GNG2101

Design Project Progress Update

<GROUP NAME AND GROUP NUMBER>

Submitted by:

Rayyan Faruquee, 300257795

Luka Flanigan, 300375708

Gurkan Emre Uslu, 300350916

Takudzwa Gwavava, 300388232

AbdulWahid Ahmed, 300331802

10 November 2024

University of Ottawa

Table of Contents

[Project Deliverable Report Instructions i](#_Toc176011349)

[Table of Contents ii](#_Toc176011350)

[List of Figures iv](#_Toc176011351)

[List of Tables v](#_Toc176011352)

[List of Acronyms and Glossary vi](#_Toc176011353)

[1 Introduction 1](#_Toc176011354)

[2 Prototype 1, Project Progress Presentation, Peer Feedback and Team Dynamics 2](#_Toc176011355)

[2.1 Prototype 1 2](#_Toc176011356)

[2.2 Project Progress Presentation 2](#_Toc176011357)

[2.3 Project plan update 2](#_Toc176011358)

[3 Design Constraints and Prototype 2 3](#_Toc176011359)

[3.1 Design constraints 3](#_Toc176011360)

[3.2 Prototype 2 3](#_Toc176011361)

[3.3 Project plan update 3](#_Toc176011362)

[4 Economic and IP Considerations 4](#_Toc176011363)

[4.1 Economics report 4](#_Toc176011364)

[4.2 Intellectual property report 4](#_Toc176011365)

[4.3 Project plan update 4](#_Toc176011366)

[5 Design Day Pitch and Final Prototype Evaluation 5](#_Toc176011367)

[6 Video and User Manual 6](#_Toc176011368)

[6.1 Video pitch 6](#_Toc176011369)

[6.2 User manual 6](#_Toc176011370)

[7 Conclusions 7](#_Toc176011371)

[8 Bibliography 8](#_Toc176011372)

List of Figures

Insert your list of figures here (right-click to update this field).

List of Tables

[Table 1. Acronyms vi](#_Toc144997103)

[Table 2. Glossary vi](#_Toc144997104)

List of Acronyms and Glossary

Provide a list of acronyms and associated literal translations used within the document. List the acronyms in alphabetical order using a tabular format as depicted below.

Table 1. Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Provide clear and concise definitions for terms used in this document that may be unfamiliar to readers of the document. Terms are to be listed in alphabetical order.

Table 2. Glossary

|  |  |  |
| --- | --- | --- |
| **Term** | **Acronym** | **Definition** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

Explain the basic context for your work and any assumptions that you have made for your work. Give an overview of the structure of your document (i.e. explain how it is organized) and summarize the purpose of the document and the scope of activities.

# Prototype 1, Project Progress Presentation, Peer Feedback and Team Dynamics

## metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim Açıklama otomatik olarak oluşturulduPrototype 1

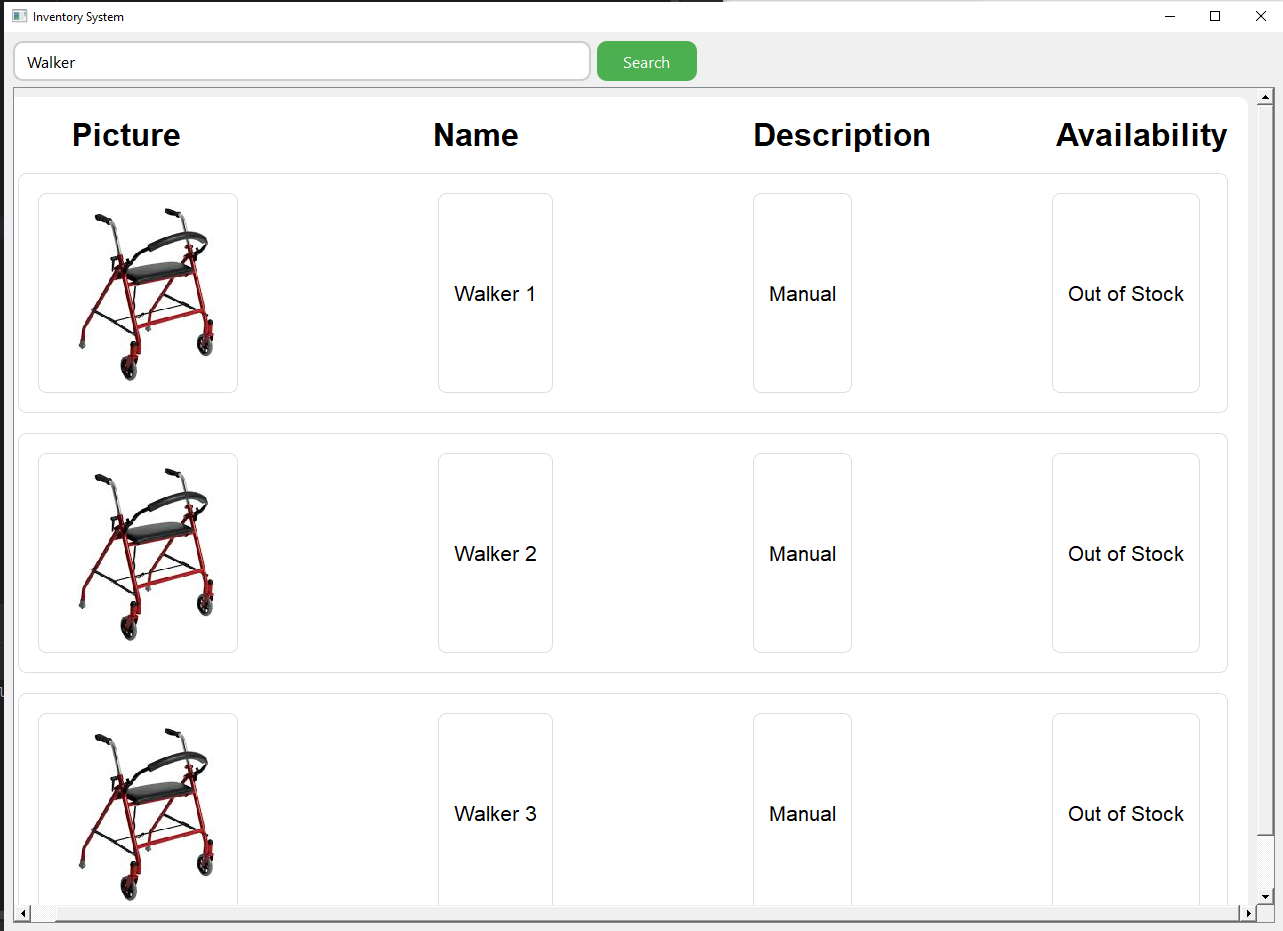
Figure 1 Coding Screen For The UI

Our code is running on pycharm using python language. There is tools to run this code PyQt6.QtWidgets, PyQt6.QtGui and PyQt6.QtCore. With the help of these tools we were able to crate our first prototype. Our main goal was to learn how to run the GUI system, see how to create a functional UI and create a basic UI that will repliacate what we are trying to acheavie in the future. In our first prototype we were able to create a UI that can do basic search for assets names, dispalye the details of the each assets.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduFigure 2 Search UI For The User

Figure 1 shows our user interface, which is designed to be straightforward and functional. This prototype was created to explore how to build an effective UI. At the top, there is a search bar where users can enter the asset name they are looking for, and it will return assets that match that name. To submit the search query, users need to click the "Search" button next to the search bar. A demonstration of the search feature can be seen in Figure Currently, we present each asset with its picture, name, description, and availability. In the future, we plan to add more details about each asset based on client feedback. The window can be minimized, maximized, or closed using the buttons in the top-right corner. On the right side of the window, there is a scroll bar that allows users to scroll up and down to view additional assets.

Figure 3 Search Fature In The UI

## Prototype 2: Asset Information Storage

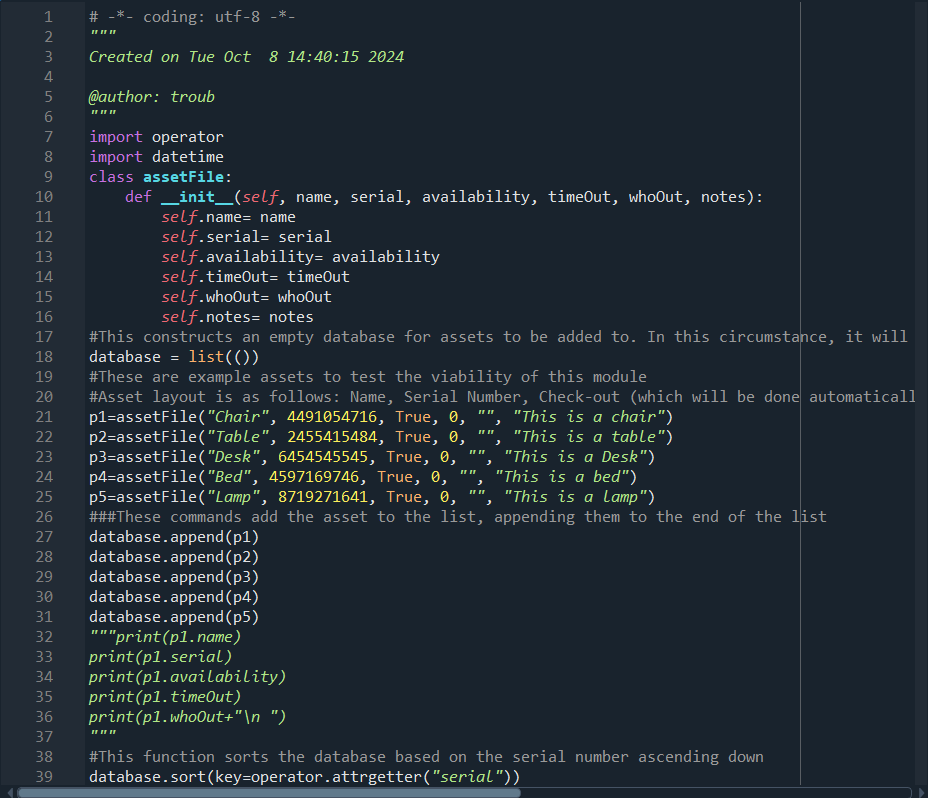
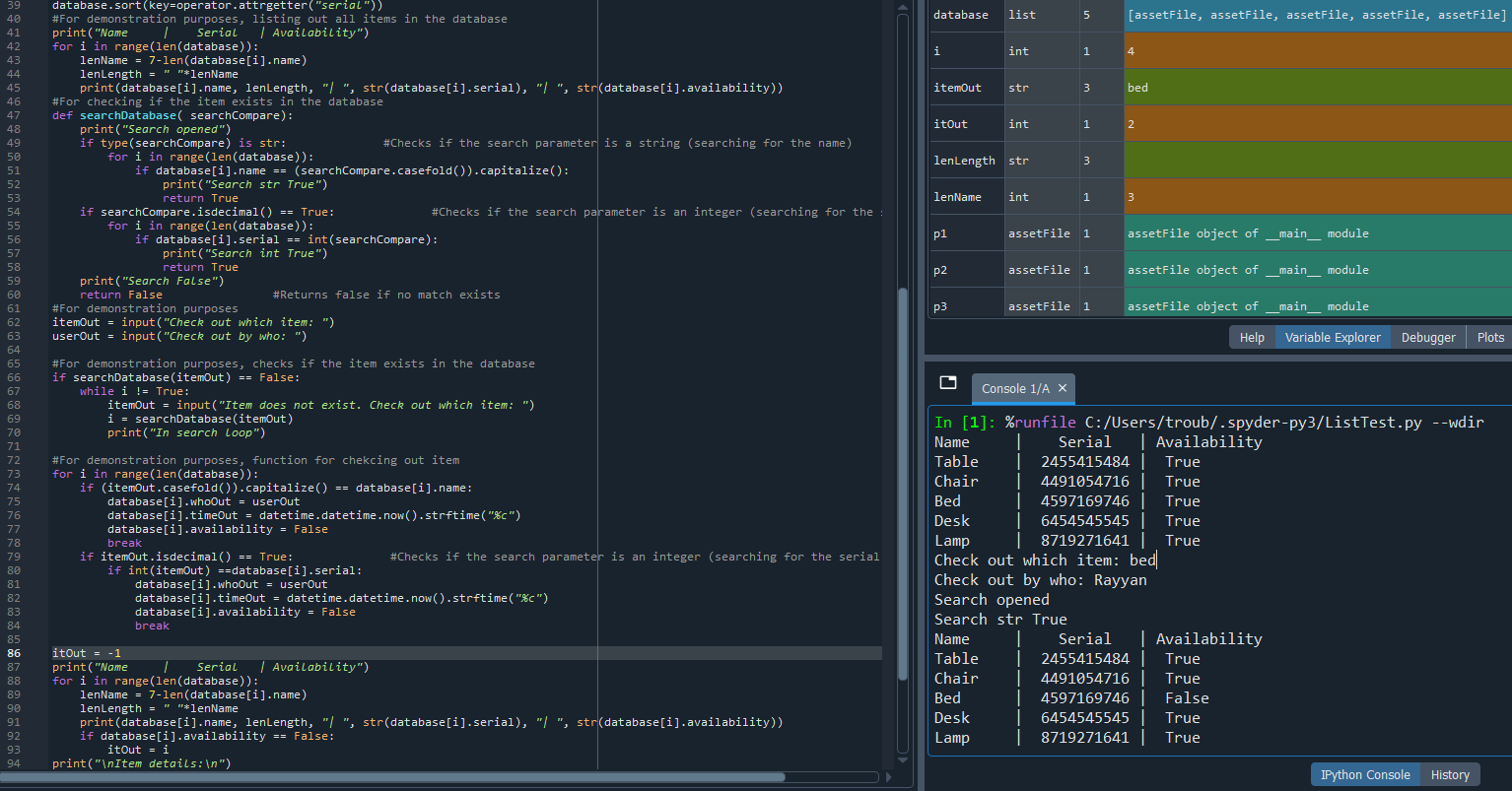
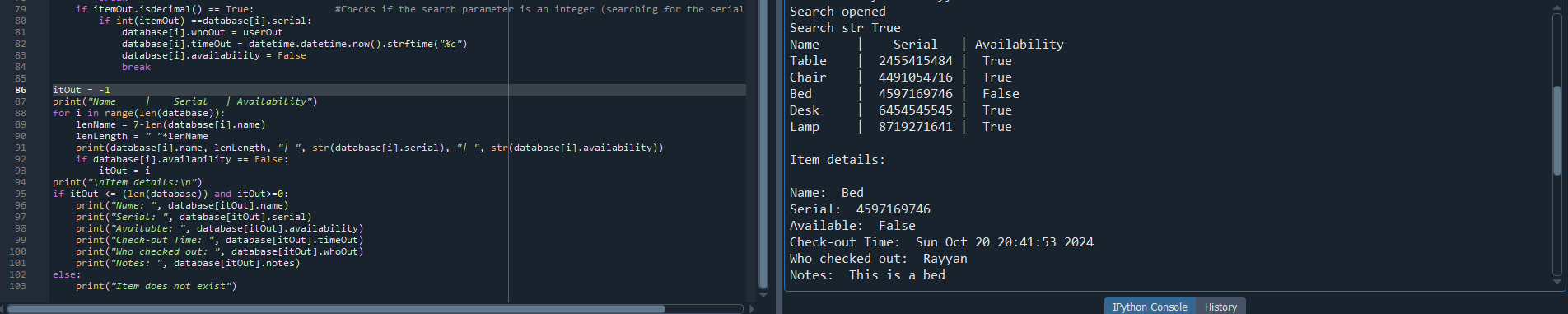


Figure 4: assetFile Class and initialisation of the database

The purpose of this prototype is to test the ability to store and search for specific information from a collection of given data in code. The code file initializes a class for the assets called “assetFile”, which currently stores the asset name, serial number, availability, and notes. If applicable, it also stores at what time it was checked out and by whom. Known as an “object”, they are then stored inside of a list called “database”, which has been initialized so that new objects can be added as desired through a method called “append”. For this prototype, five assets were generated and added to the database through the “append” method, and then sorted according to their assigned serial numbers.

Figure 5: Basic search and check-out code with test console UI

To maintain simplicity, the prototype displays object information through the console, displaying the name, serial number, and availability of each “assetFile” in a table. To test the ability to retrieve information, the code will ask the user to input the name or serial number of the asset they want to retrieve, as well as their name. The code will then determine whether the requested object exists within “database,” prompting the user for a new object request if it does not. Once the code receives a valid request, the code will print a new table, displaying that the object the user requested has been updated to be no longer available. The object information will then be displayed, showing the name, serial number, updated availability, time of being checked out, the user who checked out the asset, and the notes about the asset.

Figure 6: Code for changing asset status and updated asset information in test console UI

## Project Progress Presentation

[GNG PDE presentation.pptx](https://uottawa-my.sharepoint.com/personal/lflan013_uottawa_ca/_layouts/15/guestaccess.aspx?share=EQ7clTW8MtBGmCiof_CCescBLJuykIpOPEIJKG4dKDX_dg&e=emnpKg)

## Project plan update

Figure 7 Gnat Chart

# Design Constraints and Prototype 2

## Design constraints

## Prototype 2

During the second client meeting our team presented the progression of the project by presenting the first prototype that was developed alongside PDE. The feedback that was provided by the client was verry limited, however she did say she liked the way the development was progressing and how the UI’s appearance looked. However, she did mention that the colors appeared a bit bland. Furthermore, as the final part of the meeting we dove into our future work and explained how the goal for the second prototype was to merge the UI and the asset sign in/out systems into one functional prototype; the client did like this idea especially how it related to the previous DFX of design for testability however did express censers of the remaining time frames for the project. Finally, she expressed concerns about how easy it would be to sign in and out assets since at this moment we still had to manually input commands to sign in and out assets we expressed that the focus of design for speed would be addressed during the third prototype where we would implement scan gun technology to efficiently sign in and out assets

By the end of the meeting the client knew the exact plan of our project and how it connected to her needs expressed in the first client meeting The client seemed content about the results of prototype 1 and seemed content about the direction the second prototype was heading in which we interoperated as a green light to further our development plans for the second prototype.

The first prototype focused on designing for speed, simplicity and usability for the second prototype we decided to focus more of testing for testability by designing code checks to ensure that if an effort occurs the code check would give an idea of where the error occurs providing ease for correcting the code an ensuring that the two parts of the prototype function unanimously.

The test plans for the second prototype will be similar to the first prototype since the same features will be tested with the addition of a code check. The goals for the second prototype are to: develop an understanding of code checks, merge two codes together (and develop an understanding of how to do so) and test the efficiency of the code through tests.

Test 1: search time response: used to determine the accuracy and efficiency of the search function

Test 2: asset sign out time: used to determine the efficacy of checking in and out assets as well as any error

Test 3: code check responce: used to determine whether the code check works by purposely putting an error in the code

|  |  |  |  |
| --- | --- | --- | --- |
| Test id | Target metric | Acceptable value | Test results |
| Test 1 | Seach time <=500ms  Accuracy: 100% | Search time <= 1s  Accuracy: 90% (1 letter off) | Seach time=300ms  Accuracy: 100% |
| Test 2 | Check in/out time<=500ms | Check in/out time<=2s | Check in/out time=1s |
| Test 3 | Code check accuracy: 100% | Code check accuracy: 100% | Code check accuracy: 90% |

For the next client meeting or team would like to pivot away from the UI and focus more on integrating scan gun technology to sign people in to the system and to sign in and out assets the feedback we obtain on this aspect of the design will be critical to develop the best possible prototype for the client. Secondly since the last client meet will occur near the end of the development phase it is almost certain that some aspects from our design will have to be cut short to ensure that the quality of our product is at a high enough level to present on design day. The aspects that might be cut will relate to the client-ranked needs that were established in PDC.

## Project plan update

Add a screenshot of your gantt chart.

# Economic and IP Considerations

## Economics report

Theoretical final prototype:

The vision or our final prototype involves selling a software license to use the asset inventory system which was developed with all the features that were addressed in the detailed design of the project in PDE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | variable/fixed | material/labor/overhead | direct/indirect | Cost |
| overhead | fixed | overhead | indirect | 22000$ |
| Software license | fixed | material | direct | 1069$ |
| marketing | fixed | overhead | indirect | 30,000$ |
| labor | variable | labor | direct | 560000$ |
| Distribution cost | Variable | labor | indirect | 73.05$ |
| Legal costs | fixed | material | indirect | 6000$ |
| Development cost | fixed | material | direct | 143.6$ |

2. To determine an income statement we will need to determine a cost to sell the product. Our product has always aimed to be more of a budget version of a asset inventory app in the mid range; to determine a cost we will reflect on technical benchmarking done on PDB based on the

The market. In the deliverable the cost for a high end system (snip-it) is around $2500/year and a lower end system (Spiceworks IT Asset Management) is around 750$/year

For our product we will take a weighted average of the two asset management systems aiming for a mid range price with the difference of not charging per asset leading to our product being more on the cheaper side in the long term. This aspect is aimed to make companies want to stay with our software in the long term

cost=(750$)+(2500$))/2

cost=1625/year

|  |  |
| --- | --- |
| Revenue | Cost ($) |
| Sell price | 1625/year |
| Number of sales | 200 units |
| total | 325000$/year |

|  |  |
| --- | --- |
| Expenses | Cost ($) |
| license | 1,069 |
| Development cost | 143.6 |
| Legal cost | 6000 |
| Labor cost | 560,000 |
| Overhead costs | 2,200 |
| total | 569,412.6$ |

|  |  |
| --- | --- |
| Operating cost | Cost ($) |
| Distribution cost | 73.05$/unit |
| Operation cost | 14560 per year |
| Number of sales | 200 units |
| total | 14610+14560 $ per year= |

Assume that one worker works on bugs once a week for 8 hours at the standard rate

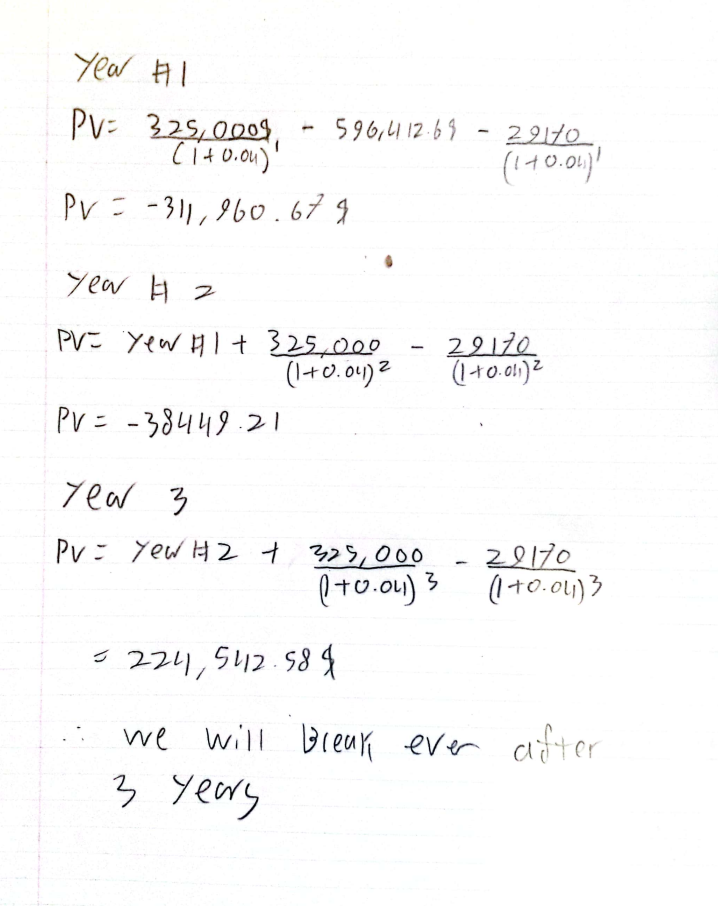
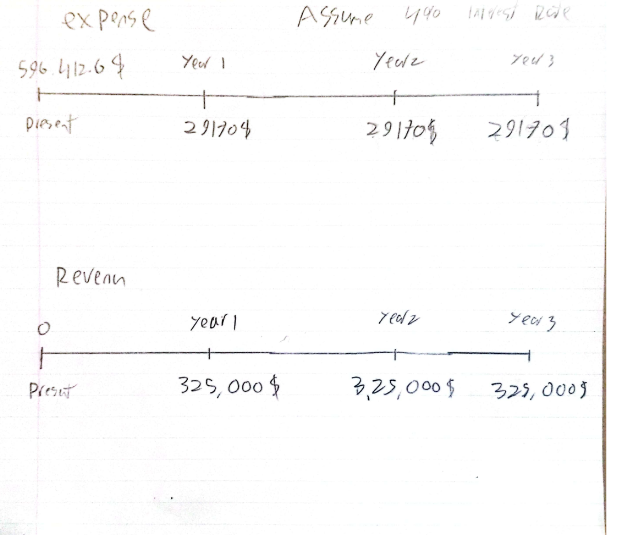
Operating cost=8\*35\*52=14560

income=revenue-expenses

income=325000$/year(3years)-569,412.6$-29170$/year(3years)

income=318,08$

3)



4) Assumptions

Based on technical benchmarking on the cost to rent out offices a good piece would be around the 5000$-6000$ per month including all electricity heating and furniture

overhead=5500$/month\*4 months

overhead=22000$

<https://www.bestinottawa.com/best-office-spaces-for-rent-ottawa/>

Assume that the app is developed in canada meaning the average salary for a software engineer based on indeed is 35$/h

<https://ca.indeed.com/career/software-engineer/salaries>

Assume that the team working on the app consists of 5 people based on user benchmarking and to relate the conditions of the course to the real world

Assume the system’s software is built with the target towards windows operating servers. The software license for Microsoft is stated in their website <https://www.microsoft.com/en-ca/windows-server/pricing>

Our final prototype will not be fully completed in the 3-month span of this course however after discussing as a team having an additional month for development would finalize our design; Furthermore, based off of feedback from software engineers the development for a team of 5-7 people takes around 4-6 months. Therefore, for simplicity we will assume the development takes 4 months. Furthermore we will assume the standard 9-5 work is used in the development meaning each member will work 40 hours a week.

Labor=40\*35$*80\*5*

Labor=$560000

<https://www.uptech.team/blog/how-long-does-it-take-to-make-an-app#:~:text=What%20is%20a%20standard%20mobile,or%20a%20mobile%20banking%20app>.

Since our product is a software app typical distribution costs will be different since transportation physical components will not be necessary. Instead, we assume an IT worker will have to help set up the software with the company's servers. Based on indeed the average salary is 24.35$/h and based on it workers feedback it typically takes an average of 3 hours to set up software to a server

Distribution cost=24.35$/h\*3h

Distribution cost=73.05$ per sale

<https://ca.indeed.com/career/it-technician/salaries>

<https://www.reddit.com/r/msp/comments/9bngav/realistic_amount_of_time_to_set_up_a_new_computer/>

We will assume our app falls under a simple complexity since an example for a medium complexity app is Instagram and snapchat both which have more complex features then the asset inventory system we developed. Based on this assumption the average cost to get a patent of a software app is around $6000 (as of 2024).

<https://gearhartlaw.com/how-much-does-a-software-patent-cost-in-2024-pricing-breakdown-from-top-patent-lawyers/#:~:text=Breakdown%20of%20Average%20Attorney%20Fees,a%20moderately%20complex%20software%20invention>.

For the development cost we will assume the team already has computers and all software required to develop the code mimicking the simulations of this course. Based on this assumption we will take the development cost for this course and add a small weight to it since we won't be finishing the complete design; this weight will be a multiplier of (4/3) since a previous assumption stated that the development cost for the complete prototype will be 4 months instead of the 3 months in this course

Development cost= (107.7$)(4/3)

Development cost=143.6$

To estimate the number of sales per year the use of technical benchmarking will be made. An asset inventory system named asset panda is a similar mid-range asset inventory system around the spot where our system would be aimed for. The difference between the ours is that we would not charge per individual asset, however we will assume we will have similar sales as them. Although the exact sales are not publicly stated anywhere using the profit and cost we can estimate that the company makes around 200 sales per year.

<https://databox.com/saas-sales-benchmarks>

<https://info.flexera.com/ITAM-REPORT-State-of-IT-Asset-Management?utm_source=google&utm_medium=ppc&utm_content=itam_sam&lead_source=PPC&cq_cmp=21426658752&cq_term=software%20for%20asset%20management&cq_plac=&cq_net=g&cq_plt=gp&gad_source=1&gclid=CjwKCAiAxea5BhBeEiwAh4t5K5WyKd2pt21r6HhgergK8jaVm6lqUBTlZT70hFnjh0KcxsFy9mUFrhoCadMQAvD_BwE>

<https://firstpagesage.com/reports/saas-benchmarks-report/>

## Intellectual property report

## Project plan update

Add a screenshot of your gantt chart.

# Design Day Pitch and Final Prototype Evaluation

Hi, my name is Luka, and we are the matrix, the codeiest team here. Have you guys ever had a slow program? It's just the worst. It gets frustrating and slows you down tremendously. Our product, which we designed, aims to eliminate the efficiency problem among asset inventory systems, which are an essential aspect across all companies.

Our problem statement describes the engineering problem at hand, which we are aiming to create the best solution for.

The staff at Bethany's Children's Health Center seek an invitation to their asset inventory system to address the limitations of their current system, focusing on usability and efficiency aspects. the problem involves developing an inventory system that provides a fast way to check in and out assets while being simple to use and incorporates a diverse set of sub-features, including a search feature, tracking feature, and an information display, while being cheap to run.

To save as much time as possible, we designed our system to have a minimal amount of inputs in order to reduce time; the use of scan gun technology is used to achieve this goal. To sign into the system, staff will scan a barcode on their badge, and then, to sign out an asset, they will simply scan the barcode on the assets. This is in contrast to many other systems that require manual inputs to sign in and search for assets, causing significant loss in time. Do a demonstration here

Moreover, a second factor differentiating us from our competitors is that our system does not charge per asset, which means it will be more affordable for businesses aiming to expand and take risks without heavily punishing them.

Furthermore, the simple method of checking in and out assets makes the design easy to use, especially when paired with its simple UI. This design focuses on its implementation in the workspace so that new hires can integrate smoothly into the workplace.

Paired with the reduced cost and simple use, our design excels in providing companies with the opportunity to expand their team and asset collection while maximizing efficiency

To conclude our time with you, our design utilises scan gun technology to improve the efficiency of asset inventory systems by providing a simple and fast way to check in and out assets; this and the concept of not charging per asset makes the best asset inventory system for companies due to the economic and efficiency factor.

# Video and User Manual

## Video pitch

Add link to video.

## User manual

See separate template for the user manual. Do not write the content here.

# Conclusions

Summarize your lessons learned and your work related to your project. Discuss any outstanding issues or implications for the project.

# Bibliography

Insert your list of references here.