Projectlogboek

Project WealthCare - Intelligent Technology for the Medical industry

*Dannique Klaver*

*500929363*

Ad Software Development

Augustus 2024

# Voorwoord

Dit projectlogboek beschrijft mijn ervaringen en bijdragen aan het project WealthCare, ontwikkeld binnen het vakgebied Intelligent Technology for the Medical industry. Het project richtte zich op het ontwerpen en ontwikkelen van een platform dat medische professionals ondersteunt bij het beheren en visualiseren van gezondheidsdata.

Mijn studie begon met de keuze van een programmeertaal en technologieën die aansluiten bij mijn toekomstige loopbaandoelen. Na onderzoek heb ik gekozen voor Java met het Spring Boot-framework voor de backend en Angular voor de frontend. Deze keuze weerspiegelde mijn ambitie om mijn vaardigheden te ontwikkelen voor een mogelijke stage bij een ICT-organisatie van de Rijksoverheid.

Tijdens dit project heb ik veel geleerd door het uitvoeren van diverse taken, waaronder:

* Zelfstudie om basiskennis op te doen van Java, Spring Boot en Angular.
* De rol van Scrum Master vervullen in de opstartfase van het project.
* Het opzetten van een gestructureerde werkomgeving in GitLab met epics, labels, een samenwerkingsovereenkomst, en een wiki.
* Een Productvergelijking maken tussen JUnit 5 en TestNG als test frameworks voor Java Spring Boot.
* Het uitvoeren van een diepgaand security-onderzoek naar XSS-aanvallen en het ontwikkelen van een prototype om deze te mitigeren.
* Een presentatie en samenvatting van C++ catastrophes en oplossingen daarvoor.

Naast mijn projecttaken heb ik deelgenomen aan meerdere stagemarkten om inzicht te krijgen in de mogelijkheden binnen de ICT-sector en mijn toekomstige carrière, waaronder de HBO ICT job en student event, Open Huis IT&IV @Rijksoverheid en de HBO ICT stagemarkt op de HvA zelf.

Ik wil mijn teamgenoten bedanken voor hun inzet en samenwerking tijdens het project, zonder hun zou dit project niet zijn wat het nu is, daarnaast wil ik Storm Verwer bedanken voor meehelpen met de zelfstudie en Britt Rood voor de hulp met het maken van verslagen en documentatie.

Tot slot bedank ik onze begeleidende docenten, Gerbrand van Dieijen en Yordi Verkroost, voor hun begeleiding en waardevolle feedback.

# Samenvatting

Dit document biedt een overzicht van mijn bijdrage aan het WealthCare-project en dient als bewijs van mijn ontwikkeling en prestaties tijdens dit semester. Het project richtte zich op het ontwikkelen van een veilige, efficiënte applicatie om medische gegevens te beheren.

In hoofdstuk 4 wordt de context van het project beschreven, evenals de bijbehorende opdracht en doelstellingen. Hoofdstuk 5 gaat dieper in op de eisen en criteria waaraan de applicatie moest voldoen, waaronder functionele eisen en beveiligingsaspecten zoals mitigatie van XSS-aanvallen.

Hoofdstuk 6 documenteert de voortgang per sprint en belicht de samenwerking binnen het team. Hier worden acties, resultaten en feedback van retrospectives beschreven. Hoofdstuk 7 bevat een handleiding voor de applicatie, met uitleg over de functionaliteiten en hoe deze door gebruikers kunnen worden ingezet.

In hoofdstuk 8 wordt een technische toelichting gegeven, waarin wordt uitgelegd hoe de applicatie is ontworpen en geïmplementeerd. Specifieke aandacht gaat uit naar de backend-opbouw, het gebruik van Spring Boot, en de implementatie van securityoplossingen.

Tot slot reflecteert hoofdstuk 9 op het technische proces, benoemt verbeterpunten en biedt suggesties voor toekomstige studenten. Hoofdstuk 10 bevat bewijsstukken zoals diagrammen, screenshots en relevante code ter ondersteuning van dit logboek.

Dit document dient als een waardevolle bron van kennis en ervaring die ik heb opgedaan tijdens mijn studie en vormt een belangrijke stap in mijn voorbereiding op een professionele carrière in de softwareontwikkeling.

# Table of Contents

[1. Voorwoord(DONE) 2](#_Toc187074580)

[2. Samenvatting(DONE) 2](#_Toc187074581)

[3. Table of Contents 2](#_Toc187074582)

[4. Context en Opdracht(DONE) 4](#_Toc187074583)

[4.1 Context 4](#_Toc187074584)

[4.2 Opdracht/probleem 5](#_Toc187074585)

[4.3 Werkwijze 6](#_Toc187074586)

[5. Requirements(DONE) 8](#_Toc187074587)

[5.1 Functionele eisen 8](#_Toc187074588)

[5.1.1 **UC-1:** Authentication Management 8](#_Toc187074589)

[5.1.2 **UC-2:** Patient Management 8](#_Toc187074590)

[5.1.3 **UC-3:** Health Analytics Visualization 9](#_Toc187074591)

[5.1.4 **UC-4:** Advanced Analytics Management 9](#_Toc187074592)

[5.2 Quality Attribute Scenarios 10](#_Toc187074593)

[5.3 Constraints 11](#_Toc187074594)

[5.4 Concerns 11](#_Toc187074595)

[6. Opleveringen(DONE) 12](#_Toc187074596)

[6.1 Sprint 1: 23 september – 4 oktober (opzet sprint) 12](#_Toc187074597)

[6.2 Sprint 2: 7 oktober – 18 oktober 13](#_Toc187074598)

[6.3 Sprint 3: 21 oktober – 8 november (3 weken sprint wegens vakantie) 14](#_Toc187074599)

[6.4 Sprint 4: 11 november – 22 november 15](#_Toc187074600)

[6.5 Sprint 5: 25 november – 6 december 16](#_Toc187074601)

[6.6 Sprint 6: 9 December – 20 december 17](#_Toc187074602)

[7. De applicatie(DONE) 18](#_Toc187074603)

[7.1 Handleiding 18](#_Toc187074604)

[7.1.1 Overzichtspagina 18](#_Toc187074605)

[7.1.2 Patiëntgegevens 18](#_Toc187074606)

[7.1.3 Grafiek overzicht 18](#_Toc187074607)

[7.1.4 Tabel overzicht 18](#_Toc187074608)

[8. Technische toelichting 18](#_Toc187074609)

[8.1 Frontend 18](#_Toc187074610)

[8.1.1 Patient Analytics Pagina 18](#_Toc187074611)

[8.1.2 PatientRecords & PatientService 18](#_Toc187074612)

[8.1.3 PatientGraphs Component 18](#_Toc187074613)

[8.1.4 HealthData Service & Component 18](#_Toc187074614)

[8.1.5 Complete analytics 18](#_Toc187074615)

[8.2 Backend 18](#_Toc187074616)

[8.2.1 Class Diagram 18](#_Toc187074617)

[8.2.2 Login functionaliteit 18](#_Toc187074618)

[8.2.3 Session Management 18](#_Toc187074619)

[8.2.4 Patient data functionaliteiten 18](#_Toc187074620)

[8.3 Security 18](#_Toc187074621)

[8.3.1 Role validation 18](#_Toc187074622)

[8.3.2 Logout functionaliteit 18](#_Toc187074623)

[8.4 Code kwaliteit 18](#_Toc187074624)

[8.4.1 Facade Pattern 18](#_Toc187074625)

[*8.4.2* Strategy Pattern 18](#_Toc187074626)

[8.5 Backend Testing 18](#_Toc187074627)

[8.5.1 Controller testing 18](#_Toc187074628)

[8.5.2 Service testing: 18](#_Toc187074629)

[8.5.3 Utility 18](#_Toc187074630)

[8.6 Frontend Testing 18](#_Toc187074631)

[8.6.1 Component Tests (Unit/Integration) 18](#_Toc187074632)

[8.6.2 Service Tests (Unit) 18](#_Toc187074633)

[8.6.3 E2E Tests (Cypress) 18](#_Toc187074634)

[9. Technische reflectie 18](#_Toc187074635)

[9.1 Dit ging goed 18](#_Toc187074636)

[9.2 Dit kon beter 18](#_Toc187074637)

[9.3 Hier moet ik aan werken 18](#_Toc187074638)

[9.4 Zo zou een student zich moeten voorbereiden 18](#_Toc187074639)

[10. Appendices(FORMAT, EN POP LEERDOELEN) 18](#_Toc187074640)

# Context en Opdracht

## Context

ITM, een afkorting van Application for Intelligent Technology for the Medical Industry, is een project dat ons werd gegeven door onze producteigenaar, Mohamed El Hallaoui. Dit project valt onder een intern project bij het bedrijf Lion’s Peak. Meer informatie over het project volgt in 3.2 (Opdracht/probleem).

Het project is gebaseerd op een probleem dat wereldwijd voorkomt: de moeilijkheden rond het beheer en delen van medische gegevens. De inspiratie hiervoor kwam uit een real-life situatie waarin een patiënt in een buitenlandse kliniek veel tijd verloor aan het beantwoorden van vragen omdat zijn medische dossier niet beschikbaar was. Dit benadrukt de uitdagingen van papieren en gefragmenteerde informatiesystemen, zoals fouten en gebrek aan toegankelijkheid, vooral in kritieke medische situaties.

Lion's Peak

Lion's Peak (KvK 90012046) is een bedrijf dat zich interesseert in het maken van duurzame en innovatieve softwareoplossingen. Het bedrijf staat geregistreerd in Amsterdam (Zilverplevierstraat 13).

## Opdracht/probleem

De opdracht binnen het ITM-project richt zich op het oplossen van problemen die ontstaan door:

* Het gebruik van primitieve informatiebeheer methoden (zoals papieren dossiers).
* Een hoge kans op fouten, die fataal kunnen zijn in medische situaties.
* De fragmentatie van informatie, waarbij gegevens vaak niet worden gedeeld tussen verschillende medische instellingen.

Het doel van de opdracht is om een oplossing te bieden in de vorm van een digitaal informatiebeheersysteem dat geschikt is voor zowel medische professionals als patiënten. Deze oplossing moet:

1. Papieren medische dossiers elimineren die vatbaar zijn voor fouten en moeilijk betrouwbaar te delen zijn.
2. Patiënten in staat stellen eigenaar te worden van hun medische dossiers via een centraal patiëntenportaal.
3. Diagnoses verbeteren door middel van gegevensinzichten, waardoor meer effectieve behandelingen mogelijk worden.

Om deze doelen te bereiken, werd het project opgesplitst in vijf modules, elk toegewezen aan een specifiek software development team:

* Patiënt Platform
* General Practitioner Health Portal
* Digital Health Record
* Dentist Portal
* Health Analytics

Mijn team, WealthCare, heeft de opdracht gekregen om een **Health Analytics** platform te ontwikkelen. Deze module richt zich op het verzamelen, analyseren en presenteren van medische gegevens, met als doel medische professionals beter inzicht te bieden in patiëntgegevens. Het platform moet betrouwbare, toegankelijke en visueel aantrekkelijke data-analysemogelijkheden bieden die helpen bij besluitvorming en diagnose.

De HA portal moet de volgende functies leveren:

* Een analysis summary per patient.
* Health Risk predictions gebaseerd op hun health data.
* Aanradingen over lifestyle changes, medication en follow-up visits.

## Werkwijze

Voor dit project maakt ons team gebruik van het Agile werkproces. Het project is opgedeeld in zes sprints (Sprint 0 - 5). Voor elk van deze sprints zal een van de teamleden aan bod komen om de Scrummaster te zijn. Dit houdt in dat dat teamlid verantwoordelijk is voor de communicatie met de product owner gedurende deze sprint. Ook zal dit teamlid binnen het team overzicht moeten houden en aan het einde van de sprint de Retrospective leiden.

***Sprint presentaties***

Tijdens sprint presentaties worden de vorderingen van het team doorsproken met de product owner. Dit zal onder andere bestaan uit:

* Het doorspreken van het Scrumboard
* Het laten zien van de gemaakte vorderingen
* Het voorstellen van ideeën en oplossingen
* Het doornemen van de feedback van de product owner

De feedback die de product owner geeft wordt tijdens deze meeting gedocumenteerd en zal worden meegenomen voor de volgende sprint.

***Retrospectives***

Aan het einde van de sprint zal een Retrospective gedaan worden waarin de volgende punten aan bod komen:

* De gemaakte actiepunten van de vorige sprint
* Dingen die tijdens de sprint opvielen en door een teamlid besproken willen worden
* Dingen waarvan teamleden vinden dat eraan gewerkt moet worden
* Eventuele oplossingen voor problemen binnen het team
* Actiepunten voor de volgende sprint

Er zal van deze meeting een opname worden gemaakt en deze opname zal ook gedeeld worden in de teams & wiki[[1]](#endnote-1)**.**

***Sprint planning***

Vanuit de gegeven sprint presentatie worden eventuele feedback punten en nieuwe issues doorsproken om mee te nemen in de aankomende sprint. Dit wordt gedaan door:

* Teamleden issues waaraan zij aan willen werken of verbetering in willen maken naar de sprint backlog te laten slepen
* Uitgeschreven issues te doorspreken
* Daar door middel van Scrum poker een gewicht aan te hangen (inclusief het nogmaals doorspreken van de redenaties van mogelijk verschillende gewichten)
* Het aanwijzen van verschillende issues aan de bijbehorende teamleden
* Het doorgeven van de gemaakte planning aan de product owner

Deze sprint planning zal door middel van de benoeming van issues en het bijhouden van het scrumboard worden bijgehouden om inzicht te creëren in de voortgang van het team.

***Issues en branching***

Voor een issue wordt gebruik gemaakt van het issue template die wij gebaseerd hebben op het template wat in het vorige jaar geleverd was, deze hebben wij uitgebreid en in het engels geschreven.   
Dit issue template is beschikbaar op de wiki van ons team[[2]](#endnote-2)**,** en zal ook in iedereens persoonlijke project beschikbaar zijn[[3]](#endnote-3).

Wij hebben afgesproken om in het Engels te werken binnen de gitlab zodat code termen en technische uitleg niet “dutchlish” worden en dat de mogelijkheid om diepere kennis te moeten beschrijven makkelijker zal gaan.

Elk uitgeschreven issue moet worden nagekeken door 2 andere developers als deze opmerkingen hebben worden deze toegepast waarna er nogmaals gereviewd word en dan door de PO of als dat niet lukt door Yordi.

Per issue wat goedgekeurd word wordt er een branch aangemaakt vanuit het issue zelf als deze in de sprint backlog staat. Hiervan uit wordt er door een teamlid aan het issue gewerkt. Het is belangrijk dat hierbij aan de afgesproken Coding Standards[[4]](#endnote-4) en de DOD(Definition of Done)[[5]](#endnote-5) wordt gehouden

Voordat een issue met de main branch mag mergen is het belangrijk dat het issue volledige goedkeuring van de pipeline heeft gekregen en goedgekeurd is door twee teamleden door middel van een merge request. Dit is verplicht. Alleen werkende code mag op de main branch gepusht worden.

# Requirements

## Functionele eisen

Op basis van de diagram ‘Use\_case\_product’(zie de diagrams folder) beschrijf ik de functionele eisen van het Patient Analytics System

### **UC-1:** Authentication Management

**Uitleg:**Medische professionals en patiënten kunnen veilig toegang krijgen tot het systeem via een login-interface. Het systeem valideert de inloggegevens, maakt sessies aan en bepaalt het gebruikerstype (Medische Professional/Patiënt) om de juiste toegangsrechten te bieden. Gebruikers kunnen ook veilig uitloggen wanneer nodig.

**Primary Actors:**Medical Professional, Patient

**Main Flow (Login):**

1. Gebruiker voert BSN/email en wachtwoord in
2. Systeem valideert de inloggegevens
3. Systeem maakt een sessie aan
4. Systeem bepaalt het gebruikerstype
5. Gebruiker wordt doorgestuurd naar het juiste dashboard

**Alternative Flow (Logout):**

1. Gebruiker selecteert uitlog optie
2. Systeem verwijdert de actieve sessie
3. Systeem verwijdert sessie cookie
4. Gebruiker wordt teruggestuurd naar login pagina

### **UC-2:** Patient Management

**Uitleg:** Medische professionals(MP) kunnen hun toegewezen patiënt-list bekijken en doorzoeken, waardoor efficiënt patiëntbeheer en gegevens toegang mogelijk is.

**Primary Actor:** Medical Professional

**Main Flow:**

1. MP bekijkt de lijst van toegewezen patiënten
2. Basis patiëntinformatie wordt getoond
3. MP kan patiënten zoeken op BSN of naam
4. MP selecteert een specifieke patiënt voor een gedetailleerd overzicht

### **UC-3:** Health Analytics Visualization

**Uitleg:** Medische professionals kunnen uitgebreide gezondheidsanalyses van hun patiënten bekijken, terwijl patiënten hun eigen gezondheidsgegevens kunnen bekijken via gevisualiseerde weergaven.

**Primary Actor:** Medical Professional, Patient

**Main Flow (MP):**

1. Gebruiker opent het analytics dashboard
2. Gebruiker selecteert specifieke gezondheidsmetingen
3. Systeem toont gevisualiseerde gezondheidsdata
4. Gevisualiseerde gezondheidsdata komt in Grafiek als dit numeriek is, en tabel als dit tekstueel is
5. MP kan toegang krijgen tot extra analytics functies

**Alternate Flow (Patient):**

1. Patiënt bekijkt direct zijn eigen analytics pagina
2. Patiënt kan alleen persoonlijke gezondheidsmetingen bekijken

### **UC-4:** Advanced Analytics Management

**Uitleg:** Medische professionals kunnen gedetailleerde health analytics genereren en analyseren via een analytics sidebar en genereer optie.

**Primary Actor:** Medical Professional

**Main Flow:**

1. MP opent de analytics sidebar
2. MP genereert nieuwe analytics
3. Systeem verwerkt en toont de resultaten

## Quality Attribute Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *Kwaliteitsattribuut (ISO 25010)* | *Scenario* | *Gerelateerde Use Case* | *Prioriteit* |
| *QA-1* | Performance Efficiency (Time behaviour) | Het systeem moet inlogverzoeken binnen 5 seconden verwerken en valideren. | UC-1 | Gemiddeld |
| *QA-2* | Security (Authenticity) | Het systeem vereist een sessie in de database en een cookie op de website om te valideren dat gebruiker ingelogd is. | UC-1 | Hoog |
| *QA-3* | Performance Efficiency (Time behaviour) | De patiëntenlijst moet binnen 5 seconden worden geladen. | UC-2 | laag |
| *QA-4* | Security (Confidentiality) | Systeem verleent alleen toegang tot functionaliteiten waartoe de gebruiker geautoriseerd is. | UC-1, UC-2, UC-3, UC-4 | Hoog |
| *QA-5* | User Engagement | Gebruikers kunnen gezondheidsdata visualisaties aanpassen (switch grafiek/tabel, aantal entries) voor optimale dataweergave. | UC-3 | Gemiddeld |
| *QA-6* | Resource Utilization | UI-componenten (lijsten, grafieken) passen zich aan aan beschikbare ruimte met scroll-functionaliteit waar nodig. | UC-2, UC-3 | Gemiddeld |
| *QA-7* | Interoperability | Systeem kan data uitwisselen met andere ITM-oplossingen via gestandaardiseerde interfaces. | UC-2, UC-3, UC-4 | Hoog |

## Constraints

|  |  |
| --- | --- |
| *ID* | *Constraint* |
| *CON-1* | Het systeem moet kunnen communiceren met zowel Java Spring Boot als C# .NET backends via REST API endpoints. |
| *CON-2* | De applicatie vereist een actieve SQL database connectie met vooraf geladen test/productie data om te functioneren. |
| *CON-3* | De frontend is gebouwd met Angular (v17+) en moet de component-based architectuur volgen met strict typed interfaces. |
| *CON-4* | Alle data manipulatie moet via services verlopen die de Single Responsibility Principle volgen. |
| *CON-5* | Components moeten modulair opgezet zijn en herbruikbaar zijn binnen de applicatie. |
| *CON-6* | Elke service moet voorzien zijn van error handling voor backend communicatie failures. |
| *CON-7* | Unit tests moeten aanwezig zijn voor alle services en components. |
| *CON-8* | De applicatie moet gebruik maken van environment files voor verschillende backend configuraties (development/testing/production). |
| *CON-9* | Authenticatie en autorisatie moet consistent werken ongeacht welke backend gebruikt wordt. |
| *CON-10* | Data modellen moeten synchroon zijn tussen frontend en alle mogelijke backends. |

## Concerns

|  |  |
| --- | --- |
| *ID* | *Concern* |
| *CRN-1* | Opstartperiode kan verlengd worden door benodigde kennis van C# en Java doordat hier eerder nog niet mee gewerkt is. |
| *CRN-2* | Database moet opgezet en gevuld worden met realistische dummy data voordat development effectief kan beginnen. |
| *CRN-3* | Integratie met verschillende backend implementaties (Java/C#) kan leiden tot inconsistenties in data handling en API responses. |
| *CRN-4* | Implementatie moet consistent werken over verschillende backends wat extra coördinatie vereist. |
| *CRN-5* | Gebrek aan initiële Requirements kan leiden tot architecturale beslissingen die later aangepast moeten worden. |

# Opleveringen

## Sprint 1: 23 september – 4 oktober (opzet sprint)

Voor deze sprint was ik, Dannique, de Scrum Master.

Ik ben direct aan de slag gegaan om met mijn team de development- en GitLab-omgeving naar wens op te zetten. We hebben epics, labels, issue boards, de DOD(definition of done) en een samenwerkingsovereenkomst opgesteld en deze vastgelegd in de wiki, zodat alles overzichtelijk beschikbaar was.   
  
Het doel van deze sprint was om een opgezette development-omgeving te leveren en een prototype te maken voor het product. Als Scrum Master heb ik gelijk een meeting ingepland met de PO om een beter beeld te krijgen van zijn verwachtingen voor het project. Tijdens deze meeting hebben we het prototype in Figma gepresenteerd. De PO was zeer enthousiast over de pagina’s die we hem hebben laten zien, waaronder de login, registratie, patient list en health analytics. We hadden oorspronkelijk ook een aparte patient overview pagina, maar aan de hand van de PO’s feedback besloten wij deze te schrappen zodat alles op één pagina gecombineerd kon worden.   
  
Daarnaast heb ik een functionele pipeline en een stappenplan opgesteld voor mijn Java Spring Boot-omgeving, die ik vervolgens heb gedeeld met de andere Java-ontwikkelaars. Dit hielp de zelfstudie efficiënter te maken en dubbele problemen te voorkomen.

**Feedback PO:**

* Controleer dat foutmeldingen gebruikersvriendelijk zijn en alleen noodzakelijke informatie tonen.
* Focus eerst op functionaliteit zoals login en registratie.
* Patient overview schrappen en deze combineren met de patient list.

**Resultaat Retrospective:**

* De samenwerking binnen het team verliep soepel, en de tools waren effectief opgezet.
* Verbeterpunt: Meer aandacht besteden aan gebruikersgericht denken bij foutmeldingen.

**Actiepunten:**

* Marcus en Rowin:   
  Kwantiteit boven kwaliteit(Verdeel de issues goed en zorg dat je niet teveel wilt in een sprint.)
* Dax en Danique:   
  Planning concreet.(in stappen opdelen.)
* Milosz en Latricha:   
  Issue board bijhouden.

## Sprint 2: 7 oktober – 18 oktober

In deze sprint was Marcus de Scrum Master.  
Dat ging wel oké, hij had problemen met veel werk willen leveren en wou een relaxtere werksnelheid waar ik een beetje tegenaan botste. Maar hij regelde PO communicatie wel naar wens alhoewel de communicatie daarvan wel beter kon.

Voor deze sprint wouden we de backend van login en sessions maken aan de hand van de feedback van de vorige sprint, enkel bleek bij de oplevering dat de PO meer de pagina’s zelf met visualisatie wou hebben, en niet enkel de functionaliteit, wat bij nader inzien ook wel prioriteit zou moeten hebben gehad, dit hebben wij meegenomen voor de volgende sprint.

Deze sprint waren we ook druk bezig met de Productvergelijking, wat zorgde voor minder tijd voor het project, wel ben ik tevreden over de functionaliteiten die we geleverd hebben, omdat we ook aan de database hebben moeten werken zodat de sessies goed bewaard werden wat ook wat tijd kostte.

**Feedback PO:**

* Zorg dat de frontend een visueel werkende login toont om progressie te demonstreren.
* Prioriteit geven aan een werkende pagina boven backend-verbeteringen.

**Resultaat Retrospective:**

* Het team leverde technische resultaten op, maar de visuele progressie bleef achter.
* Verbeterpunt: Meer tijd en aandacht aan de frontend-functionaliteiten besteden.

**Actiepunten:**

* Milosz & Latricha:  
  De Front-End bijhouden (Zorgen dat we als team bij onze backend ook bijbehorende gaan front-end hebben)
* Dannique & Marcus:  
  Scrum Proces (Zorgen dat we Scrum blijven & dat issues grondiger gereviewd worden)
* Rowin & Dax:  
  Po Communicatie (Zorgen dat de PO beter geïnformeerd wordt)

## Sprint 3: 21 oktober – 8 november (3 weken sprint wegens vakantie)

In deze sprint was Dax de Scrum Master.   
Hij deed het behoorlijk goed; hij had niet het micromanagen waar ikzelf soms schuldig aan ben, maar ook niet het gebrek aan teamcommunicatie dat Marcus af en toe toonde.   
  
Tijdens deze sprint lag onze focus op het verder ontwikkelen van de gebruikersinterface en het uitbreiden van de backend-functionaliteit om gebruikers meer mogelijkheden te bieden. Op de frontend hebben we de inlogfunctionaliteit verder ontwikkeld, zodat zowel patiënten als medische professionals veilig kunnen inloggen. De inlogpagina communiceert effectief met de backend en maakt gebruik van het sessiebeheer dat we in de vorige sprint hebben opgezet. Hierdoor blijft een gebruiker ingelogd gedurende zijn sessie, en kunnen toekomstige acties worden gekoppeld aan het ingelogde account.

Om de navigatie binnen het product te verbeteren, hebben we een navigatiebalk geïmplementeerd. Deze stelt gebruikers in staat om eenvoudig tussen verschillende pagina’s te navigeren. Daarnaast hebben we rolgebaseerde authenticatie toegevoegd, waardoor medische professionals toegang krijgen tot extra opties zoals het inzien van patiëntgegevens en gezondheidsanalyses, terwijl patiënten alleen de functies zien die voor hen relevant zijn. Dit maakt de interface overzichtelijk voor patiënten en biedt medische professionals de tools die zij nodig hebben.

Het implementeren van de rolgebaseerde weergave was wel uitdagend. Voor het koppelen van data aan gebruikersrollen vereiste meer technische kennis over waar deze data beschikbaar was, wat voor enige frustratie binnen het team zorgde. Ook moesten we extra aandacht besteden aan beveiliging en authenticatie om ervoor te zorgen dat alleen bevoegde gebruikers toegang krijgen tot specifieke gegevens.

Tijdens de sprint review toonden we de nieuwe functies aan de Product Owner, die positief verrast was door de vooruitgang en de mogelijkheden van de rolgebaseerde interface.   
  
Ook heb ik in deze sprint veel data in de database gezet waardoor de visualisatie van het product nog beter ging.

**Feedback PO:**

* Zorg dat het sessiebeheer volledig visueel wordt getoond, inclusief cookie- en foutafhandeling.
* Focus op een werkende pagina die compleet is met design en functionaliteit.

**Resultaat Retrospective:**

* Grote vooruitgang in backend- en frontend-integratie.
* Verbeterpunt: Meer structuur aanbrengen in teamcommunicatie tijdens langere sprints.

**Actiepunten:**

* Hele team:  
  Code Kwaliteit
* Rowin & Marcus:

Tijdsnood (inschatten hoeveel tijd er nodig is en overblijft voor project werk)

* Dannique & Milosz:  
  Issue Board Bijhouden
* Latricha & Dax:  
  Naamgeving van code

## Sprint 4: 11 november – 22 november

In deze sprint was Rowin de Scrum Master.

Rowin was niet iemand die bepaalt veel liefde had voor de Scrum methodiek dus ik verwachtte niet veel, maar om Plato te quoten 'Only those who do not seek power are qualified to hold it.'[[6]](#endnote-6)

Net zoals toen ik scrum master was heeft Rowin gelijk initiatief genomen om een indeling te maken van wat er moest gebeuren (enkel hadden we in de 4e sprint een beter beeld hiervan).

De taken die we uiteindelijk hebben opgepakt, waren voornamelijk gericht op de frontendpagina’s, zoals de Health Analytics, Registratie, en de Navbar. Daarnaast moesten we alle verslagen voor school afronden, zoals het Logboek, het Securityonderzoek, en eventuele extra STARR-verslagen indien nodig. Hierbij hadden de verslagen prioriteit omdat deze bepalen of we de rubriek volledig kunnen invullen. Dit is cruciaal om het semester te halen.

Wat betreft de verdeling van de taken hadden we het volgende afgesproken:

* **Register**: Marcus
* **Patient Analytics**:
  + Één grafiek met dummy data en een tabel: Rowin
  + Lijst met gezondheidsdata: Dannique
  + Analytics-overzicht en invoer: Latricha
* **Search Bar**: Dax
* **Logout-functionaliteit**: Milosz

Ik had de lijst met gezondheidsdata aangewezen gekregen, deze functionaliteit bied de gebruiker een lijst waarin alle mogelijke gezondheidsdata visueel weergegeven wordt en klikbaar moet zijn zodat deze gevisualiseerd kan worden.  
  
De implementatie hiervan ging sneller dan verwacht, waardoor ik extra tijd heb genomen om de testing uitgebreider te maken. Ook heb ik het op mijn genomen om de service te maken die alle data op zal halen die gebruikt zal worden voor de visualisatie, dit was een leuke en uitdagende taak.  
  
Daarnaast heeft iedereen zijn toegewezen taken op tijd afgerond. Dit betekende dat de frontendpagina’s compleet waren en we ons konden richten op het finaliseren van het product, zoals het integreren van alle patient analytics componenten met elkaar, de PO gaf in zijn feedback ook aan dat hij daar een focus op wou hebben voor de volgende sprint.

Doordat we deze sprint ook een securityonderzoek hebben moeten maken en daarnaast aan het Projectlogboek hebben moeten werken ben ik heel tevreden over het eindresultaat van deze sprint.  
  
**Feedback PO:**

* Voeg meer visuele feedback toe voor medische professionals, zoals grafieken en knoppen.
* Zorg dat health analytics een centrale rol speelt in de volgende sprint.

**Resultaat Retrospective:**

* Het team heeft concrete stappen gezet richting een werkende applicatie.
* Verbeterpunt: Meer aandacht besteden aan de afronding van taken binnen één sprint.

**Actiepunten:**

* Dannique en Marcus:  
  Rubric doorlopen (per persoon)
* Rowin en Latricha:  
  Toekomst planning
* Dax:  
  Leerdoelen progressie
* Milosz:  
  Wiki/Gitlab opschonen

## Sprint 5: 25 november – 6 december

In deze sprint was Latricha de Scrum Master.  
Deze sprint was heel wazig omdat er een grote prioriteit lag op het maken van het Projectportfolio voor feedback, hierdoor hebben we besloten om minder intensief aan het project te werken, wat ook te merken is aan de beloofde functionaliteiten deze sprint.

De taken in deze sprint waren gericht op het afronden van essentiële functionaliteiten, zoals:

* **Logout-functionaliteit:** voltooid door Milosz.
* **Zoekbalk-functionaliteit:** waarmee gebruikers patiënten kunnen zoeken op naam of andere criteria, ontwikkeld door Dax.
* **Dynamische grafieken:** deze functionaliteit maakt het mogelijk om per gebruiker persoonlijke data visueel weer te geven. Ontwikkeld door Mijzelf in samenwerking met Rowin

De Dynamische grafieken maken gebruik van de functie die de health data ophaalt en de selectie van een specifieke health data optie vanuit de lijst die ik in de vorige sprint geïmplementeerd heb. Door deze te verbinden aan de grafiek, in samenwerking met Rowin om de juiste format te gebruiken voor de data, is de grafiek volledig dynamisch gebaseerd op wat de gebruiker selecteert.

Dit had op de laatste dagen van de sprint nog wel aardig wat bug-fixes nodig omdat ik niet bekend was met de grafiek module, maar door goede communicatie met Rowin over de werking hiervan is alles goed gekomen.

Ondanks de tijdsdruk is het ons gelukt om alle doelen van deze sprint te behalen. De communicatie binnen het team verliep ook soepel.

De Product Owner was tevreden met de resultaten, enkel was er wel een wissel van Product Owner omdat Mohammed meerdere persoonlijke omstandigheden had, dit was frustrerend maar een bekend probleem waar al vanuit gegaan was.

**Feedback PO:**

* Rond alles af en zorg dat je alle testing hebt.
* Implementeer extra security als er tijd over is.

**Resultaat Retrospective:**

* We hebben alles zo goed als af in de frontend.
* We moeten nu alles een beetje afronden

**Actiepunten:**

* De rubric afronden (iedereen)
* Refactoring van laatste veranderingen (iedereen)
* Testing finaliseren.

## Sprint 6: 9 December – 20 december

In deze sprint was Milosz de Scrum Master.

Milosz voert de taken over het algemeen wel uit, enkel omdat er zoveel zieken zijn deze sprint gecombineerd met het feit dat iedereen eigenlijk alle tijd gebruikt om aan het projectportfolio te werken zorgt ervoor dat er weinig issues gemaakt werden.

Ik, Milosz en Rowin zijn 50% van de sprint ziek geweest, Latricha en Marcus zijn een halve week ziek geweest. Dit was niet optimaal voor de al kleine hoeveelheid tijd die we wouden besteden aan de applicatie.

De gehele sprint verliep vrij stilletjes doordat er veel zieken waren en dat we vooral bezig waren met het afronden van het Projectlogboek en het protfolio in elkaar zetten, hierdoor lieten we een aantal beloftes aan de PO liggen doordat we alle focus op het portfolio legde.

Ik ben wel heel tevreden over mijn gemaakte diagrammen, welke te vinden zijn in dit logboek.

Verder was de PO tevreden over ons product, en gaf hierbij minimale verbeterpunten die nog geïmplementeerd konden worden.

**Feedback PO:**

* Alles ziet er netjes uit, als er extra tijd was zouden jullie betere security kunnen implementeren voor het selecteren van een patiënt.

**Resultaat Retrospective:**

* Het team is tevreden over het uiteindelijke product, en de algemene samenwerking binnen het team.
* Verbeterpunt: Meer aandacht besteden aan de documentatie en rubric vanaf het begin van de sprint.

**Actiepunten:**

* Dannique: meer nadenken over wat ik wil gaan bespreken.
* Team: Vanaf dag 1 aan de rubric en documentatie werken.

# De applicatie

## Handleiding

In dit hoofdstuk wordt stap voor stap uitgelegd hoe je de applicatie gebruikt als medische professional. Aan de hand van ondersteunende afbeeldingen krijg je een overzicht van de functionaliteiten en leer je hoe je patiëntgegevens kunt bekijken en analyseren.

De handleiding richt zich op het gebruik van de cliëntkant van de applicatie. Hierdoor krijg je inzicht in hoe de ontwikkelde functies samenwerken met de gebruikersinterface.  
  
Inhoud van dit hoofdstuk

1. **Overzichtspagina**: Een overzicht van patiënten en hoe je een specifieke patiënt selecteert.
2. **Patiëntgegevens**: Hoe je gedetailleerde informatie van een geselecteerde patiënt bekijkt en analyses uitvoert.
3. **Grafiek Overzicht**: Visualisatieopties en grafiekinstellingen voor gezondheidsparameters.
4. **Tabel Overzicht**: Weergave van gegevens in tabelvorm en sorteeropties.

### Overzichtspagina

Na het inloggen als een medische professional word je doorgestuurd naar de Overzichtspagina van de patiënten. Hier zie je een lijst met alle patiënten die aan jou zijn toegewezen (zie afbeelding 1). Voor elke patiënt worden de volgende gegevens weergegeven:

* Naam
* Geboortedatum
* Een knop "View" om de details te bekijken.

Stappen:

1. Zoek een specifieke patiënt met behulp van de zoekbalk bovenaan.
2. Klik op de knop "View" naast de naam van de patiënt om naar de gedetailleerde gegevens van de patiënt te gaan.

*A screenshot of a computer

Description automatically generated*

*Afbeelding 1: Patiënt overzichtspagina.*

### Patiëntgegevens

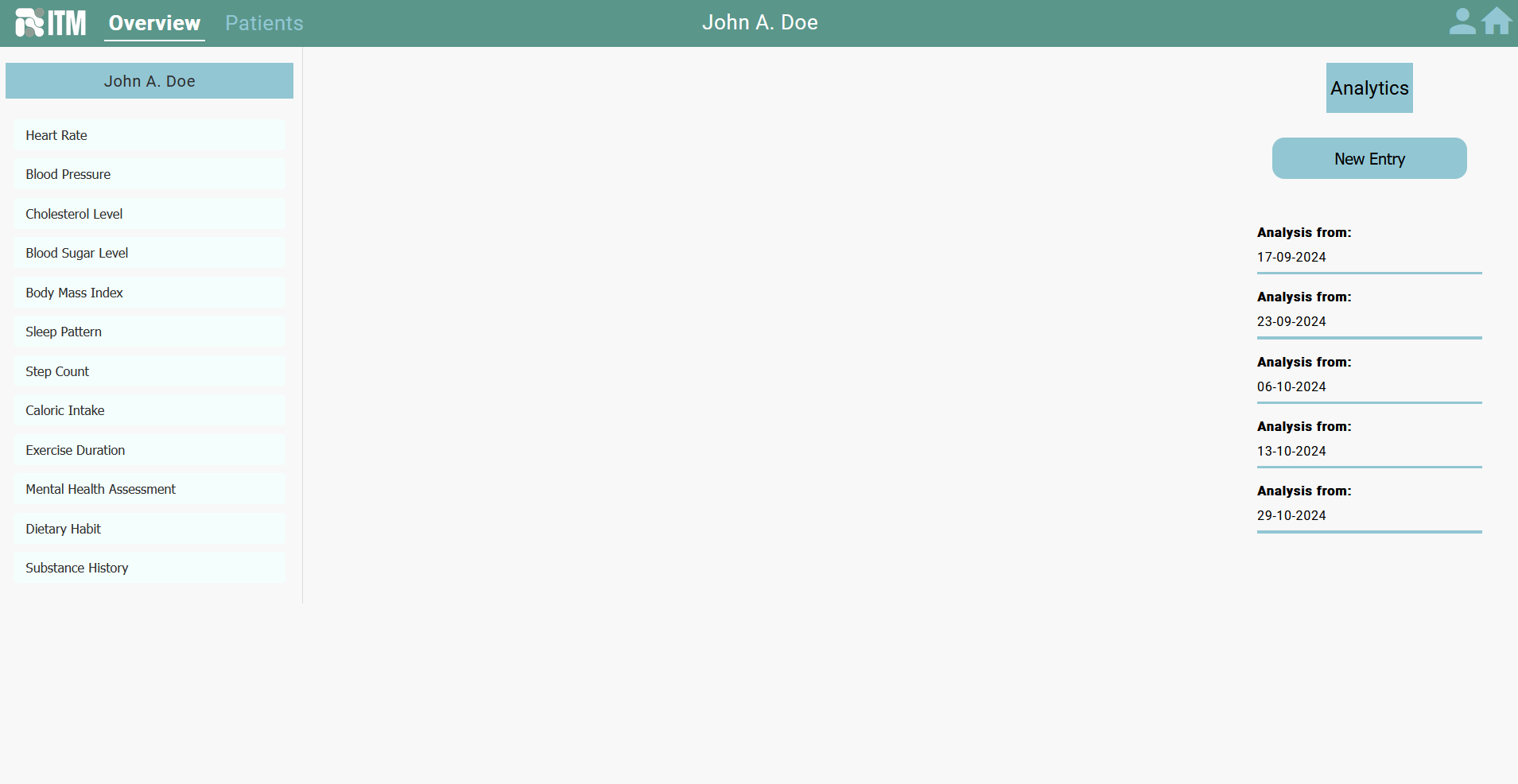
Na het selecteren van een patiënt wordt je doorgestuurd naar de Patient Analytics-pagina (zie afbeelding 2). Op deze pagina wordt de naam van de geselecteerde patiënt weergegeven in de header en in de zijbalk.

De pagina bevat de volgende elementen:

* **Gezondheidsdata Zijbalk**: Hier kun je verschillende gezondheidsstatistieken bekijken, zoals hartslag, bloeddruk, cholesterolniveau, en meer.

**Stappen:**

1. Selecteer een specifieke gezondheidsparameter in de zijbalk om meer details te bekijken.



*Afbeelding 2: Patientgegevens startpagina.*

### Grafiek overzicht

**Opties voor Visualisatie**

Bovenaan de pagina, direct boven de grafiek of tabel(zie afbeelding 3), zijn twee knoppen beschikbaar:

* **Show Graph**: Toont de gegevens als een lijn-grafiek. Dit is de standaardweergave voor gezondheidsparameters met numerieke waarden (bijv. hartslag, bloeddruk).
* **Show Table**: Toont de gegevens als een tabel. Deze optie wordt automatisch geactiveerd als de parameter geen numerieke gegevens bevat, zoals bij tekstuele metingen (bijv. Mental Health Assessment, Sleep Pattern).

De naam van de geselecteerde parameter wordt ook weergegeven in een wit vlak linksboven de grafiek of tabel, net onder de visualisatie-opties.

**Aanpassingen aan de Grafiek**

De grafiek biedt verschillende aanpassingsopties om de dataweergave beter af te stemmen op je behoeften:

* **Hoeveelheid Entries**  
  Rechtsboven in de grafiek bevindt zich een dropdown-menu waarmee je kunt selecteren hoeveel data-items je wilt weergeven. De standaardinstelling is "8", waarbij de laatste 8 metingen worden getoond. Je kunt echter ook kiezen uit:
  + **All**: Toont alle beschikbare gegevens.
  + **8, 16, 32, 64, 128**: Toont een specifiek aantal recente metingen.
* **Sorteren van Gegevens**  
  Onder de dropdown-menu vind je een **Sort By**-knop.
  + **Default (Latest)**: Toont de meest recente metingen bovenaan.
  + **Oldest**: Toont de oudste metingen eerst.  
    Wanneer je de knop selecteert, verandert de donkere kleur van de knop naar een lichtere tint om aan te geven dat de sorteeroptie is gewijzigd.

**Opmaak van de Grafiek**

De grafiek is ontworpen om duidelijke en nauwkeurige informatie weer te geven:

* **Y-as (Verticale As)**  
  Aan de linkerzijde van de grafiek zie je een reeks dalende nummers die de waarde van de gezondheidsparameter vertegenwoordigen. Deze schaal past zich automatisch aan op basis van de meetwaarden. Bijvoorbeeld: Bij hartslag is de schaal op de afbeelding ingesteld van 80 tot 180, maar dit wordt aangepast als de waarden daarbuiten vallen.
* **X-as (Horizontale As)**  
  Onderaan de grafiek worden de datums weergegeven van de metingen. Elke kolom in de grafiek vertegenwoordigt een specifieke datum, en de lijn in de grafiek verbindt deze punten om trends te tonen.

A graph on a computer screen

Description automatically generated

*Afbeelding 3: Visualizatie van de patient’s heart-rate gegevens binnen een grafiek.*

**Voorbeeld: Hartslaggegevens**

In de grafiekweergave van hartslag (zie afbeelding 3):

* De verticale as toont waarden van 80 tot 180.
* De horizontale as toont datums van de laatste 8 metingen.
* Door het aantal entries te wijzigen naar "16" of "All", wordt de grafiek uitgebreid om meer metingen te tonen (zie afbeelding 4).

A graph with a line

Description automatically generated

*Afbeelding 4: een grotere visualisatie(16 entries) van de hartslag metingen.*

### Tabel overzicht

Bij het klikken op de Show Table-optie verandert de gegevensweergave van een grafiek naar een tabel (zie afbeelding 5). De tabelweergave biedt dezelfde functionaliteit als de grafiek, maar presenteert de gegevens in een overzichtelijke, tekstuele indeling. Dit is met name handig voor parameters waarbij trends minder belangrijk zijn of waarbij tekstuele gegevens worden geregistreerd.

**Structuur van de Tabel**

De tabel bevat twee hoofdkolommen:

* **Date**: De datum van de meting.
* **[Naam van de Parameter]**: De waarde van de geselecteerde gezondheidsparameter. Dit kan bijvoorbeeld "Heart Rate" zijn, maar het verandert automatisch als je een andere parameter selecteert.

De naam van de gezondheidsparameter blijft zichtbaar in het witte vlak linksboven, net zoals bij de grafiekweergave.

**Sorteren van de Gegevens**

Net als bij de grafiekweergave biedt de tabel een Sort By-knop onder de dropdown-menu:

* **Default (Latest)**: Toont de laatste metingen eerst, met de nieuwste entry bovenaan de tabel.
* **Oldest**: Toont de oudste metingen eerst, met de oudste entry bovenaan de tabel.

Bij een wijziging in de sorteeroptie verandert de knop van een donkere naar een iets lichtere tint, zodat je direct ziet dat de sorteerinstelling is aangepast.

**Beperking van Entries**

De hoeveelheid weergegeven entries in de tabel wordt geregeld via het dropdown-menu rechtsboven. Hier kun je, net als bij de grafiek, kiezen uit: All, 8, 16, 32, 64, 128.

De standaardinstelling toont de laatste 8 entries.

**Scrollbare Tabel**

Als de geselecteerde hoeveelheid entries groter is dan **16**, wordt de tabel automatisch scrollbaar. Dit voorkomt dat er een overvloed aan gegevens in één keer zichtbaar is, waardoor de weergave overzichtelijk blijft voor de gebruiker. Je kunt door de gegevens scrollen om oudere of nieuwere metingen te bekijken.

**Weergave bij tekstuele data**

De tabelweergave toont 8 sleep pattern-metingen (zie afbeelding 6), aangezien de gegevens niet geschikt zijn voor visualisatie in een grafiek. De optie "Show Graph" is niet selecteerbaar, en de tabel bevat twee kolommen: **Date** en **Sleep Pattern**. De meest recente metingen worden standaard bovenaan weergegeven dankzij de "Latest"-sorteeroptie. Omdat er slechts 8 metingen zijn, is de volledige tabel zichtbaar zonder scrollmogelijkheid.

A screenshot of a computer

Description automatically generated

*Afbeelding 5: de hartslag metingen van de geselecteerde patient binnen een tabel.*

A screenshot of a computer

Description automatically generated

*Afbeelding 6: de sleep pattern metingen van de geselecteerde patient binnen een tabel*

# Technische toelichting

Dit hoofdstuk beschrijft de technische implementatie van het WealthCare platform. Als ontwikkelaar was ik verantwoordelijk voor zowel frontend als backend componenten, met specifieke focus op de Patient Analytics functionaliteit en de onderliggende security infrastructuur.

In de frontend lag de nadruk op het ontwikkelen van de Patient Analytics pagina met bijbehorende componenten voor data visualisatie. De backend werkzaamheden omvatten het implementeren van cruciale functionaliteiten zoals authenticatie, sessiemanagement en patiëntdata-beheer.

De ontwikkeling volgde een test-driven development aanpak, wat resulteerde in een robuuste codebase met uitgebreide test coverage. Door gebruik te maken van moderne frameworks (Spring Boot & Angular) en best practices is een schaalbare en onderhoudbare architectuur gerealiseerd.

## Frontend

Mijn deel van de frontend bestond uit de Patient Analytics pagina, met focus op de health data lijst, data ophalen voor geselecteerde patiënten en integratie van verschillende componenten.

**Sprint 1-2 (23 Sep - 18 Okt)**

* Minimale frontend betrokkenheid
* Focus lag op backend ontwikkeling
* Frontend basis werd opgezet door teamgenoten

**Sprint 3 (21 Okt - 8 Nov)**

* Patient Health Data interface ontwikkeling
* Onderzoek naar benodigde components voor visualisatie
* Opzet data structuur tussen components

**Sprint 4 (11 Nov - 22 Nov)**

* Implementatie dynamic health data buttons
* Ontwikkeling data management systeem voor grafieken
* Gewerkt aan health-data-select-list
* Integratie health data met graph components

**Sprint 5 (25 Nov - 6 Dec)**

* Frontend testing afgerond
* API endpoints gestandaardiseerd
* Frontend/backend integratie voltooid
* Focus op analytics page components:
  + - Health data lijst
    - Graph component
    - Analytics pagina layout
    - Patient analytics entries display

**Sprint 6 (9 Dec - 20 Dec)**

* Documentatie bijwerken
* Bug fixes
* Code opschoning
* Final testing

Notities:

* Sterke focus op component integratie
* Regelmatige afstemming met team over backend integratie
* Uitgebreide testing implementatie
* Gebruik van moderne Angular practices

### Patient Analytics Pagina

De **PatientAnalyticsComponent** (wealthcare-front-end/src/app/components/pages/patient-analytics/patient-analytics.component.ts), deze is verantwoordelijk voor:

1. Het tonen van de health-data-list in een sidebar

2. Het tonen van een error message als er geen patiënt geselecteerd is

3. Het beheren van de geselecteerde patiënt via de **PatientService** (wealthcare-front-end/src/app/services/patient.service.ts)

4. Het updaten van de *selectedFieldData* (wealthcare-front-end/src/app/components/pages/patient-analytics/patient-analytics.component.ts) wanneer er een veld wordt geselecteerd

De HTML structuur bestaat uit:

- Een sidebar met de health-data-list

- Een error message sectie

- Een content sectie met de grafieken/tabellen

### PatientRecords & PatientService

De **PatientRecordsComponent** (wealthcare-front-end/src/app/components/elements/patient-records/patient-records.component.ts) werkt samen met de **PatientService** (wealthcare-front-end/src/app/services/patient.service.ts) voor:

1. Het selecteren van een patiënt via de *selectPatient()* methode

2. Het doorgeven van de geselecteerde patiënt aan de service via *setSelectedPatient()*

3. Navigatie naar de analytics pagina na selectie

4. Het bewaren van de BSN en naam van de geselecteerde patiënt

Deze data wordt beschikbaar gesteld via een *BehaviorSubject* (wealthcare-front-end/src/app/services/patient.service.ts) in de service zodat andere componenten kunnen subscriben op wijzigingen.

### PatientGraphs Component

De **PatientGraphsComponent** (wealthcare-front-end/src/app/components/elements/patient-graphs/patient-graphs.component.ts) functionaliteiten:

1. Subscribe op *selectedFieldData$* van de **HealthDataService** (wealthcare-front-end/src/app/services/health-data.service.ts)

2. Verwerkt de ontvangen data voor weergave

3. Bepaalt of data numeriek is voor grafiek weergave

4. Past minimum en maximum waarden aan voor de y-axis

5. Schakelt automatisch naar tabel-view voor niet-numerieke data

De HTML biedt:

- Toggle buttons voor grafiek/tabel weergave

- Grafiek sectie met ngx-charts-line-chart

- Tabel sectie met datum/waarde kolommen

- Scroll indicator voor grote datasets

### HealthData Service & Component

De **HealthDataService** (wealthcare-front-end/src/app/services/health-data.service.ts) functionaliteiten:

1. Data ophalen via API calls met BSN in een POST

2. Formatteren van ruwe data naar bruikbaar formaat

3. Beheren van geselecteerd gezondheidsveld

4. Error handling en logging

De **HealthDataListComponent** (wealthcare-front-end/src/app/components/elements/health-data-list/health-data-list.component.ts) features:

1. Toont lijst van beschikbare gezondheidsgegevens

2. Markeert geselecteerd veld

3. Emits selectie events

4. Toont patiëntnaam in header

HTML structuur:

- Patiëntnaam header

- Lijst van klikbare velden

- Styling voor actieve selectie

### Complete analytics

Deze componenten werken samen om een complete analytics ervaring te bieden:

1. Patiëntselectie → PatientService update
2. Health data ophalen → HealthDataService
3. Veld selectie → HealthDataList Component
4. Visualisatie → PatientGraphs Component

Het systeem gebruikt observables en services voor state management en data delen tussen componenten.

## Backend

**Sprint 1 (23 Sep - 4 Okt)**

Backend infrastructuur opgezet

* + Spring Boot project basis gecreëerd
  + Database connectie geïmplementeerd
  + Basis models en DTOs gemaakt
  + Pipeline met testing opgezet
  + Layered architectuur geïmplementeerd

**Sprint 2 (7 Okt - 18 Okt)**

Login functionaliteit

* + Login systeem gebouwd
  + Password hashing toegevoegd
  + User authenticatie geïmplementeerd
  + Testing voor login routes toegevoegd
  + Integratie met frontend login

**Sprint 3 (21 Okt - 8 Nov)**

Session management

* + Sessie aanmaak/validatie gebouwd
  + User role checks toegevoegd
  + Session DTOs gemaakt
  + Testing voor session management
  + Sessies gekoppeld met frontend

**Sprint 4 (11 Nov - 22 Nov)**

Patiëntenlijst functionaliteit

* + Ophalen patiëntenlijst geïmplementeerd
  + Zoekfunctie toegevoegd
  + Patient data DTOs gemaakt
  + Testing voor patiënt routes
  + Security implementatie verbeterd

**Sprint 5 (25 Nov - 6 Dec)**

Logout functionaliteit

* + Logout route geïmplementeerd
  + Sessie beëindiging toegevoegd
  + Logout testing gecreëerd
  + Gekoppeld met frontend logout

**Sprint 6 (9 Dec - 20 Dec)**

UML Diagrammen

* + System sequence diagrams gemaakt
  + Class diagrams toegevoegd
  + Documentatie bijgewerkt

Eindopschoning en documentatie

Opmerkingen:

* Continu focus op security verbeteringen en bug fixes
* Nadruk op authenticatie, session management en patiëntgegevens
* Uitgebreide test coverage gehandhaafd
* Regelmatige integratie met frontend componenten

### Class Diagram

(zie file class\_backend in de diagrams folder)

**Uitleg:**

De backend van de applicatie bestaat uit verschillende met elkaar verbonden componenten die samen de kernfunctionaliteiten realiseren. Wanneer een gebruiker bijvoorbeeld inlogt, begint de flow bij de [LoginController](#_Login_functionaliteit), die een HTTP request ontvangt met de inloggegevens. Deze controller werkt samen met de LoginServiceImpl, die op zijn beurt het UserRepository en PasswordRepository gebruikt voor authenticatie.

Na succesvolle authenticatie wordt via de SessionServiceImpl een nieuwe sessie aangemaakt en opgeslagen in het SessionRepository. Deze sessie wordt vervolgens gebruikt voor alle vervolgverzoeken, waarbij de [SessionInterceptor](#_Session_Management) elk request controleert op een geldige sessie.

Voor het ophalen van patiëntgegevens werkt de [PatientController](#_Patient_data_functionaliteiten) samen met zowel de PatientServiceImpl als de HealthDataServiceImpl. De PatientServiceImpl haalt basis patiëntgegevens op via het PatientRepository, terwijl de HealthDataServiceImpl gezondheidsgegevens ophaalt via het HealthDataRepository.

Medische professionals hebben toegang tot extra functionaliteit via hun BIG-nummer. De MedicalProfessionalServiceImpl valideert deze toegang en werkt samen met het MedicalProfessionalRepository voor het ophalen van specifieke informatie.

Bij het uitloggen zorgt de LogoutController ervoor dat de actieve sessie wordt beëindigd via de SessionServiceImpl, die vervolgens het SessionRepository gebruikt om de sessie te verwijderen uit de database.

Deze samenhang tussen componenten zorgt voor een gestructureerde dataflow, waarbij elke component zijn eigen specifieke verantwoordelijkheid heeft binnen het grotere geheel van de applicatie.

### Login functionaliteit

(zie file sequence\_login in de diagrams folder)

**Uitleg:**

De login route in de backend van de applicatie zorgt voor de authentication van gebruikers en het aanmaken van sessions. Wanneer een gebruiker een login verzoek indient via de LoginController, wordt het verzoek doorgestuurd naar de LoginService. Deze service valideert de inloggegevens door de ingevoerde BSN of e-mail en het wachtwoord te controleren tegen de gegevens in de database.

De LoginService maakt gebruik van de UserRepository om de gebruiker op te zoeken en de PasswordRepository om het bijbehorende wachtwoord op te halen. Vervolgens wordt het ingevoerde wachtwoord gevalideerd. Als het wachtwoord correct is, wordt een nieuwe sessie aangemaakt via de SessionService. Deze service verwijdert eventuele bestaande sessies voor de gebruiker en genereert een nieuwe sessionId, die samen met de BSN en een vervaldatum wordt opgeslagen in de SessionRepository.

De LoginController retourneert uiteindelijk een cookie met de sessionId en een JSON-object met het userType naar de gebruiker. Bij een foutieve login wordt een foutmelding geretourneerd. Deze route zorgt voor een veilige en efficiënte manier om gebruikers te authenticeren en sessies te beheren.

### Session Management

(zie file sequence\_validation in de diagrams folder)

**Uitleg:**

De /role query in de backend van de applicatie zorgt voor het ophalen van de gebruikersrol op basis van een geldige sessie. Wanneer een gebruiker een verzoek indient via de UserController voor de endpoint /user/role, wordt het verzoek eerst onderschept door de SessionInterceptor om de sessie te valideren.

De SessionInterceptor controleert de cookies in het verzoek om de sessionId te vinden. Vervolgens roept het de SessionService aan om de sessie te verifiëren en de bijbehorende BSN op te halen. De SessionService gebruikt de SessionRepository om de sessiegegevens uit de database op te halen en valideert de sessie.

Na succesvolle validatie haalt de SessionService de gebruikersinformatie op via de UserRepository en bepaalt het type gebruiker. Deze informatie wordt teruggestuurd naar de SessionInterceptor, die de BSN en het gebruikersroltype toevoegt aan de verzoekattributen. Als de sessie niet geldig is, retourneert de SessionInterceptor een foutmelding met de status 401 (Unauthorized). Bij onverwachte validatiefouten wordt een status 500 (Internal Server Error) geretourneerd.

### Patient data functionaliteiten

(zie file sequence\_health\_data in de diagrams folder)

**Uitleg:**

De /patient/health-data endpoint in de backend van de applicatie zorgt voor het ophalen van gezondheidsgegevens op basis van een BSN. Wanneer een gebruiker een verzoek indient via de PatientController, wordt het verzoek eerst onderschept door de SessionInterceptor om de sessie en gebruikerstype te valideren.

Na succesvolle validatie stuurt de PatientController het verzoek door naar de HealthDataService. Deze service gebruikt de HealthDataRepository om de gezondheidsgegevens van de patiënt op te halen uit de database. Voor patiënten wordt automatisch hun eigen BSN gebruikt, terwijl medische professionals een specifieke BSN kunnen opgeven.

De HealthDataRepository voert een geoptimaliseerde query uit om alle relevante health data op te halen. Deze gegevens worden vervolgens door de HealthDataService verwerkt en gestructureerd in een formaat dat geschikt is voor frontend visualisatie. Bij een succesvol verzoek wordt een list van health data geretourneerd, bij een mislukt verzoek of wanneer er geen data gevonden wordt, retourneert de controller een 404(not found) HTTP-status.

## Security

### Role validation

**Gitlab issue:** <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/issues/6>

(zie file sequence\_validation in de diagrams folder)

**Issue:** naast het niet toegankelijk maken van de pagina’s kan de gebruiker mogelijk nog gebruik maken van bepaalde query’s ook al zou hij hier geen rechten toe hebben.

**Oplossing:** door bij elke query een preHandle te gebruiken die de users rechten valideert zou de user geen gebruik moeten kunnen maken van query’s die niet toegestaan zijn.

**Uitleg:** zie [8.2.2](#_Validation_bij_role)

### Logout functionaliteit

**Gitlab issue:** <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/issues/7>

(zie file sequence\_logout in de diagrams folder)

**Issue:** Het ontbreken van een logout functionaliteit creëert een security issue omdat actieve sessies toegankelijk blijven tot ze automatisch verlopen. Dit betekent dat ongeautoriseerde gebruikers toegang kunnen krijgen tot het systeem via een nog actieve sessie.

**Oplossing:** door de logout functionaliteit te implementeren kan de gebruiker zijn actieve sessie verwijderen.

**Uitleg:**De logout route in de backend van de applicatie zorgt voor het veilig beëindigen van gebruikerssessies. Wanneer een gebruiker een logout verzoek indient via de LogoutController, wordt eerst de sessionId uit de cookies van het verzoek gehaald. Deze sessionId is essentieel voor het identificeren van de actieve sessie die beëindigd moet worden.

De LogoutController maakt gebruik van de SessionService om de sessie te verwijderen uit de database. Dit gebeurt via de deleteSessionBySessionId methode, die ervoor zorgt dat de sessie permanent wordt verwijderd. Na succesvolle verwijdering van de sessie, creëert de controller een nieuwe cookie met een levensduur van nul (Duration.ZERO) om de bestaande sessionId cookie te overschrijven en ongeldig te maken.

De controller retourneert vervolgens een succesbericht samen met de header die de cookie verwijdert. Bij fouten tijdens het uitlogproces, zoals een ontbrekende sessionId of databasefouten, wordt een gestandaardiseerde foutmelding teruggegeven. Deze route waarborgt dat gebruikers veilig kunnen uitloggen en dat hun sessiegegevens correct worden opgeruimd, wat essentieel is voor de beveiliging van de applicatie.

## Code kwaliteit

### Facade Pattern

A screenshot of a computer

Description automatically generated

**Beschrijving:**

LoginServiceImpl fungeert als facade omdat deze complexe interacties tussen repositories verbergt.

Hierdoor biedt het een vereenvoudigde interface voor de controller en beheert het alle transacties en businesslogica.

### Strategy Pattern

A screenshot of a program

Description automatically generated

**Beschrijving:**

SessionService definieert de strategy-interface die SessionServiceImpl implementeert

Dan gebruikt SessionInterceptor de strategie waardoor verschillende sessiebeheermethoden mogelijk zijn. Deze methoden zijn ook flexibel in te zetten bij toekomstige toevoegingen.

## Backend Testing

A screen shot of a computer

Description automatically generated

### Controller testing

**LoginController Tests (E2E/Integration)**

* **Succesvol Inloggen**: Test of een gebruiker kan inloggen met geldige inloggegevens en een sessie krijgt
* **Mislukt Inloggen**: Test of inloggen faalt met ongeldige wachtwoorden
* **Ongeldige Input**: Test verschillende ongeldige inloggegevens formaten (property-based)

A screenshot of a computer

Description automatically generated

**LogoutController Tests (E2E/Integration)**

* **Succesvol Uitloggen**: Test of een gebruiker succesvol kan uitloggen met een geldige sessie
* **Ongeldig Uitloggen**: Test of uitloggen faalt met een ongeldige sessie
* **Sessie Formaat**: Test verschillende sessie ID formaten voor consistente foutafhandeling

**A screenshot of a computer program

Description automatically generated**

**PatientController Tests (E2E/Integration)**

* **Zoeken op Naam**: Test of patiënten gevonden kunnen worden op naam
* **Zoeken op BSN**: Test of een specifieke patiënt gevonden kan worden op BSN
* **Lege Zoekresultaten**: Test correcte afhandeling van zoekopdrachten zonder resultaten
* **Patiëntenlijst per MP**: Test ophalen van patiënten gekoppeld aan een arts
* **Lege Patiëntenlijst**: Test correcte afhandeling wanneer een arts geen patiënten heeft

**A screenshot of a computer program

Description automatically generated**

**UserController Tests (E2E/Integration)**

* **Gebruikersrol Ophalen**: Test of de juiste rol wordt teruggegeven voor een ingelogde gebruiker
* **Ongeldige Sessie**: Test foutafhandeling bij het ophalen van een rol met ongeldige sessie
* **Autorisatie Controle**: Test of toegang wordt geweigerd voor ongeautoriseerde gebruikers

A screenshot of a computer

Description automatically generated

**SessionController Tests (E2E/Integration)**

* Succesvolle sessie verwijdering
* Ongeldige sessie afhandeling
* Randgevallen (lege input)

A screenshot of a computer

Description automatically generated

### Service testing:

**LoginService Tests (Unit)**

* **Succesvolle Login**: Test of een gebruiker kan inloggen met geldige logingegevens
* **Validatie Wachtwoord**: Test of een wachtwoord correct wordt gevalideerd tegen de opgeslagen hash
* **Ongeldige Credentials**: Test foutafhandeling bij ongeldige inloggegevens
* **Email/BSN Formaatvalidatie**: Test verschillende formaten van inloggegevens

**A screenshot of a computer

Description automatically generated**

**SessionService Tests (Unit)**

* **Sessie Aanmaken**: Test of een nieuwe sessie correct wordt aangemaakt
* **Sessie Validatie**: Test of een bestaande sessie correct wordt gevalideerd
* **Sessie Verwijderen**: Test of een sessie succesvol wordt verwijderd
* **Gebruikersrol Ophalen**: Test of de juiste gebruikersrol wordt teruggegeven

**A screenshot of a computer

Description automatically generated**

**PatientService Tests (Unit)**

* **Patient Zoeken**: Test zoekfunctionaliteit op naam en BSN
* **Validatie Zoekopdracht**: Test validatie van zoekparameters
* **Patiëntenlijst Ophalen**: Test ophalen van patiënten voor een arts
* **BIG Nummer Validatie**: Test validatie van BIG nummers

**A screenshot of a computer

Description automatically generated**

**HealthDataService Tests (Unit)**

* **Gezondheidsdata Ophalen**: Test ophalen van gezondheidsdata voor een patiënt
* **Data Formaat**: Test correcte formattering van gezondheidsdata
* **Lege Resultaten**: Test afhandeling van niet-bestaande gezondheidsdata

*A screenshot of a computer

Description automatically generated*

### Utility

**Data Generatie (hulp bij testen en niet zelf tests)**

**TestLogin**: Genereert test login data

* Valid/invalid login credentials
* Gehashte wachtwoorden voor tests
* BSN/email combinaties

**TestUser**: Maakt test gebruikers

* Verschillende user types (medical\_professional, patient)
* Valid/invalid user data
* Test gebruikers met specifieke eigenschappen

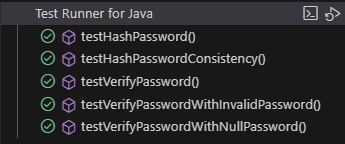
**TestSession**: Creëert test sessies

* Valid/invalid sessie tokens
* Sessies met verschillende user types
* Expired/active sessies

**Helper Utils**

**TestPassword**: Tests voor de password verificatie en hashing

* Standaard test waardes
* Random data generatie
* Edge case test data

****

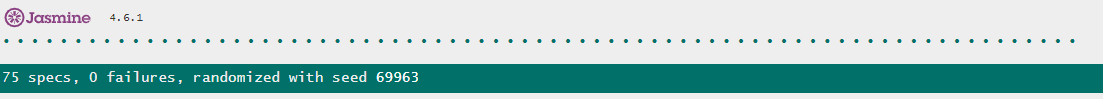
**TestRoleBasedAccess**: Validatie helpers voor role access

* Input validatie checks
* Format validatie
* Boundary testing helpers

A screenshot of a computer program

Description automatically generated

## Frontend Testing



### Component Tests (Unit/Integration)

**HeaderComponent**:

* + Navigatie links correct weergeven per gebruikersrol
  + Login/logout knoppen conditionale weergave
  + Patient naam weergave validatie

A white background with black text

Description automatically generated**LoginComponent**:

* + Validatie van inlogformulier
  + Error handling bij ongeldige credentials
  + Redirect na succesvolle login

**A close up of text

Description automatically generated**

**AnalyticsComponent**:

* + Gezondheidsdata selectie
  + Grafiek rendering
  + Data transformatie validatie

**A screenshot of a computer

Description automatically generated**

**HealthDataListComponent:**

* Patient informatie selectie
* Inladen van gezondheidsdata

**A white background with black text

Description automatically generated**

### Service Tests (Unit)

**PatientService**:

* + Patiëntdata ophalen
  + Zoekfunctionaliteit
  + Error handling

**A screenshot of a computer

Description automatically generated**

**RoleService**:

* + Gebruikersrol ophalen
  + Authenticatie status updates
  + Error handling bij netwerk problemen

A white background with black text

Description automatically generated

**HealthDataService:**

* Gezondheidsdata ophalen
* Gezondheidsdata formatteren
* Error handling bij verschillende returns.

**A screenshot of a computer

Description automatically generated**

### E2E Tests (Cypress)

**Patient Analytics Flow:**

* Navigatie naar analytics
* Data visualisatie
* Interactie met tablen en grafiek visualizatie

**A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated**

# Technische reflectie

## Dit ging goed

Tijdens mijn afstudeerproject zijn er verschillende aspecten goed verlopen. Een van debelangrijkste successen was de implementatie van de login functionaliteit. Dit blijkt uit de commits waarin ik de login route heb opgezet, getest en uiteindelijk gemerged naar de main branch. De samenwerking met het team verliep ook soepel, zoals blijkt uit de vele code reviews en merge requests die ik heb goedgekeurd en waar ik feedback op heb gegeven. Daarnaast heb ik een gestructureerde aanpak gehanteerd voor het opzetten van de backend infrastructuur, wat heeft geleid tot een robuuste en onderhoudbare codebase.

**Bewijzen:**

* Commentaar van Marcus Kaagman: "Very cool, I'd say it's all presto" en "Really just hitting all rubric required tests in this file huh? Nice bwahaha"[[7]](#endnote-7)
* Commentaar van Milosz Nawrocki: "The code looks awesome. It looks like some guy with 30 years of experience in his 20 has build it. Approved..."[[8]](#endnote-8)
* Commentaar van Rowin Schoon: "Your code looks very well structured, even if Java is not my language I can tell what is going on. Good use of Jsdocs. No further comments."[[9]](#endnote-9)
* Commentaar van Latricha Seym: "The JSdoc describes really well what each function does and what it returns, props to that! Great use of constants as well. The rest of the code looks clean to me!"[[10]](#endnote-10)

Deze positieve feedback van mijn teamleden bevestigt dat mijn code goed gestructureerd en begrijpelijk was, zelfs voor teamleden die minder bekend zijn met Java. Dit heeft bijgedragen aan een efficiënte samenwerking en een succesvolle implementatie van de functionaliteiten.

## Dit kon beter

Bij de start van het project heb ik niet voldoende onderzoek gedaan naar de versies van de verschillende dependencies die ik wilde gebruiken. Hierdoor liep ik tegen problemen aan met de Spring Boot versie en de bijbehorende security implementaties. Dit heeft geleid tot extra werk omdat ik de project setup opnieuw moest doen met de juiste versies.

Een ander punt van verbetering was de implementatie van de sessiemanagement. In eerste instantie probeerde ik een eigen token-gebaseerd systeem te implementeren, wat eigenlijk een hybride werd tussen een session en een JWT token. Dit was niet de juiste aanpak en zorgde voor onnodige complexiteit. Na feedback van Gerbrand, andere studenten en verder onderzoek bleek dat een eenvoudige cookie-based session met Spring's ingebouwde functionaliteit veel efficiënter en veiliger zou zijn.

**Bewijzen:**

* Commit: fixed connection with frontend...[[11]](#endnote-11) file: src/main/java/com/project/health\_analytics/controller/LoginController.java
* Starr 1.5 - Bij het opzetten van mijn backend de verkeerde versie van Java gebruikt waardoor de pipeline en andere features niet bruikbaar waren.

## Hier moet ik aan werken

Tijdens mijn project heb ik ervaring opgedaan met het ontwikkelen van custom oplossingen voor authenticatie en autorisatie, maar ik wil mij verder specialiseren in het Spring Boot framework. Specifiek wil ik mij richten op twee hoofdgebieden:

Ten eerste ga ik via LinkedIn Learning en zelfstudie Spring Boot cursussen volgen. Spring Boot biedt veel ingebouwde functionaliteiten die ik nog niet volledig benut, zoals de security filters en role-based access control via annotaties. Door deze kennis op te doen kan ik in toekomstige projecten efficiënter werken en de volledige kracht van het framework benutten.

Ten tweede wil ik mij verdiepen in JWT (JSON Web Tokens) implementatie. Hoewel ik nu een custom session management systeem heb gebouwd, biedt JWT een modernere en schaalbaardere oplossing voor authenticatie. Door praktische ervaring op te doen met JWT kan ik toekomstige projecten beter beveiligen volgens de laatste industry standards.

## Zo zou een student zich moeten voorbereiden

Voor studenten:

Bij het maken van functionaliteiten zorg eerst dat je een kleine basis hebt waarvan je weet dat het werkt, waarna je dat opbouwt. Als je begint met het beeld van een volledige implementatie en je gaat ergens de fout in kan het gebeuren dat je niet weet waar het fout gaat en dan is je gehele implementatie ‘fout’, het scheelt enorm veel tijd om dit dus stapsgewijs op te bouwn.

Voor docenten:

Het zou fijn zijn om een Product Owner te hebben die geen verstand heeft van technische zaken, zelfs al moeten we bijvoorbeeld backend functionaliteiten beschrijven voor goedkeuring. Dit is omdat we ons op deze manier beter kunnen voorbereiden op managers of dergelijke waaraan wij onze bezigheden moeten uitleggen. Door veel technische kennis toe te staan in de sprint reviews, en ook technische vragen te laten stellen heb ik naar mijn beeld minder ervaring om met niet-technische leidinggevende uitleg te geven dan gewenst.

# Appendices(FORMAT, EN POP LEERDOELEN)

Bewijs uit Semester 3:

1. **Security implementatie**

* Security samenvatting + presentatie (7/10/24 - 9/10/24)
* XSS-aanvallen onderzoek en prototype
* Implementatie van beveiligingsmaatregelen in de backend

1. **Testing Methodologies**

* Productvergelijking tussen JUnit 5 en TestNG
* Implementatie van verschillende testmethodologieën
* Documentatie van testresultaten en vergelijkingen

1. **Aanvullende Vaardigheden:**

* Scrum Master rol in Sprint 1 (23/09/24 - 4/10/24)
* GitLab repository management
* Team communicatie en coördinatie
* Code review processen

Note: Alle aanvullende bewijsstukken zijn gedocumenteerd in de rubric in deze folder en kunnen worden ondersteund door commit history en merge request feedback.

1. *GitLab van Team Wealthcare*. (n.d.). <https://gitlab.fdmci.hva.nl/groups/ad-software-development-2425/201/team-6/-/wikis/home> [↑](#endnote-ref-1)
2. *Issue Template frontend · GitLab*. (n.d.). https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-front-end/-/tree/main/.gitlab/issue\_templates [↑](#endnote-ref-2)
3. *Issue Template backend · GitLab*. (n.d.). https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/tree/main/.gitlab/issue\_templates [↑](#endnote-ref-3)
4. *Team Wealthcare coding standards · GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/groups/ad-software-development-2425/201/team-6/-/wikis/coding-standards> [↑](#endnote-ref-4)
5. *Team Wealthcare DoD· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/groups/ad-software-development-2425/201/team-6/-/wikis/definition-of-done> [↑](#endnote-ref-5)
6. *A quote by Plato*. (n.d.). <https://www.goodreads.com/quotes/10158279-only-those-who-do-not-seek-power-are-qualified-to> [↑](#endnote-ref-6)
7. *Team Wealthcare backend Dannique Merge request 6 comment· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/merge_requests/6#note_1078886> [↑](#endnote-ref-7)
8. *Team Wealthcare backend Dannique Merge request 6 comment· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/merge_requests/6#note_1078814> [↑](#endnote-ref-8)
9. *Team Wealthcare backend Dannique Merge request 4 comment· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/merge_requests/4#note_996994> [↑](#endnote-ref-9)
10. *Team Wealthcare backend Dannique Merge request 4 comment· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/merge_requests/4#note_997101> [↑](#endnote-ref-10)
11. *Team Wealthcare backend Dannique Commit· GitLab*. (n.d.). <https://gitlab.fdmci.hva.nl/ad-software-development-2425/201/team-6/wealthcare-back-end/dannique/-/commit/2528cc07cec170d377e2ff8f64c516808f4a1247> [↑](#endnote-ref-11)