**MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, (MUMBAI)**

**A**
**Project Report**
**on**

# " Nitro Studio IDE "

***Submitted by:***


| Sr. No. | Name of Student | Exam Seat No. |
|---|---|---|
| 1) | Aditya More (Leader) | : 202709 |
| 2) | Rushabh Kumat | : 202705 |


***Under the Guidance of:***
**Name of Guide: Mrs N.S.Gite**


**Department of Information Technology**
**K. K. WAGH POLYTECHNIC, NASHIK**


**Academic Year 2021-22**

Karmaveer Kakasaheb Wagh Education Society's

# KARMAVEER KAKASAHEB WAGH POLYTECHNIC

Hirabai Haridas Vidyanagari, Amrutdham, Panchavati, Nashik-3

## Certificate

This is certify that Mr./Ms. **Aditya Purushottam More** From **K. K. Wagh Polytechnic, Nashik** having Enrolment No. **1900780094** has completed Capstone Project Planning Report of the Fifth Semester having project title **Nitro Studio IDE** in a group consisting of 2 candidates under the guidance of the faculty guide.

*(Name & Sign of Internal Guide)*
Project Guide

Ms. M. S. Karande
HOD, Information Technology

# INDEX


| Sr. No. | Name of Topic | Page no. |
|---|---|---|
| 1 | Abstract | |
| 2 | Introduction | 1 |
| 3 | Literature Survey | 6 |
| 4 | Proposed Methodology | 16 |
| 5 | Software Requirement | 17 |
| 6 | Project Related Diagrams | 18 |
| 7 | References | 19 |

# List of Figures


| Sr. No. | Name of Figure | Page no. |
|---|---|---|
| 2.1 | Types of software developed by the developers | 7 |
| 2.2 | Company Size | 8 |
| 2.3 | Average rating of importance of qualities | 10 |
| 2.4 | Overall satisfaction with IDE | 11 |
| 2.5 | Average rating of satisfaction with IDE qualities | 12 |
| 2.6 | Limitation of choice effect Vs overall satisfaction | 13 |
| 2.7 | Comparing importance and satisfaction | 14 |
| 6.1 | Flow Diagram | 18 |
| 6.2 | Use Case Diagram | 19 |
| 6.3 | ER Diagram | 20 |


# List of Table


| Sr. No. | Name of Table | Page no. |
|---|---|---|
| 2.1 | Company size & IDE used | 8 |
| 2.2 | Company size and who made the choice | 9 |

# Abstract

IDEs (Integrated Development Environments) support software developers in their implementation work. However, embedded software has specific requirements, so an off-the-shelf IDE for this purpose does not exist. In such a case, this paper recommends developing a customized IDE based on open source software. We present a case study of developing such an IDE for the languages C#, Python and JavaScript. We used several open source projects with varying project status as a basis for our development. We analyzed developer communication within these open source projects and identified the benefits and the potential pitfalls for the case study. Moreover, we present the effort made in terms of person months and that reuse of open source software improves cost-efficiency for the development of such IDEs.

# Chapter 1 Introduction

This Project will develop for developers, now days developers are using various IDE like Vs code, Subline text but problems is that to set workspace for creating project is very complex. Configure for project any programming language & link compiler is also complex process & one more problem in colleges/schools Turbo C uses for C/C++ development. Turbo C is very old IDE which discontinued 20 year ago. If user want write same turbo c code on modern ide that not possible. Because of C/C++ is updated but for learning language how condition users use Turbo c. Configure for project any language & link compiler is also complex process & one more problem in colleges/schools Turbo C uses for C/C++ development. Turbo C is very old IDE which discontinued 20 year ago. If user want write same turbo c code on modern ide that not possible. So, solve this all issues we are developing Nitro Studio IDE. Nitro IDE will be future of development. this having Modern UI, Light Weight, Plug & Play. IDE having two modes 1st Nitro Mode which design for professionals, for big scale projects. This mode having features of Nitro Teams for build with your team, Virtual assistant for help. 2nd mode Turbo mode for new Learners, code Intelligence will be disabled by default because if you really want to learn any language you should don't use code suggestions. You can enable it in settings. This ide will be most light weight ide. This will be one of the Best IDE of World.

## 1.1 Aim

Now days users using another IDE. Turbo C is very old IDE which discontinued 20 year ago. If user want write same turbo c code on modern ide that not possible. So, solve this all issues we are developing Nitro Studio IDE. To create a user-friendly IDE for Turbo C users & other IDE users with enhanced features & benefits for developers in ide. Create world & India's own all in one IDE

## 1.2 What is IDE?

It is a software application that defines the visual representation of the location of the files easily and makes it more understandable for the user. It contains development tools such as text editors, code libraries, compilers, and test platforms and consists of at least build automation tools and a debugger. Net Beans and Eclipse are good examples of IDE, which contains a compiler, interpreter, or both; other IDE's such as Sharp Develop and Lazarus do not include these tools. IDE's have the capability of using the functionality of multiple programming processes in a single process. Some IDE's will work on a specific programming language, and also, they contain cross-language capabilities. IDE's such as Eclipse, Active State Komodo, IntelliJ IDEA, My Eclipse, Oracle JDeveloper, Net Beans, Codenvy and Microsoft Visual Studio supports multiple languages

## 1.3 What we can do with IDE

With IDE, you can develop applications such as standalone or dynamic web applications, etc. IDE includes a code editor, a compiler or interpreter, and a debugger to access the graphical user interface and allow the user to write and edit the code editor's source code.

IDE can develop software applications by using a set of tools, which makes it easier to write programs. The main objective of using IDE is that it allows coding quickly and efficiently. IDE includes built-in compilers, which convert the program into machine-level code or byte code and saves a lot of time. You can also select multiple programming languages of your choice. Some IDE's include server like Net Beans or GlassFish server to test the web applications easily.

IDE's have some common features as listed below:

- Text editor: It provides a text editor to write and manage source code.
- Debugger: It uses debugging tools to identify the errors in the source code.
- Compiler
- Code Completion
- Programming language support
- Integration and use of plug-ins

## 1.4 Why do we need IDE?

It provides inclusive facilities to a programmer for the development of software. Below are some points which describe why we should use IDE in application development: It has the ability to debug your program and compile your code. It makes it easy to see a visual representation of the location of program files. They provide support for external plug-ins, and you can use them by providing interfaces to external tools like debugging tools. They provide a console to see the execution result, and if you find any errors, you can easily debug the errors and fix them. In the C++ example, you can stop the program and check the value of variables. You can set breakpoints to pause the program when it reaches a certain line of code. IDE uses Graphical User Interface to use the graphical interface of IDE to enter the required specifications of the application.

## 1.5 Advantages

- IDE's can be used to create software applications, drivers and utilities.
- It allows developing software in any programming language without spending much time on language syntax.
- IDE has the ability to correct syntaxes, gives a warning about memory leaks, assist in writing quality of code, etc.
- It has increased efficiency, where you can code faster with less effort, and its features help organize resources, prevent mistakes, and provide shortcuts.
- It supports collaboration, in which a group of programmers can easily work together within an IDE.
- It provides program resources that are easy.
- When creating applications, IDE manages resources such as library files, header files, etc., at specified locations.
- This includes pre-installed libraries for a specific programming language.
- That makes development easier by using syntax highlight features.
- It makes the creation of database applications easily.

- IDE's can able to translate code from high-level languages to the object code of the targeted platform at the compile or build stage.
- It helps to keep track of the code, generates code and allows searching.

## 1.6 Background

In this chapter, we will present some of the key concepts that will be discussed in this thesis. The concepts are explained briefly to help understanding the thesis work and the motivation behind it.

### 1.6.1 User experience

UX is a term that is meant to broadly describe everything that the user is going to experience with a software system, how it feels to use as well as how practical and functional it is. One of the first to introduce the term was Donald Norman in the 90's and it has since become widely adopted by both researchers and practitioners. However, despite UX being widely addressed there is a lack of consensus on a precise definition and what it does and does not cover What is generally agreed is that "UX is dynamic, context-dependent, and subjective. ". 4 There exists a model for describing UX called the hedonic-pragmatic model. This model splits the UX of a product into two parts: hedonic and pragmatic attributes. Pragmatic attributes refer to the perceived usability of the product, how much the product can do and how easy it is to perform tasks. Hedonic attributes refer to the beauty of the product, how the product looks and how stimulating it is to use. The hedonic-pragmatic model will be used in this thesis to discuss the qualities of IDEs and the developers view on them.

### 1.6.2 Developer experience

Previous research on UX has mainly focused on consumers. Developers are users of software themselves and as such, the previous research on UX should also apply to developer experience. However, developer experience is different from UX since it covers more than just the "use" of software. Developer experience is a relatively new concept and refers to how the developer perceives and feel about her own activities in the development environment, as defined by Fagerholm and Münch: "a means for capturing how developers think and feel about their activities within their working environments". With UX the perceived end goal is to use an application or service, while developers perceived end goal is to create an application or service. The concept of

developer experience was introduced to better understand developers' perceptions and feelings with the assumption that an improved developer experience would have a positive effect on the development itself. We will use the concept of developer experience to elaborate on the outcomes of our study.

### 1.6.3 Motivation in software development

There has been an increased interest in studying motivation among software developers over the last couple of years. In software development, human resources often constitute the biggest expenditure. Since human resources are such a big expense, companies expect efficiency from them. This makes human motivation a very important factor in the software industry, apart from employees' wellness, for productivity and software development success rate. However, one of the problems with human motivation is that it is hard to quantify. Furthermore, despite an increase in number of studies on this specific subject, overall understanding of what motivates software developers has not changed much in the latest years. Further studies are needed to more clearly define the relation between motivation and outcome, and this thesis is among them.

## 1.7 Excepted Outcome

Turbo C code will support in new gen Nitro IDE. Users can use chatbot at the time developing program. Chat bot will help developers to get information & syntax of any language

# Chapter 2 Literature Survey

In this chapter, we describe the process of preparing the questionnaire and the tool used. Then, we describe how we designed the survey. The topic and questions to be answered by the survey were formulated based on the literature and discussions with the this is supervisor. After formulating a set of suitable questions, a preliminary questionnaire was prepared and sent to an expert who provided feedback in terms of missing IDE qualities and needed clarifications. After improving the questionnaire according to the feedback, a pilot survey was run with three of the participants. The feedback from the pilot survey suggested some improvements in the layout. The survey was performed online using Google Forms, a free survey tool provided by Google. The reasoning behind using Google Forms is that it is easy to setup, the questionnaire is accessible online and participants can reply directly through the web anonymously. Questions can easily be added or modified and the results are clearly presented and summarized. The questionnaire is composed of 27 close-ended questions and 6 open-ended questions. We chose to mainly use close-ended question in the questionnaire for two reasons. Firstly, close ended questions are more focused and in general easier to answer, making it more likely for the question to be answered. Secondly, close-ended questions are simpler to analyze and synthesize with little room for misinterpretation. The questionnaire begins with a brief explanation about the survey study and its purpose. The participant is also informed about the ethical and societal considerations, asserting that we take the responsibility for all gathered data and that no information that can be linked back to the participants is disclosed. The questionnaire asks a set of questions about the demographics of the participant such as: years of work experience in software development, the type of development work she usually performs, the size of the company she work for, her primary choice of IDE as well as any additional IDEs she uses. The open-ended question concerning the type of development work she usually performs is formulated as “What type of software products do you develop (e.g. web application, control software, etc.)?”. The questionnaire also asks whether the choice of IDE was made by the participant or her employer, using the close-ended question “Was the choice of IDE yours or your employer's?”. The rationality for the questions in this part is to gain a better understanding of the participant and her background the questionnaire focuses on the participant’s preferences regarding IDE qualities through questions about the importance of qualities such as

reliability, ease of use, versatility, stability and customizability. For close-ended questions, we use a five-point Likert scale for rating their preference In addition to the rating, the participant is also asked about her rationale behind the choice of IDE, which is formulated as the open-ended question "Describe shortly the reasons for

## 2.1 Results:

The participants of this survey are twenty software developers with at least one year of work experience in software development projects. Eleven of the participants have an experience of over seven years, while the remaining participants have on average four years of experience working professionally with software development. This means that we can assume that participants in this study have a great deal of experience working with IDEs in a professional environment. By having experienced users of IDEs as participants, we can be reasonably certain that the provided answers are not affected by inexperienced users. The type of development work done by the participants varied, with eight of the participants (about 37%) working with software applications, six (about 27%) working on software systems and four (about 18%) focusing on software tools. The remaining four (about 18 %) did not specify the nature of their work, as seen in Figure 1.

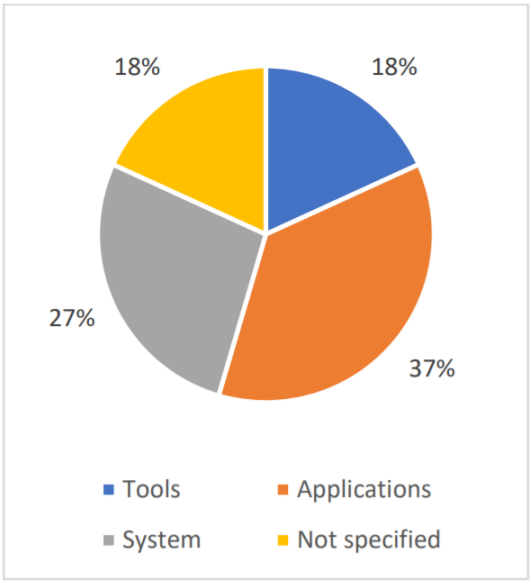


**Fig 1. Types of software developed by the developers**

Regarding the size of the company the participants work at, twelve of the participants (about 60%) work in a small company (1-99 employees), five (about 25%) in a medium sized company (100-1000 employees), and three (about 15%) in a large company (1000+ employees).

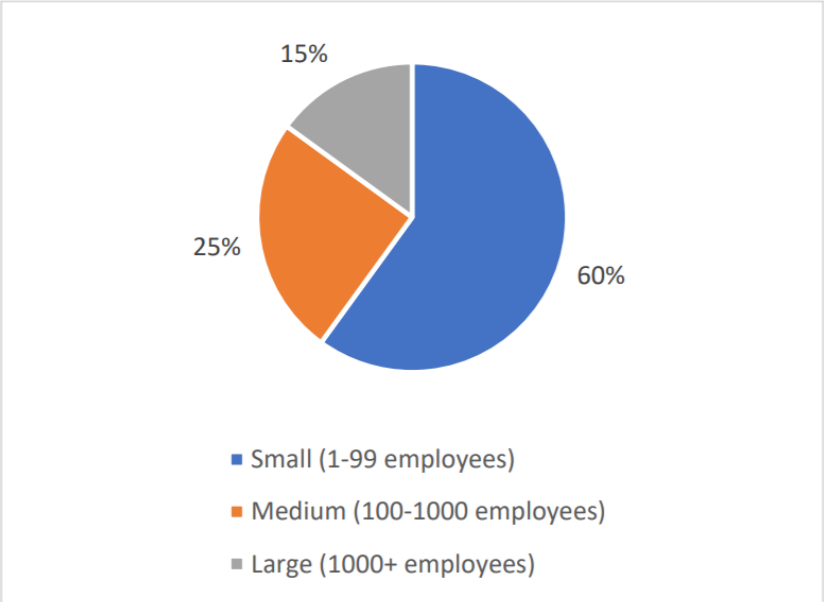


**Fig 2. Company Size**

Regarding the used and rated IDE, it is interesting to notice that only two primary IDEs were mentioned: Eclipse and Microsoft Visual Studio. Table 1 shows the result provided by participants. The amount of Eclipse users (14 of 20) were more than double the amount of Microsoft Visual Studio users (6 of 20).

In addition to the participants primary IDE, twelve participants also mentioned using other IDEs such as: IntelliJ, XCode, BBedit, PyCharm, Visual Studio Code, Rubus-ICE, Notepad++ and Atom.

| IDE | Small | Medium | Large |
|---|---|---|---|
| Eclipse | 12 | 1 | 1 |
| Visual Studio | 0 | 4 | 2 |

**Table 1. Company size & IDE used**

Every respondent that worked in a small company primarily used Eclipse (see Table 1). When asked about whether the choice of IDE had been made by the participant or the employer, thirteen participants (about 65%) answered that they had chosen their own IDE (see Table 2).

| Choice of IDE | Small | Medium | Large |
|---|---|---|---|
| Developer | 7 | 3 | 3 |
| Employer | 5 | 2 | 0 |

**Table 2. Company size and who made the choice**

## 2.2 IDE qualities

"Reliability" refers to the ability of an IDE to function correctly under an extended time without losing or altering files. "Ease of use" refers to how intuitive the IDE is to use and navigate without explanation. "Versatility" refers to the range of programming languages, additional components (plug-ins) and platforms supported by the IDE. "Aesthetic design" is the perceived appeal of the layout, icons and other visual elements of the IDE. "Efficiency" represents the speed of an IDE, as well as how efficient the IDE is to use (number of steps required to access different elements etc.). "Customizability" is the ease of customizing the IDE. "Update rate" refers to the frequency of software updates to an IDE. "Support for different operating systems" is the ability to use the IDE on different operating systems. "Support for integration with other tools" addresses the IDEs support for using different tools together with the IDE. "Support for collaborative development" refers to the tools and support that an IDE provides to ease sharing and participating in software development collaborative work with other users.

## 2.3. Satisfaction with IDE and needed improvements

Developers were asked to rate their overall satisfaction with the chosen IDE. Figure 5 shows the distribution of the provided answers. The results are clear, with most participants (about 70%) being "very satisfied" about their IDE.

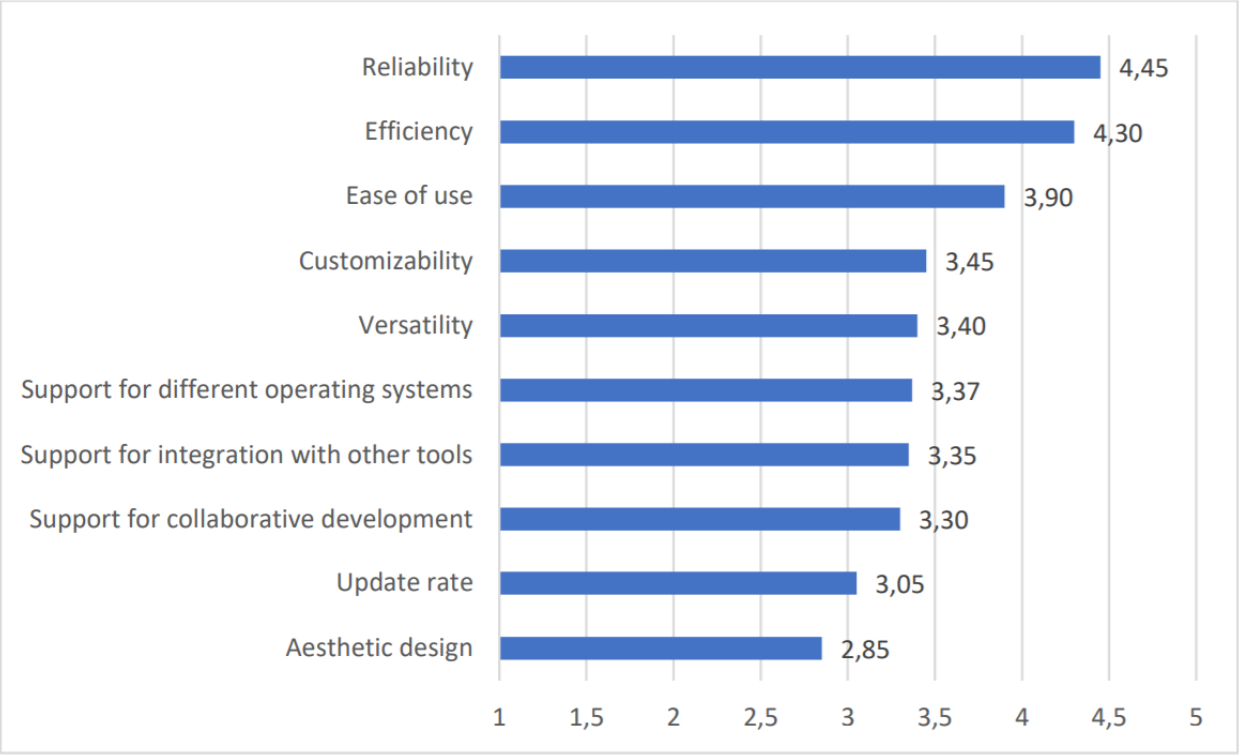


**Fig 3. Average rating of importance of qualities**

The highest rated quality of an IDE on average according to the participants was “reliability” (4,45 of 5), followed by “efficiency” and “ease of use”. On the other hand, “aesthetic design”, “update rate” and “support for collaborative development” were the least prioritized qualities. In addition to the rating, participants who had chosen their IDE themselves were also asked about the rationale for adopting it. Six out of the fourteen participants asked answered this question. Four, who chose Eclipse themselves, mentioned “customizability”, “versatility” and “reliability” as the primary reasons. The remaining two participants mentioned that they were somewhat limited in their choice of IDE due to the specific platform they were working with.

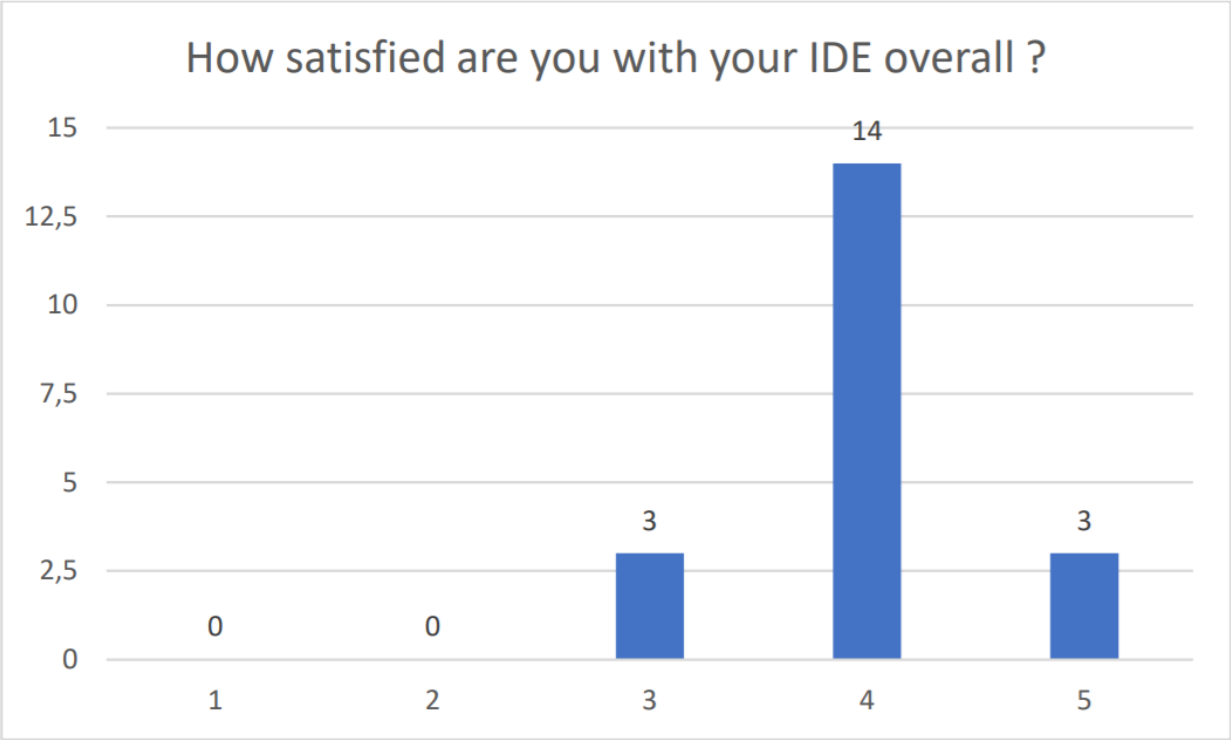


**Fig 4. Overall satisfaction with IDE**

Additionally, the participants were asked to rate how well their IDE satisfied the ten IDE qualities mentioned before. Figure 6 shows a synthesized view on how satisfied participants were about the different qualities in their chosen IDE.

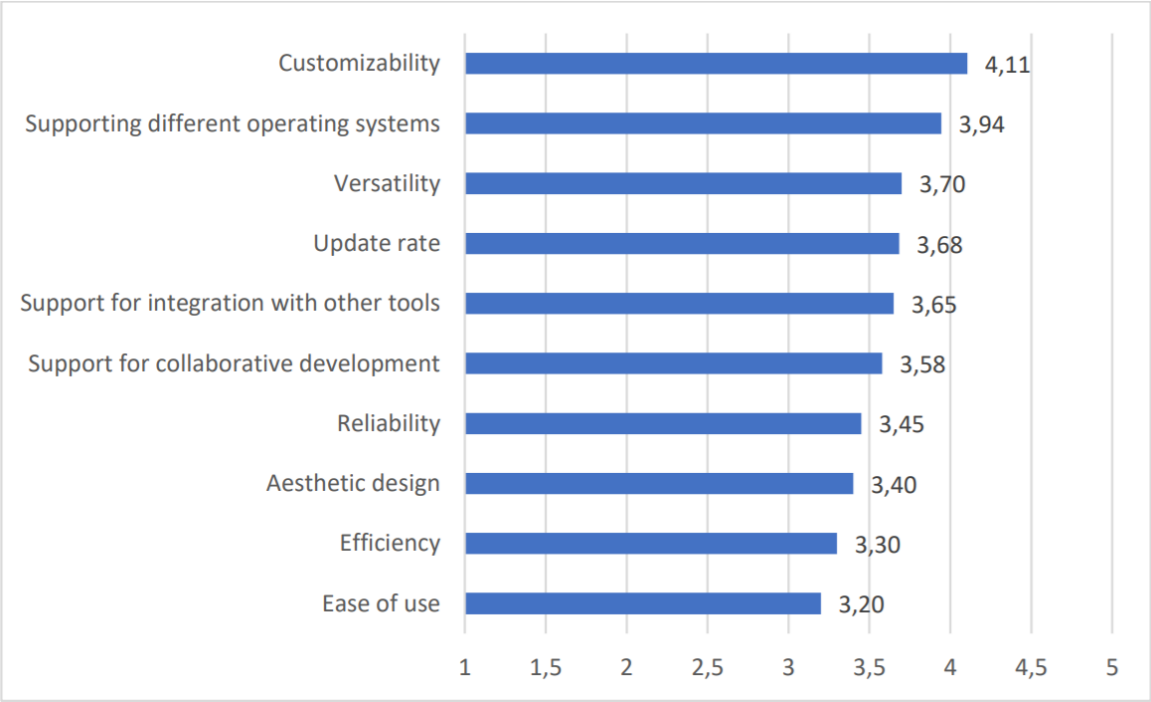


**Fig 5. Average rating of satisfaction with IDE qualities**

The most satisfactory quality overall according to the participants was “customizability” (4.11 of 5), followed by “support for different operating system” and “versatility”. On the other hand, qualities such as “ease of use”, “efficiency” and “aesthetic design” were perceived as least satisfactory.

## 2.4 Additional analysis

First, to better understand how developers are affected when they are limited in their choice of IDE, we analyzed the satisfaction between participants who could choose their IDE themselves and those who could not. The results did quite surprisingly not show any remarkable variation in overall satisfaction between the two groups. Moreover, the average ratings of satisfaction for each of the IDE qualities (see Figure 7) were very similar between the two groups too: participants who had chosen the IDE themselves rated their satisfaction with IDE qualities 6% higher than the other group on average. The only major differences were found when comparing “aesthetic design” and “versatility”, which participants who had chosen their IDE rated on average 29% and 36% higher than the other group, respectively.

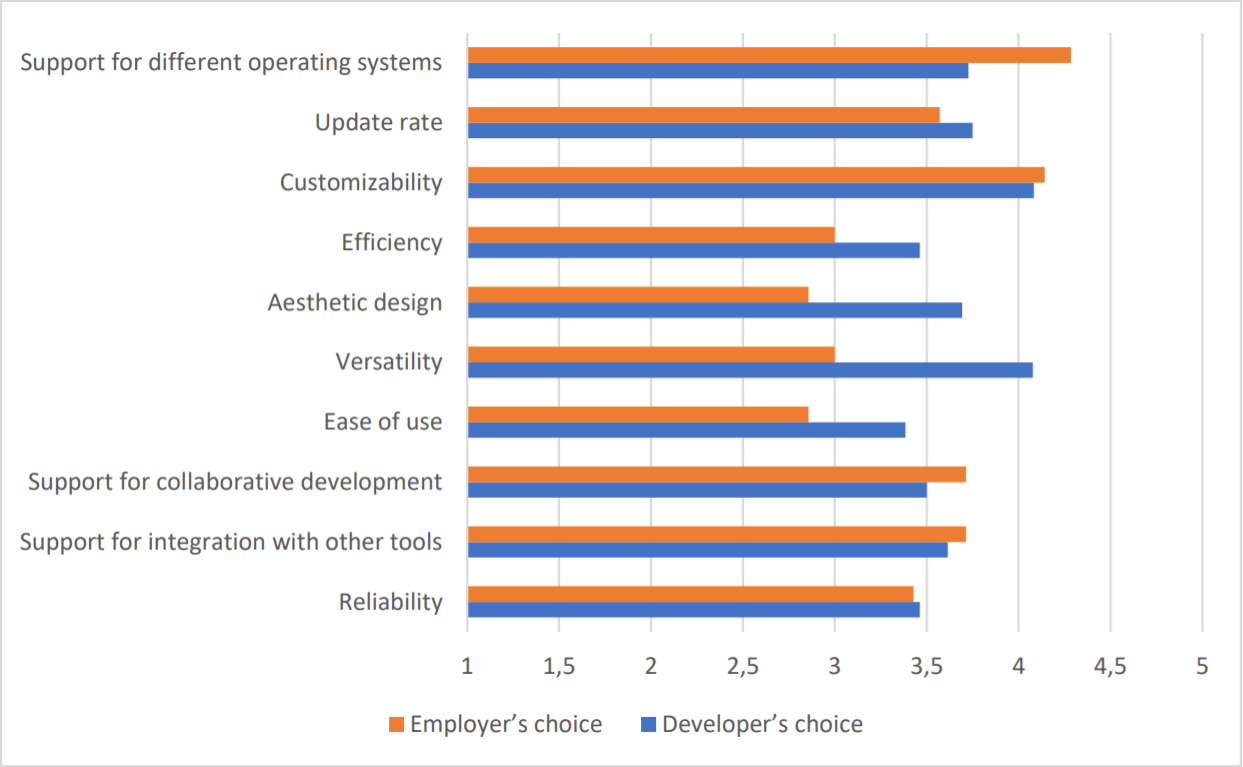


**Fig 6. Limitation of choice effect Vs overall satisfaction**

Comparing between the two most popular IDEs in the survey (Eclipse and Microsoft Visual Studio) shows that users feel the same satisfaction in general towards their respective IDE. However, users of Eclipse rate their satisfaction with the "efficiency" on average 20% lower compared to those using Visual Studio. On the other hand, Eclipse users felt on average 40% more satisfied with the "support for different operating systems" compared to those using Visual Studio. The survey results show that users of Eclipse and Visual Studio weigh the same IDE qualities in different ways. Eclipse users placed more importance on "support for collaborative development" (59% higher rating on average) and "integration with other tools" (37% higher rating on average). On the other hand, Visual Studio users felt that the "support for different operating systems" was more important than for Eclipse users. This result could be a consequence of the intrinsically low capability of Visual Studio in that specific matter. This is somewhat confirmed by the fact that Visual Studio users did not show high satisfaction with how the IDE provides that kind of support.

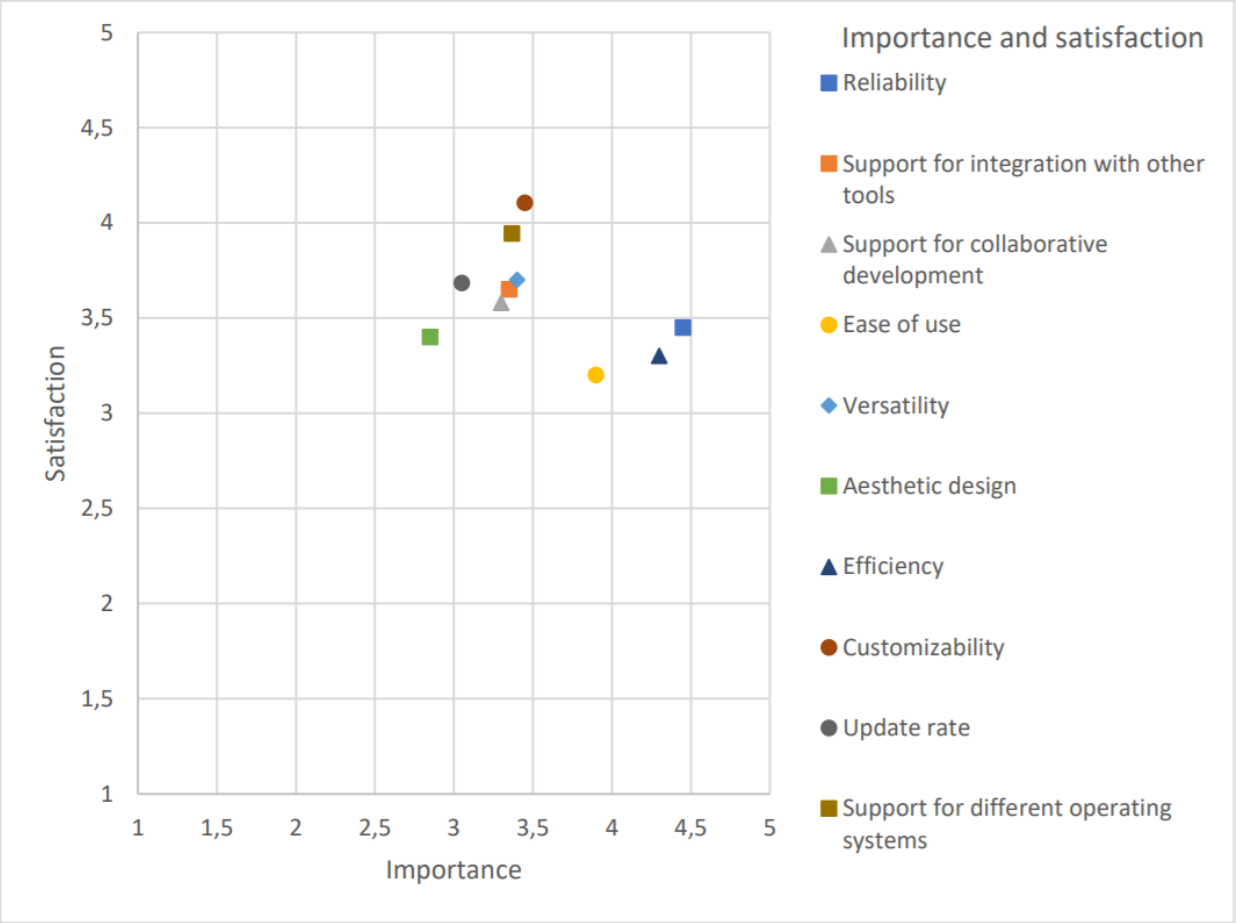


**Fig 7. Comparing importance and satisfaction**

## 2.5 Limitation of IDE

The participants seem very satisfied with their IDEs, where users of Visual Studio rated their overall satisfaction on average slightly higher than Eclipse users. The survey also allowed for the question to be answered in a scale from 1 to 5, and it is unlikely that the participants would feel completely satisfied in all aspects of their IDE with no room for improvement. On the other hand, an overall rating of 4 on a 1-5 scale shows very good satisfaction level with some room for improvement. Amongst the survey responses, the ability to use the IDE on different operating systems had a very high standard deviation. One possible reason why there is such a big dispersion among the answers could be that this quality is very dependent on the individual needs of developers. The 16 quality is also binary in nature, since a IDE can either have support for different operating systems or not. The survey could have been improved by using ranking questions. The results show that most participants tend to rate each quality as at least “very important” since it is

common to think that most qualities are necessary, but only a few could be considered “extremely important”. As a result, most qualities were valued similarly with only a few deviations. This trend could have been mitigated by structuring the questionnaire in a way that the potential differences could have been more evident. For instance, instead of asking the participants to rate each quality from “not at all important” to “extremely important”, the participants could have ranked the qualities from lowest priority to highest. By asking the participants to place each quality in order after their priorities, they would have been required to reflect more on how they prioritize qualities, thus making answers better reflect their actual thoughts.

## 2.6 Problem Definition:

On this basis on of the survey we get detailed information about how ide works their compilers, advantages, disadvantages after that we found some issues like GUI is good enough for log-time use for developers. Then Multiple Ide required for programming language. Code editors like vs code subline text, etc. require configure for that particular programming language download compilers separately then add to system path this very complex process. No one again gain configure ide. Now days developers are using various IDE like Vs code, Subline text but problems are that to set workspace for creating project is very complex. Configure for project any language & link compiler is also complex process & one more problem in colleges/schools Turbo C uses for C/C++ development. Turbo C is very old IDE which discontinued 20 year ago. If user want write same turbo c code on modern ide that not possible. So, solve this all issues we are developing Nitro Studio IDE

# Chapter 3 Proposed Methodology

Gathering Information for development like on which target platform we will use Identify problems. Then analyses others IDE software their issues, pros, cons with the help of them build our solution. After that we will plan our own solution its features, name, tools. Then getting project approved by faculty. Then after project approved discuss with team, distribute works with team members. Then Set-up workspace for development like installing SDK, tools. We will build blueprints of software then we will design GUI of our IDE according to the blueprints. Design: Frontend Environment – Splash, Intro, Create Project, Main Ide, settings, Sub-Menu, etc. Then main work developments first we give logic all software windows to link each other order them. Then link compilers/SDK to IDE & Final Touch for small events. After that Testing IDE with various platforms, languages. We will Deploy IDE in public.

## Action Plan:

| Sr. No. | Week | From | To | Description |
|---|---|---|---|---|
| 1 | 1st | 15/9/2021 | 18/9/2021 | Gathering Information for development |
| 2 | 2nd | 20/9/2021 | 25/9/2021 | Analyse others IDE software |
| 3 | 3rd | 27/9/2021 | 02/10/2021 | Communication: Planning solution, its features |
| 4 | 4th | 04/10/2021 | 09/10/2021 | Getting project approved by faculty |
| 5 | 5th | 11/10/2021 | 16/10/2021 | Communication: discuss with team, work distribute |
| 6 | 6th | 18/10/2021 | 23/10/2021 | Set-up workspace for development |
| 7 | 7th | 25/10/2021 | 30/10/2021 | Design: building blueprints of software |
| 8 | 8th | 01/11/2021 | 06/11/2021 | Design: Frontend Environment – Splash, Intro, Create Project, Main Ide, settings, Sub-Menu, etc |
| 9 | 9th | 08/11/2021 | 13/11/2021 | Constructions: Backend- Splash, Intro screens |
| 10 | 10th | 15/11/2021 | 20/11/2021 | Constructions: Backend- New Project screen |
| 11 | 11th | 22/11/2021 | 27/11/2021 | Constructions: Backend- Main Ide Environment, Sub menu |
| 12 | 12th | 29/11/2021 | 04/12/2021 | Constructions: link SDK with main ide |
| 13 | 13th | 06/12/2021 | 11/12/2021 | Constructions: Final Touch for small events |
| 14 | 14th | 13/12/2021 | 18/12/2021 | Testing: Testing IDE with various platforms, languages |
| 15 | 15th | 20/12/2021 | 25/12/2021 | Report Writing |
| 16 | 16th | 27/12/2021 | 31/12/2021 | Deployment: Announcing IDE in public |

# Chapter 4 Software Requirement

## Resources and consumable required

- **Software:** 1. Visual Studio 2022
  2. Net 6.0 SDK
  3. Adobe XD – Design

- **Hardware Requirement:**

  Processors: 4 core CPU, 2 GHZ
  Memory: 16 Gb
  HDD: 50Gb minimum
  Video Memory: Supports minimum 720p, Recommended 1080p
  Operating System: Windows 11, Windows 10

- **Used Hardware:**

  Processors: 2 core CPU, 2.30 GHZ
  Memory: 12 Gb
  HDD: 1TB
  Video Memory: 1920*1080p Display
  Operating System: Windows 11

# Chapter 6 Project Related Diagrams

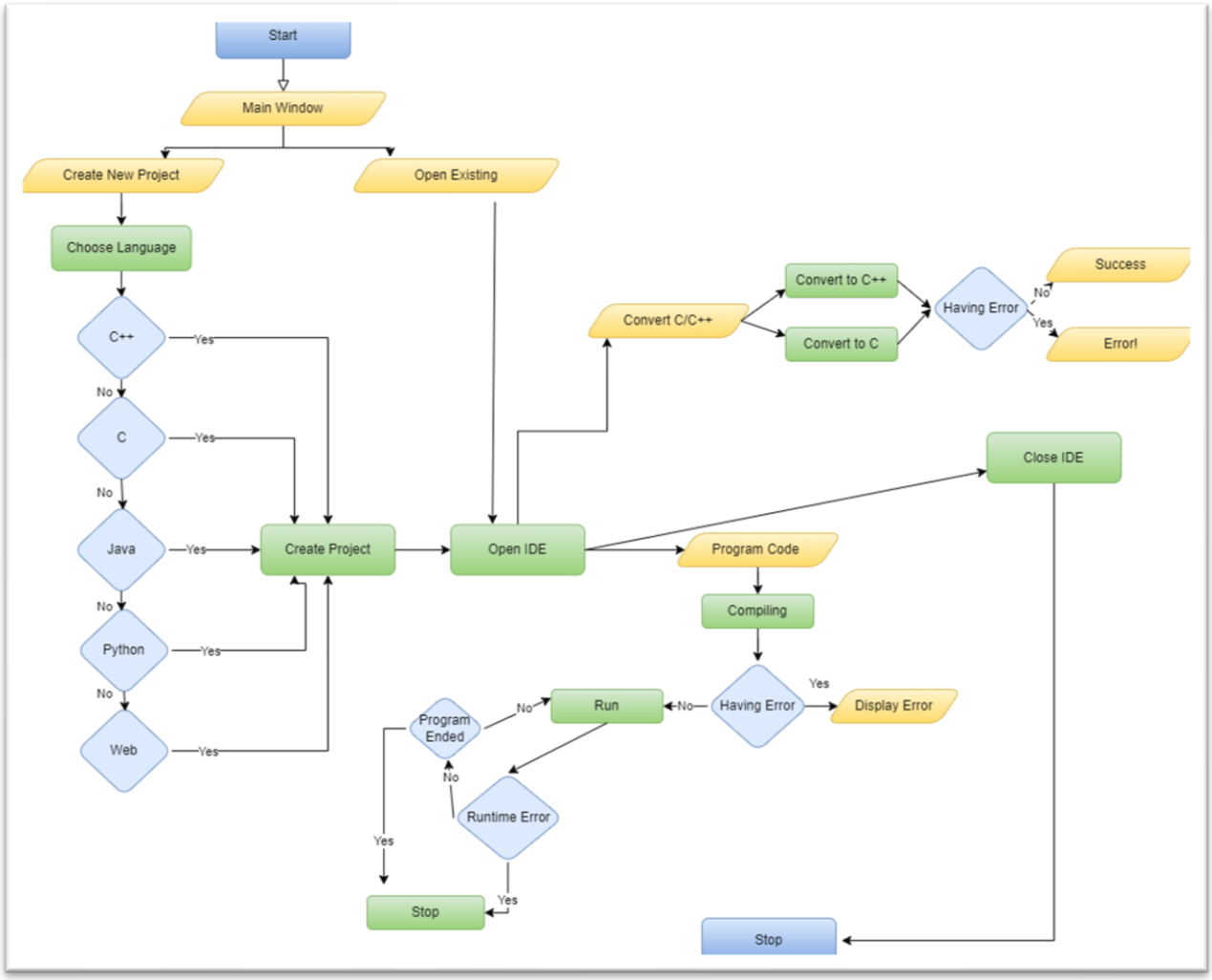


**Fig 8. Flow Diagram**

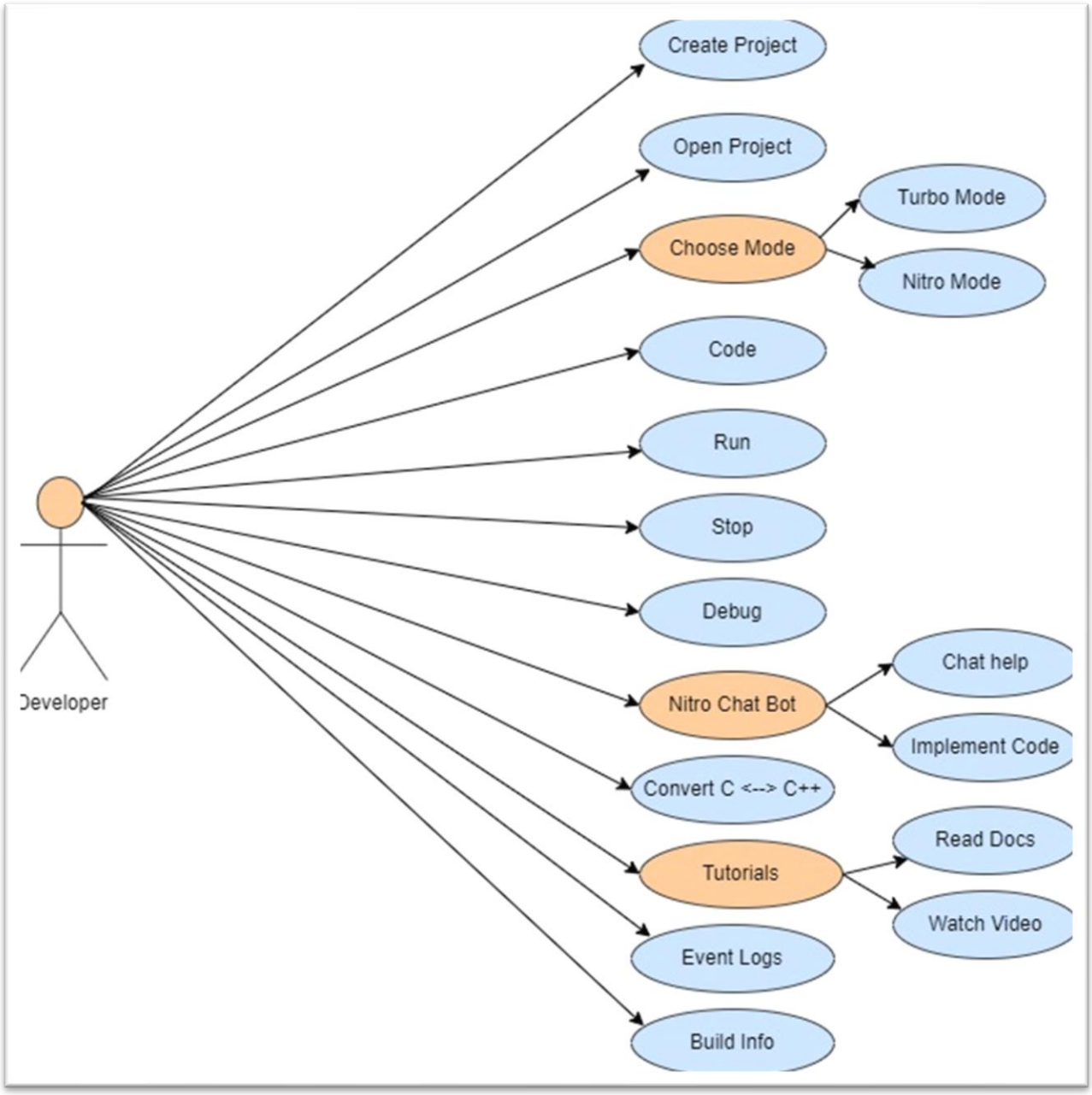


**Fig 9. Use Case Diagram**

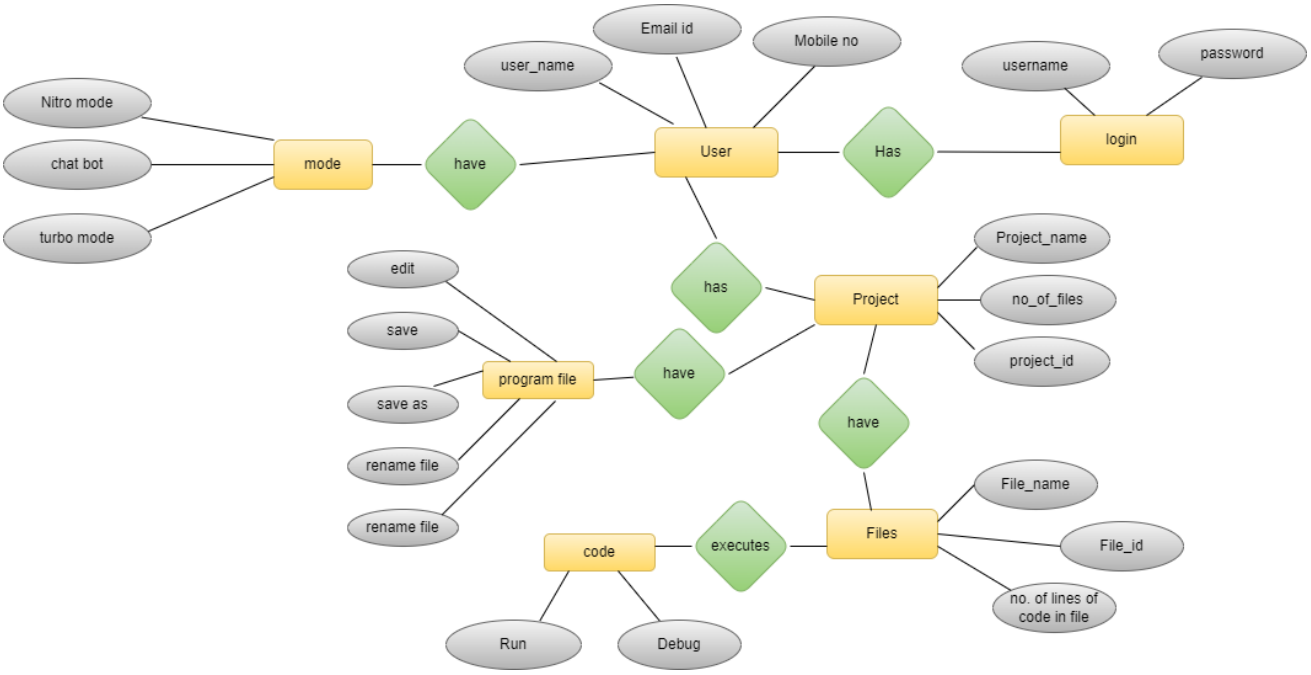


**Fig 10. ER Diagram**

# References


1. E. Foundation, "Eclipse Oxygen", Eclipse.org, 2018. [Online]. Available: http://www.eclipse.org/.
   [Accessed: 04-Dec-2021].
   "Code::Blocks", Codeblocks.org, 2018. [Online]. Available: http://www.codeblocks.org/.
   [Accessed: 04-Dec-2021].
2. D. Stroud, "Visual Studio IDE, Code Editor, VSTS, & App Center", Visual Studio, 2018. [Online]. Available:
   https://www.visualstudio.com [Accessed: 04-Dec-2021].
4. I. Sommerville, Software engineering. Boston: Addison-Wesley, 2011.
5. K. Kuusinen, H. Petrie, F. Fagerholm and T. Mikkonen, "Flow, Intrinsic Motivation, and