|  |  |
| --- | --- |
| Kristianstad University  SE-291 88 Kristianstad  +46 44-250 30 00  www.hkr.se |  |

Independent project (degree project), 15 credits, for the degree of Bachelor of Science (180 credit) with major in Computer Science  
Spring Semester 2021  
Faculty of Natural Science

An analysis and comparison between the three main prevalent user interfaces from an efficiency standpoint under specific cases

Subheading if used

Suzanne Zomer and Malek Alabed

Author (style Abstract Headline)

Suzanne Zomer, Malek Alabed

Title

An analysis and comparison between the three main interfaces from an efficiency standpoint under specific cases

Supervisor

Kamilla Klonowska

Examiner

Daniel Einarson

Abstract (maximum 250 words)

Abstract text.

Keywords (5–8 words)

Keyword1, keyword2, keyword3, keyword4, keyword5

Content

[1. Introduction 1](#_Toc102909960)

[1.1. Background 1](#_Toc102909961)

[1.2. Problem Statement 2](#_Toc102909962)

[1.3. Research Questions 3](#_Toc102909963)

[1.4. Limitations 3](#_Toc102909964)

[1.5. Thesis Structure 4](#_Toc102909965)

[2. Method 4](#_Toc102909966)

[2.1. Literature Review 4](#_Toc102909967)

[2.2. Experiment specifications 7](#_Toc102909968)

[2.2.1. Command Line Interface 7](#_Toc102909969)

[2.2.2. Graphical Interface 7](#_Toc102909970)

[2.2.3. Touchscreen Interface 9](#_Toc102909971)

[2.2.4. System Hardware Specifications 9](#_Toc102909972)

[1. Literature Study 10](#_Toc102909973)

[1.1. Theory 10](#_Toc102909974)

[1.1.1. Graphical Interface 10](#_Toc102909975)

[1.1.2. Command Line Interface 10](#_Toc102909976)

[1.1.3. Touchscreen Interface 10](#_Toc102909977)

[1.2. Related Work 10](#_Toc102909978)

[1.3. Results Literature Study 12](#_Toc102909979)

[2. Experiment 13](#_Toc102909980)

[2.1. Experiment Description 13](#_Toc102909981)

[2.1.1. Identify the problem – what are we trying to do? 13](#_Toc102909982)

[2.1.2. Designing a solution – How is it going to be done? 14](#_Toc102909983)

[2.1.3. Writing the program – Can we run it? 16](#_Toc102909984)

[2.1.4. Check the solution – Is it solving the problem? 16](#_Toc102909985)

[2.1.5. Wireframes 16](#_Toc102909986)

[2.1.6. Command Line Interface Development 18](#_Toc102909987)

[2.1.7. Recording Elapsed Time 19](#_Toc102909988)

[2.2. Results Experiment 20](#_Toc102909989)

[3. Results and Analysis 21](#_Toc102909990)

[3.1. Comparison 21](#_Toc102909991)

[4. Discussion 22](#_Toc102909992)

[4.1. Ethical Aspects 22](#_Toc102909993)

[4.2. Future Work 22](#_Toc102909994)

[5. Conclusion 23](#_Toc102909995)

[6. References 23](#_Toc102909996)

[7. Appendix 23](#_Toc102909997)

# Introduction

With growing supply of software, information technology is penetrating all areas of human activity and affecting people's lifestyles. Ever since the conception of personal computing, approximately four decades ago. And later, with the introduction of User interfaces concurrently with the evolution of computing, interfaces such as, Command line interface (CLI, or CLUI) [1] Graphical interface (GUI) [2] and Touch screen Interface (TSI) [3] became the predominant way to interact with a device. The stated methods will aim to further expand on and compare these interfaces under specific environments, such as ease of learnability, and efficiency of completing tasks. By applying the same program across the three interfaces, the program will be created on a webpage, an android application, and a command line application. Then Specific tools will be used to test and compare them, tools that provide information such as time spent to perform a certain task, time taken to move to the next step, Power consumption by each interface and so on. The data will be displayed through tables and heatmap figures. The defined solution aims to provide a better judgment path to each interface in aim to reconsider the approach that is currently taken by developers to funnel all consumer-based applications and software towards graphical user interface.

## Background

Early on the main and only method of communication with a device was the keyboard, with devices such as the Altair 8800 Microcomputer [5] with a Memory size of only 256-byte [6]. Briefly after, pointer devices were introduced in the form of trackballs, and mouses with devices such as The Xerox Alto [7] that provided a memory capacity of 128KB with is a 49900% increase in comparison with the Altair 8800 [5], which transfigured personal computing as we know, and introduced a new type of interface besides Command Line user interface (CLUI), which was graphical user interface (GUI) which resulted in mass adoption of personal computers due to the ease and simplicity of use.

“GUI is a means of enabling user interaction with electronic devices such as computers or hand-held devices” [8]

Later, with the introduction of the first capacitive touch screen devices aimed for consumption in 2006 through the LG Prada Phone [9], Touchscreen interface was presented permitted technology to touch even more aspects of everyday lives. Every day, consumers face a huge flow of information. They need to select the information they need and absorb it quickly. Whether we are talking about websites, mobile applications, operating systems for various devices and other ways of communicating information, in order to be true to its purpose a UI must be quick to understand and intuitive, regardless of one's education, social differences or age. Therefore, there is a tendency for developers of applications, operating systems, and other software to give up unnecessary functions, trying to provide only the most necessary information to the user. In this study we provide a deep understanding of each of the three main interfaces (Command line, Graphical, and Touchscreen) and their purpose, and we test each interfaces efficiency by developing a program that applies across all three interfaces, and testing which is more demanding on our devices, and the speed it takes for a user to complete a certain number of tasks within each interface. And based on that information we will produces a statement of which is the most productive when it comes to developing a program and using it. and if we should reconsider Command line as an interface that still holds a place in consumer-usage.

## Problem Statement

The aim is to compare the three main interfaces in terms of efficiency, learnability, complexity, and processing power usage. Graphical user interface in both of its forms is becoming the dominant type of interface expeditiously. And that has its effects on the industry and consumption behavior. With rising standards of web pages, and phone applications the bar to entry is continuously rising, and with new software technology being introduced and being set as an immediate standard such as Web Assembly [10] developing a purposeful application can be restricted to specific interfaces and will consume a large amount of time to develop. And that will be done through excessive research and practical testing.

## Research Questions

Research Questions:

**R1:** *Which of the main three types of user interfaces is superior in a larger number of applications and under specific cases compared to the others from an efficiency standpoint?*

**R2:** *What is each user interface, and under which area is each interface used and how efficient is each one, and what system problems does each interface have?*

## Limitations

In this study our focus lies to test each of the interfaces under an environment which we created. The limitations lie in the lack of resources, in other words, restriction in time, development periods, and number of programs written. Our sample pool is limited by its nature, by picking a specific program type that will be applied across all three interfaces, we are inherently blocking different types of programs to influence our results, therefore our results will be specific to that type of program, but the bases of the results can be applied generally in their essence. It is needless to say that if resources were to be abundant, multiple programmers would’ve been made to test each interface, therefore providing us with a better scope of results. Moreover, the study will be performed in a specific environment that will have an influence that’s not within our control, we will not be able to assess the influence that our operating system, hardware, computing specification, different tools, programming language versions, and machines management process of allocating resource will hold on the accuracy of our result. What can be done however is for us to specify these machine and programming language specifications in order to give the reader a better sense of understanding of our results.

## Thesis Structure

# Method

## Literature Review

The research questions were broken down into conceptual parts deriving related terms and keyword synonyms. The first part consists of the keyword “user interface” and the second part the different types of interfaces related to the thesis project. The final part is made up of keywords in search of finding useful documents in relation to user interfaces statistics or research.

**Keywords:**

* User interface
* Command Line, Menu-driven, Graphical, Touchscreen, Text-based
* Comparison, analysis, research, performance, efficiency

The search for literature was conducted with Summon@HKR, Högskolan Kristianstad’s scholarly search tool. Each search was conducted under the same specifications to provide relevant results:

**Specifications:**

1. Publication date: 5 last years
2. Content type: Journal article
3. Discipline: Computer science, Engineering
4. Language: English
5. Limit to: Peer reviewed publications

Three searches were conducted to narrow the keywords down to gain a wanted number of relevant results, around 50 to 30. Table 1 displays the combination of keywords with different searching techniques.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Keywords** | **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** |

Table 1.

**First search – 434 results**

* User interface (title)

**Second search – 97 results**

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

**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)

Each article gathered from the search results was scanned in a certain order to determine the usefulness to the project.

**Article scanning process:**

1. Title
2. Abstract
3. Conclusion
4. Introduction
5. Deeper dive

Diagram

Description automatically generated

*Figure 1.*

## Experiment specifications

The experiment consists of testing the program on three different user interfaces. The applications are developed for the command line interface, graphical interface, and touchscreen interface. The goal of the experiment is to gather further testing information on each user interface and make out the differences and similarities. The development of each application identifies specifications such as the platform, and programming language used.

### Command Line Interface

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.

### Graphical Interface

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.

### Touchscreen Interface

The touchscreen user interface is developed for Android. The choice of Android over Apple’s iOS correlates to our knowledge on the development of Android applications at our studies at Högskolan Kristianstad. In addition to the convenience of having access to an Android mobile device to test on, specifications presented in the next section *2.2.4. System Hardware Specifications*. The IDE used for development of the Android application is Android Studio, built on JetBrains' IntelliJ IDEA. The support for programming languages is Kotlin, Java and C++. However, since May 7th, 2019, the most used language for Android development remains Kotlin, and with that our choice of programming language.

### System Hardware Specifications

The case study was performed on an Asus ROG Zephyrus G14 laptop, with the following specifications:

**Device model:** GA401IH-HE072T

**Processor:** AMD Ryzen 7 4800HS

**Graphics card:** NVIDIA GeForce GTX 1650 4GB

**RAM:** 16,0 GB

**Storage:** 1TB PCIe 3.0 NVMe M.2 SSD

**Operating system:** Windows 10 Home 64-bit

The Android mobile device, Huawei P Smart, used for the touchscreen user interface, with the following specifications:

**Device model:** FIG-LX1

**Processor:** HiSilicon Kirin 659 (Octa-core)

**Graphics card:** Mali-T830 MP2

**RAM:** 3,0 GB

**Storage:** 32.0 GB

**Operating system:** Android 9.0 (Pie)

# Literature Study

## Theory

### Graphical Interface

**What is it?**

Graphical user interface (GUI) has numerous design principles that oblige to the model–view–controller software pattern, which separates internal representations from external representation of information from the manner in which information is shown to the user, and that results in a platform where users are shown which functions are possible at a given time rather than requiring the input of command codes. In other words GUI is able to strap itself to natural manipulation rather than countless command sequences that a user has to memorize. Users interact with information by controlling visual widgets, which are designed to respond in accordance with the type of data they preserve and support the actions necessary to complete the user’s intended task.

The appearance, or the “skin” of an application software or operating system may be redesigned at any point due its nature of graphical user interfaces being independent from application functions. Applications usually implement their very own unique graphical user interface that display elements in addition to graphical user interface elements that are already present on the existing operating systems. A typical graphical user interface also includes a standard of formats that concern themselves with representing graphics and text, that results in it making it possible to share data between many applications running under common graphical user interface design software.

When it comes to testing the Graphical user interface it refers to the systematic process that consists of generating test cases in hopes to evaluate the functionality of the system that’s being tested and its design elements. The testing tools for Graphical user interface, which are divided into two types, manual or automated and usually performed and implemented by third-party operators, are available under an array of protected licenses and are supported by many platforms. The most popular examples include: Tricentis Tosca, Squish GUI Tester, and Maveryx.

**History**

The GUI is an interface developed in the 1970s by the Alto research laboratory, it concerns itself mainly with human to computer interactions through an outer coating that is presented to the user. It was deployed commercially first in Apple’s Macintosh systems and then Microsoft windows. Its design was a direct response to the problem of inefficient usability in early computer usage, where text-based command-line interfaces were the main interface for the common user, a user that gets confronted with a blank screen that holds only a few prompts. It initially solved the blank screen problem that confronted the early computer users (Norman 1988). Early on the user sat down in front of a computer and was faced with a blank screen, with only a simple prompt facing them. The computer gave the user no indication on what the user should do. GUIs are an attempt to solve that screen problem. At a conceptual level, a computer human interface is a "means by which people and computers communicate with each other" (Norman 1988). A popular analogy is between a a car's steering wheel and computer system's GUI. The wheel directly connects the driver to the operation and functionality of their vehicle. When driving, a drivers focus should not be on the steering wheel. In the same sense, the GUI connects the user of the computer system to the operation and potential of the computer system (Bonsiepe 1993). A good GUI design removes the impediment and anyway hurdles of communication with the computer system and to allow the user to work directly on the problem at hand (Norman 1988). In more traditional computer science terms, the GUI is a visual operation that occurs on the monitor of the computer operator (Harding 1989). More specifically, a GUI is a specification for the look and feel of a computer system (Bonsiepe 1993). In most cases GUIs have common characteristics such as windows, icons, menus, and pointers (WIMP). In simpler terms. WIMPs are pictures that bring a certain action or an action space. The user issues a series of commands via the GUI to the computer applications. In most cases GUIs usually have three major components. These three components are (Hayes and Barab 1989): a windowing system, an imaging model, and an application program interface (API). The system builds the windows, menus, and dialog boxes that appear. The imaging model creates the fonts and graphics that appear on the screen. Later after its creation the Graphical user interfaces (GUI) would become the standard interface of user-centered design in software application programming. Providing users the capability to intuitively operate computer software, hardware and other electronic devices through the direct manipulation of graphical icons, icons such as scroll bars, buttons, menus, windows, tabs, , cursors, and the mouse pointers. Many modern graphical user interfaces have switched there accessibility to focus on touch screens

**Examples of graphical user interface:**

The GUI interface success today can be accredited to the simplicity of its input and output between the computer and user, which are work in completely different ways. But that doesn’t mean all GUI’s are good interfaces. Lets look at examples.

**Good Examples**

#1 – Android General Interface

At the turn of the century when miniaturization became a driving force, there were many attempts at functional but user-friendly small scale operating systems were attempted. The J2ME (Java platform, micro edition) system that older mobile designs still use is an example of more awkward iteration of that.

The Android General interface is able to pretty much captures the same essence of a PC operating system, and it adds a touch friendly mostly flat aesthetic that has almost no learning curve to. And line Is sometimes blurred by Laptops which support a touch screen.

#2 – Google Drive

On the cloud and web side of the operation, Google realizes how SaaS and web services should work. And that is just like traditional software, if the desktop trope is exchanged for a dashboard page from which it launches everything. Again, these interfaces are based on what the user is used to, which are the Windows or Mac operating systems, which practically diminishes the learning curve.

**Bad Examples**

#1 – Windows 8

In designing their next release of their flagship product, Microsoft managed to release a product that received its fair share of deserved backlash. In a bold attempt to be a lot more tablet friendly, Windows 8 failed to appeal to desktop users, who were more comfortable with the Start menu, the standard Desktop, and other familiar features of Windows 7. Windows 8 was held as a radical departure from its predecessor. Rather than slowly transition users to the new format they envisioned, Microsoft unleashed a slew of abrupt drastic changes all at once. The Start button has been removed, the Start menu was replaced by a mundane Start screen, an awkward Desktop mode for users, edges that weren’t intuitive, and an uneasy combination of the new Settings app and the old Control Panel.

All the users who had grown accustomed to Windows 7 interface now suddenly didn’t know where to go or what to do to accomplish their tasks in Windows 8. Which increased the learning curve of a program that relied on keeping the curve as small as possible. At that point Consumers didn’t have the patience to learn something new. And many businesses had neither the time nor the resources to train employees on a different operating system. In the end, Windows 8 was held as a failure of an interface with consumers and corporations alike. With Windows 8.1, Mi-crosoft tried to fix some of the flaws by throwing in a Start button and a more usable Desktop mode, but it was late.

**Statistics**

According to a study conducted by the blog Raymond.cc where 10 popular websites were tested for CPU Usage, they found that all of the websites' CPU usage peaked around the 70% mark.

REF: <https://www.raymond.cc/blog/battle-of-the-browsers-in-cpu-and-memory-usage/>

**Strong points, weak points**

Weak points

The main weak points which concern GUI is mostly related to the system in which the interface operates on rather than the intrinsic to the interface, for example, it doesn’t give the user a lot of flexibility when it comes to operating, as the instructions are already programmed, an extension of that issue is how system functionalities are already set and does not allow much room for customization. Another weak point is the heavy dependency on the devices storage capacity and processing power relative to other interfaces, that weak point can lead to slow loading times and longer interval inputs.

Strong points -

Graphical user interface has many strong points that outscale its weaker ones. For example, its ease of use and comprehensive inclusion, and that is delivered through appealing designs. The low learning curve for new users allows for rather complex tasks to be achieved with ease, for example, navigation through files, easy path layout, and clear set of instructions for progression or regression.

In order to provide the best user experience for users. There are many standards a programmer can follow that will influence numerous factors, such as efficiency and learnability.

### Command Line Interface

Command line interface (CLUI) or Text-Based Interface

**What is it?**

A command line interface (CLI) is a text-based user interface used to interact with the system. The components of the interface consist of a text field where the user can write command prompts, the history of the current session and the feedback from the system.

Command line interface CUI allows for direct manipulation of the software or the program by inputting texts (commands) that trigger certain actions assigned to that specific command.

**History**

Most commonly the graphical user interface is the default used interface for programs, and in most cases the CLI is used by people involved in programming or technology.

A command-line User interface (CLUI) is an user interface that utilizes texts to perform commands and run programs(UI), by allowing users to insert commands to manage computer files and interact with the computer. Command-line user interfaces are also called command-line interfaces, character user interfaces and console user interfaces. CLUIs only allow input commands that are entered by keyboard, the commands that are invoked at the command prompt are then performed by the computer. CLUI has its roots in the 1960, back when the only means of input was the standard keyboard, when only text information could be displayed. At the time CLUI had no competition as it was the only way to that was natural to communicate with the device.

The command line dropped in popularity following the introduction of GUI-based personal computer OSes like Microsoft Windows and Apple's "classic" Mac OS in the 1980s. The command line remains an important tool for IT professionals, software developers, sys admins, network administrators and many others who prefer a more precise and reproducible interface to their systems, and due to its high performance compared to the GUI in some cases, for example, the CLUI requires less resources to perform tasks, due to the nature of its input and output, also its considerably more friendly when it comes to repetitive tasks and commands, for examples when handling a large number of files within a folder CLUI allows you to use a single command to automate an operation that otherwise would be unnecessarily long and repetitive on other interfaces, also in its command and prompts its shares universal prompts and commands that users can use across software and programs that use the interface. The CLUI provides high precision by allowing the user to target specific destinations with ease. CLUI is also powerful in the sense that it gives the user full control on the system, otherwise users will be more restricted to perform certain tasks that might require manipulation with the systems core processor, that’s not the case with CLUI.

**Strong points, weak points**

Weak points -

The main weak points of CLUI revolves around users experience with the interface, for example, users with low experience on CLUI or command shell,will find the interface confusing to use at first as it requires knowledge of commands that trigger certain actions, therefor having a large learning curve for new users.. A weak point that concerns both user types (expert, and novice) is case sensitivity, error handling, and accuracy of spelling; one spelling error can cause the command line to fail, with no indication of where the error exactly occurred.

Strong points -

The Strong points of command line definitely out weight the weak points, that can only happen when the learning curve is overcame, regardless of user inputs, command line still has strong points that revolve around the interface itself, for example, command line interface by nature, requires less memory space to operate in comparison of the other interfaces, and a result of that is fast performance, low process requirement, and low resolution monitors requirement. The strong points that can come with user experience of the command line interface are vast as well. For example, a knowledgeable user can complete tasks with more efficiency, especially repetitive tasks.

### Touchscreen Interface

Touch screen interface (TSI) is present on devices that require the existence of physical contact to generate responses.

**What is it?**

Touchscreen interface is a combined display and input device. With the use of touching the screen for input and displaying on the screen for output, the touchscreen interface provides a modern take on human computer interaction. It is not necessary to provide additional interactive components, for example buttons or a physical keyboard, to function.

The anatomy of touchscreen devices can be split into four parts.

1 - Front Bezel: the bezel is the shell of the device, which is usually made of metal in order to protect the other parts.

2 - Touch sensor: The sensor is a protective glass panel that is touch sensitive, which allows for pressure from finger tips to influence what's behind the panel. The sensor is placed in a position which allows coverage over the entire screen.

3 - Touch Controller: A small microcontroller is placed between the touch sensor and system controller. The controllers role is take in the information from the touch sensor which is sent as pressure from finger tips and translates it as information that the device is able to take in

4 - System Software: The system software is heavily dependent on the type of device, but its role in all of the devices is common, which is to work between the sensor and the controller to work together in order to interpret the information coming in and out of the device.

**History**

The first proposal of touchscreen interaction was made in 1965 by Eric Arthur Johnson. The interest in developing the first finger driven device was to control air traffic. Subsequently the first touchscreen interface was put into practise in 1976 and became commercially produced in 1983. Since then, touchscreens became increasingly more popular and greatly improved over time. With the release of the iPhone in 2007 by Apple, a new age for touchscreens began. Competing electronic companies started releasing their own mobile phones with a focus point of containing a touchscreen. The need for these devices expanded rapidly, resulting in a variety of improvements and new technologies over the years.

One of the reasons touch screens gained popularity so quickly is because people learn how to interact and touch starting as young as being a baby. It is a natural action and movement to point, pinch, scroll and draw with the use of our fingers. Touchscreens can already be used by young children to explore, play and learn.

Mobile devices are commonly used for single actions. It is more complicated to follow a large number of steps on these devices. Usually personal computers are preferred over mobile devices for actions related to writing papers, accounting, video or 3D processing, and high storage capacity. Some actions need computationally intensive resources and extended use of those resources or a keyboard and mouse to complete more extensive and complicated actions. Touchscreens commonly transpire in smaller formats on mobile devices which makes programs with a great quantity of functions difficult to convey. However, touchscreens provide an effortless interaction with the user that can not be given by non touch devices.

**Statistics**

According to a statistic that was performed by the StatsCounter GlobalStats [1] in April of 2022, shows that Mobile phones have a larger market than Desktop devices, Mobile phones have a market share of 58.03%, while desktop devices have a market share of 39.5%, the remaining 2.46% percent is for tablet devices [2], which use touch screen interface as well! So from that statistic we can see that touch screen interface devices are more present in our society by a large 10%.

**Strong points, weak points**

Weak points-

However just like any other interface the touch screen medium comes with its own advantages and disadvantages, such as the small screen size comparably to a desktop device, the size of the screen can limit the number of icons or information displayed to a user which might require them to perform an extra input to get the same result as a desktop user would, the size of screen allows for the users hand to obscure the screen easier. Also due to the size of the device that has the touch screen interface, the processing power of said devices will be limited therefore causing it to underperform when it comes to executing certain demanding tasks, due to the size of it, which enables it to be more portable. The size as well affects the amount of information that can be inputted at once, so inputting a large amount of data is not desirable on touchscreen devices. numerous natural elements can come into place which might hinder the user experience and efficiency, for example dust particles that might shroud the screen resulting in lower visibility scenarios for the user and might cause touch inputs to not register or delay them, or strong sunlight that can make screen visibility difficult. Another point is the lack of haptic feedback on touch screen interface, mostly web-based application, on a desktop device a user will get haptic feedback in the form of mouse or keyboard clicks, which triggers the sense of progression without visual progression ques, such as window changes, so the user can depend on two forms of feedback when communicating on a desktop graphical user interface, however on the touchscreen side those ques are limited to only visual ques which slows down the overall efficiency of a user, the user will have to perform a touch gesture and must wait on a visual cue that confirm the action. Touch screen devices have almost no real-world application usage, meaning no profession relies on specifically touch screen devices to perform their goal, which puts touch screen devices in an awkward place.

Positive-

On the other hand, touch screen devices have almost no learning curve in their interaction, a completely inexperienced user can pick up on how to operate touch screen devices almost instantly, Using a touch screen interface requires little thinking as it is a form of direct manipulation, that directly affect the way applications display and take information, meaning its always simple and user friendly. The use of touch screen interfaces or devices, virtually eliminates any type of user error, meaning due to its nature the information that is displayed to them is only the information that is available, users observe the information from clearly defined menu, unlike desktop devices with a graphical user interface, the user is not restrained to their interaction therefore the margin for error is high. various touch screen interfaces enable users to perform what is called “Multi-touch” functions and inputs which are more intuitive such as zooming in or out

**Examples**

Touch screen interface is mainly present in mobile devices, but it does exist in other hardware such as laptops although not as the predominant language of communication between the device and user. Real life examples where a touch screen interface is the sole way of communication that are not smartphones do exist. For example in modern vehicles a touch screen interface is present and preferable, due its nature of allowing the user to perform actions intuitively with their finger, and that is especially important for a user that has another main task at hand, in this case controlling a vehicle. Another example that requires the existence of a touch screen interface to fully deliver its purpose is drawing, tablets that only serve the purpose of drawing do exist, they mimic the use of finger touch by using a pen, which allows for more precise movements and control.

## Related Work

Finding related work that directly tackles the essence of our topic has proven to be difficult, due to our topic not only covering a big scope by focusing on the three main interfaces, but due to the fact that we are observing very specific factors within that scope. The related work that has been found cover branches of our topic. therefor we will gather them here, and draw conclusions from each one in order to draw a larger conclusion.

*“Comparing Text-based and Graphic User Interfaces for Novice and Expert Users”* [1] is an article that was published in the AMIA Annu Symp Proc [1]. In this article a comparison is made between text based (CLI) and graphical (GUI) in a medical environment. The test depends on two types of users, an expert user and a novice user, that were evaluated in regards of time required to perform certain tasks. Subsequent to setting guidelines in which both types of users will operate such as, a specific sequence of steps in which both users partake in, The following criteria were compared whether the experience on the interface can have an effect on efficiency of completing the tasks. The paper concluded with a conclusion that states *“GUI was not necessarily better than TUI for an expert; rather they were better for novice users”* [1]. Two flaws that was observed in the mentioned work is that the test was done only once for each user type. And only one person of each user type partook in the test. These two flaws are avoided in our paper, due to the facts that each test was held three times, and we had multiple users for each user type participate.

Another factor in the paper that we considered with caution is their graphical interface. As it is doesn’t represent the graphical interface we are using fully, as we are using web interface. Keeping that in mind, the interfaces the paper used still hold the key elements that makes it graphical! Elements that both our and their interface share. Elements such as direct manipulation of the interface, the use of buttons, symbols, visual metaphors, pointers, and requirement for text input set to a minimum. The article also touched on problems they faced with Text -based interface, and the main one being lack of information for novice users, which lead to a lot of time being wasted trying to understand the current state of the program.

Graphical user interface

Description automatically generated

*Figure 1.*

The second related work that proved to be extremely relevant, is a thesis published by Borislav Lazarov and Rafael Zerega through Södertörn University [3] Titled “To Touch or not to Touch” [4] The thesis compares traditional interface (GUI) and Touch screen interface (TSI), the two main criteria’s of testing in this paper are, time taken to complete for every task, and errors made in that procedure. The test that was held consisted of participants inputting six strings of 6, 19, and 44 characters. That was done in both interfaces. Each task was done three consecutive times to get better results. The paper concluded by stating that touch screen interface was proved to be slower with a higher rate of error than graphical interface, due to the finding that it took users 1.4 times longer using touch screen than graphical interfaces (by the use of a physical keyboard), and the average rate of error made was 1.4 on graphical interface (by the use of a physical keyboard) and 1.8 for touch screen. The flaw in this paper, is the nature of their test, as it was mainly focused on the means on input that concern that two interfaces, and the small scope of their test, by only requiring inputting strings, a lot of the elements of both interfaces are not put to use. The test was repeated multiple times, and multiple participants were involved, which resulted in concrete results. The thesis as well touches on problems that are inherent to each interface, for example for touch screen interface, one of the problems mentioned was the users fingertips covering small objects causing the user to touch on areas by mistake, other problems that were discovered after holding interviews with the participants are, screen problems, for example screen sensitivity, and selection problems, due to friction with the fingertips.

## Results Literature Study

With two main related works in mind, we decided to incorporate elements of those studies that were more aligned with our problem statement, for example we incorporated the element of having two user types, expert, and novice, from the *“Comparing Text-based and Graphic User Interfaces for Novice and Expert Users”* [1] article, as that approach allows our paper to be more applicable towards readers, as they can, based on their own knowledge, derive whether they consider themselves to be a part of a expert or novice group. Another element that effected our practical part is making sure that both user types perform the exact same tasks, and that was done by breaking down a number of steps with instructions that both user types can follow, that way there will no inconsistency in testing. The second related work mentioned, the Titled *“To Touch or not to Touch”* [4] had some elements that we incorporated that the first article didn’t. for example, each test on each interface for each user was repeated three consecutive times, and three participants of each user type participated in the test, we decided to implement those elements into our paper as it will provide us with more solid results.

Keeping in mind the two research questions, through literature review we can confidently answer both questions, The first question: ***“Which of the main three types of user interfaces is superior in a larger number of applications and under specific cases compared to the others from an efficiency standpoint?”*** from the first related work paper, we can observe from their result that graphical is far more superior than command line in efficiency, as that was the case for novice users, which most people are, and from the second related work paper, we can observe that graphical was as well superior than touch screen interface, in many areas, one of which is efficiency.

The second research question states: ***“What is each user interface, and under which area is each interface used and how efficient is each one, and what system problems does each interface have?”*** as stated in the related work section each of two papers share the same definition of each interface with our paper, as that information is common knowledge, and no disputes are held on that. Problems that were found in each of the interfaces were common as well. Problems such as, touching the wrong area in the touch screen interface, and the difficult time novice users find in command line, or the minimal struggle of expert users in graphical comparatively with command line.

# Experiment

## Experiment Description

Each user interface is to be developed withing the same boundaries and steps to maintain an equal testing environment. With this comes the responsibility of making a fair comparison, and so the details of the program that each interface had to follow are presented in the section below.

### Identify the problem – what are we 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.

**Results:**

* 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. **ADD** product to cart
7. **DISPLAY** confirmation
8. **DISPLAY** cart
9. **READ** full name and address
10. **DISPLAY** confirmation
11. **END**

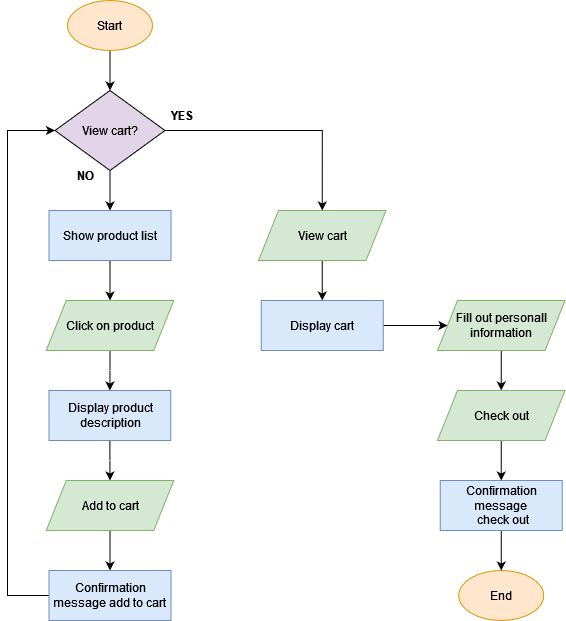


Figure 2.

### Writing the program – 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.

### 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.

### Wireframes

The development for the program in relation to the graphical and touchscreen 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.

**Graphical Interface:** *Figure 1.2* displays the three frames and the correlations 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 divider, 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.

Diagram

Description automatically generated

*Figure 4.*

**Touchscreen Interface:** The wireframes, *Figure 2*, are similar to the frames created for the graphical user interface for the web application. The biggest differences are the positioning of components and the amount of screen space each component requires. The homepage of the android application displays all the product listings with each containing: a picture of the product, the product name, and part of the description followed up with dots to indicate there is more to read. Because the screen size is limited each of these components relatively takes up more space and less products are shown at the same time. To view more products the user can scroll down until the end of the list. The functionality of the application remains as close to the application for the web as possible, with the same clickable items and number of activities/pages.

Shape

Description automatically generated

Figure 5.

### Command Line Interface Development

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.

As seen in *Figure 1*, each step in the program maintains a clear consistent structure by providing visual differences between: current menu title, command options, information listings, and user input field. The menus are separated with the use of uppercase letters and the use of dashes, the command options are displayed with most possible clear instructions, the products listings are provided with the use of a single dash character, and the user input field is displayed with the ‘>’ symbol that is commonly seen in command line interfaces.

A picture containing text

Description automatically generated

*Figure 2.*

### Recording Elapsed Time

During every test taken by the test subjects, the elapsed time is recorded with each step taken. The application stores the current time in a variable at the initiation of the step and when the end point of the step is reached, the new current time is stored in another variable. The program continues to calculate the difference between start and end point and the value is stored in an array with proper naming. At the end of the test run, the elapsed time for individual steps is displayed to the user. The method for recording the elapsed time difference on each user interface because of language used for example, however, follows the same principle.

The command line application has a unique function, which is the use of a keyboard to enter commands. The time between each menu or page is recorded, as described earlier, however, there is an extra step recorded with each user interaction. This is the recorded time elapsed for each user input. Before the user initiates inputting the command, the program waits for a press on the Enter key, and afterwards the command can be written out. An additional time recording takes place from the moment the Enter key is pressed, till the user input for the command is being received. See *Figure 1*, the program writes out “Press Enter to write...” continues by “>” symbol to clarify the expected user input. The additional recorded time is added to the array.

## Results Experiment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage name** | **Time (Seconds)** | | | | | |
| **Command line interface** | | **Graphical**  **interface** | | **Touchscreen interface** | |
| **Expert user** | **Novice user** | **Expert user** | **Novice user** | **Expert user** | **Novice user** |
| first steps | 4.476 | 15.765 | 10.078 | 14.563 | 14.11 | 18.455 |
| viewing the products | 30.439 | 71.836 | 39.546 | 45.381 | 42.6 | 57.648 |
| adding the products to cart | 36.059 | 43.784 | 39.684 | 39.924 | 45.628 | 51.712 |
| checking out | 13.515 | 38.755 | 8.233 | 10.969 | 12.87 | 19.622 |
| **Total time:** | **84.489** | **170.14** | **97.541** | **110.837** | **115.208** | **147.437** |

Table 2.

|  |  |  |
| --- | --- | --- |
| **Step #** | **Step description** | **Cognitive distribution** |
| Step 1 | Observing the program | Internal |
| Step 2 | Inputting “products” | External |
| Step 3 | Read menu | Internal |
| Step 4 | Inputting product name | External |
| Step 5 | Read product description | Internal |
| Step 6 | Inputting “return” | External |
| Step 7 | Inputting “add” | External |
| Step 8 | Inputting “view cart” | External |
| Step 9 | View cart | Internal |
| Step 10 | Inputting “check out” | External |
| Step 11 | Input name | External |
| Step 12 | Input address | External |
| Step 13 | Input confirmation | External |

Table 3.

# Results and Analysis

## Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| **Stage name** | **Time difference between Expert and Novice user (Seconds)** | | |
| **Command line interface** | **Graphical**  **interface** | **Touchscreen interface** |
| first steps | 11.289 | 4.485 | 4.345 |
| viewing the products | 41.397 | 5.835 | 15.048 |
| adding the products to cart | 7.725 | 0.24 | 6.084 |
| checking out | 25.24 | 2.736 | 6.752 |
| **Total time difference:** | **85.651** | **13.296** | **32.229** |

*Table 4.*

Chart, line chart

Description automatically generated

*Figure 1.*

# Discussion

## Ethical Aspects

The purpose of this study was to provide a result on the comparison of the three main interface types through an efficiency standpoint. The aspect of this paper that is of concern with ethical elements are, the programmers developed by us, and the environment where the testing was done. The 3 programmers that were developed by us have all been uploaded on GitHub and available for public download. All testing was done on the same device to ensure equal grounds between all testing matters and subjects.

## Future Work

Future work on this study can help expand its scope and lead it to be more applicable across different fields, the points that can be enhanced on for future works are the following:

* The expansion of testing different applications, towards more specific ones. Our study focused on a single type of application; it would be more desirable to create more applications that are field-specific, for example, dentist checking in devices, transportation ticket machines, messaging applications, and public access applications, such as libraries.
* Following up on the previous point, when testing different application types, new interface types will be tested on, interfaces that are derived from existing ones, and new interfaces like menu driven interfaces.
* Testing elements such as participants and devices, that will allow for more concrete and robust data.
* Modification to our application for it to be more complex to allow for more real-world relatability.

# Conclusion

# References

# Appendix

Table

Description automatically generated

Figure 4.

|  |  |  |
| --- | --- | --- |
| **Command Line Interface** | | |
| **Step #** | **Average Time (Seconds)** | |
| **Expert** | **Novice** |
| First steps: | | |
| Step 1 | 2.243 | 6.453 |
| Step 2 | 2.233 | 9.312 |
| Viewing products: | | |
| Step 3 | 2.233 | 5.780 |
| Step 4 | 1.577 | 1.126 |
| Step 5 | 4.988 | 16.084 |
| Step 6 | 4.155 | 7.462 |
| Step 3 (repeating) | 1.697 | 1.493 |
| Step 4 (repeating) | 1.031 | 3.634 |
| Step 5 (repeating) | 4.882 | 12.743 |
| Step 6 (repeating) | 1.552 | 5.737 |
| Step 3 (repeating) | 0.936 | 3.895 |
| Step 4 (repeating) | 0.981 | 2.326 |
| Step 5 (repeating) | 5.534 | 9.531 |
| Step 6 (repeating) | 0.873 | 2.025 |
| Adding products to cart: | | |
| Step 3 (repeating) | 0.842 | 1.059 |
| Step 4 (repeating) | 1.557 | 1.286 |
| Step 5 (repeating) | 6.062 | 6.085 |
| Step 7 | 1.979 | 2.551 |
| Step 3 (repeating) | 3.143 | 4.439 |
| Step 4 (repeating) | 2.016 | 2.637 |
| Step 5 (repeating) | 6.579 | 9.704 |
| Step 7 (repeating) | 1.154 | 1.649 |
| Step 3 (repeating) | 4.119 | 3.865 |
| Step 4 (repeating) | 1.791 | 1.606 |
| Step 5 (repeating) | 5.102 | 6.599 |
| Step 7 (repeating) | 1.715 | 2.304 |
| Checking out: | | |
| Step 3 (repeating) | 1.069 | 4.261 |
| Step 8 | 1.159 | 7.547 |
| Step 9 | 2.023 | 5.974 |
| Step 10 | 1.051 | 8.375 |
| Step 11 | 3.805 | 3.281 |
| Step 12 | 3.094 | 3.498 |
| Step 13 | 1.314 | 5.819 |
| **Total** | **84.489** | **170.14** |

Table 5.

|  |  |  |
| --- | --- | --- |
| **Command Line Interface Errors** | | |
| **Step #** | **User Type** | **Error Type** |
| Step 6 | Expert | Misspelled command |
| Step 2 | Novice | Wrong command |
| Step 2 | Novice | Misspelled command |
| Step 5 | Novice | Misspelled command |
| Step 6 | Novice | Misspelled command |
| Step 8 | Novice | Wrong command |
| Step 10 | Novice | Wrong command |
| Step 10 | Novice | Misspelled command |
| Step 13 | Novice | Wrong command |

Table 6.

|  |  |  |
| --- | --- | --- |
| **Graphical User Interface** | | |
| **Step #** | **Average Time (Seconds)** | |
| **Expert** | **Novice** |
| First steps: | | |
| Step 1 | 1.199 | 2.331 |
| Step 3 | 8.879 | 12.232 |
| Viewing products: | | |
| Step 5 | 9.388 | 10.401 |
| Step 3 (repeating) | 5.630 | 5.731 |
| Step 5 (repeating) | 8.729 | 10.145 |
| Step 3 (repeating) | 5.972 | 9.071 |
| Step 5 (repeating) | 9.827 | 10.033 |
| Adding products to cart: | | |
| Step 3 (repeating) | 3.599 | 3.178 |
| Step 5 (repeating) | 8.627 | 9.866 |
| Step 7 | 3.702 | 3.763 |
| Step 3 (repeating) | 2.756 | 2.562 |
| Step 5 (repeating) | 6.344 | 6.373 |
| Step 7 (repeating) | 2.037 | 2.742 |
| Step 3 (repeating) | 3.805 | 2.216 |
| Step 5 (repeating) | 6.356 | 6.497 |
| Step 7 (repeating) | 2.458 | 2.727 |
| Checking out: | | |
| Step 9 | 3.319 | 3.235 |
| Step 11 | 1.070 | 2.327 |
| Step 12 | 3.331 | 4.435 |
| Step 13 | 0.513 | 0.972 |
| **Total** | **97.541** | **110.837** |

Table 7.

|  |  |  |
| --- | --- | --- |
| **Touchscreen User Interface** | | |
| **Step #** | **Average Time (Seconds)** | |
| **Expert** | **Novice** |
| First steps | | |
| Step 1 | 1.798 | 3.496 |
| Step 3 | 12.312 | 14.959 |
| Viewing | | |
| Step 5 | 10.983 | 11.603 |
| Step 3 (repeating) | 7.014 | 10.028 |
| Step 5 (repeating) | 9.863 | 14.181 |
| Step 3 (repeating) | 6.317 | 9.587 |
| Step 5 (repeating) | 8.423 | 12.249 |
| Adding products to cart | | |
| Step 3 (repeating) | 4.339 | 5.219 |
| Step 5 (repeating) | 8.465 | 9.067 |
| Step 7 | 4.159 | 5.762 |
| Step 3 (repeating) | 3.517 | 4.380 |
| Step 5 (repeating) | 7.335 | 7.529 |
| Step 7 (repeating) | 3.868 | 4.665 |
| Step 3 (repeating) | 4.116 | 5.617 |
| Step 5 (repeating) | 6.670 | 6.392 |
| Step 7 (repeating) | 3.159 | 3.081 |
| Checking out | | |
| Step 9 | 5.079 | 6.794 |
| Step 11 | 2.089 | 4.794 |
| Step 12 | 4.601 | 6.282 |
| Step 13 | 1.101 | 1.752 |
| **Total** | **115.208** | **146.437** |

Table 8.