Technical Overview

May 25, 2023

Team name: Storskov inc.
Company: Systematic
Teaching assistant: Mikkel Lippert
Agile coach: Chamara

Role	email	Name
PO	Miboj@systematic.com	Mico Bøje
SM	isam@itu.dk	Isabella Amstrup
dev	ahad@itu.dk	Adam Temsamani
dev	bath@itu.dk	Baldur Thomsen
dev	ehel@itu.dk	Emilia Helsted
dev	jakst@itu.dk	Jakob Storskov
dev	jklo@itu.dk	Johan Skoven
dev	kmsa@itu.dk	Katrine Sandø
dev	masc@itu.dk	Malte Schlack
dev	biha@itu.dk	Maxime Havez
dev	olfw@itu.dk	Oliver Wilhjelm

Contents

1 Product

The product is an anomaly detection service, that uses a machine learning model to analyse given log-messages for anomalies. Users are then informed and given the option to handle them through a web-app.

2 Software Qualities

Below is a list of software qualities we intend to uphold throughout the project.

Usability:
The dashboard should be intuitive to use for IT professionals. Important information such as alerts and dashboard should be the first thing users are presented with.
Performance:
The system should be able to handle at least 3.000 log messages per minute.
Security:
Logins must be handled in a secure fashion, meaning that passwords are at least hashed.
Scalability:
The system should be able to handle increasing workloads by scaling components. This means that everything except the database should ideally be able to run in multiple instances.
Maintainability:
The code must be documented, commented and preferably achieve at least a B in maintainability on a Sonarqube scan.
Testability:
The code should be tested in a CI/CD pipeline, like GitHub workflow.

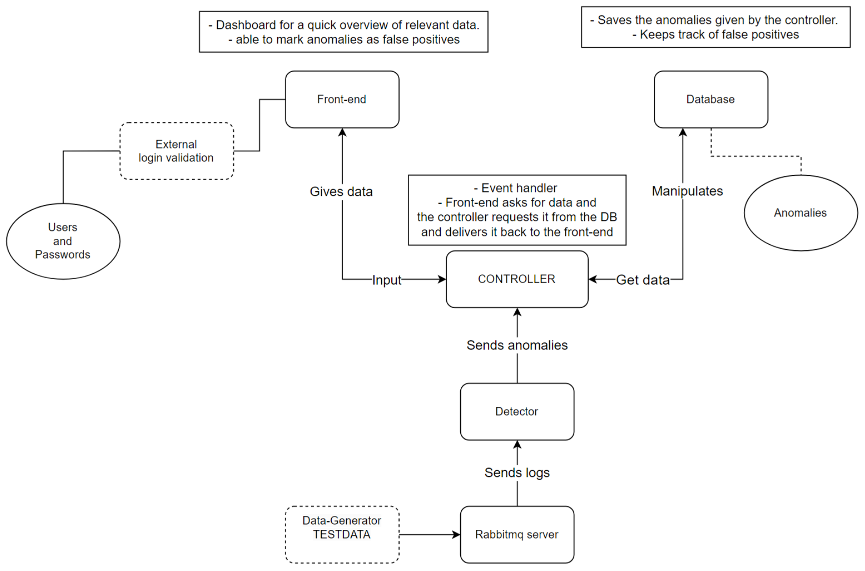
2.1 Ensuring Software Qualities

Usability:
To ensure good usability, we have done user testing[1] and used the given feedback to improve the user-experience
Performance:
The product has achieved this software quality (assuming adequate hardware) through use of the RabbitMQ framework's load-balancing abilities.
Security:
The product uses the Keycloak solution to handle user management and authorisation to achieve this. Further masking of sensitive information is considered a nice to have feature, which could be solved using Istio.
Scalability:
The product has achieved this software quality through use of the RabbitMQ

[1] 5.1

framework's load-balancing abilities.
Maintainability:
We have set up pre-commit hooks and GitHub actions to help with maintainability, along with a pull request template. Together, they enforce and remind us of upholding a clean codebase for easier maintainability. The codebase has been thoroughly documented and the final product has achieved acceptable results in a SonarQube scan.
Testability:
Testing has primarily been done through API-calls via Postman, as most of our program is dependent on these. The front-end has been tested further through use of user-tests.

3 Architecture Design



- Dashboard for a quick overview of relevant data.
- able to mark anomalies as false positives
- Saves the anomalies given by the controller.
- Keeps track of false positives
Front-end
Database
External login validation
Users and Passwords
Gives data
Manipulates
Anomalies
- Event handler
- Front-end asks for data and the controller requests it from the DB and delivers it back to the front-end
Input
CONTROLLER
Get data
Sends anomalies
Detector
Sends logs
Data-Generator TESTDATA
Rabbitmq server

Figure 1: Overall architecture of the system, updated for exam submission

Figure 1 shows the software architecture of the program. We have based our architecture on a Model View Controller architecture.

The system uses a machine learning algorithm build in Python to analyse log messages. The model we got from our PO is written in Python. It was therefore decided that the primary language for the entire codebase is Python to maintain consistency.
The system uses a Postgres database, to store our data. This database was

given to us along with the initial codebase, and there was no reason to change the type of database. The system uses the open source solution Keycloak for user management, as Systematic is already using it for other products.
It also uses RabbitMQ as a message broker. This ensures no messages are lost in case of a system crash, and enables load balancing by having multiple active anomaly detectors. The RabbitMQ framework can hence be used to scale the product horizontally.
Lastly the system uses Docker for containerization as it is a industry standard and was desired by our PO.

4 Technical Specifications

The system is divided into a front-end and back-end. The front-end is on a single docker container and made primarily with python dash and pandas frameworks. This includes functions and callbacks with API-calls that connects the front-end to the back-end.
The back-end is further divided into 6 different docker containers, handling different parts of the system: anomaly-detector, controller, Keycloak, RabbitMQ, Postgres, and a data-generator.

Anomaly-detector

The anomaly-detector contains the machine-learning model which handles analyzing given log-messages for anomalies. If the analyzed message is considered an anomaly it is given to the controller through an API call.

Controller

The controller handles communication between the front-end and the back-end, with the exception of the Keycloak container. The controller also handles communication between the anomaly-detector and the database. This is done through API-calls, which are made with the FastAPI framework.

Keycloak

The Keycloak container handles user-management authentication of user credentials. The container uses the open-source Keycloak solution, which is made for handling user identity and access. The Keycloak container communicates directly with the front-end through methods made using the Python Keycloak library.

RabbitMQ

The RabbitMQ container handles message-queuing and load-balancing through the use of the open source RabbitMQ message broker software. The container divides the data between the amount of initialized anomaly-detectors. It also prevents data loss in case of system failure.

Postgres database

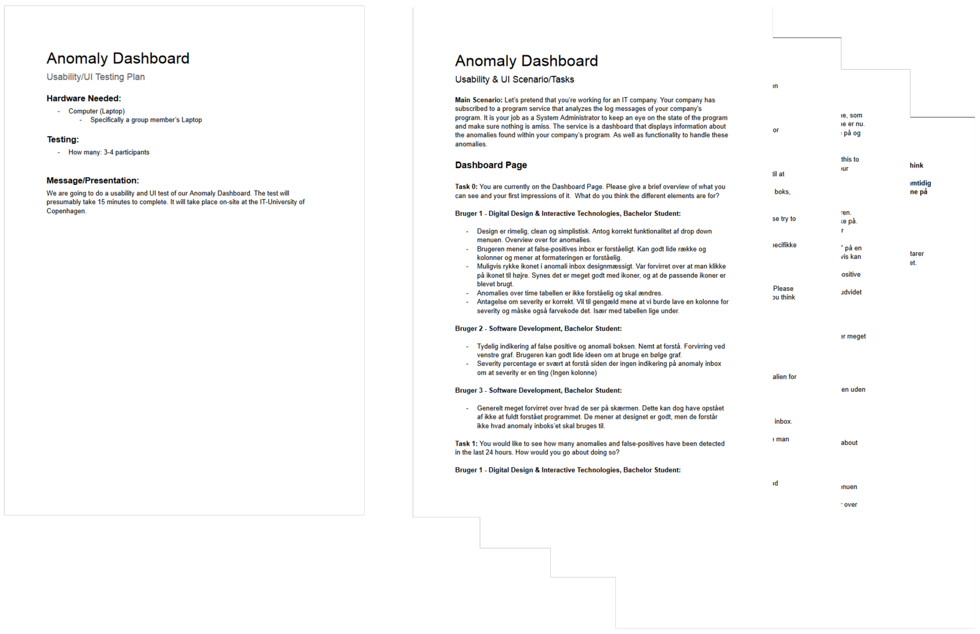
The Postgres container is our Postgres database, which stores anomaly-messages including all relevant information. It is accessed by the controller through CRUD operations.

Data-generator

The data-generator sends a constant stream of data to the RabbitMQ container. It currently simulates real life data but it is intended to be replaced actual data eventually.

5 appendix

5.1 Usertest



Anomaly Dashboard

Usability/UI Testing Plan

Hardware Needed:

- Computer (Laptop)
    - Specifically a group member's Laptop

Testing:

- How many: 3-4 participants

Message/Presentation:

We are going to do a usability and UI test of our Anomaly Dashboard. The test will presumably take 15 minutes to complete. It will take place on-site at the IT-University of Copenhagen.

Anomaly Dashboard

Usability & UI Scenario/Tasks

Main Scenario: Let's pretend that you're working for an IT company. Your company has subscribed to a program service that analyzes the log messages of your company's program. It is your job as a System Administrator to keep an eye on the state of the program and make sure nothing is amiss. The service is a dashboard that displays information about the anomalies found within your company's program. As well as functionality to handle these anomalies.

Dashboard Page

Task 0: You are currently on the Dashboard Page. Please give a brief overview of what you can see and your first impressions of it. What do you think the different elements are for?

Bruger 1 - Digital Design & Interactive Technologies, Bachelor Student:

- Design er rimelig, clean og simplistisk. Antog korrekt funktionalitet af drop down menuen. Overview over for anomalies.
- Brugeren mener at false-positives inbox er forståeligt. Kan godt lide række og kolonner og mener at formateringen er forståelig.
- Muligvis rykke ikonet i anomali inbox designmæssigt. Var forvirret over at man klikke på ikonet til højre. Synes det er meget godt med ikoner, og at de passende ikoner er blevet brugt.
- Anomalies over time tabellen er ikke forståelig og skal ændres.
- Antagelse om severity er korrekt. Vil til gengæld mene at vi burde lave en kolonne for severity og måske også farvekode det. Især med tabellen lige under.

Bruger 2 - Software Development, Bachelor Student:

- Tydelig indikering af false positive og anomali boksen. Nemt at forstå. Forvirring ved venstre graf. Brugeren kan godt lide ideen om at bruge en bølge graf.
- Severity percentage er svært at forstå siden der ingen indikering på anomaly inbox om at severity er en ting (Ingen kolonne)

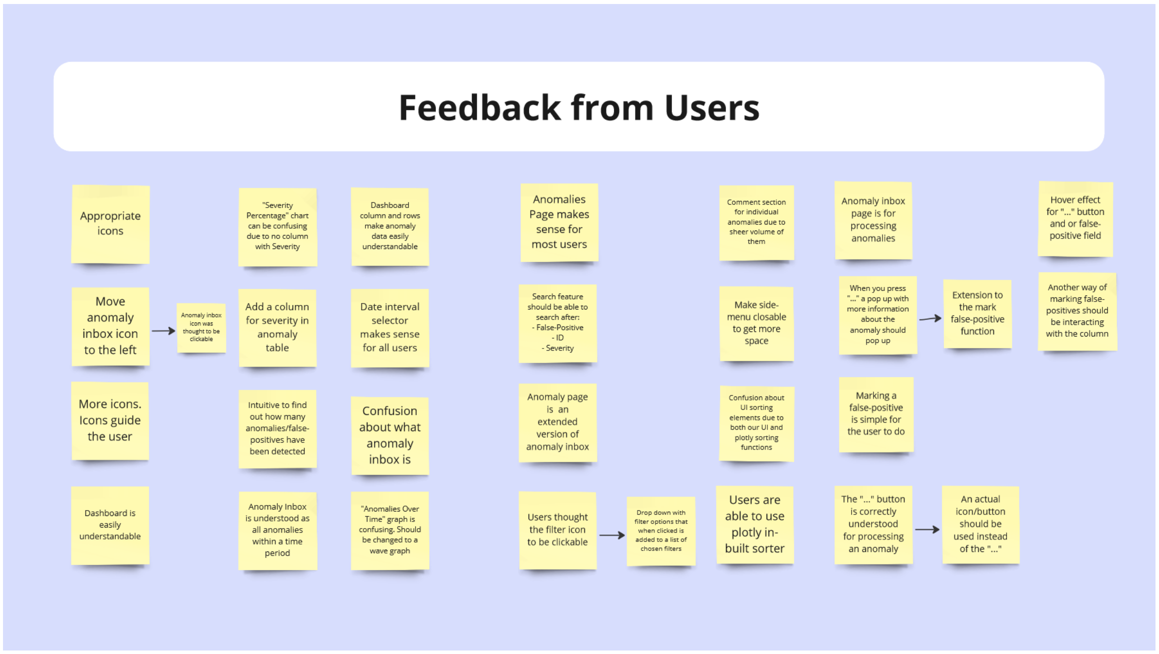
Bruger 3 - Software Development, Bachelor Student:

- Generelt meget forvirret over hvad de ser på skærmen. Dette kan dog have opstået af ikke at fuldt forstået programmet. De mener at designet er godt, men de forstår ikke hvad anomaly inboks'et skal bruges til.

Task 1: You would like to see how many anomalies and false-positives have been detected in the last 24 hours. How would you go about doing so?

Bruger 1 - Digital Design & Interactive Technologies, Bachelor Student:

Figure 2: Questions for user tests



Feedback from Users

Appropriate icons

Move anomaly inbox icon to the left → Anomaly inbox icon was thought to be clickable

More icons. Icons guide the user

Dashboard is easily understandable

"Severity Percentage" chart can be confusing due to no column with Severity

Add a column for severity in anomaly table

Intuitive to find out how many anomalies/false-positives have been detected

Anomaly Inbox is understood as all anomalies within a time period

Dashboard column and rows make anomaly data easily understandable

Date interval selector makes sense for all users

Confusion about what anomaly inbox is

"Anomalies Over Time" graph is confusing. Should be changed to a wave graph

Anomalies Page makes sense for most users

Search feature should be able to search after:
- False-Positive
- ID
- Severity

Anomaly page is an extended version of anomaly inbox

Users thought the filter icon to be clickable → Drop down with filter options that when clicked is added to a list of chosen filters

Comment section for individual anomalies due to sheer volume of them

Make side-menu closable to get more space

Confusion about UI sorting elements due to both our UI and plotly sorting functions

Users are able to use plotly in-built sorter

Anomaly inbox page is for processing anomalies

When you press "..." a pop up with more information about the anomaly should pop up → Extension to the mark false-positive function

Marking a false-positive is simple for the user to do

The "..." button is correctly understood for processing an anomaly → An actual icon/button should be used instead of the "..."

Hover effect for "..." button and or false-positive field

Another way of marking false-positives should be interacting with the column

Figure 3: Feedback from users

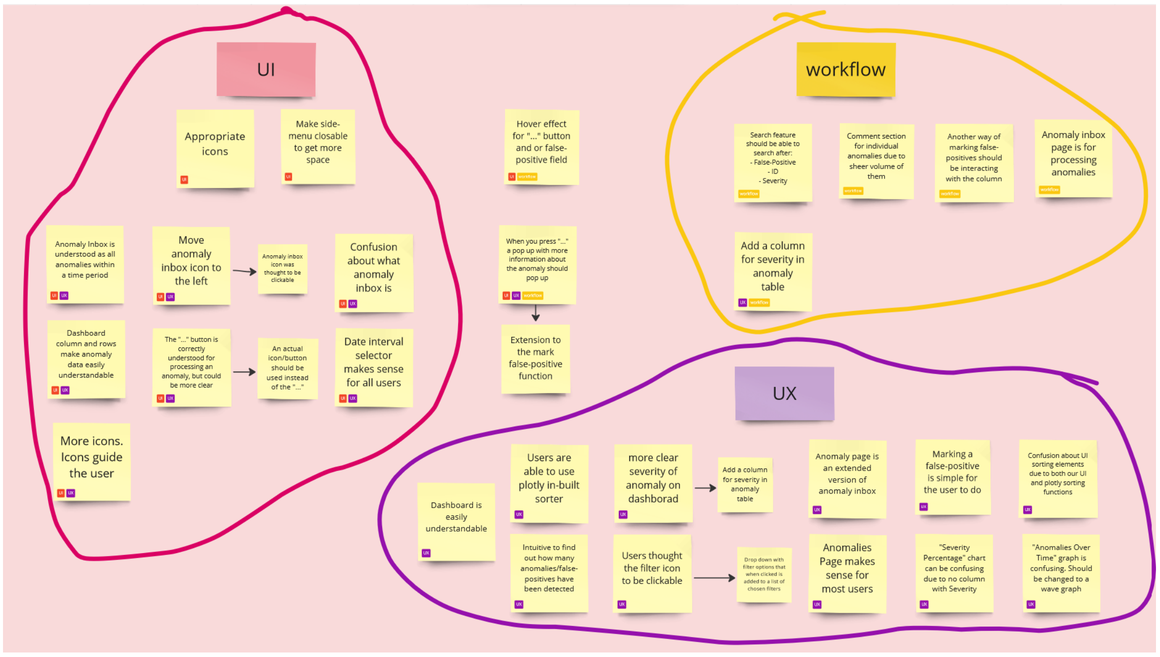

UI
Appropriate icons
Make side-menu closable to get more space
Anomaly Inbox is understood as all anomalies within a time period
Move anomaly inbox icon to the left
Anomaly inbox icon was thought to be clickable
Confusion about what anomaly inbox is
Dashboard column and rows make anomaly data easily understandable
The "..." button is correctly understood for processing an anomaly, but could be more clear
An actual icon/button should be used instead of the "..."
Date interval selector makes sense for all users
More icons. Icons guide the user
Hover effect for "..." button and or false-positive field
When you press "..." a pop up with more information about the anomaly should pop up
Extension to the mark false-positive function
workflow
Search feature should be able to search after: - False-Positive - ID - Severity
Comment section for individual anomalies due to sheer volume of them
Another way of marking false-positives should be interacting with the column
Anomaly inbox page is for processing anomalies
Add a column for severity in anomaly table
UX
Dashboard is easily understandable
Users are able to use plotly in-built sorter
more clear severity of anomaly on dashborad
Add a column for severity in anomaly table
Anomaly page is an extended version of anomaly inbox
Marking a false-positive is simple for the user to do
Confusion about UI sorting elements due to both our UI and plotly sorting functions
Intuitive to find out how many anomalies/false-positives have been detected
Users thought the filter icon to be clickable
Drop down with filter options that when clicked is added to a list of chosen filters
Anomalies Page makes sense for most users
"Severity Percentage" chart can be confusing due to no column with Severity
"Anomalies Over Time" graph is confusing. Should be changed to a wave graph

Figure 4: Grouping of user data

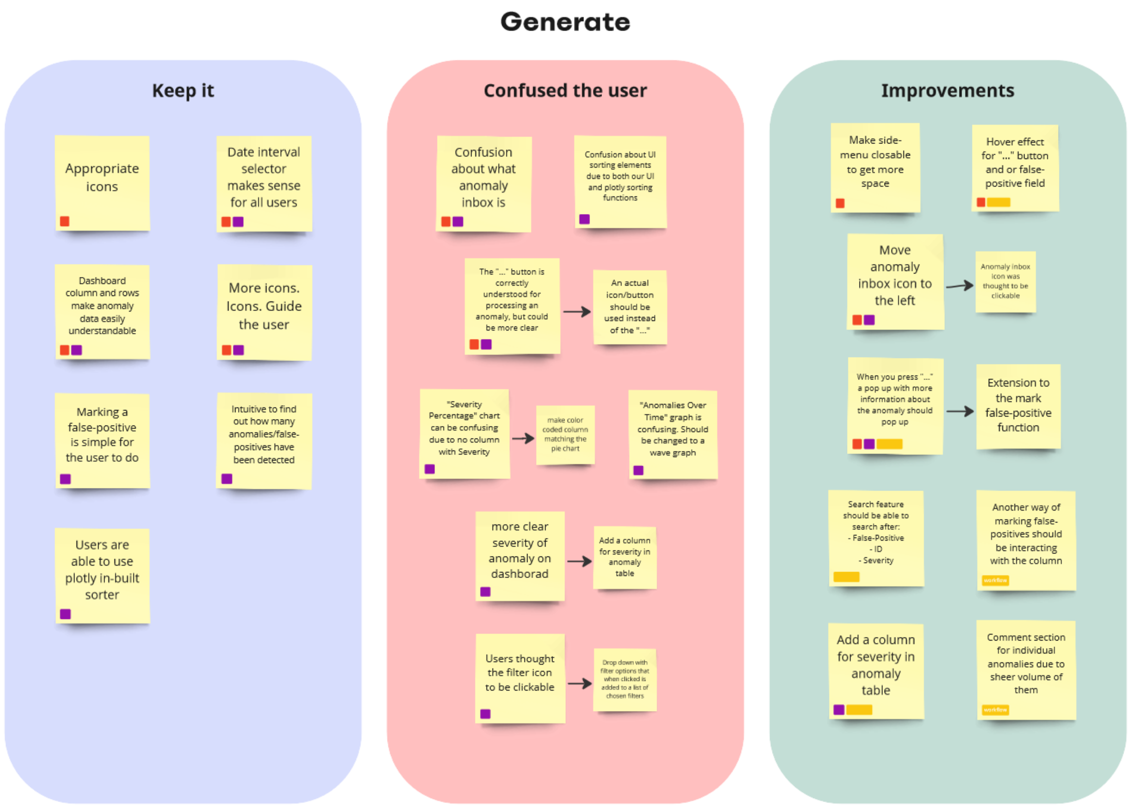

Generate
Keep it
Appropriate icons
Date interval selector makes sense for all users
Dashboard column and rows make anomaly data easily understandable
More icons. Icons. Guide the user
Marking a false-positive is simple for the user to do
Intuitive to find out how many anomalies/false-positives have been detected
Users are able to use plotly in-built sorter
Confused the user
Confusion about what anomaly inbox is
Confusion about UI sorting elements due to both our UI and plotly sorting functions
The "..." button is correctly understood for processing an anomaly, but could be more clear
An actual icon/button should be used instead of the "..."
"Severity Percentage" chart can be confusing due to no column with Severity
make color coded column matching the pie chart
"Anomalies Over Time" graph is confusing. Should be changed to a wave graph
more clear severity of anomaly on dashborad
Add a column for severity in anomaly table
Users thought the filter icon to be clickable
Drop down with filter options that when clicked is added to a list of chosen filters
Improvements
Make side-menu closable to get more space
Hover effect for "..." button and or false-positive field
Move anomaly inbox icon to the left
Anomaly inbox icon was thought to be clickable
When you press "..." a pop up with more information about the anomaly should pop up
Extension to the mark false-positive function
Search feature should be able to search after: - False-Positive - ID - Severity
Another way of marking false-positives should be interacting with the column
workflow
Add a column for severity in anomaly table
Comment section for individual anomalies due to sheer volume of them
workflow

Figure 5: Final results of processing the user feedback from the user tests.

References

[1] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.