PROJECT PLAN

INDUSTRIAL ROBOT MONITORING FOR OPIFLEX AB

GROUP 5

2017-11-28

VIKTING GUSTAV LINDBERG, WILLIAM ACHRENIUS, NOÉ BAYLAC JACQUÉ, NAIDA DEMIROVIC, ANDREAS HEMLÉN, ERIK MARTIN CAMPAÑA, ELAHEH AILINEJAD, PONTUS WEDÉN

1 Introduction

The client for this project is the Västerås based company OpiFlex AB. They produce mobile robot platforms, which enables increased production flexibility for their customers. The platform can be moved easily, which means a robot can always be used where automation is needed.

OpiFlex also produces fenceless security systems for robots. Manufacturing robots are often inside a fence when they operate, since a robot in active operation poses a personal safety hazard. This means that operators cannot get close to an actively working robot. OpiFlex provides what they call a “Fenceless Safety Solution”, which instead of physical fences, uses laser scanners to detect if someone is close to the robot. If an obstruction is detected, the robot's speed of operation will be slowed to a safe level.

What OpiFlex have asked us to provide is an application and a back-end server for monitoring the status of their customer’s robots. The application should have information about, for example, input material levels, remaining room for output, and total items produced for a robot. The goal is that with this application, the need for operators to always be on site will be reduced, since information can be viewed remotely.

2 Project Organization

2.1 Project Group

- Viking Gustav Lindberg (Project manager)
    - vlg15001@student.mdh.se
- William Achrenius
    - was15001@student.mdh.se
- Noé Baylac Jacqué (Configuration manager)
    - nbe17002@student.mdh.se
- Naida Demirovic
    - ndc17001@student.mdh.se
- Andreas Hemlén
    - ahn13008@student.mdh.se
- Erik Martin Campana (Document templates)
    - ema16003@student.mdh.se
- Elaheh Aalinejad (Document templates)
    - ead17002@student.mdh.se
- Pontus Wedén (Client communication responsible)
    - pwn15002@student.mdh.se

2.2 Organization and Communication

The meetings are coordinated by the Project Manager. Every member must participate in all the meetings. The Project Manager sends the message to each team members to set the meeting time: everyone must see and respond to it. The group has planned to meet each day we have project meeting with the steering group. Depending on the number of activities, we will arrange how many hours the meeting will take. Whenever it is needed, the project manager will arrange another time in the week to meet and solve problems. We have agreed to hold meetings remotely on Skype when the team members are away on vacation, for instance at the end of the year. Moreover, we have meetings with the client. They are also scheduled to be once in a week and the contact person sets the time via email.

The Project Manager has created a Facebook group and the members can contact each other via Facebook messenger and email. The reports and presentations are shared in a folder on Google Drive, so everyone can edit and review it in advance.

We have created a Trello board and every member can access it and create new cards. It applies for the project backlogs and all the activities, which include to do list, what is being done and what we have done.

We have created a Git repository and invited every member there. So, the group members can access the project and start doing and syncing their assigned tasks.

2.3 Planned Effort

The planned effort for the entire group assumes that we should put 20 hours a week on the project and its different activities. Of course, this might not be reached to optimal performance every week as other courses and problems might hinder progress. Although, the main goal of the team is to reach 20 hours per member each week (roughly 160 hours of effort per week).

As we have planned currently, we are trying to move to the design phase as soon as possible as we have planned enough for the future. Our current plan is to start the design and implementation phase by week 46 so that we have a head start into the development of the product. This is advantageous to us as this can potentially give us a head start into development, it also allows us to encounter obstacles earlier so that we can solve those.

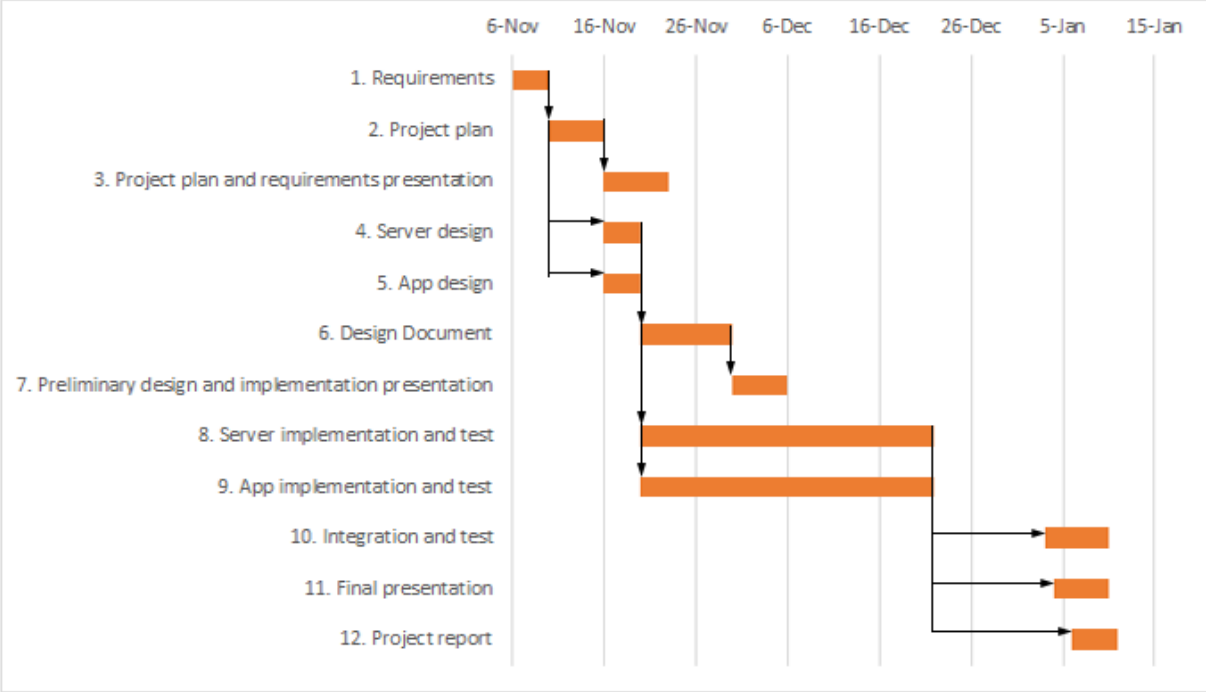
One known absence that might make the planned effort be inconsistent is the holidays this upcoming December. To counter this we have decided that the group, thanks to Git, Trello and messenger can work individually on our set assignments, and if necessary a meeting can be scheduled through Skype, as physical meetings become impossible during the holidays.

2.4 Deliverables, Deadlines, and Milestones

We gathered some requirements that are enough to start the project with an initial meeting with OpiFlex. We have assigned initial responsibilities in the group (project manager, client communication, document templates, configuration management). We have 2 weeks for planning and initial requirements. During this period, we need to decide on forms for the group work and assign responsibilities, which we have already finished. Also, we have met the client and formulated requirements. First deadline for delivering the project plan is on 16th November. On 22nd November is the deadline for our first presentation Project plan and requirements. We need to deliver product and design description no later than 6th December. We have 4 weeks to finish design and implementation. And the final project report is on 10th January.

2.4.1 Activities and GANTT chart

Activities	Allocation	Dependencies	Duration(days)	Start date	End date
1. Requirements	8	none	4	6-Nov	10-Nov
2. Project plan	8	1	6	10-Nov	16-Nov
3. Project plan and requirements presentation	3	2	7	16-Nov	22-Nov
4. Server design	3	1	4	16-Nov	20-Nov
5. App design	2	1	4	16-Nov	20-Nov
6. Design Document	5	4,5	10	20-Nov	30-Nov
7. Preliminary design and implementation presentation	3	6	6	30-Nov	6-Dec
8. Server implementation and test	4	4	32	20-Nov	22-Dec
9. App implementation and test	4	5	32	20-Nov	22-Dec
10. Integration and test	8	8,9	7	3-Jan	10-Jan
11. Final presentation	8	8,9	6	4-Jan	10-Jan
12. Project report	8	8,9	5	6-Jan	11-Jan



2.5 Quality Assurance

To ensure that we deliver the correct product with a good quality documentation, we have set up some routines to guarantee that our delivered work is according to our quality standards:

- The documentation will be clear, easy to understand, and correct regarding to our final product.
- Every document written will be reviewed by all the team members before delivery to the steering group.
- Source code will be reviewed by at least another team member to check for any potential errors and inconsistencies with the agreed architecture and the documentation.
- Regular communication between the customer, the steering group and among all the team members will be kept to report on the progress of the project and to make sure everyone's view and suggestion has been taken into account in the project work.
- The final product must respect the requirements defined in this document. Acceptation tests will be carried out to check if the customer is satisfied with the features implemented in the final product.

- Efforts will be put on security and data protection to make sure customer's data will be kept safely and delivered to the right customer.

3 Description of the System

3.1 High Level Description of the Domain and the Problem

The problem OpiFlex has today is that information about a robot is presented on a device that must be physically connected to the robot. This means that it is both impossible to see this information during operation, and someone must be present at a robots' location to view the information when it is available. Presently, there is also no way to log errors produced by a robot, and if it existed on the current device, OpiFlex thinks there is a risk that the operators could become frustrated by writing longer logs on the present device.

The requested system from OpiFlex that we will implement contains the main parts shown in Figure 1. The server will receive data from robots at a customer location, and act as the main database of information. The database stores information about robots for several companies, and differentiate between them with a company id. The information stored on the server will then be presented on the application. Each customer has their own account, so the application will only fetch information about that customer's robots. This allows for operators to check a robot's status from off-site. The application should also provide an interface for displaying/editing error logs.

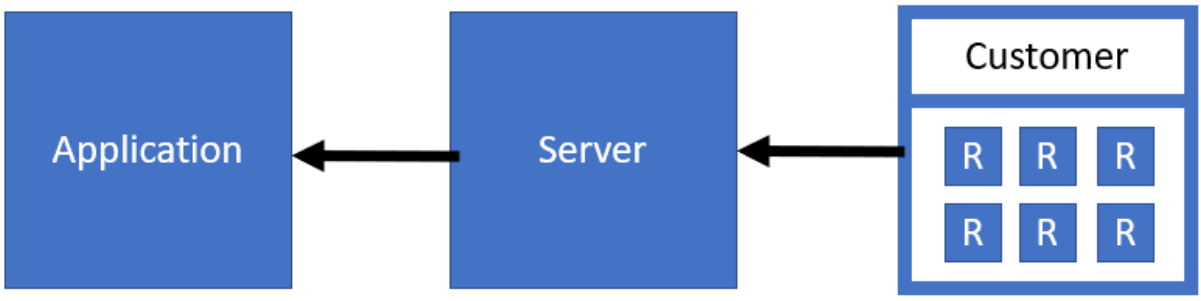


Figure 1. The basic design of the requested system. A customer has several robots (R) that send data to a server.

The target demographic for this system, according to OpiFlex, will be primarily male machine operators in their middle ages. The secondary group, belonging to the same profession, are mostly younger males.

OpiFlex has forwarded wishes from their customers regarding that the system should be rewarding to use. This would mean adding some form of acknowledgment for the operator when he/she accomplishes different tasks. These tasks could include writing error logs and measuring how many items the operator has helped manufacture. Given the scope of the project (10 weeks including the holidays), our focus will be producing an app that displays the requested information properly, meaning that this reward system will be a secondary priority.

3.2 Existing Systems

The program to develop will interact with an existing robot although the only consideration about this interaction will be the reception of data coming from it. The platform that will send this data to the server is still not designed and the program will instead be designed around some simulated data given by the company.

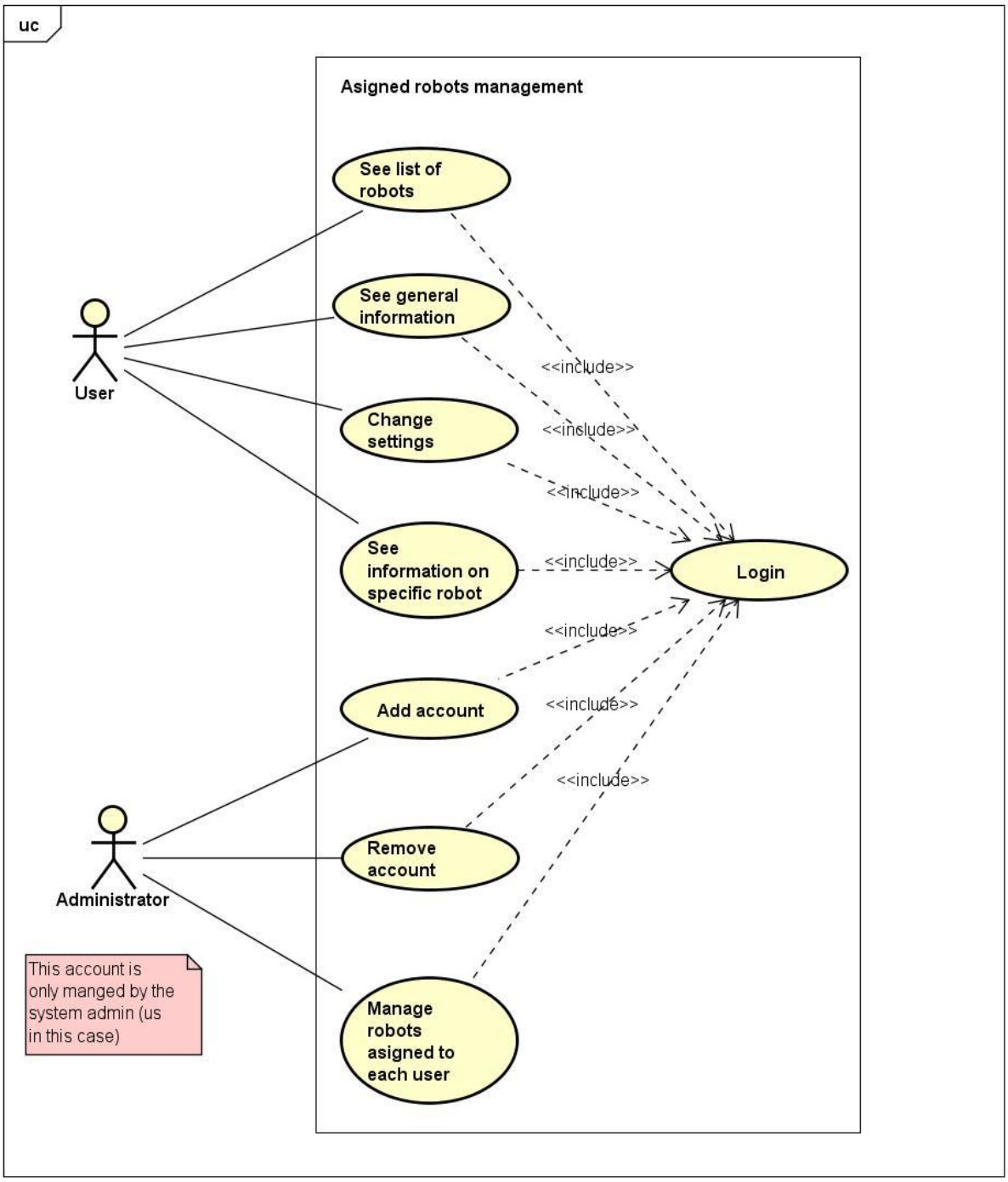
3.3 High-level Description of the Desired Functionality

- The system will have two differentiated parts, a server and a mobile app client.

    - For security reasons, the information will be stored in the server and the client will only get access to the information related that's related to them.
- The server will be able to store production data from a robot:
    - Robot Id
        - Identifies who is sending the data.
    - Robot cell Id
        - From which robotcell its sending data.
    - Product name
        - Name of the product currently being produced.
    - Start time
        - The start time for the currently running product, represented in UNIX time.
    - Current time
        - Timestamp of the current time, represented in UNIX time.
    - Running status
        - Boolean that indicates whether the robot is running.
    - Number of produced units
        - How many products the robot has produced since start.
    - Produced units target
        - How many products the robot aims to produce, -1 if it will produce until stopped.
    - Idle time
    - Cycle time
        - How many seconds it takes for the robot on average to produce one product.
    - Station
        - The current position of the robot.
    - Error
        - Boolean that indicates if the machine has encountered an error.
    - Log messages (examples)
        - Slowing down due to object entering low speed zone.
        - Input pallet empty.
        - Output pallet full.
        - Gripper dropped detail.
        - Detail not found.
        - Scanners detected unknown object in stop zone.
    - Event code
        - Identifier for specific event associated with the log message. For example, if it's an error or a normal log message.
    - Input/output pallet
        - Remaining items in the robots' input/output pallet.
    - Speed
        - The current speed of the robot.
- There will be a login by user/company that will only allow the user to see the data that he/she should be able to see (i.e. should not see data from other companies' robots).
    - There will only be an administrator account that will be managed by the server owner. This account will be the one allowed to add and remove users and modify their permissions.
- There will be a list of robots to connect to (and optionally the ability to pick certain robots as favorites).

- That list will allow to see more information about each of the listed robots individually.
- There will be a list of errors in a log. Errors or events could be entered, edited and documented by a user.

3.3.1 Use-Case Diagram of Desired Functionality



4 Initial Project Backlog

	We want to...	so that...	Notes	Priority	Effort
1	make an initial app design	we can start implement it		1	3

2	make an initial design of the server	we can start implement it		2	3
3	create server tables for storing the data	we store data in the server		3	1
4	store some dummy data in the server	we can use it for the app implementation	Create the necessary tables	4	1
5	setup server and app communication	we can continue to implement the communication protocols	Using JSON objects	5	2
6	create a login page for the app	the app can connect to the database	Secure, hash passwords.	6	2
7	create page for listing robots in the app	the company can choose what robot to view		7	2
8	create login functionality in the database	different companies can only access their own data	No SQL injections. Possibly SHA-256, (username:password:salt). Firebase might solve all these problems.	8	1
9	create page to display data in the app	we can display the database data		9	4
10	create settings page for the app	the user can choose what data to display		10	3
11	create an error log in the app	the user can view errors		11	3
12	make it possible to type notes for each error in the error log	the user can save notes related to the error		12	3

4.1 Additional Requirements Not Covered by the Backlog

- All the necessary operations will be performed in the server, the client will only serve as a way get the information from the server.
- Make the client user friendly and enjoyable (if possible):
    - Use gamification to motivate users to log events.
    - Fixing idle times faster could give “points of efficiency”.
- Optionally there will be the possibility to solve possible errors and search through the log to find previous solutions.