Lazy Lab Inventory

Namn: Simon Lindqvist, Casper Aborres, Mohammed Kweder, Abdalrahman Mohammed, Mohammed Isak Obaid

Program: Civilingenjör i AI och Maskininlärning

Kurs: Systemutveckling

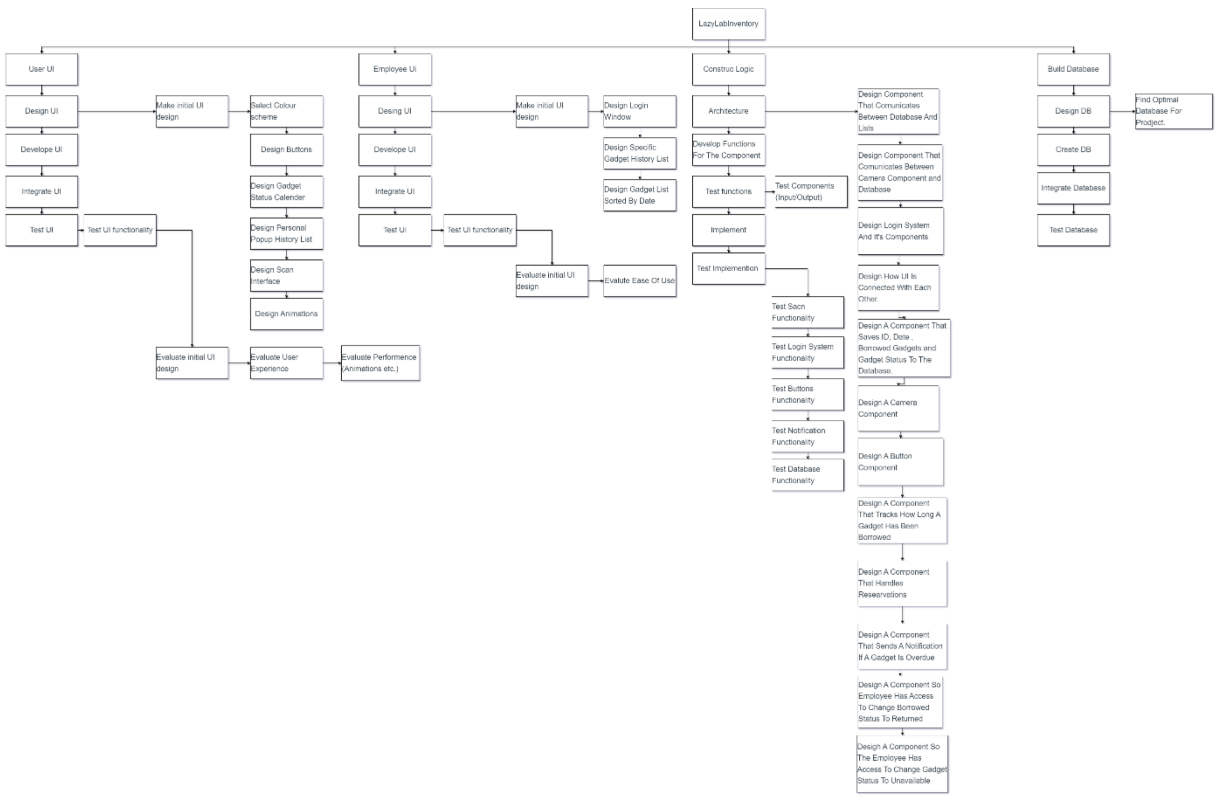
Handledare: Andreas Bauer, Waleed Abdeen

Datum: 2023-03-13

1. Innehåll

2. Work Breakdown Structure

In the first version of the plan our plan wasn't divided into enough smaller components. To fix this issue we did a greater breakdown in our work breakdown structure. We are however not allowed to change anything in the first six weeks of the GANTT chart. This means that everything in the work breakdown structure not included in the GANTT chart is assumed to have been done in the first six weeks of the project in the parts we are not allowed to alter.


LazyLabInventory
User UI
Design UI
Develope UI
Integrate UI
Test UI
Test UI functionality
Make initial UI design
Select Colour scheme
Design Buttons
Design Gadget Status Calender
Design Personal Popup History List
Design Scan Interface
Design Animations
Evaluate initial UI design
Evaluate User Experience
Evaluate Performence (Animations etc.)
Employee Ui
Desing UI
Develope UI
Integrate UI
Test UI
Test UI functionality
Make initial UI design
Design Login Window
Design Specific Gadget History List
Design Gadget List Sorted By Date
Evaluate initial UI design
Evalute Ease Of Use
Construc Logic
Architecture
Develop Functions For The Component
Test functions
Test Components (Input/Output)
Implement
Test Implemention
Test Sacn Functionality
Test Login System Functionality
Test Buttons Functionality
Test Notification Functionality
Test Database Functionality
Design Component That Comunicates Between Database And Lists
Design Component That Comunicates Between Camera Component and Database
Design Login System And It's Components
Design How UI Is Connected With Each Other.
Design A Component That Saves ID, Date , Borrowed Gadgets and Gadget Status To The Database.
Design A Camera Component
Design A Button Component
Design A Component That Tracks How Long A Gadget Has Been Borrowed
Design A Component That Handles Researvations
Design A Component That Sends A Notification If A Gadget Is Overdue
Design A Component So Employee Has Access To Change Borrowed Status To Returned
Desigh A Component So The Employee Has Access To Change Gadget Status To Unavailable
Build Database
Design DB
Find Optimal Database For Prodject.
Create DB
Integrate Database
Test Database

3. Initiation

3.1. Team Members

We are a group consisting of five members. Contact information is provided below.

- Abdalrahman Mohammed: abmm22@student.bth.se
- Simon Lindqvist: siln22@student.bth.se
- Casper Aborres: caab22@student.bth.se
- Mohamad Isak Obaid: isob22@student.bth.se
- Mohammed Kweder: mokw22@student.bth.se

3.2. Establishing Project Goals

The goals of the project are to allow students to borrow gadgets with a simple process that avoids complicated steps, e.g., creating additional accounts in order to borrow items. The lab shall have a tablet which functions as the interface to the user. The user can use to scan his/her ID-card (BTH-card) to login and gadgets with QR-cod to be able to be scanned and borrowed.

3.3. Time Calculation

In this project we have about 3 months to finish it. Each member should work about 40/week. Members are to freely dispose their time during the week to meet their goals, even if this means working on weekends. The total weeks are 12 which means that every member should work 480 hours. Summary of all members´ workings time is 2400 hours. After the stakeholders demanded changes after the first six weeks had passed, the project was assigned an additional month of extra time. We used past experiences in time estimating. The experience came from past projects and university courses we've had that involved software design and development.

4. Planning

4.1. Refined Requirements

4.1.1. User

- REQ1: The system shall be able to automatically identify a specific gadget.
- REQ2: The system shall not require the user to create an account and save user data.
- REQ3: The system shall be able to display gadgets borrowed by the user.

4.1.2. Employee/Admin

- REQ1: The system shall have an option to login to the employee account.
- REQ2: The system shall be able to view the borrowed gadgets sorted by date.
- REQ3: The system shall be able to let an admin see the history of a specific gadget.

4.1.3. System

- REQ1: The system shall be able to store data on if a gadget is borrowed or not.
- REQ2: The system shall be able to display the status of all gadgets.
- REQ3: The system shall be able to save the dates when gadgets are borrowed.

- REQ4: The system shall be able to create/delete employee accounts.

4.1.4. Additional Requirements

- The system shall keep track of how long a gadget could be borrowed.
- The system shall be able to show a calendar in which gadgets current and future status are shown.
- The system shall be able to place reservations on gadgets for a specific time.
- The system shall be able to let employees change the borrowed status on gadget to returned.
- The system shall be able to send notifications to employees if a gadget is overdue.
- The system shall be able to set the status of a gadget to unavailable.

4.2. GANTT

We have created a GANTT chart (see submitted excel document) which provides us a clear overview over the project and helps us finish the project in time. By breaking down the project into small tasks and get these tasks time-limit, we will have a better chance to get the project finished, with correct time estimation.

4.3. Risks analysis

Our definition of risk levels. Risk levels determine how probable an event is to occur. High levels of risks must be prioritized first during the process.

- **Low**: It means that the situation has a low probability of happening and minimal chances of causing harm or danger. Approximately a risk of happening of 0-20%.
- **Medium**: It means that the situation has a greater probability of happening and could cause larger issues and harms than low risks. Approximately a risk of happening of 20-50%.
- **High**: It means that the situation is very likely to happen and could cause big issues and harms than other risks. Approximately a risk of happening of 50-100%.

Our definition of consequence levels. The meaning of consequences in this report is the negative effect that happens when a risk has happened. The higher the consequence level is, the more negative effect it has on the project. The consequences with high levels of consequence should be handled with a priority over lesser levels.

- **Low**: Affects the progress in a negative way but will hardly venture the success of the whole project.
- **Medium**: The effect after medium consequences could cause some time losses and affect the project process in a highly negative way. Medium consequence levels may evolve into further problems in the future.
- **High**: This means that the project is badly affected after a risk has occurred. In such a situation, we must take quick action or otherwise the whole project is at stake.

As we were discussing with the group about risks and problems that may show up during the work, we found out some risk the are low and some others that we should focus on. Observe that after every risk we put a possible strategy to avoid it.

- Storage management. low risk, high consequences: The database may grow larger over time or be subject to attacks substantially increasing storage use to brick the system. This is a low risk, especially if doing an offline application. The consequence of this would at the worst be loss of data regarding borrowed items and potentially loss of stock. **Solution**: Use a database that removes old enough data entries or removes old entries to preserve a certain maximum database file size.

- Cyberattacks. low risk, low consequences: If the application is connected to the internet, cyber-attacks are a threat. This could result in loss of data and leakage of data such as admin passwords and personal information such as personal numbers (this is public records though). **Solution**: Avoid internet connection for the application

- Impersonation. low risk, high consequences: (By using a copy of an admin's bth card you can impersonate an employee and whitelist unauthorized users) Solution: Require a password when logging in to the admin interface.

- Users falling to understand the user interface. medium risk low consequences: (If not developing a good enough user interface users are at risk of not properly using the software. This has no real consequences except frustration) **Solution**: Test different UI layouts to make sure to optimize it.

- Users not understanding the language. low risk low consequences: (If using either Swedish or English there may be people unable to understand everything) Solution: Either assume students know English or have multiple languages available.

- Time estimation. low risk medium consequences: (There's always the risk of not estimating the time correctly, which can cause the project to run over the deadline. This will cause additional expenses) **Solution**: Plan more rigorously and/or have a buffer of time in the planning to account for sudden delays.

4.4. Dependencies

4.4.1. Libraries

The project will need a few libraries function properly and to make development as easy as possible. Some of the more notable dependencies (libraries) we figure might be needed are:

- Date/Almanac library to help the implementation of reservation functionality.
- Mail server library to help implementation of notification functionality.
- Graphics library to implement the UI properly.
- QR-code handling library for scanning functionality.

While libraries might make the development easier, it also introduces a form of uncertainty to development. When using libraries there's always a risk of the owner of the libraries changing something your own product depends on, thus breaking your product and making further development time needed. To help mitigate this, common libraries that rarely changes their vital parts are recommended to be chosen.

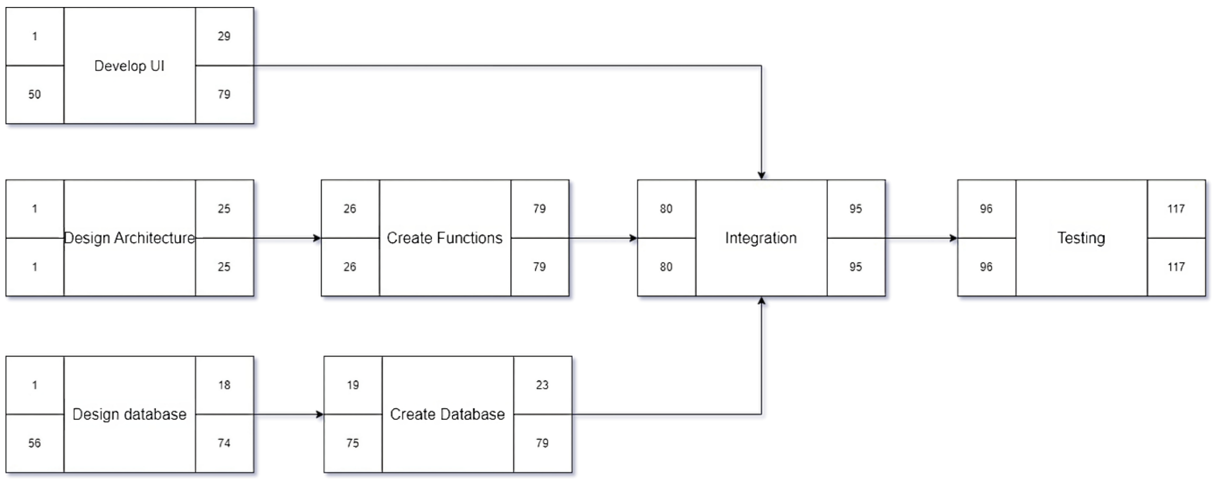
4.4.2. Dependencies in development

In the GANNT chart, dependencies can be read as, if a part starts after another; it is dependent on that other part. For example: The implementation of anything cannot begin until all other parts have been developed. Thus, development have to wait until the last part "Develop Functions New Components" is completed (see GANTT chart).

5. Memo/Discussion with Owner

- Asked owner if we had to account for possible theft of items. Owner said such precautions were unnecessary.
- Asked owner if a limit of maximum simultaneous borrowed items was needed. Owner said such precautions were unnecessary.
- Asked owner if a maximum amount of time an item could be borrowed (and potentially consequences) was to be implemented. Owner said it wasn't necessary.
- Asked owner if an admin interface really was necessary, since all information in requirements could be public without consequence. Owner said a functionality for an admin to explicitly login was needed.

6. Critical Path



7. Contribution

The entire team has been involved throughout the whole process, since all progress has been made where the entire group was present (and everything was discussed in group). The contribution of individual team members is therefore deemed to be equal.

8. Reflections

8.1. Deviations Cause

The main cause of the deviations was bad management and an inexperienced planer. Management missed to plan for some potential risks, like the cyber-attack we experienced early on. The attack introduced a delay in the project which could have been partially negated if proper countermeasures were in place. Avoiding cyber-attacks can be hard or plain impossible, but ensuring that data is backed up, that slack is present in the GANTT chart and that the general plan can handle an amount of deviations, will minimize the impact major setbacks such as cyber-attacks has on the project.

In the first draft management also forgot to include outside dependencies and what to do if those dependencies caused any problems. Those dependencies produced some deviations in the first 6 weeks but was properly addressed in the second draft of the plan. This led to a major setback when one of the libraries we depended on changed their syntax, and we had to adjust all of our code to see it working again.

Management completely left out any planning of the testing strategy which caused delays throughout the entire project. They did however plan how long time they thought the testing would take and left the method of testing to the individual developer. If management also planned what method to test with, they could have minimised or completely removed the deviation caused. It's therefore advised to include method of testing in future projects, in order to further optimize the time estimation capability of the project plan, which can reduce eventual time over deadline and save the company precious funds and reputation.

What Management could not have foreseen was a co-worker falling sick and needed a full week to recover. The other developers tried to make up for the loss of labour force, but it still caused delays in the project.

Management also left out any thought of how to integrate different part of the project together. What they did do was to plan that the integration needed to be worked on simultaneously and the did give a lot of time for the developers to sort out how to do it.

Management left the inside dependencies unclear in the first draft by only making them visible in the GANTT chart but that was fixed in the second draft, but it still caused some slight deviations.

8.2. What we have learned

We have got a lot of experience during this workshop. The main thing we have learned that we need to plan how to start and end a project in time. It is not easy because we need to plan the main plan, then break it down to small pieces, and above all always be ready for unexpected events. Being in a group and being active was also one of the reasons that made us work very well. We have discussed about different ideas and always chose the best one based on what majority vote.

We also learned other, different, ways and strategies to draw a plan, like planning and break it down, making a critical path and making GANTT charts. When we compare the first revision with the last one, we could see a big difference. We learned from our mistakes and fixed it which helped us to make the projects plausibility and chance of success if we ever come to plan a similar project for real.

9. Time deviation and additional costs

After altering the GANTT chart with the time modifiers gained from the last workshop, our "Lazy Lab" project had run overdue by an amount of eight days. Fictionally this was caused by the reasons mentioned in the reflections section. Eight days overdue translates to an amount of extra money spent for the project by the formula:

$$n * 10000 = m$$

, where n equals the number of days over deadline, and m equals the amount of additional money required for the project. This means that the project requires further 80 000kr, according to the calculations:

$$n * 10000 = m = 8 * 10000 = 80000$$