PRÜFUNGSAMT
IU.DE

INTERNATIONALE
HOCHSCHULE

PORTFOLIO

Assignments for the course: Data Engineering (DLMDSEDE02)

TABLE OF CONTENTS

1. TOPICS AND TASKS

Data volumes and data availability keep increasing rapidly, allowing for more sophisticated applications to generate more insightful outcomes. While this can contribute to more productive work processes and better understanding of phenomena and systems, at the same time this confronts Data Engineers with the challenge of handling vast amounts of data in very short amount of time. What is more, as data becomes more business critical in most organizations, the requirements for data security, governance and protection become more central considerations. Furthermore, the need for data-intensive applications to be robust, reliable, scalable, and maintainable increases.

In this portfolio project, you will design and implement a data architecture for a data-intensive large-scale application. While doing so, you will adapt principles such as Infrastructure as Code (IaC) and Microservices (MS). At the end of this project, you will verifiably have obtained and practised the data engineering skills necessary to build the backend of data-intensive applications while using state-of-the-art concepts and methods.

Within the framework of this course, one of the following topics must be selected. Given the nature of canonical data pro-cessing architectures, the two topics to choose from are similar but are differentiated by the requirement to process data in batches (topic 1) or as real-time streams (topic 2).

1.1. Task 1: Build a batch-processing-based data architecture for a data-intensive application

As the backend of a data-intensive machine learning application, you design, build, and provide the underlying data infra-structure. You will build a system which is able to ingest massive amounts of data, store the data effectively, pre-process parts of it and aggregate the data for direct usage in a machine learning application (which is not covered by this project). As the frontend application will run once per quarter to generate the new versions of a machine learning model, you will design the data infrastructure in a way that data processing is scheduled to be conducted in batches. During your journey to a state-of-the-art data processing architecture, you will be able to adapt common data engineering principles. You will also be acquainted with canonical software components and frameworks.

Task: Implementation of a reliable, scalable and maintainable batch-processing data system.

Implement this system in the following 3 phases:

1.1.1. Conception phase

This part of the process is the most important. Anything that is overlooked or forgotten in this phase has a negative effect on the implementation later and will lead, in the worst case, to useless results.

The first step is to **familiarize yourself with common software components** to build a data-intensive application, for example Apache Kafka, Hadoop or Spark. You do not have to know all details of each component, but you should be able to associate each software component with a specific task within your system. At this stage, you should also consider as to how you will implement techniques which **ensure reliability, scalability, and maintainability** of your system. Furthermore, you should reason how you will **ensure data security, governance, and protection** of your system.

Right from the start, you should keep in mind, that everything you design and build should be reproducible. Therefore, while choosing your working environment, you should consider its suitability to **use version control** (e.g. Git Hub) and implement **Infrastructure as Code (IaC)** as far as possible.

For reasons of security and failover recovery, you design your system in a way that all software components work together but are **isolated and independent microservices**. To achieve that, you should familiarize yourself with the basic principles of the **containerization** framework, Docker, and choose suitable images for your system components from Docker Hub.

You should also consult **best-practice architectures** which achieved similar goals as the ones you are about to achieve with your system. Good starting points for this are Blog posts or other similar online formats.

Furthermore, you choose a **sample data source** for your project. A good starting point is Kaggle where you can find numerous open sample data sets. The requirements for the sample data you choose, are that it should contain large amounts of data which should be at least 1,000,000 data points and that the data is time referenced, i.e. containing a timestamp for each data point.

By the end of this phase, you should be able to answer the following questions:

- Which microservices will take care of data ingestion to my system?
- Which microservices will take care of data storage in my system?
- Which microservices will take care of data pre-processing and aggregation?
- Which microservices will take care of data delivery to the frontend machine learning application?
- Which techniques will I use to implement reliability, scalability, and maintainability to my system?
- Which techniques will I use to ensure data security, governance, and protection?
- Which docker images will I use to build the system and must they be modified?
- Which data will I use for my project?
- At which frequency will my system ingest (e.g. monthly), process, aggregate and deliver data (e.g. quarterly)?

For each bullet point, justify your choice. These bullet points are for you to structure your work. They do not have to be answered literally in your submission.

Finally, you draft and visually present your architecture to be built in a concise, comprehensive, and appealing **flow chart**. For inspiration, you can visit the Microsoft Azure **Reference Architecture** site at https://docs.microsoft.com/en-gb/azure/architecture/browse/ [last accessed at 11.11.2020]. You should also discuss **advantages and disadvantages** of your conceptual draft.

Throughout the process, online meetings provide an opportunity to talk, share ideas and/or drafts, and obtain feedback. In the online meetings, exemplary work, that has been previously submitted, can be discussed with the tutor. Here, everyone has the opportunity to get involved and learn from each other's feedback. There are also other channels available for you to address questions to the tutor and/or to your fellow students, such as the CourseFeed. Through the latter you will obtain feedback, tips and advice before submitting your examination performance. **It is recommended to make use of these channels to avoid errors and to make improvements.** You submit work after making use of all the above-mentioned tutorial and informative media. This will be followed by a feedback from the tutor and the work in the second phase can begin.

1.1.2. Development phase/reflection phase

In this phase you practically implement your data processing system. You set up a **Git Repository** which will contain all the code you generate during this project. You will create your data processing infrastructure as microservices as it was drafted in the conception phase. To do that, you **deploy docker containers** which might also be modified for your system. You will not deploy to cloud services but locally to your machine for development purposes. If necessary, you **adapt your system to ensure reliability, scalability, maintainability, data security, governance, and protection**. After all microservices are running and communicating with each other, you

ingest the data you chose in the conception phase to your system. Make sure that the data is ingested, pre-processed, and aggregated as expected. At the end of this phase, you have a working environment which you can use to reproducibly build your data processing infrastructure. Your code resides in a version-controlled Git repository which can also be used as a portfolio in itself and shown to potential future employers.

Throughout the process, online meetings and other channels provide the opportunity to profoundly discuss ideas and/or drafts and to get sufficient feedback, tips, and hints. **It is recommended to use these channels to avoid errors and to improve your work.** Once this is done, you can hand in your second phase for evaluation. Following a feedback from the tutor, your work on the final draft will continue in the third phase.

1.1.3. Finalization phase

In this phase, you finalize your project. First, you finetune your code which builds your infrastructure. Revise the whole process and note where things went smoothly and where problems emerged.

- Does your system fulfill the technical requirements?
- What went wrong and why?
- Is your system reliable, scalable, and maintainable?
- What measures for data security, governance and protection can be added?
- What could you do in the next project to improve your workflow?
- What are the major steps you took and what are the three most valuable technical skills you learned during the project?
- What are the three most valuable “soft” skills you learned during the project?

You should **discuss which strategy** you could pursue in order **to introduce a second data pipeline to your system** which is able to process real-time streaming data.

In the “finalization phase”, the online meetings and other channels also provide the opportunity to obtain sufficient feedback, tips, and hints before the finished product is finally handed in. **It is recommended to use these channels to avoid errors and to make improvements.** The finished product is submitted with the results from Phase 1 and Phase 2 and together with the materials mentioned above. In addition, it is mandatory to write an abstract which describes the solution of the task in terms of content and concept and presents a short breakdown (making of) of the technical approach in a sober and informative way. Following the submission of the third portfolio page, the tutor submits the final feedback which includes evaluation and scoring within six weeks.

1.2. Task 2: Build a real-time data backend for a data-intensive application

As the backend of a data-intensive real-time streaming report, you design, build, and provide the underlying data infrastructure. You will build a system which is able to continuously ingest massive amounts of data, pre-process parts of it and aggregate the data for direct usage in a real-time streaming report (which is not covered by this project). During your journey to a state-of-the-art data real-time processing architecture, you will be able to adapt common data engineering principles. You will also be acquainted with canonical software components and frameworks.

1.2.1. Conception phase

This part of the process is the most important. Anything that is overlooked or forgotten in this phase has a negative effect on the implementation later and will lead, in the worst case, to useless results.

The first step is to **familiarize yourself with common software components** to build a real-time data-intensive application, for example Apache Kafka and Spark. You do not have to know all details of each component, but you should be able to associate each software component with a specific task within your system. At this stage, you should also consider as to how you will implement techniques which **ensure reliability, scalability, and maintainability** of your system. Furthermore, you should reason how you will ensure **data security, governance, and protection** of your system.

Right from the start, you should keep in mind, that everything you design and build should be reproducible. Therefore, while choosing your working environment, you should consider its suitability to **use version control** (e.g. Git Hub) and implement **Infrastructure as Code (IaC)** as far as possible.

For reasons of security and failover recovery, you design your system in a way that all software components work together but are **isolated and independent microservices**. To achieve that, you should familiarize yourself with the basic principles of the **containerization** framework, Docker, and choose suitable images for your system components from Docker Hub.

You should also consult **best-practice architectures** which achieved similar goals as the ones you are about to achieve with your system. Good starting points for this are Blog posts or other similar online formats.

Furthermore, you choose a **sample data source** for your project. A good starting point is Kaggle where you can find numerous open sample data sets. The requirements for the sample data you choose, are that it should contain large amounts of data which should be at least 1,000,000 data points and that the data is time referenced, i.e. containing a timestamp for each data point. You can also find open streams of data or you can run a small local application on your machine which simulates, e.g. sensor data which are generated continuously. Alternatively, you can simulate the real-time feature of your system by connecting a static data source and make changes manually which should be visible throughout your system and finally in the produced aggregated outcomes.

By the end of this phase, you should have answered the following questions:

- Which microservices will take care of data ingestion to my system?
- Which microservices will take care of data pre-processing and aggregation?
- Which microservices will take care of data delivery to the frontend machine learning application?
- Which techniques will I use to implement reliability, scalability, and maintainability to my system?
- Which techniques will I use to ensure data security, governance, and protection?
- Which docker images will I use to build the system and must they be modified?
- Which data will I use for my project and how do I ensure real-time streaming features?
- Which aggregation and windowing functions do I use to aggregate the data stream?

For each bullet point, justify your choice. These bullet points are for you to structure your work. They do not have to be answered literally in your submission.

Finally, you draft and visually present your architecture to be built in a concise, comprehensive, and appealing flow chart. For inspiration, you can visit the Microsoft Azure **Reference Architecture** site at https://docs.microsoft.com/en-gb/azure/architecture/browse/ [last accessed at 11.11.2020]. You should also discuss advantages and disadvantages of your conceptual draft.

Throughout the process, online meetings provide an opportunity to talk, share ideas and/or drafts, and obtain feedback. In the online meetings, exemplary work, that has been previously submitted, can be discussed with the tutor. Here, everyone has the opportunity to get involved and learn from each other's feedback. There are also other channels available for you to address questions to the tutor and/or to your fellow students, such as the CourseFeed. Through the latter you will obtain feedback, tips and advice before submitting your examination performance. **It is recommended to make use of these channels to avoid errors and to make improvements.** You submit work after making use of all the above-mentioned tutorial and informative media. This will be followed by a feedback from the tutor and the work in the second phase can begin.

1.2.2. Development phase/reflection phase

In this phase you practically implement your data processing system. You set up a **Git Repository** which will contain all the code you generate during this project. You will create your data processing infrastructure as microservices as it was drafted in the conception phase. To do that, you **deploy docker containers** which might also be modified for your system. You will not deploy to cloud services but locally to your machine for development purposes. If necessary, you adapt your system to **ensure reliability, scalability, maintainability, data security, governance, and protection**. After all microservices are running and communicating with each other, **you ingest the data** you chose in the conception phase to your system. Make sure that the data is ingested, pre-processed, and aggregated as expected. At the end of this phase, you have a working environment which you can use to reproducibly build your data processing infrastructure. Your code resides in a version-controlled Git repository which can also be used as a portfolio in itself and shown to potential future employers.

Throughout the process, online meetings and other channels provide the opportunity to profoundly discuss ideas and/or drafts and to get sufficient feedback, tips, and hints. **It is recommended to use these channels to avoid errors and to improve your work.** Once this is done, you can hand in your second phase for evaluation. Following a feedback from the tutor, your work on the final draft will continue in the third phase.

1.2.3. Finalization phase

In this phase, you finalize your project. First, you finetune your code which builds your infrastructure. Revise the whole process and note where things went smoothly and where problems emerged.

- Does your system fulfill the technical requirements?
- What went wrong and why?
- Is your system reliable, scalable, and maintainable?
- What measures for data security, governance and protection can be added?
- What could you do in the next project to improve your workflow?
- What are the major steps you took and what are the three most valuable technical skills you learned during the project?
- What are the three most valuable “soft” skills you learned during the project?

You should **discuss which strategy** you could pursue in order **to introduce a second data pipeline to your system** which is able to store and process data in batches.

In the “finalization phase”, the online meetings and other channels also provide the opportunity to obtain sufficient feedback, tips, and hints before the finished product is finally handed in. **It is recommended to use these channels to avoid errors and to make improvements.** The finished product is submitted with the results from Phase 1 and Phase 2 and together with the materials mentioned above. In addition, it is mandatory to write an abstract which describes the solution of the task in terms of content and concept and presents a short breakdown (making of) of the technical approach in a sober and informative way. Inserting an additional zip folder is not necessary in this course. Following the submission of the third portfolio page, the tutor submits the final feedback which includes evaluation and scoring within six weeks.

2. TUTORIAL SUPPORT

In principle, several channels are open to attain feedback for the portfolios. The respective use is the sole responsibility of the user. The independent development of a product and the work on the respective portfolio parts is part of the examination performance and is included in the overall assessment.

On the one hand, the tutorial support provides feedback loops on the portfolio parts to be submitted in the context of the conception phase as well as the development and reflection phase. The feedback takes place within the framework of a submission of the respective part of the portfolio. In addition, regular online tutorials are offered. These provide you with an opportunity to ask any questions regarding the processing of the portfolio and to discuss other issues with the tutor. In the course on “myCampus” there is also a forum available to clarify course-specific questions with all course participants and the tutor. The tutor is also available for technical consultations as well as for formal and general questions regarding the procedure for portfolio management.

Technical questions regarding the use of “PebblePad” should be directed to the exam office via mail.

3. EVALUATION

The following criteria are used to evaluate the portfolio with the percentage indicated in each case:

Evaluation criteria	**Explanation**	**Weighting**
Problem Solving Techniques	*Capturing the problem *Clear problem definition/objective *Understandable concept	10%
Methodology/Ideas/Procedure	*Appropriate transfer of theories/models *Clear information about the chosen Methodology/Idea/Procedure	20%
Quality of implementation	*Quality of implementation and documentation	40%
Creativity/Correctness	*Creativity of the solution approach *Solution implemented fulfils intended objective	20%
Formal requirements	* Compliance with formal requirements	10%

The design and construction of the portfolio should take into account the above evaluation criteria, including the following explanations:

Problem Solving Techniques: Correct reflection and implementation of data engineering concepts such as reliability, scalability, maintainability, data security, governance and protection

Methodology/Idea/Procedure: Correct technical requirements and reasonable argumentation of the conceptual data architecture

Quality of implementation: Fulfillment of the technical requirements, the work in terms of latencies, reproducibility with the help of Infrastructure as Code (IaC) to other systems in relation to the implemented system as well as concise and comprehensive documentation

Creativity/Correctness: Creative approach to the problems and correct task fulfillment by the implemented system

Formal requirements: Compliance with the formal requirements for submission (see below)

4. FORMAL GUIDELINES AND SPECIFICATIONS FOR SUBMISSION

4.1. Components of the examination performance

The following is an overview of the examination performance portfolio with its individual phases, individual performances to be submitted, and feedback stages at one glance. A template in "PebblePad" is provided for the development of the portfolio parts within the scope of the examination performance. The presentation is part of this examination.

Stage	Intermediate result	Performance to be submitted
Conception phase	Portfolio part 1	• Concept presentation in written form, which concisely shows that you thought about the questions stated in the task description (200 words; approx. 1/2 page directly written into the corresponding PebblePad field) • An attached file (PNG) showing a concise and comprehensive visual draft of the data architecture including explanatory remarks.
		Feedback
Development phase/ reflection phase	Portfolio part 2	• Explanation of implementation in written form including a link to a Git Hub repository containing your code (200 words; approx. 1/2 page directly written into the corresponding PebblePad field) • A document (TXT) with a link to the Git Hub repository containing your code. This code does not have to be production ready and completely errorless. It should show that it can be used to build the main microservices as the backbone for your system.
		Feedback
Finalization phase	Portfolio part 3	• A short abstract of your work summarizing the key solutions which you implemented during this project (50 words directly written into the corresponding PebblePad field) • The full abstract as an attached 2 page PDF file including the project documentation as well as a technical and personal project reflection. • A document (TXT) with a link to the Git Hub repository containing your code. As this code is the final product, i.e. the reproducible data architecture, the code should be transferable and executable on another machine using docker containers. • Result from phase 1 • Result from phase 2

Feedback + Grade

4.2. Format for Digital File Submission

Conception phase

Recommended tools/software for processing	Git Hub, IDE of choice (VS Code), Docker for your local machine
Permitted file formats	visual architecture draft as PNG
File size	as small as possible
Further formalities and parameters	Files must always be named according to the following pattern: **For the performance-relevant submissions on "PebblePad":** Name-FirstName_MatrNo_Course_P(hase)-1_S(ubmission) Example: Mustermann-Max_12345678_ Data Engineering _P1_S

Development/reflection phase

Recommended tools/software for processing	Git Hub, IDE of choice (VS Code), Docker for your local machine
Permitted file formats	link to your Git Hub repository in a TXT
File size	as small as possible
Further formalities and parameters	Files must always be named according to the following pattern: **For the performance-relevant submissions on "PebblePad":** Name-FirstName_MatrNo_Course_P(hase)-2_S(ubmission) Example: Mustermann-Max_12345678_ Data Engineering _P2_S

Finalization phase

Recommended tools/software for processing	Git Hub, IDE of choice (VS Code), Docker for your local machine
Permitted file formats	full abstract as PDF, link to your Git Hub repository in a TXT
File size	as small as possible
Further formalities and parameters	The practical work must be submitted as printable PDFs in Hi-Res (300dpi resolution, CMYK color mode). Please make sure that you either embed the images (and fonts, if any) linked in your document or to place them in the respective directory. Otherwise your documents cannot be opened completely and therefore cannot be assessed! Files must always be named according to the following pattern: **For the performance-relevant submissions on "PebblePad":** Name-FirstName_MatrNo_ Course _P(hase)-3_S(ubmission) Example: Mustermann-Max_12345678_ Data Engineering _P3_S

4.3. Format of Abstract

Length	2 pages of text
Paper size	DIN A4
Margins	Top and bottom 2cm; left 2cm; right 2cm
Font	General Text - Arial 11 pt.; Headings - 12 pt., Justify
Line Spacing	1,5
Sentences	Justified; hyphenation
Footnotes	Arial 10 pt., Justify
Paragraphs	According to mental structure - 6 pt. after line break
Affidavit	The affidavit shall be made in electronic form via "myCampus". No submission of the examination performance is possible before it. Please follow the instructions for submitting a portfolio on "myCampus".

If you have any questions regarding the submission of the portfolio, please contact the exam office via mail.

Please also note the instructions for using PebblePad & Atlas!

Good luck creating your portfolio!