Introduction

This project overview contains details about two separate projects – Virtual Human Package and SpendSense. While one project is developed individually (by me) and the other one is being built by an entire team, I will utilize both of them to demonstrate my knowledge and skills necessary to pass this semester.

Project 1 – SpendSense

In this section you will find a description of the first project I will be working on throughout the semester. Below I will describe the context and goal of this project, along with the way of working and different approaches I might employ.

**Context**

Personal finances are one of the most important resources in today’s world, yet a big portion of our society’s individuals fail to manage it properly. Most people are not only unable to resist the temptation of meaningless purchases, but they also succumb to a bigger underlying issue – financial illiteracy. The goal of this project is to develop an application that gives insights into its users’ spending habits, and to help them improve their financial intelligence.

**Objective**

The aim of this project is to develop a useful and reliable product. The main deliverable is a secure, scalable and responsive web application that allows users to upload their expenses, set budgets and receive reports on their spendings. The application aims to promote financial discipline and provide users with tools to achieve their financial goals.

**Development Approach**

This project will adopt an agile way of working. Since it requires a lot of research, the development process will be heavily guided by frequent iterations and stakeholder feedback.

Project 2 – Virtual Human Package

In the paragraphs below you can find a detailed overview of the group project of the semester. In this section you can read the context and goal of this project, along with the intended way of working.

**Context**

MindLabs and its partners, under the "Digital Innovation District for Society" initiative, are exploring the development of interactive virtual humans and their applications in society. The project focuses on creating a realistic, dynamic interaction between AI-powered virtual humans and real users.

**Objective**

The goal of this project is to create a realistic and functional proof of concept for a virtual human. This application will use a microservices architecture to enable the dynamic deployment of AI models. The aim is to convert an existing Docker Swarm installation to Kubernetes, create deployment scripts, and develop a monitoring dashboard.

**Development Approach**

The team working on this project will employ the SCRUM methodology. We will use sprints to organize the development workflow and we will hold stand-up meetings to track progress. Furthermore, we will use retrospectives to reflect on our process to ensure an efficient way of working.

Professional Standard

Working on SpendeSense and Virtual Human Package, I face various challenges. I always strive to perform thorough research on order to find the answers to my issue and I do my best every time deliver an effective and functional ICT solution. I apply structured research methodologies to validate my decisions and provide well-founded advice to stakeholders.

**Research and Methodologies**

For each of the two projects I conduct applied research to ensure I am developing a quality application. The main topic that I had to research for the group project (Virtual Human Package) so far was the deployment of AI models to Kubernetes. After completing a short course online, I managed to deploy the model in Docker and Minikube, and I am just a few steps away from deploying it in Kubernetes as well.

The research that I had to perform for my individual project (SpendSense) also involved some practical exercises, but a large portion of it was dedicated to reading materials and watching courses online. The aim of this research is to introduce me to the best practices for developing the architecture for cloud-native applications. More details about the scope and context of my research you can read in my Research Plan. So far, thanks to my research I have managed to create a few microservices, deployed in Docker and communicating with databases also in Docker.

I had to perform more research in order to make my application more secure, efficient and reliable. I have broadened my technical implementations and now I am working on documenting my findings in the Research Document.

**SCRUM**

For both the individual and group project I am following an agile approach – the SCRUM methodology. For both projects I am working in 3-week sprints. For the individual project I try to have one or two feedback sessions with my teachers every week and for the group project we try to have 1-2 progress meetings weekly. For both projects we also have deliveries every 3 weeks where we receive feedback and plan our next steps.

**Group Communication**

The team for the group project this semester has a clear vision of the expected workflow – every teammate is aware that we should set meetings with the client where we discuss expectations and progress and meetings within the team where we plan tasks, distribute them and help each other.

We had many of these meetings throughout the semester, we discussed the goals we have for the project, set up some tasks and described them in Jira. We showed each other what we have achieved and discussed challenges. I am content with this side of the team’s communication.

What I feel like we had trouble agreeing on was the level of independence expected from the team. This project was not clear from the start so our team had to be very proactive in order to receive more information. My intended approach was to come up with features that we believe the client might want and implement them before being asked to do so specifically. The downside of this approach is that some features might end up not so useful, in which case our time would not have been well spent. I believe that this would have been an acceptable downside in our situation as the benefit is that our implementations would make it clear for the client what direction we have chosen for this project and he would be able to easily give us feedback on whether he agrees with our ideas or not.

Some teammates, however, were in favor of a different approach, which relied on stronger communication with the client. This approach involved detailed meetings with the client where clarity on the tasks was achieved before the start of the implementation. Although I agree that this approach is more time efficient for the group, I do not think it suits the specifics of this project. This method involves frequent discussions with the client, but in our case, he was busy and expected more independence on our side. I also believe that this approach narrows the research opportunities as it requires more information to be provided to the team from the start.

**Virtual Human Research**

As it is already well-known by now the group project of this semester is challenging. For a considerable part of the semester our group had difficulty understanding the project, but also the technical details of the assignment. In order to bring some clarity on the topic we had multiple meetings where we discussed our interpretations of the information we had gathered so far – explanations from client, project description document and deliverables from previous groups. Since our sources sometimes pointed to different conclusions we tried to et up more meetings with our client. Although, sometimes we received responses a bit slowly, we managed to participate in a few key sessions that shed light on the goals and turned the course of this project. In these sessions we discussed our understanding of the tasks, but we also received a clearer explanation of what the finished product must consist of.

After we received this information, we divided the tasks for research and completion. My task was to work on the OCI service along with a few of my teammates. After reading about the topic online, I understood how this service has to work. My teammates and I managed to develop a small program that downloads an AI model and makes an image out of it. In this image it also adds a script that permits the user to interact with the model. The benefit of this implementation is that it creates a strong foundation for the student of the other group that will later build upon this code in order to fully implement the dynamic deployment of the models.

Personal Leadership

In both of the projects of this semester I take ownership of my professional development. I set clear goals and I take initiative to grow both technically and personally. I seek feedback often, analyze it and apply it.

**Planning and Goals**

I have set personal goals to broaden my knowledge on how to build a secure, scalable and reliable application. I have also created a rough plan of the activities this semester in which I have set different technical goals to achieve each sprint. You can find that planning in my Project Plan.

I am content with the progress I have made on my technical goals so far. I now have an application with multiple microservices and databases. I took steps to ensure it is secure, reliable and efficient by implementing authentication and authorization, SAGA pattern for GDPR regulations, monitoring, static checks, load tests, API gateway, message bus and more. I now need to focus on moving my application to the cloud, I need to deploy some microservices to Kubernetes and others to AWS Lambda. I also need to put my databases in the cloud.

**Future Preparations**

Throughout the past semester I have noticed that I have strong interest in Backend programming, but I also like implementing the Frontend of an application. This is why I am considering a career as a full-stack developer.

My education at Fontys enriches my portfolio with a lot of experience – 2 internships and multiple university projects developed for real companies. It has also prepared me very well to investigate topics by myself and acquire technical knowledge for various positions. I am confident I can find a job position here in the Netherlands.

I have not, however, ruled out a career in Bulgaria. I was worried that there technical knowledge and specifically mathematics and algorithms are valued more. I have strategically picked a minor that is interesting, but will also help me strengthen my portfolio in these aspects – Quantum-Safe Cryptography. It is an exciting new field, that offers unique research opportunities, but also requires the knowledge of certain types of mathematics and algorithms. I am very excited for this semester.

**Feedback**

I proactively research the best practices for developing cloud-native applications to make sure that my skills and delivered solution are up to the industry standards. To take it even further, I seek feedback often to ensure that I am moving in the right direction. Once received, I write down my feedback in Feedpulse, I analyze it and then implement everything necessary.

**Semester Reflexion**

To be frank I am finding this semester to be pretty challenging. I have already mentioned the difficulties my team experienced with the group project, but on top of that the topics of the semester are very new to me. All previous semesters were about implementing functional requirements, which I was accustomed to. This semester, however, includes a lot of infrastructure aspects, which I find interesting, but I have never been naturally good with. This means that I need to put more effort into understanding the material and completing my tasks. I also have to be very time and energy efficient, because this semester I am a working student. I didn’t want to rely solely on my university to enrich my portfolio, so I found a job as a software engineer in Vanderlande. It is a unique and interesting project, but unfortunately it is quite different than what we have to do this semester at university.

Considering all of the challenges mentioned above, I believe I am doing pretty well this semester. I fell slightly behind a few times, but for the most part I was told by my teachers that I am on track. I work at Fontys every required day and for the rest of the week sometimes I work during the self-study days (when I prioritize the group project to make sure I never slow my teammates down), others – during the weekend (when I focus on my individual, as I find its schedule slightly more flexible). I have implemented the bigger part of my projects, and as long as I stay focused these last few weeks, I should finish all of my tasks on time.

One thing that I did not do this semester, and that I need to do is learning to change the initial plan and accept that some strategies need to be left behind. For my individual project I had planned a specific structure for how the login should work. When I started implementing it, I encountered countless errors. Every time I would think that it is the last issue I need to fix, but always another one would come up afterwards. At the end I managed to implement the login, but it took me around 3 weeks. This naturally put me a bit behind, and now I am thinking that maybe I should have switched to a different login approach that would be easier to implement.

Scalable Architectures

In both applications – Virtual Human Package and SpendSense I strive to design scalable and reliable architecture that prioritizes key software quality requirements such as performance, scalability and security.

**Design for Quality Requirements**

For my individual project – SpendSense I have explicitly defined the non-functional requirements such as scalability, security and reliability (you can find them in my Project Plan). Using these NFRs as a guide I have also designed a plan for the architecture of my application that you can see in my Architecture Diagram.

So far, I have implemented a few microservices that I have deployed in Docker along with their respective databases. This is the first step to scalability as it allows individual components to be independently deployed and managed.

My next step is to implement asynchronous communication between the microservices to ensure system responsiveness and fault tolerance.

For the group project – VHP I have also managed to deploy an AI model and currently I am working on putting it in Kubernetes, which also brings the project a step closer to scalability.

**Load Testing**

To assess the scalability and performance of my SpendSense microservices architecture, I conducted a series of load tests using K6, as detailed in my [Load Testing Report]. These tests simulated realistic user traffic on key API endpoints of the Expense and Budget microservices, evaluating how the system behaves under increasing load. The results showed stable performance, with response times remaining within acceptable thresholds and error rates staying below 1%, even at peak loads. This validates that the architecture can handle concurrent users and highlights potential bottlenecks in write operations, which I plan to address through optimizations such as database indexing and horizontal scaling and caching. The insights gained from this process directly informed my architectural decisions, ensuring that the system not only meets its functional requirements but is also resilient, reliable, and scalable as usage grows.

Kubernetes and Cloud Deployment  
At first, I ran my SpendSense app using Docker and Docker Compose on my local machine. This helped me develop and test each part of the app separately. Later, I moved to Minikube so I could learn and use Kubernetes, which helps manage and scale applications more easily. After testing it locally, I deployed the app to Azure using Azure Kubernetes Service (AKS), which made it easier to run the app in the cloud and handle more users.

**API Gateway**

In the early stages of development, when I was running my application locally using Docker Compose, I used Traefik as the API gateway. Traefik was a great fit for that setup because it integrates well with Docker and automatically discovers services, making routing simple and fast to configure. However, once I moved to Minikube and eventually deployed the app on Azure Kubernetes Service (AKS), I transitioned to using NGINX Ingress Controller. NGINX worked better in my new setup because it’s widely used with Kubernetes and has strong support for features like Ingress routing and HTTPS. It made it easier to set up secure access to my services and gave me more control over how traffic is handled. Switching to NGINX helped make my app more reliable and easier to manage in the cloud.

Development and Operations (DevOps)

For both SpendSense and VHP I plan on implementing DevOps practices to ensure smooth development, deployment and monitoring of the software. My goal for these projects is to My goal is to create automated, scalable, and reliable deployment pipelines.

**Continuous Integration & Continuous Deployment (CI/CD)**

So far I have set up the GIT repository for my individual project, but I haven’t applied CI/CD yet.

Initially, I created a GitLab repository for my individual project. There I tried to implement a CI/CD pipeline, but I had issues with the runners. After a few unsuccessful attempts to set them up, I decided to follow my colleagues’ advice and move my project to GitHub. There, using Actions I managed to implement a CI pipeline very easily. For now my pipeline consists of only 2 stages – build and test, but I plan on extending it once I deploy my application.

**Finished** **CI/CD Pipeline**  
To automate my development workflow, I set up a CI/CD pipeline using GitHub Actions. The pipeline includes several key stages: first, it builds and tests the code to make sure everything runs correctly. Then it uses SonarQube to scan the code for quality issues like bugs, code smells, and potential security problems. Once the code passes these checks, a Docker image is built and pushed to Docker Hub. Finally, the image is deployed to Azure Kubernetes Service (AKS). This setup helps me catch errors early, keep the codebase clean, and deliver updates quickly and reliably. Automating these steps also reduces manual work and ensures that every change goes through the same process before reaching production.

**Static Quality Check**

I am monitoring the behavior of my application, but I also wanted to make sure that I do not have any issues in the code, libraries and dependencies itself. For this I used Snyk – a security platform that scans your application and suggests fixes. The first time I scanned my application the results were not good – I had 121 issues per microservice and only 67 of them were with low severity. According to Snyk this vulnerability was due to the base image I was using in my Dockerfiles – it was outdated. I followed Snyk’s suggestion and replaced my images with newer versions. This helped me reduce my issues to 35 low severity ones and only 1 critical.

**Monitoring**

Since I believe it is important to monitor the activity in my application, I decided to implement Prometheus and Graphana. I use these tools to track the successful GET requests, requests per second and active connections in Traefik and the response time of my microservices. I also implemented the SAGA pattern for deleting user data, so I monitor the total and successful attempts. If more than 5 unsuccessful attempts are made in the span of 5 minutes, an alarm in Prometheus will go off.

Cloud Native

I am developing my software with cloud-native principles in mind, using Spring Boot and deploying components in a cloud environment. While my knowledge is still growing, I am working on researching and implementing best practices in developing cloud-native applications.

**Initial Cloud-Native Implementations**

VHP is hosted on Azure and Netlab environments which provides a scalable infrastructure. Regarding SpenSense I have planned an architecture in which 2 microservices will be deployed on Kubernetes and one will be serverless (for more details, please refer to my ).

**Final Architecture**

My application is divided in 2 parts, I have some microservices and databases that uploaded in Kubernetes, while other components are serverless. At the beginning of the project, I planned to deploy SpendSense using AWS because of its popularity and wide range of services. However, after exploring both platforms, I decided to switch to Microsoft Azure. One of the main reasons was that Azure offered better integration with the tools I was already using, such as Visual Studio Code and GitHub. I also found Azure’s learning resources and documentation more beginner-friendly, which helped me set up Kubernetes (AKS) and Azure Functions more easily. Additionally, the Azure student credits made it more cost-effective for development and testing during the project. This choice allowed me to build and deploy the application in a more efficient and manageable way.

*AKS*

My microservices, databases and front end are deployed on Kubernetes. Initially the project was deployed to Minikube, but eventually I moved it to Azure Kubernetes Services. Some components are automatically scaled to ensure the reliability of the application under heavy load. The third microservice – SAGA Delete calls the second part of my application – the serverless functions. For a better illustration of my AKS architecture, please refer to my SpendSenseAKS diagram.

*Azure Functions*

In this part of my application, I have 3 serverless functions that communicate via Azure Message Busses and Topics. The process begins when the AKS microservice is called. Its job is to put a message in the first message que. For an illustration of the process please refer to SpendSenseServerless.

Security by Design

In both of the applications that I am working on this semester I prioritize security at every stage of the development. I research common strategies that malicious actors use and best practices on how to build a safe application to make sure I develop a non-vulnerable product.

**Analysis and Built-in Security**

So far, I have analyzed the OWASP top 10 vulnerabilities and researched ways to mitigate the risks. For the development of SpendSense I have decided to use SpringBoot which provides automatic protection against 4 out of the 10 top vulnerabilities - Cross-Site Request Forgery Protection (enabled by default in Spring Security), Cross-Site Scripting Prevention (through built-in input validation mechanisms), SQL Injection Protection (by using Spring Data JPA with parameterized queries), Secure Default Headers (e.g., Content Security Policy, HSTS).

**Authentication & Authorization**

In order to protect the data and activity of my users, I have implemented authentication and authorization. In the current structure of my application instead of routing requests directly to the microservices, my API gateway sends them to Aouth2 – a reverse proxy that checks if the user is authorized to make that request. If they have not yet been authenticated, Aouth2 sends the user to Keycloak – an ID provider that lets them log in. If they provide correct credentials, users are allowed to continue with their request.

**Security Challenges During Deployment**  
While migrating my application to Minikube, I encountered a technical issue with the authentication flow—specifically, OAuth2 failed to redirect users to the Keycloak login page. This issue appears to be related to how OAuth2 handles redirect URIs in a Kubernetes environment.

In my local Docker setup, services could communicate using container hostnames, and redirects worked correctly. However, in Minikube, service discovery and external access are handled differently—often requiring ingress controllers or NodePort services to expose routes externally. OAuth2 was attempting to redirect to an internal service name (e.g., http://keycloak:8080/auth), which is only resolvable from within the cluster and not accessible from the user's browser. Since OAuth2 couldn’t reach Keycloak’s login page externally, the flow broke before authentication could complete. I suspect the root cause lies in misconfigured redirect URIs or missing ingress rules that would expose Keycloak securely to external traffic via a stable hostname.

If time permits, I plan to switch to Azure Active Directory for user login, as it integrates more naturally with AKS and avoids manual redirect and exposure configuration, simplifying both security and deployment.

**HTTPS**

In order to protect the data of my users I am now using https for my application. I bought a domain for my app and requested a certificate from letsencrypt. After running a script and verifying that this is my address, I received a proper, not self-signed, certificate. This ensures that the data is encrypted and therefore travels safely between the browser and server.

After moving my application to Azure, I made sure it still used HTTPS to keep all communication between users and the server secure. While Azure does offer automatic ways to manage certificates using tools like Azure Front Door or Application Gateway, I chose to set everything up myself. I used a custom domain I had already bought and uploaded an SSL certificate I got from Let’s Encrypt into Azure. Then, I connected this certificate to my Kubernetes setup on Azure (AKS) by configuring an Ingress resource that handles incoming traffic.

To do this, I created a TLS (Transport Layer Security) section in the Ingress configuration and stored the certificate and private key inside a Kubernetes secret, which the cluster can use to enable HTTPS. This made sure that any data shared between the user and my application was encrypted and safe from being seen or changed during transmission. By setting this up manually, I was able to understand how secure networking works in Kubernetes and make sure that both my development and production environments protect user data properly.

**Encrypted critical data**

At the start of my project, I stored sensitive values like database credentials and API keys directly inside the Kubernetes YAML files. While this was quick and easy for testing, I realized it was not secure for production use. As I moved my application to Azure Kubernetes Service (AKS), I improved this setup by using Kubernetes Secrets instead. Now, sensitive data is separated from the deployment files and stored securely within the cluster. These secrets are mounted into my containers as environment variables, making the application safer and more maintainable without exposing sensitive information in version control.

**Security Check Using OWASP Top 10**

While working on SpendSense, I did a full security check following the OWASP Top 10 list, which shows the most common web app security problems. I wrote an OWASP report that looks at the main risks in my app and how I handle them. This includes checking the microservices, databases, API gateway, login system with Keycloak, and message system with RabbitMQ. I used tools like Snyk to find weak spots in the code and libraries, and I’m using SonarQube for ongoing security checks. The report showed me what’s working well, like protecting against database attacks and using HTTPS for secure data transfer, and what still needs work, like locking down monitoring tools and making sure internal services are safer. This report helped me improve the app’s security and shows I take building safe software seriously.

Distributed Data

My projects of this semester require the handling and storing of data in a structured and a secure way. I will also make sure to take into account legal and ethical considerations. I am focusing on data distribution, security, and compliance with regulations such as GDPR.

**Data Decentralization**

For my project SpendSense, I have decided to follow the Database per Service Pattern. I have connected two of my microservices to two separate databases. This approach ensures that the databases are independent and changes to one will not affect the other. It also allows each database to be scaled individually and makes the isolation of sensitive data easier.

**Distributed Data and GDPR**

In order to comply with GDPR regulations, an application needs to permit their users to delete their account, but also all of the information related to it. In my application the user can be deleted easily through Keycloak, but that will not remove all the data from the databases of the other microservices. In order to ensure complete data removal, I employed the SAGA pattern. I now have an Orchestration microservice, that is responsible for initiating the deletion of the data across different databases.

**Improved SAGA and GDPR**

In the current version of my application a user can request the deletion of their account. In compliance with GDPR regulations this would result in all data from all databases related to this user to be deleted. I achieve this functionality by using the SAGA pattern where all data from ExpensesDB is SOFT deleted first. If everything goes alright the same happens with the data from BudgetsDB as well. If there is an issue with the deletion of BudgetsDB, however, the data from ExpensesDB will be restored. The communication between the microservices responsible for these databases happens through Azure busses and topics. You can find a diagram of this process in SpendSenseFunctions.

**Automatic Scaling**

In my project, I set up automatic scaling for the microservices and the database to handle different loads efficiently. For the microservices running in Kubernetes, I use horizontal scaling, which means adding or removing instances (pods) automatically based on the current demand, like CPU or memory usage. This helps the app handle more users without slowing down. For the database, I use vertical scaling, where the database gets more resources like CPU or memory when needed.