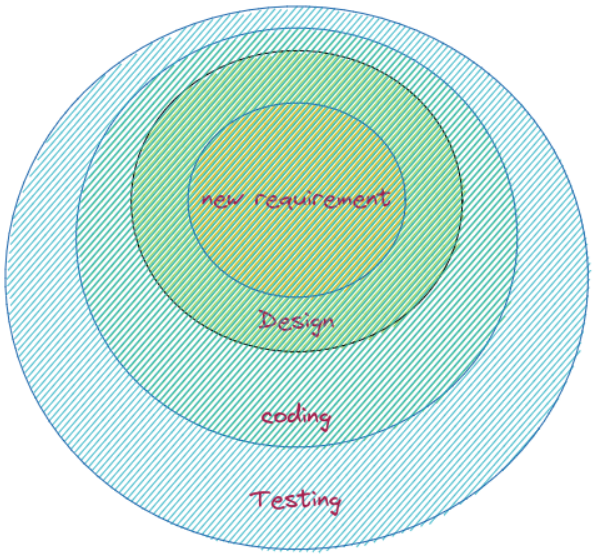


Documentation

Getting started

- Upload the file using the file selector in home page, then click upload
  Do follow the suggestions in the web page itself. Order of uploads does matter
- After doing for 2 files, then click the Download button to download the file
- As a good measure do **clear!**

Development cycle



Code using Go and Javascript
Test using Jest and go standard library
Hosting Azure Web apps, AWS EC2 server
Continuous Testing, Integration, Staging and deployment using canary rollouts

Development Cycle = ~1 month

Software Quality Attributes

- Availability: it should be 24/7
- Correctness: correctly merge 2 pdf's

Current Work

1. No 2 people can see each other's uploaded PDF Issue Link
2. More than 2 PDFs merge
3. Rotate PDF’s
4. Write more test cases