**Front**

* Abstract
* Table of contents
* Keywords

**First half**

1. Introduction

* Background
* Problem statement
* Research questions
* Limitations
* Thesis structure (what will be covered in each chapter)

1. Method

* Literature review (how we found, keywords etc)
* Experiment (general experiment description, hardware, software)

1. Literature study

* Theory, background, explain what everything is, facts (need more stats)
* Related work (other research results)
* Results from literature study (possible answer to research question, how did the literature study effect the experiment)

1. Experiment

* Detailed desc of experiment (how we did the experiment)
* Results from experiment? Without analysis

1. Results and Analysis

* Make the comparison, answer research questions

1. Discussion

* What happened during study, experiment
* Ethical aspects
* Future work

1. Conclusion

* Summarizing

1. References
2. Appendix

Missing:

* Chapter 1: Background
* Chapter 3: Literature study (statistics)
* Chapter 4: Results (android data, description of data)
* Chapter 5: Results and analysis
* Chapter 6: Discussion
* Chapter 7: Conclusion

**Tables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | First steps | viewing the products | adding the products to cart | checking out |
| Command line | 11,289 | 41,397 | 7,725 | 25,24 |
| graphical | 4,485 | 5,835 | 0,24 | 2,736 |
| touchscreen | 4,345 | 15,048 | 6,084 | 6,752 |

­

# **Method**

# **2.1. Literature study (how we found, keywords etc)**

**2.1.1. Keywords**

* User interface
* Command Line, Menu-driven, Graphical, Touchscreen, Text-based
* Comparison, analysis, Efficiency, performance, productivity, effectiveness

**Specifications:**

* Publication date: 5 last years
* Content type: Journal article
* Discipline: computer science, engineering
* Language: English
* Limit to: peer reviewed publications

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Key words | Keyword relation | Full text results | Title only results | First Search | Second Search | Third Search |
| 1 | user interface | OR | 76,028 | 434 | Title only | Title only | Title only |
|  |  | AND |  |  |  |  |  |
| 2 | command line, menu-driven, graphical, touchscreen, text-based | OR | 97 | 44 | - | Title only | Title only |
|  |  | AND |  |  |  |  |  |
| 3 | comparison, analysis, performance, research, efficiency | OR | 43 | 15 | - | - | Full text |
| Total matches: | | | | | 434 | 97 | 43 |

1. **First search - 434 results**

* User interface (title)

1. **Second search - 97 results**

* User interface (title) AND
* Command Line OR Menu-driven OR Graphical OR Touchscreen OR Text-based (title)

1. **Third search - 43 results**

* User interface (title) AND
* Command Line OR Menu-driven OR Graphical OR Touchscreen OR Text-based (title) AND
* Comparison OR analysis OR performance OR research OR efficiency (full text)

<https://hkr.summon.serialssolutions.com/#!/search?ho=t&include.ft.matches=f&fvf=ContentType,Journal%20Article,f%7CDiscipline,computer%20science,f%7CDiscipline,engineering,f%7CLanguage,English,f%7CIsPeerReviewed,true,f&rf=PublicationDate,2017-05-04:2022-05-04&l=en&q=(TitleCombined:(User%20interface))%20AND%20((graphical)%20OR%20(command%20line)%20OR%20(menu-driven)%20OR%20(text-based)%20OR%20(touchscreen))>

<https://hkr.summon.serialssolutions.com/#!/search?ho=t&include.ft.matches=f&fvf=ContentType,Journal%20Article,f%7CDiscipline,computer%20science,f%7CDiscipline,engineering,f%7CLanguage,English,f%7CIsPeerReviewed,true,f&rf=PublicationDate,2017-05-04:2022-05-04&l=en&q=(TitleCombined:(User%20interface))%20AND%20((TitleCombined:(graphical))%20OR%20(TitleCombined:(command%20line))%20OR%20(TitleCombined:(menu-driven))%20OR%20(TitleCombined:(text-based))%20OR%20(TitleCombined:(touchscreen)))%20AND%20((comparison)%20OR%20(analysis)%20OR%20(performance)%20OR%20(research)%20OR%20(TitleCombined:(Efficiency)))>

<https://hkr.summon.serialssolutions.com/#!/search?ho=t&include.ft.matches=f&fvf=ContentType,Journal%20Article,f%7CDiscipline,computer%20science,f%7CDiscipline,engineering,f%7CLanguage,English,f%7CIsPeerReviewed,true,f&rf=PublicationDate,2017-05-04:2022-05-04&l=en&q=(TitleCombined:(User%20interface))%20AND%20((TitleCombined:(graphical))%20OR%20(TitleCombined:(command%20line))%20OR%20(TitleCombined:(menu-driven))%20OR%20(TitleCombined:(text-based))%20OR%20(TitleCombined:(touchscreen)))%20AND%20((comparison)%20OR%20(analysis)%20OR%20(performance))>

Program description

**Identify the problem: - what are you trying to do?**

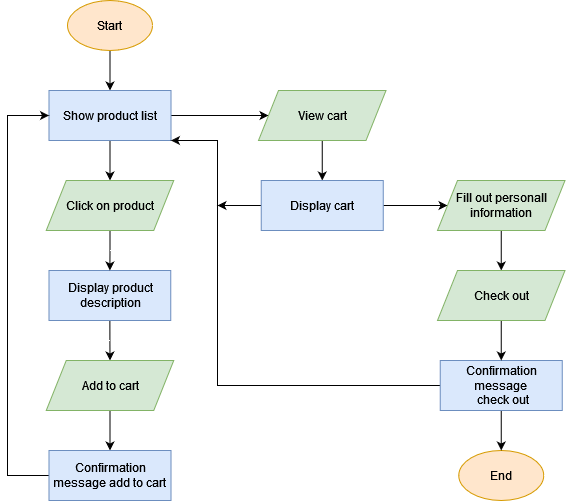
The first stage of developing the program is to identify the problem and solution. In this case the problem presented is the ability to purchase product(s) through a webstore in the shortest amount of time possible. The two steps to find the solution are identifying the requirements and specifications. The requirements consist of the necessary components of the program to reach the goal. The specifications consist of the steps of the program to fulfill the requirements.

* Requirements:
  + Display product list
  + Display product description
  + Add product(s) to cart
  + Check out
* Specifications:­­
  + List all the available product with name and possible image
  + Select a product to view detailed information
  + Add the selected product to the cart
  + View the cart with the products that have been added
  + Fill out information about full name and address
  + Check out
  + Display confirmation messages after actions take place

**Designing a solution: - how is it going to be done?**

The second stage is designing a solution for the problem. In an effort to implement the solution, the steps the program has to take to reach the goal are formulated. This is done in pseudocode and a flowchart, see figure 1.1. Presented below are the functional steps the program takes to go from start point to end point. In the most basic scenario, the user will view the lists of products, select a product, add the selected product to the cart, view the cart and finally check out. The main actions the program takes are displaying, reading, and storing product information.

1. **BEGIN**
2. **DISPLAY** product list
3. **READ** product selection
4. **DISPLAY** product description
5. **READ** product name
6. **STORE** product to cart
7. **DISPLAY** confirmation
8. **DISPLAY** cart
9. **READ** full name and address
10. **DISPLAY** confirmation
11. **END**



**Writing the program: - what are the specifications? Can we run it?**

For the program to be developed accordingly, it must be coded, compiled, and debugged. Coding the program will depend on the platform and act in accordance with the designed solution. However, it is crucial that the programmer follows guidelines: write as few lines as possible, use appropriate naming, split code into decent size sections and progress minimal error handling. The program must be compiled before it can be run. The computer does not understand the high-level programming language and must be converted into low-level binary language, made up of 0’s and 1’s. Finally, the code must be debugged before the program is finished. This indicates the removal of any existing or potential errors inside the code. If any errors occur in the compiled code, the program might potentially behave unexpectedly or crash. It is of utmost importance to prevent any delays in the testing portion of the thesis project.

Every step the user takes to navigate through the program has to be simple and clear. This implies that the instructions must be formulated properly so that the user can understand its meaning and purpose. Visual components must be displayed with relevant size and position in accordance with web standards. The use of buttons and text fields must be of average size with text size between 18 - 23 px. Each page or section must be clearly divided either by containing an appropriate title or spacing.

* Specify what the layout should look like, interacting should be like – reference

**Check the solution: - is it solving the problem?**

Every listed step the program performs must be tested by the programmer. The next step is not pursued until the previous step is completed. After each step is completed, the programmer tests the finished program to perceive if the requirements meet. The main question declares if the program is serving a solution to the problem.

* Confirm that the program has a solution
* Match requirements to the solution – make comparison

**Command Line interface development**

The command line interface (CLI) is written in java. Java is a high-level object-oriented programming language. For the computer to understand the code and run it, the code first must be compiled by the javac compiler. The compiler transforms the source code (java files) into byte code (class files). The byte code can be run on all platforms that support Java Virtual Machine (JVM). Each operating system has a different JVM, however the output they produce after execution of byte code is the same across all operating systems.

The integrated development environment (IDE) used for developing the program for the command line is IntelliJ IDEA, by JetBrains. IntelliJ is the most used IDEA for development of java programs because of features such as coding assistance, build in tools and integration, plugin ecosystem, language support, frameworks, and connection to version control.

The program consists of a single class written in Java, see figure 1.1. Each method in the java class performs a unique task requested by the user. Each of these methods can be called several times, for example to return to the products list after adding a product to the cart, or when the user enters an invalid command. The user interacts with the program by reading the text-based menu and can input menu selections or details.

* Selection method

The deployment of the software:

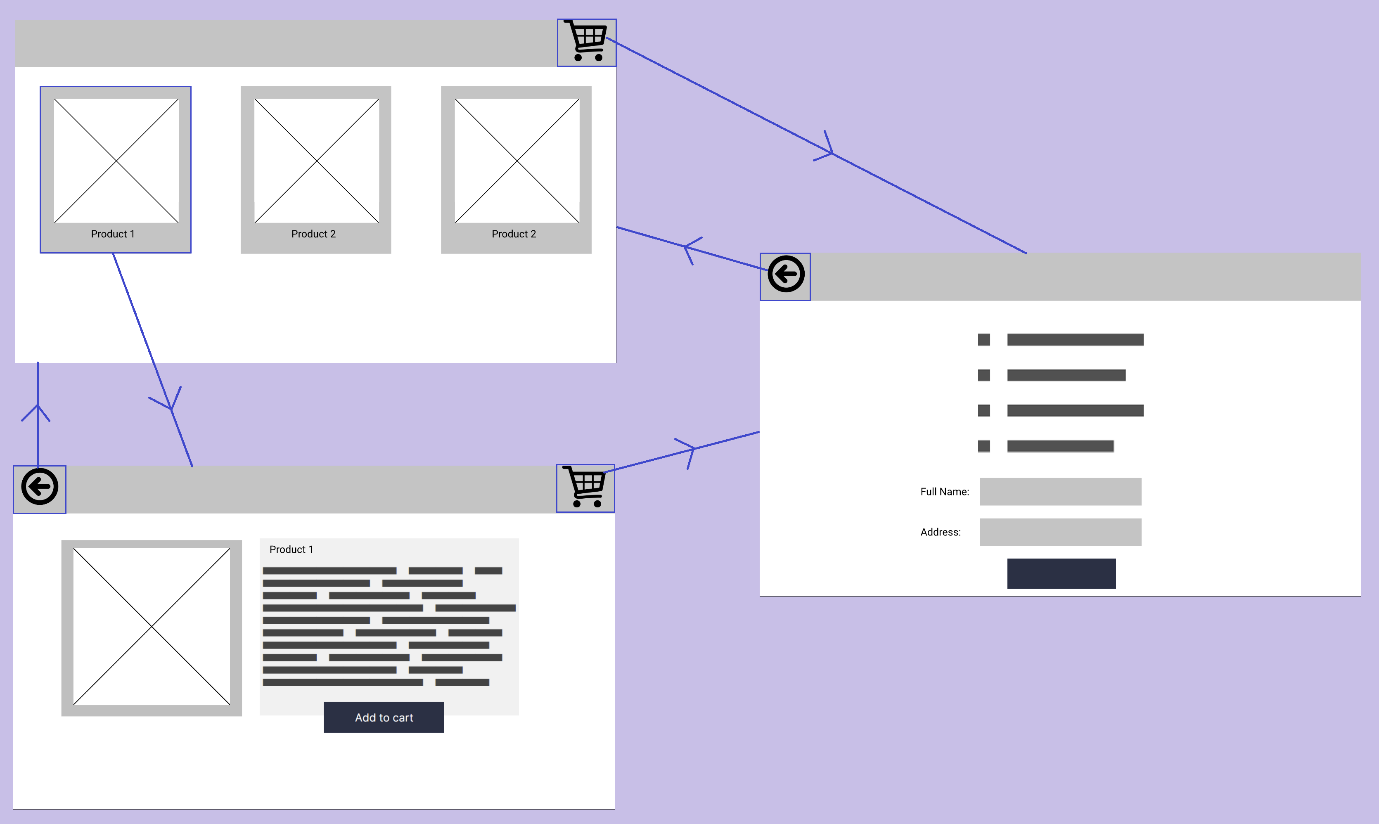
1. Navigate with cmd or terminal to the project folder
2. Run the program from the command line ‘java main’

Table

Description automatically generated

**Graphical user interface development**

The development for the program in relation to the graphical user interface starts by the creation of wireframes. Wireframes are a screen blueprint for the basic functionality and structure of the layout. The frames contain visual representation of pictures, text, buttons, menu bar and context area. Figure 1.2 displays the three frames and the connections between them. The first frame (top-left) represents the product listings each with a picture and title of the product. The user can click on the product div, the section of the HTML document containing the product listing, to be redirected to the second frame (bottom-left) to add the product to the cart and view the product description. To view the third frame (middle-right) the user can click on the shopping cart icon located in the top right inside the navigation bar. The icons used are representative of the function performed when clicked. For example: the return button, made from an arrow key pointing backwards with a circle to indicate it is a button. The product description and cart list fill a portion of the page, making sure they are readable and gain attention of the user by the placement in the center of the page.



The languages used for developing the front-end for the graphical user interface are HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript. HTML is the standard markup language for browsing the web and is used for the page structure containing its plain text, title, buttons, images, and much more. CSS makes up the styling for the HTML elements, giving them shape and color. Finally, JavaScript is used to change the page contents, handle page interactions, make external calls and process data (calculate, manipulate, validate).

The back end of the graphical user interface is served by a web server on Node.js built on Google Chrome's JavaScript Engine (V8 Engine), developed in 2009 by Ryan Dahl. Node.js is an open-source asynchronous event-driven runtime environment that executes JavaScript outside the web browser. This allows the website to be run locally from our own server. The interaction between back end and front-end are both written in the same language, making it easy and compatible to work with both sides.

Node.js includes a package manager. A package manager is a collection of software tools that automates processes that evolve around packages. A package is a third-party bit of software usually written by someone else to solve a problem. Installing a package may include sub-dependencies. The package manager takes care of correctly installing/uninstalling packages, managing storage locations, including packages correctly into the project, handling duplicate packages and more. Npm, short for Node Package Manager, is the default package manager for Nodejs. Npm consists of a command line client and an online database for both public and private packages. There are alternative package managers such as ied, pnpm, npmd and Yarn. The back end uses a npm module Express, a minimal and flexible Node.js web application framework that provides a robust set of features for web applications. This framework helps organize the web application into an MVC (Model-view-controller-model) architecture on the server side.

Node.js is provided with a rich library of various JavaScript modules and depending on the integrated development environment (IDE) can be supplemented with various extensions. Some of the most used IDEs for web development are: Visual Studio Code, Atom, Sublime Text, PyCharm, Ruby Mine and NetBeans. Choosing a fitting IDE depends on the language used for the project and the requirements of the project, for example the need of certain extensions or compatibility with external connections. For the development of the graphical user interface project the choice of IDE is VS Code (Visual Studio Code), the most popular choice. The software is open-source, simplistic layout, minimal design, rich extensions support and works perfectly for a smaller project.

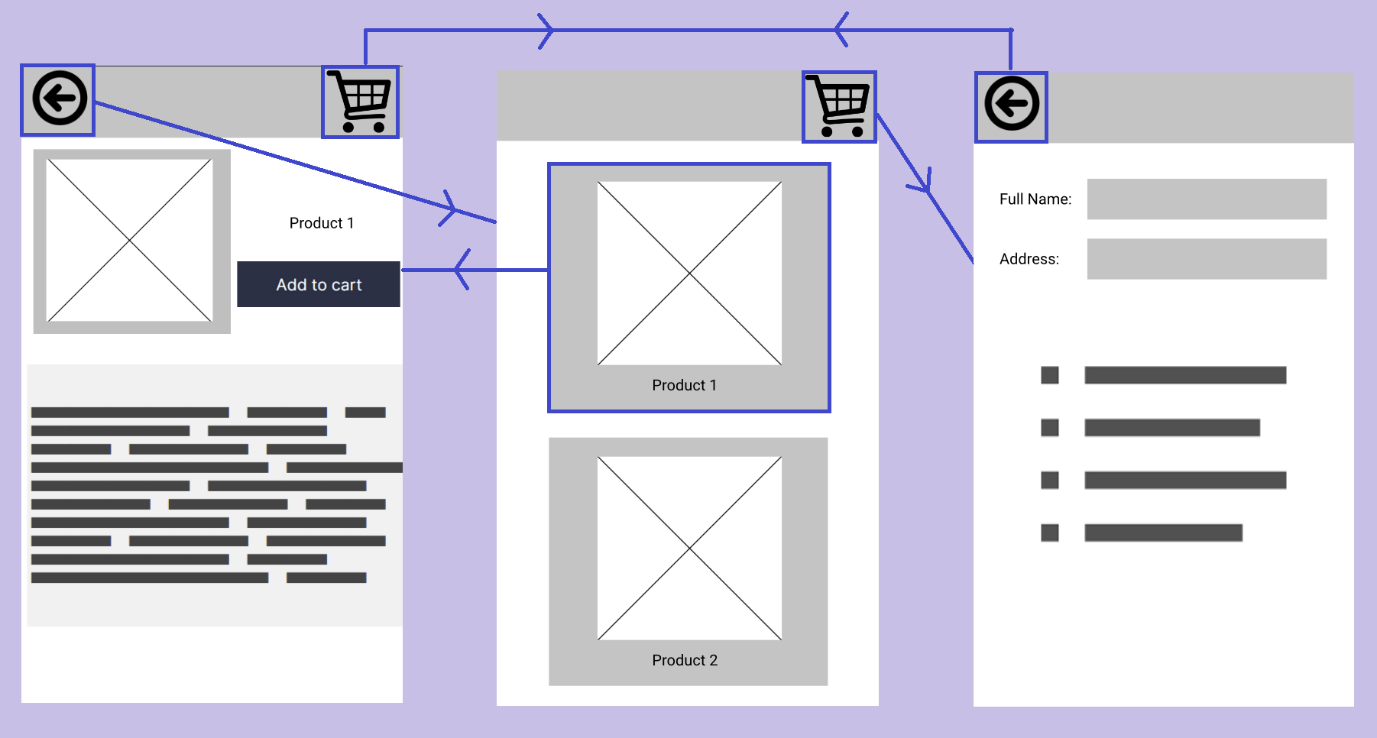
* Compiling, debugging
* The development and design of the program matches the requirements as well as the wireframes.

The deployment of the software:

1. Navigate with cmd or terminal to the project folder and start the server ‘*node server.js’*
2. Open the web browser and browse to *‘https://localhost:3000’*. Replace ‘*3000’* if another port is specified.

**Touchscreen interface development**

The development for the program in relation to the touchscreen user interface starts by the creation of wireframes. The wireframes are similar to the ones created for the graphical user interface for the web, section 1.1. The biggest differences are the positioning of components and the amount of screen space each component needs.



* Command line is much better for expert in comparison to novice
  + Graph difference is huge
  + Misspelled mistakes with commands by novice user
  + Literature research was similar – expectations
* Graphical user interface intermediate
  + Difference is small between users
  + Fastest for novice
  + Medium for expert
  + Refer to related work reference