# Assignment 3

Steven Schenk, Cees-Jan Nolen, Robin Bakker, William de Visser, Robert Kraaijeveld – Groep 1 INF2C

Inhoudsopgave

[Assignment 3 1](#_Toc432363775)

[A. 3](#_Toc432363776)

[B. 9](#_Toc432363777)

[C. 11](#_Toc432363778)

[Implementation plan 11](#_Toc432363779)

[Doel van het project 11](#_Toc432363780)

[Indeling 11](#_Toc432363781)

[Timeline (les)week 7-10 11](#_Toc432363782)

[Timeline (les)week 11-14 11](#_Toc432363783)

[Timeline (les)week 15-16 11](#_Toc432363784)

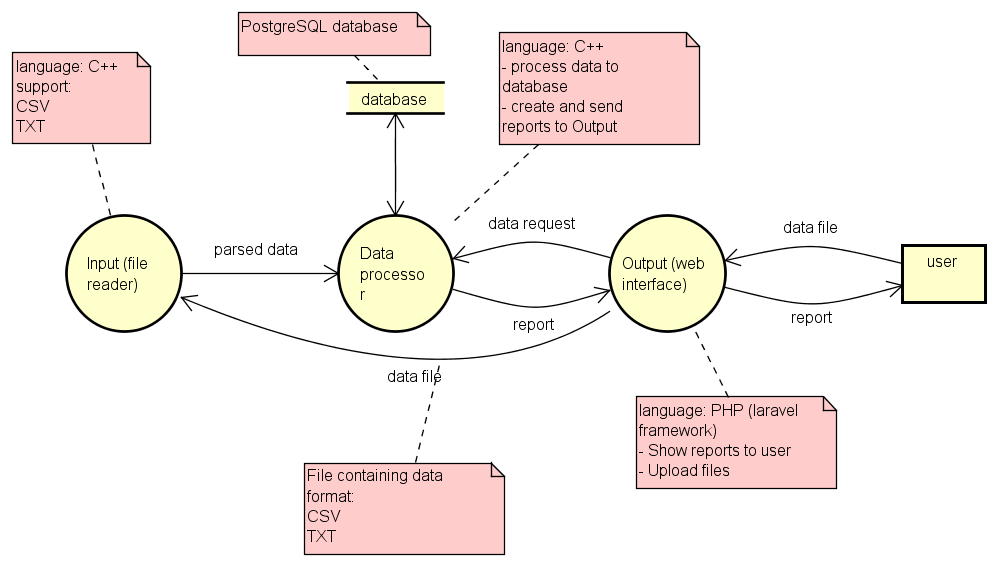
[Rolverdeling en verantwoordelijkheden 11](#_Toc432363785)

[Technieken 12](#_Toc432363786)

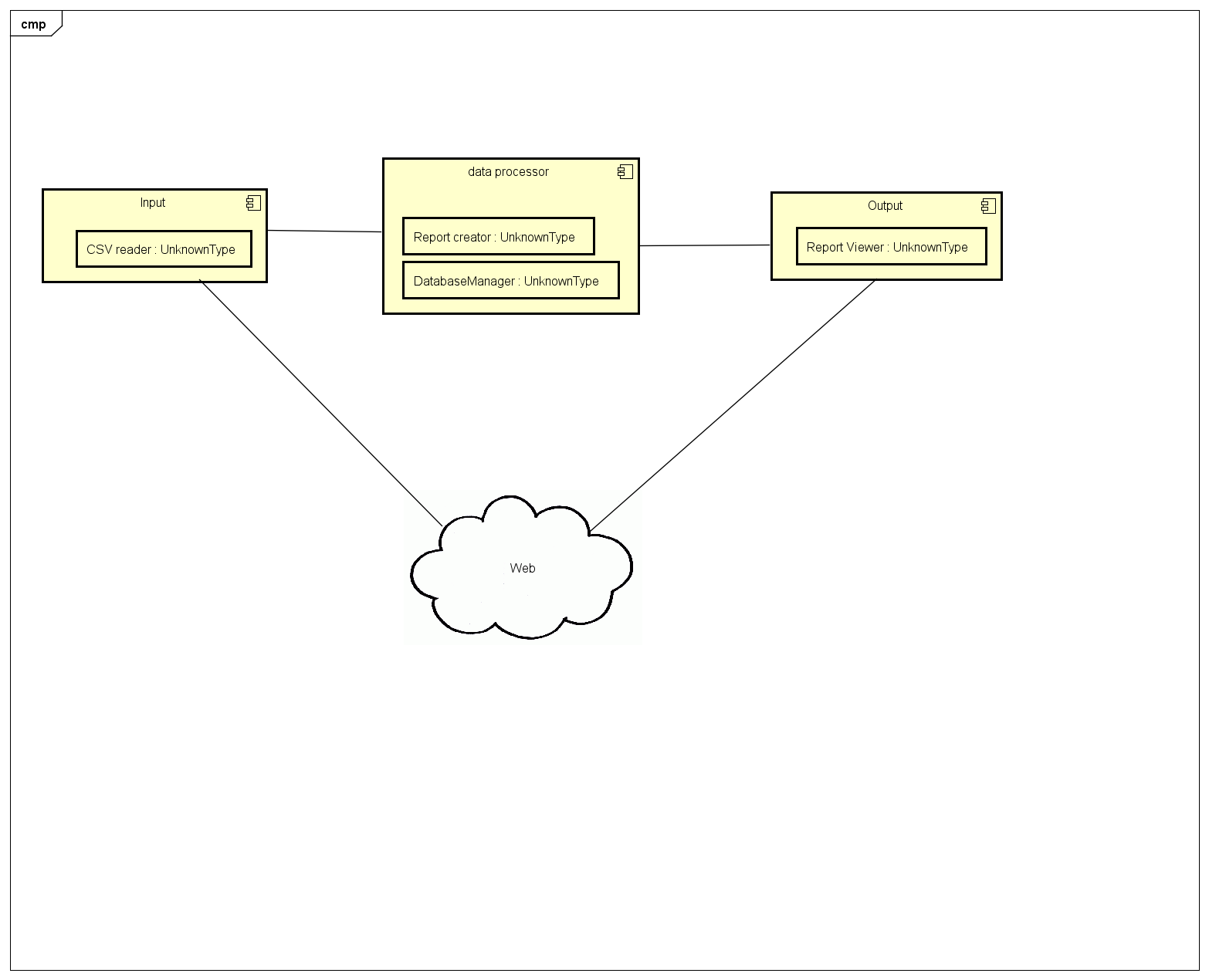
[Communicatie 12](#_Toc432363787)

## A.

Wij hebben naast onze al bestaande dfd/contextdiagrammen die u hieronder kunt vinden sequence diagrammen gemaakt, die elk bij 1 van onze Use Cases horen. (Zie voor deze use cases Assignment .)

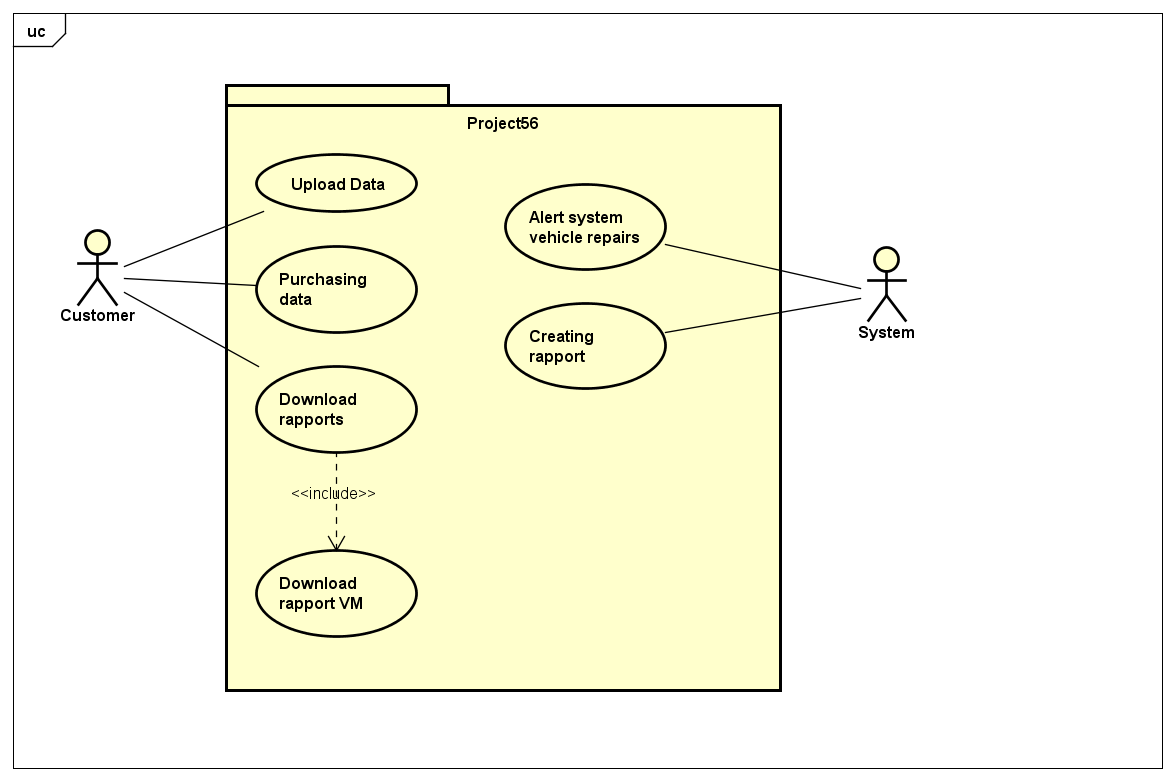


Figuur 1: DFD van het systeem

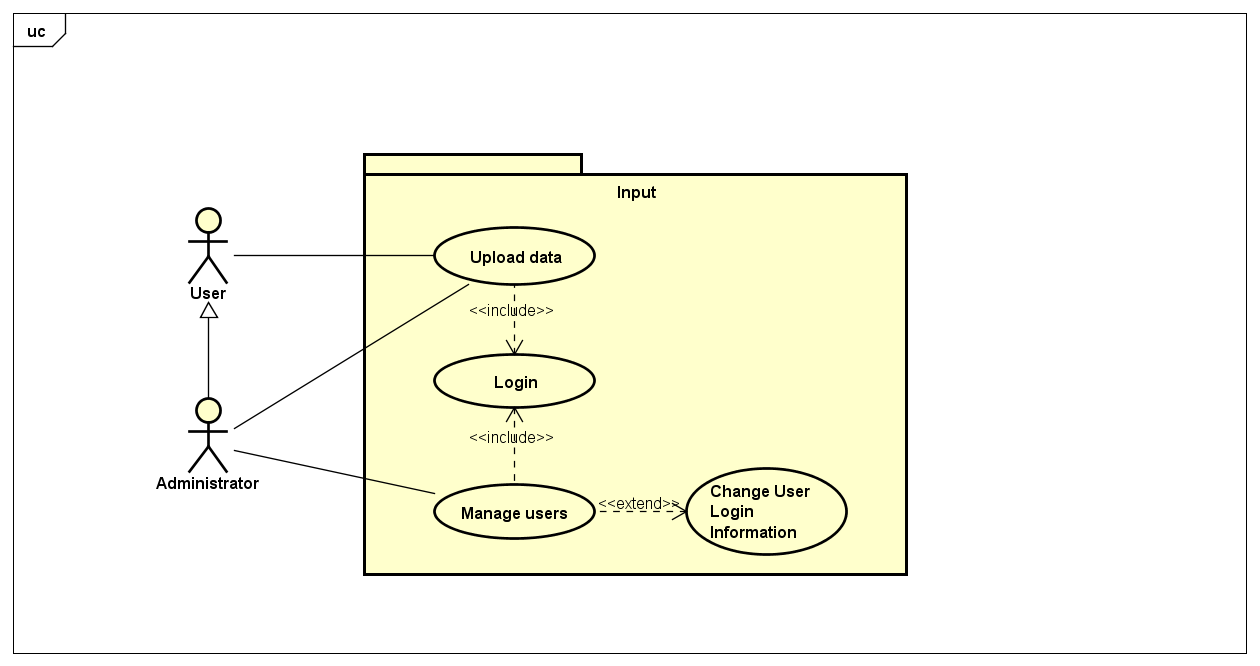


Figuur 2: High level system architecture diagram

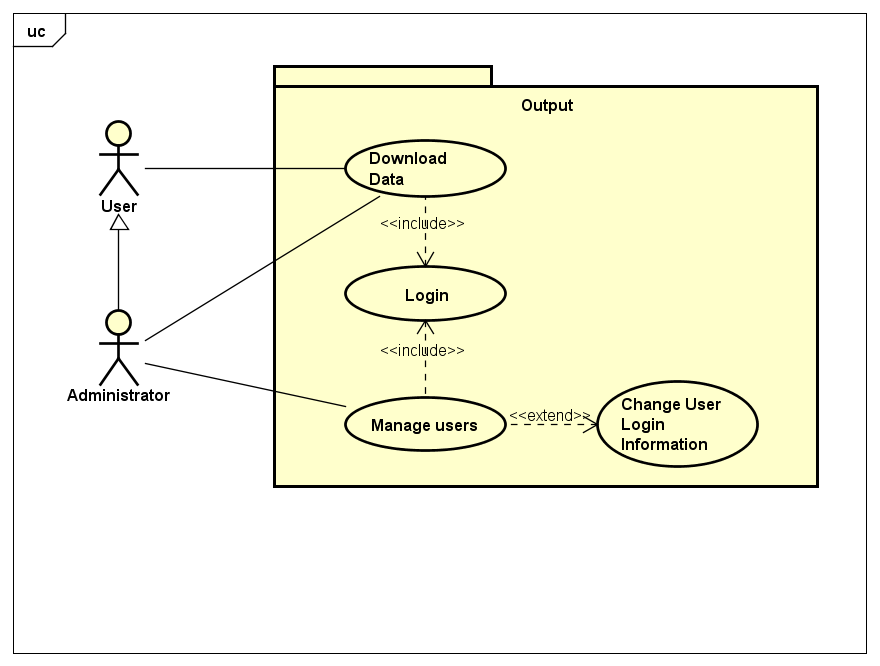
*Use cases diagram:*



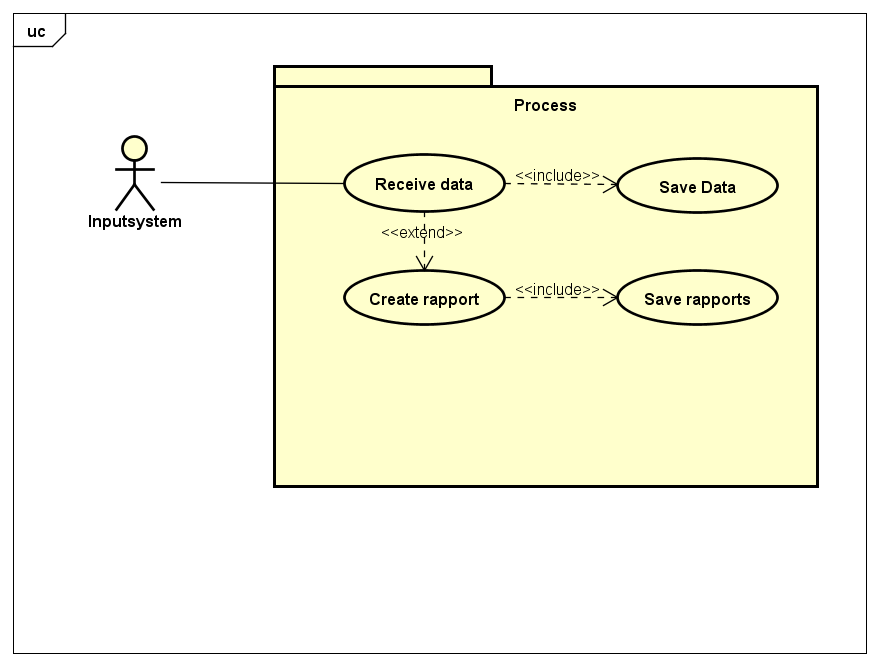
Figuur 3: Use Case Overview Diagram



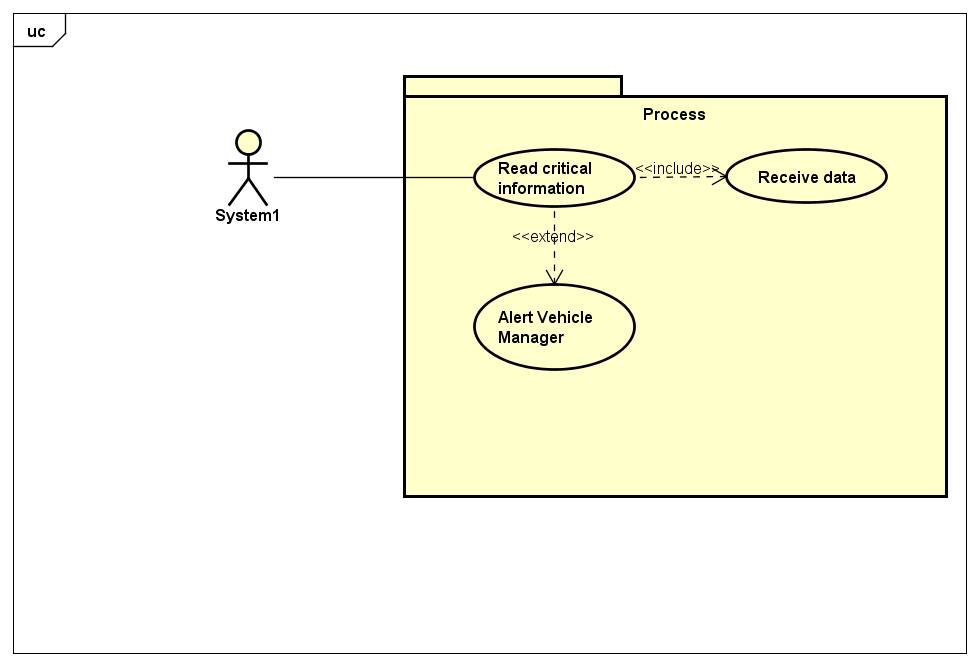
Figuur 4: Use Case Diagram, UC1



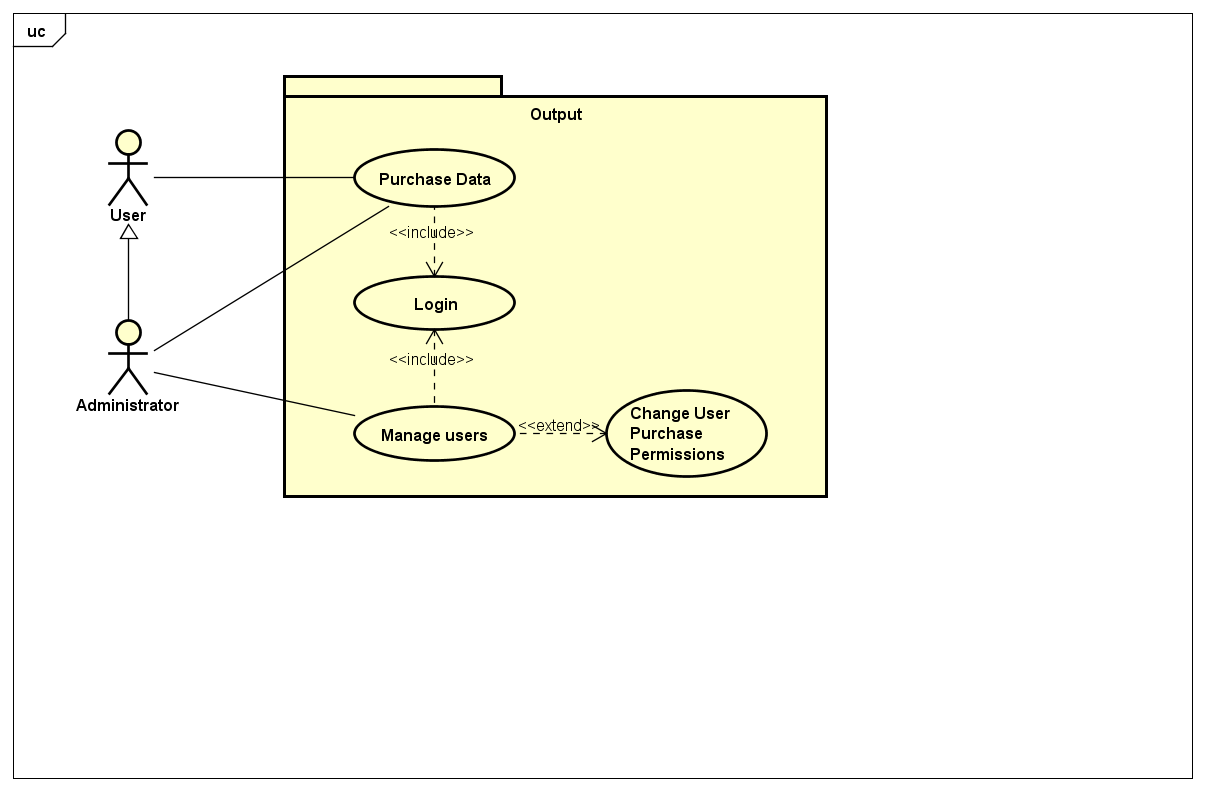
Figuur 5: Use Case Diagram, UC2



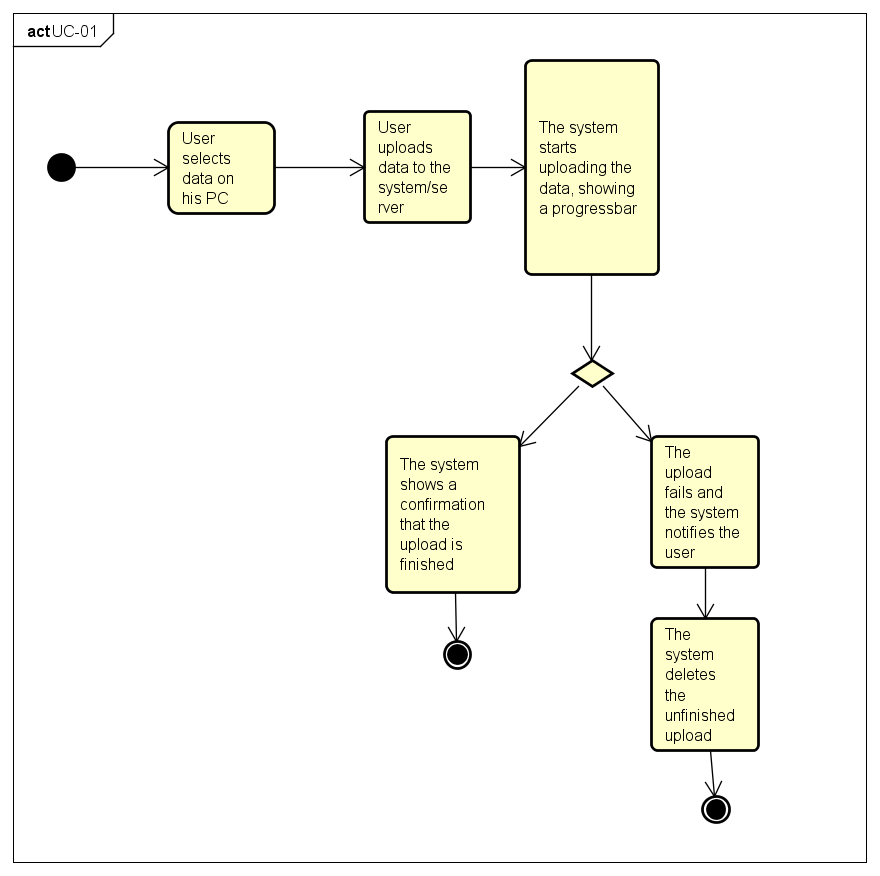
Figuur 6: Use Case Diagram, UC3



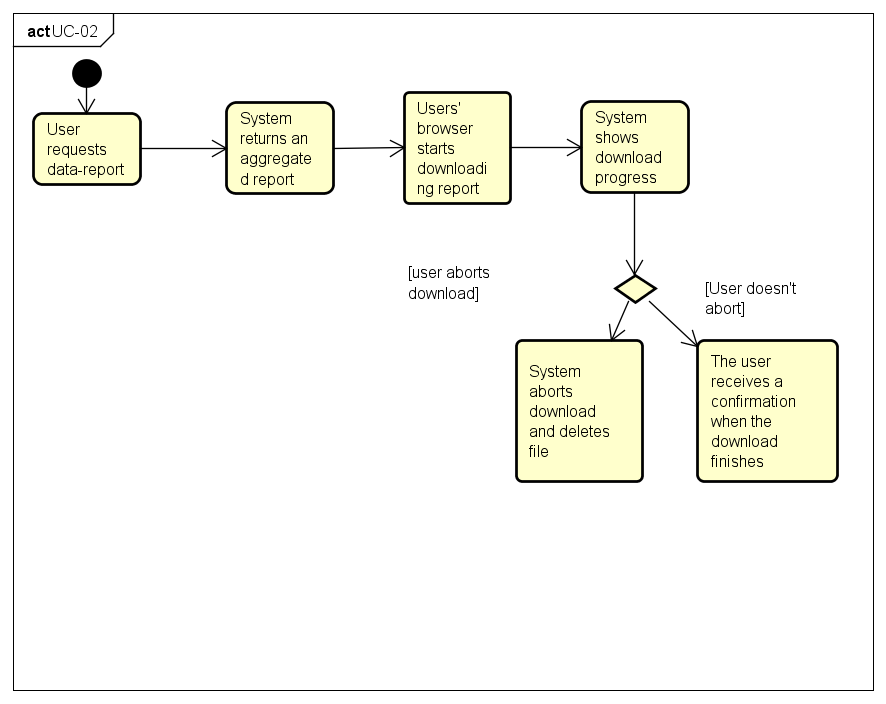
Figuur 7: Use Case Diagram, UC5



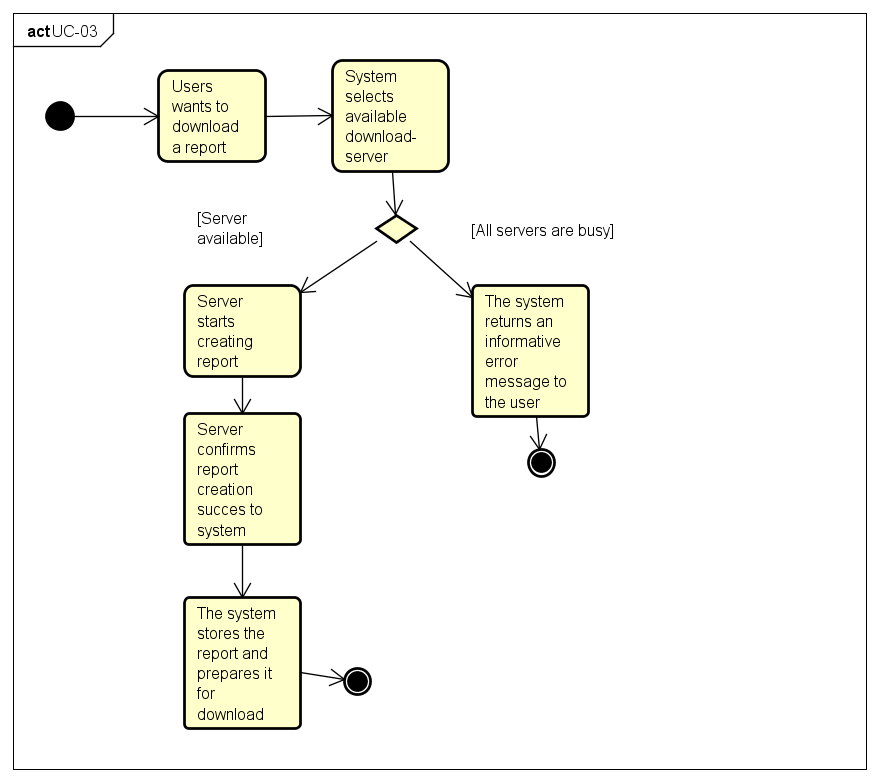
Figuur 8: Use Case Diagram, UC6



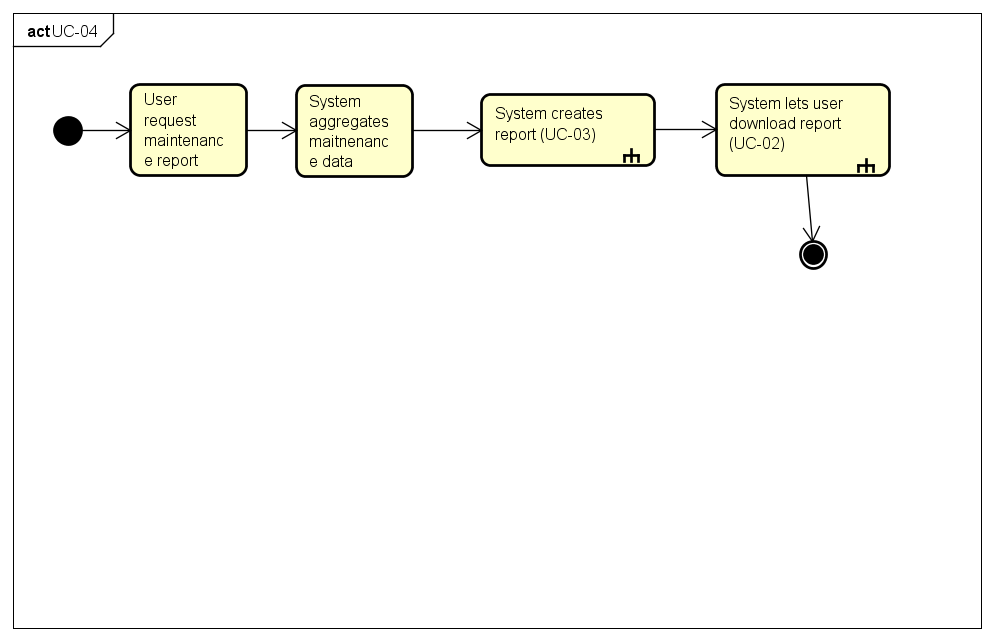
Figuur 9: Activity Diagram, UC1



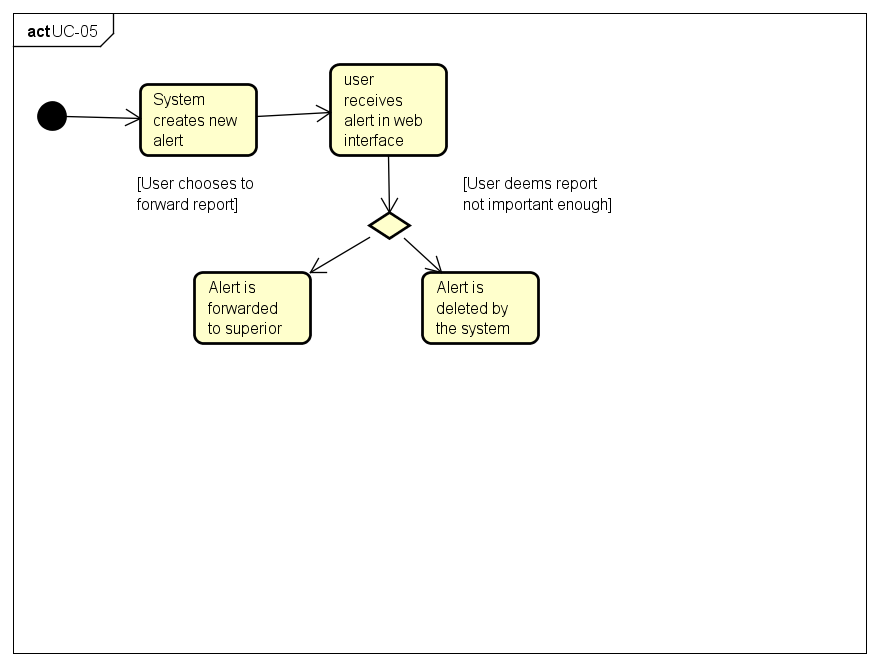
Figuur 10: Activity Diagram, UC2



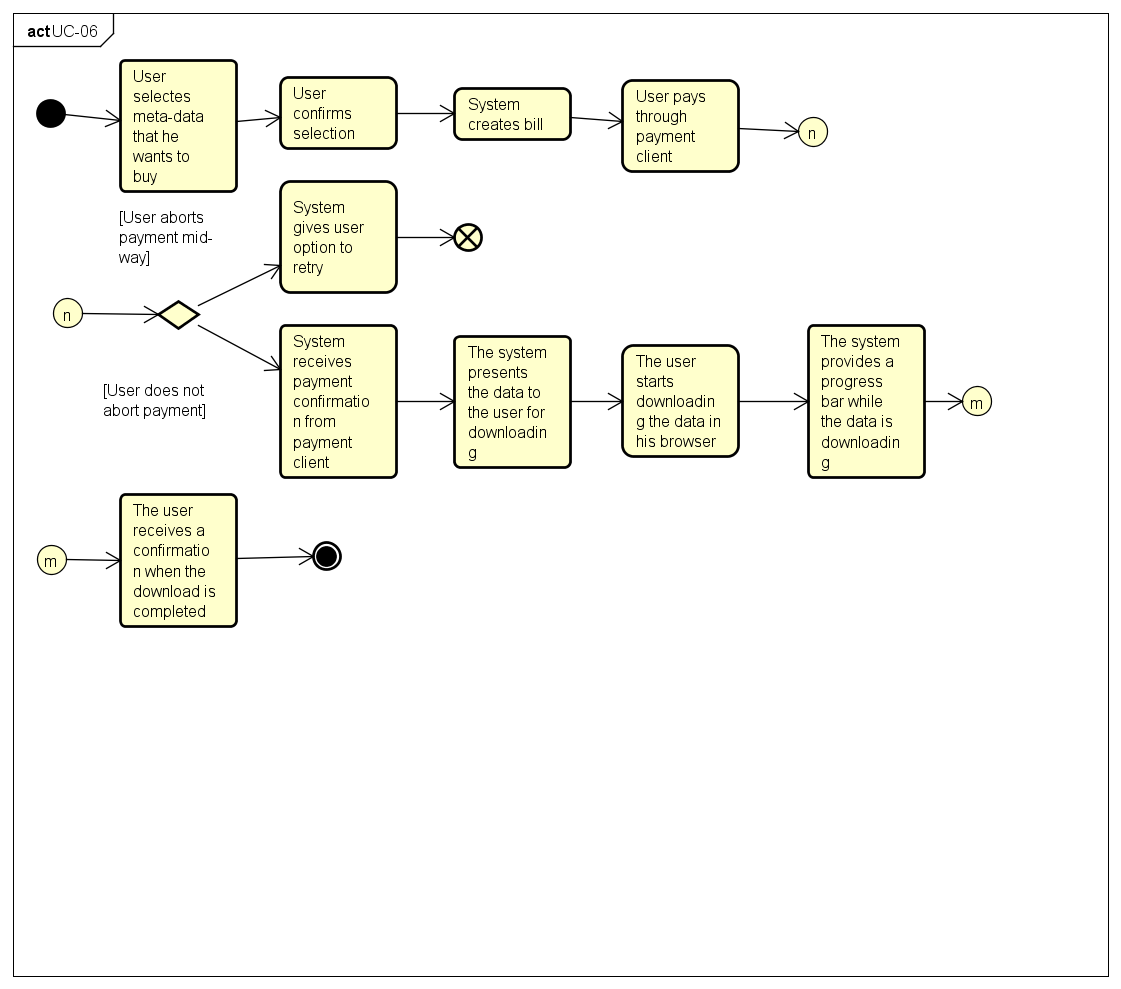
Figuur 11: Activity Diagram, UC3



Figuur 12: Activity Diagram, UC4



Figuur 13: Activity Diagram, UC5



Figuur 14: Activity Diagram, UC6

## B.

Als eerste hebben wij enkele standaardvragen mbt. Tot de opzet van de architectuur van ons systeem beantwoord: Onze antwoorden op deze vragen scheppen duidelijkheid over enkele beslissingen die wij omtrent onze systeem-architectuur hebben gemaakt.

**Vraag 1: Hoe zullen de structurele componenten van het systemen onderverdeeld worden in sub-componenten?**

Het systeem zal in 3 componenten verdeeld worden: Input, verwerking en output. Deze 3 componenten zullen als losse, geïsoleerde services geimplementeerd worden zodat zij onafhankelijk

**Vraag 2: Welke fundamentele aanpak zal gebruikt worden om het systeem te structureren?**

Jazeker, namelijk het onderverdelen van het systeem in de Input, Verwerking en Output componenten; dit gebeurt vaak in veel verschillende software-projecten. Het output-component zal ook gebruik maken van de MVC (model-view-controller) architectuur.

**Vraag 3: Is er een generieke applicatie architectuur die als sjabloon voor het design van het systeem gebruikt kan worden?**

Wij zullen SOA(Service Oriented Architecture) gebruiken. De sub-componenten zullen geisoleerd van elkaar gebouwd worden zodat zij makkelijk over verschillende servers verdeeld kunnen worden.

**Vraag 4: Hoe zal het systeem verdeeld worden over hardware-cores of processors?**

De specifieke verdeling van het systeem over de beschikbare hardware-infrastructuur zal op een later moment tijdens de implementatie van het systeem besloten worden.

**Vraag 5: Wat voor architecturele stijlen of patronen zullen er (wellicht) gebruikt worden?**

- SOA (Service Oriented Architecture).

- Het Client-Server model.

- MVC (model-view-controller)

**Vraag 6: Hoe zal de architectuur van het systeem gedocumenteerd worden?**

Het zal in het SRS-document gedocumenteerd worden, vooral door middel van DFD diagrammen.

**Vraag 7: Wat voor strategie zal er gebruikt worden om de operaties van decomponenten van het systeem te besturen?**

Welke strategieën precies gebruikt zullen worden zal pas in de implementatie-fase beslist worden . Zeker is echter dat MVC gebruikt gaat worden, aangezien het PHP framework Laravel deze strategie gebruikt. Ook zullen wij waarschijnlijk pThreads voor C++ gaan gebruiken om de Input en Verwerkings-componenten onderling met elkaar te laten communiceren.

**Vraag 8: Wat voor architecturele organizatie is het beste om de non-functional requirements van het system te creeëren?**

We hebben onze nonfunctional requirements gemaakt volgens het SRS formaat.

# C.

# Implementation plan

## Hergebruik

## Abstractie-niveau:

Op een abstract niveau van hergebruik zullen wij gebruik maken van de eerder besproken design patterns/architecturele patterns, SOA (Service Oriented Architecture), Het Client-Server model en MVC (model-view-controller).

## Object-niveau:

Aangezien onze backend-taal C++ een groot aantal verschillende ingebakken libraries kent, is het niet mogelijk om op dit moment al precies vast te stellen welke libraries we zullen gebruiken. Wel zullen gezien de aard van dit project waarschijnlijk wel veel String- en IO-libraries gebruiken.

## Component-niveau:

Voor ons backend maken wij (voor nu) gebruik van pThreads, om tussen processen te kunnen communiceren. Daarnaast gebruiken wij voor ons front-end het Laravel PHP Framework, en waarschijnlijk ook het bootstrap-framework voor opmaak.

## Systeem-niveau:

Wij zullen voor dit project geen volledige systemen hergebruiken.

## Configuratie-management.

## Versie-management

Als versie-management zullen wij Git en een Github-repository gebruiken.

## Systeem-integratie

Wij zullen geen systeem-integratie gebruiken.

## Probleem-tracking

Wij zullen een simpele vorm van problem tracking gebruiken, namelijk een contactpagina op het front-end van ons systeem waarin gebruikers eventuele bugs via een email-formulier kunnen doorgeven.

## Host-target development

## Hardware en software requirements per component

De input- en processingcomponenten maken gebruiken van C++ inclusief enkele libraries die in Linux-installaties standaard beschikbaar zijn, maar op Windows later gedownload moeten worden. Het Output-component heeft enkel de Laravel- en Bootstrap Javascript, CSS en PHP bestanden nodig die eenvoudig cross-platform beschikbaar zijn.

## De beschikbaarheidseisen van het systeem

Op het moment denkt het ontwikkel-team nog na over de eventuele implementatie van een ‘failsafe’. Later in de implementatiefase zal de beslissing genomen worden om wel of niet een failsafe te bouwen.

## Communicatie tussen componenten

Details over de fysieke verdeling van het systeem over hardware kunt u in ons SRS-document vinden.