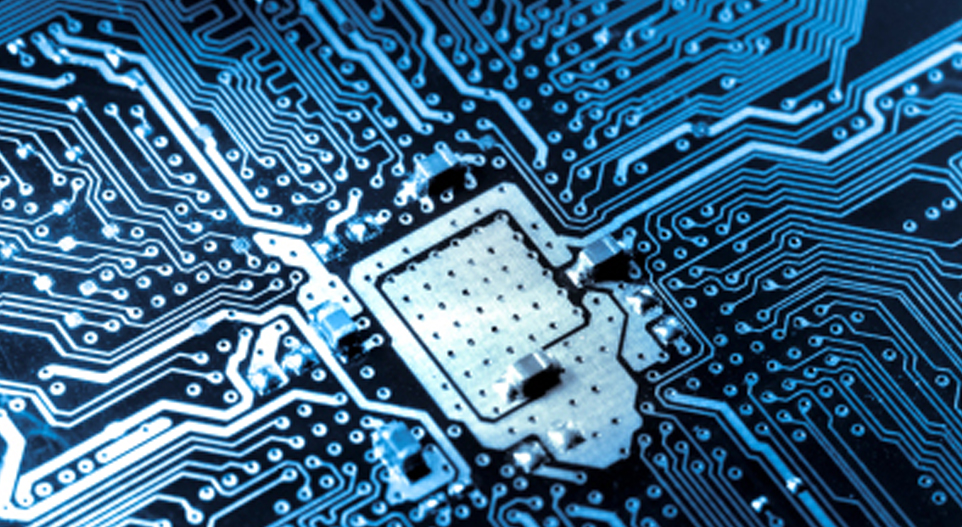
Walden Medical Center



|  |
| --- |
| ***Gruppe: Nikolaj Dyring, Ian Frost, Ahmed Dönmez*** |

Indhold

[Description 3](#_Toc492886662)

[IT Strategy 3](#_Toc492886663)

[Walden Medical Center 3](#_Toc492886664)

[Inception Phase 3](#_Toc492886665)

[Use Cases 3](#_Toc492886666)

[Use Case Diagram 4](#_Toc492886667)

[Use Case Texts 5](#_Toc492886668)

[Domain Model 7](#_Toc492886669)

[Design Sequence Diagram 8](#_Toc492886670)

[Design Class Diagram 9](#_Toc492886671)

# Description

Vi har i perioden fra d. 28/8 – 15/9 beskæftiget os med projektet Walden Medical Center, hvor vi skulle oprette et IT-system som skulle automatisere nogle af de funktioner i virksomheden, som gjorde de var bagud i forhold til deres konkurrenter. Vi har via interviews fået af vide kundens krav og opfyldt dem.

I vores beskæftigelse med projektet har vi haft 3 interviews med Walden Medical Center. I vores første interview fik vi kundens første krav om hvad IT-systemet skulle indeholde.

I vores andet interview fik vi en mere detaljeret forklaring om hvordan patient proceduren fungerer. Vi fik på den måde også vores use cases mere specialiseret.

I det tredje interview fik vi feedback på nogle af de ideer vi havde. Her fandt Walden Medical Center ud af at deres IT-system også skulle tjekke om deres patient var forsikret. Dette skulle tilføjes til vores IT-System. Funktionen ”Check Insurance” blev også tilføjet her, hvilket gjorde at vi skulle opdatere alle vores artifacts.

Programmer vi vil beskæftige os med under projektforløbet er følgende: Microsoft Visual Studio, Microsoft Visio, GitHub, Microsoft Office Word

Det endelige IT-system skal kunne:

* Registrer patienter
* Udstede hospitalkort ID
* Lave aftaler (booking)
* Opdatere Medical Records
* Oprette regning for patientens behandling

# IT Strategy

Vi tager udgangspunkt i den forretningsanalyse vi har fået af kundens forretningsanalytiker. IT-strategien er udarbejdet på baggrund af analysens kriterier.

## Walden Medical Center

*Programming Language:*

* C#

*Architecture:*

* MVVM

*Non-Functional Requirements*

* User Friendly
* Security

# Inception Phase

## Use Cases

* Register Patient

Vi har Use Casen Register Patient, da Walden Medical Center stadig har deres patient record i papirform og det hele foregår manuelt. Walden Medical Center Vores system skal gøre det hele nemmere, overskueligt og automatiseret.

## 

## Use Case Diagram

Use Case Diagram illustrer hvem der interagere med de forskellige use cases.  
På højre side af vores scope (firkanten) ser man den primære aktør, som bruger den funktion vi skal skabe. På venstre side af vores scope ser vi vores sekundære aktør. Vores scope skal illustrerer vores use cases, og hvad vores IT-system skal indeholde.



## Use Case Texts

Når man laver Use Case tekst illustrere man kundens krav og hvad man skal gøre for opnå opfyldelse af kundens krav. Dette gør det nemmere for os at lave vores Domain Model og use case Diagram.   
Use Case tekst bliver oprettet ud fra ens case, som kommer via interviews.

|  |  |
| --- | --- |
| Use Case | Register Patient |
| ID: | UC\_01 |
| Prioritet: |  |
| Summary: | Opret patient med personlig information for at oprette Hospital ID. |
| Primary Actor: | Registration Clerk |
| Secondary Actor: | Insurance Company & Bank |
| Stakeholders: | * Patienten: Mangler en service fra hospitalet * Medical Staff: har brug for patientens ID for at give en service * Accounting: Skal have patientens forsikringsoplysninger. |
| Generalization: |  |
| Include: | Issue Hospital Card, Verify Insurance Plan |
| Extend: | Verify Credit card |
| Pre-Conditions: | Personlig information skal nedskrives |
| Trigger: | Patienten har brug for behandling |
| Normal Flow: | 1. Registrerings kontoristen får patientens forsikringskort, og tjekker om forsikringen er valid ved at tjekke det i systemet. 2. Registrerings kontoristen skriver eller opdatere patientens personlige oplysninger:  * Skriver eller opdatere Patientens CPR Nummer * Skriver eller opdatere Patientens Navn * Skriver eller opdatere Patientens Adresse * Skriver eller opdatere Patientens Telefon nummer * Hvis patienten er under 18 skriver eller opdatere registreringskontoristen patientens tætteste familiemedlems Navn, adresse og telefon nummer.  1. Hvis Patienten ikke har en forsikring, registrere registrerings kontoristen patientens kreditkortoplysninger. 2. Registrerings kontoristen giver patienten et hospital kort. |
| Alternate Flow: | Hvis patienten allerede er registreret, kan han ikke blive registreret igen. |
| Post-Conditions: | Patienten er blevet oprettet. |
| Author: | Nikolaj Dyring, Ian Frost & Ahmed Dönmez |
| Revision & Date: | 04/09-2017 |

|  |  |
| --- | --- |
| Use Case | Make Appointment |
| ID: | UC\_02 |
| Priority:: |  |
| Summary: | Patienten kan få en tid, for at få noget hjælp. Det kunne være f.eks. diagnose, medicin, scanning osv. |
| Primary Actor: | Registration Clerk, Appointment Clerk, Referral Source |
| Secondary Actor: |  |
| Generalization: |  |
| Include: |  |
| Extend: |  |
| Pre-Conditions: | Personen skal være registreret som patient, for at kunne få en appointment. Udover det SKAL de have et Hospital ID-kort. |
| Trigger | En patient skal have en tid. |
| Normal Flow: | Patienten viser sit ID og spørg om en ledig tid. Derefter finder Clerken en ledig tid, og aftaler med patienten. |
| Alternate Flow: |  |
| Post-Conditions: | Aftale er blevet oprettet til patienten. |
| Author: | Nikolaj Dyring, Ian Frost & Ahmed Dönmez |
| Revision & Date: | 30/08-2017 |

## Domain Model

En domain model illustrerer værdifulde conceptual objects i problem domain. (problem domain er vores use cases. Vi ønsker altså at kigge på de problemer der kan opstå inden for dette scope). Ud fra vores case tekst finder vi de navneord som enten skal være en class eller en attribute i vores domain model.



## Design Sequence Diagram

Design Sequence Diagram er en realisering af vores use cases. Dens formål er at se hvilke beskeder vores primære aktør sender til systemet, og hvilke beskeder vores system returnere.

I Design Sequence diagram ”åbner” vi den såkaldte ”black box” og ser hvordan de forskellige objekter i systemet arbejder sammen med hinanden.



## Design Class Diagram

Når man skal lave et Class Design Diagram kigger man på Design Sequence Diagram her ser man hvilke klasser der sender hvilke metoder og hvilke klasser der har referencer til hinanden.