**NUMBER SYSTEM CONVERSION AUTOMATION PROJECT**

**COURSE: SPC2207**

**GROUP MEMBERS**:

MOSES RUNANU CT101/G/22122/24 (RunanuM)

BRIAN MUNENE CT101/G/21785/24 (brianmunenee)

AUGUSTINE OWAYA CT101/G/23524/24 (Silver-Incognito)

CLAIRE JEPNG’ETICH CT101/G/22839/24 (Clairetarus)

QUEEN ESTHER CT101/G/22453/24 (Esther-qu33n)

**GROUP ROLES**

Moses Runanu- Project Manager/ Backend (Logics)

Queen Esther- Process Documentation

Augustine Owaya- Backend (Logics)/Testing& debugging

Claire Jepng’etich- Design

Brian Munene- Frontend (UI)

**INTRODUCTION**

Number systems are a technique to represent numbers in the computer system architecture, every value that you are saving or getting into or from computer memory has a defined number system. The number system is used in the computer system for better communication and representation. We type letters or words into a computer system but computers do not understand words and letters; therefore, it converts every letter and word into numbers for better understanding and processing.

The digital computer converts or translates all data and information like audio, video, graphics, and text into binary form, i.e., 0s and 1s, which is easily read and understood by computers for better functionality and proper outputs. The output generated by computers is in human-readable form.

The different number systems are all equally important in a computer system for each of the number systems have their own functions. This is therefore why number system conversion is important. Moving between these number systems is very common, but doing it manually is slow and can lead to mistakes.

This project aims to make the process easier by creating a mobile application that can automatically convert numbers from one system to another. The app works on both Android and iOS devices and gives users a simple way to type in a number in one system and instantly see the result in the system they want all this aiming to provide fast, accurate and reliable results.

Apart from solving a practical problem, the project also helps the team put what we’ve learned in class into practice giving us a chance to experience how real software is built. Working as a group also teaches us teamwork, problem solving, and the use of project management methods.

Therefore, this project is not only about converting numbers but also preparing ourselves with the skills needed in future technology work.

**PROBLEM DEFINITION**

The main problem in application of the number systems in the computer system is moving between these systems. Manual conversion requires several steps and can be confusing, especially when dealing with large values or fractional numbers making mistakes to be more common and the process slow, making it unsuitable for real-world applications where speed and accuracy are important.

There is a need for a simple, accessible, and cross-platform solution that can automatically and accurately convert numbers between different bases since the ones that are being used commonly nowadays are either not reliable, user-friendly or available online for example calculators or online tools.

**OBJECTIVES**

The objective of this project is to design and implement a mobile application that automates number system conversions between binary, decimal, octal, and hexadecimal providing fast, accurate, and user-friendly results on both android and iOS devices by: -

1. Analyzing the requirements for number system conversions and identifying the common numeral bases to be supported.

2. Designing algorithms that can handle both integer and fractional number conversions across different bases.

3. Developing a mobile application that works on both android and iOS

4. Implementing a user interface that allows users to easily input numbers, select bases, and view results.

5. Integrating input validations and error handling to prevent incorrect conversions and improve reliability.

6. Testing the application thoroughly to make sure it is accurate and performs well.

7. Documenting the development process and program details for academic and practical purposes

8. Deploying the final application on android and iOS platforms for accessibility and use by learners and professionals

**TOOLS AND TECHNOLOGIES USED**

1. Programming Language & Framework

* Python
* Kivy

2. Development Environment

* Visual studio code

3. Version Control & Collaboration

* Git – for version control and tracking changes in source code.
* GitHub – for hosting the repository, team collaboration, issue tracking, and project management.

4. Testing & Debugging Tools

* Kivy Built-in Debugger/Logger

5. Project Management & Documentation

• GitHub Projects / Issues – task tracking and agile workflow.

• Google Docs / MS Word – writing process and program documentation.

• Edraw – creating flowcharts, data flow diagrams (DFDs), and system models.

• Figma/Canva – for designing simple UI mockups.

6. Build & Deployment Tools

* Linux operating System
* Buildozer- Bundles the kivy app into an android APK file that can be installed on the mobile devices

**STEP BY STEP EXPLANATION OF PROJECT WORK**

The team started off by assigning different roles to each member as stated above. The documentation team first wrote and submitted an introduction of the project and outlined the main objectives of this project and also together with the name of each member along with their roles. The design team then came up with a design layout/plan of how the mobile application will look like which she presented in form of a diagram which was also submitted in the github repository. The backend team then started off building the logic of the app by implementing the algorithm using python language. To implement the prototype the frontend and design team first needed to design the user interface for the input ,output,values, dropdowns for selecting base to target base,convert button that triggers the conversions. They validated the input by checking if the digits are valid for the given base. They also checked conversion functions from integers,fractions and validation. They then refined the prototype through styling,(colors and responsive layout). Accessibility through labels,use user friendly messages for error handling. The backend team then used kivy to create the app using python language. They also checked for errors and then finally deployed the app on android.

**ALGORITHM USED**

BEGIN

FUNCTION ConvertToDecimal(number, sourceBase)

SET decimalValue ← 0

SET power ← 0

REVERSE number // process digits from right to left

FOR each digit d in number

IF d is a letter (A-F or beyond) THEN

CONVERT d to its numeric equivalent (A=10, B=11, ...)

ENDIF

decimalValue ← decimalValue + (d \* (sourceBase ^ power))

power ← power + 1

END FOR

RETURN decimalValue

END FUNCTION

FUNCTION ConvertFromDecimal(decimalValue, targetBase)

SET result ← empty string

WHILE decimalValue > 0

remainder ← decimalValue MOD targetBase

IF remainder > 9 THEN

CONVERT remainder to corresponding letter (10=A, 11=B, ...)

ENDIF

result ← remainder + result // prepend digit

decimalValue ← decimalValue DIV targetBase

END WHILE

RETURN result

END FUNCTION

// MAIN PROGRAM

INPUT number

INPUT sourceBase

INPUT targetBase

decimalNumber ← ConvertToDecimal(number, sourceBase)

convertedNumber ← ConvertFromDecimal(decimalNumber, targetBase)

PRINT "Converted Number = ", convertedNumber

END

**CHALLENGES FACED WHEN DOING THE PROJECT**

During the development of the Number Systems Converter application, several challenges were encountered, particularly in the areas of environment setup, dependency management, and APK compilation. The following section outlines the key issues faced and the solutions implemented to overcome them.

1. ***Compatibility Issues with Non-Debian Linux Distributions***

**Challenge:**

The initial development environment was based on Fedora Linux, which is not Debian-based. Since Buildozer and Python-for-Android are primarily designed for Debian environments, several dependency errors occurred during installation and compilation. Required packages such as the Android SDK, NDK, and certain Python dependencies failed to configure properly, causing repeated build interruptions.

**Solution:**

The development environment was migrated to Ubuntu 20.04.6 LTS, a Debian-based distribution. This change ensured compatibility with Buildozer’s default package management system and significantly reduced dependency conflicts. The switch to Ubuntu allowed smoother integration of the Android SDK, NDK, and other essential components.

2. ***Build Failures on the Latest Ubuntu Versions***

**Challenge:**

Initial builds attempted on newer Ubuntu releases resulted in repeated errors caused by incompatibility between Buildozer, Python-for-Android, and the system’s updated libraries. Some dependencies had been renamed or deprecated, which led to missing modules and failed installations.

**Solution:**

A stable environment using Ubuntu 20.04.6 LTS was adopted instead of the latest Ubuntu version. Additionally, Python 3.10 was installed and pinned as the active interpreter, as it matched the version officially supported by both Kivy and Buildozer. This approach stabilized the build environment and eliminated most version-related errors.

3. ***Package and Module Dependency Errors***

**Challenge:**

Several dependency issues occurred during setup, including missing modules such as setuptools, distros, and Cython. Buildozer’s automated dependency installation occasionally failed due to outdated repositories or incomplete installations, especially when running inside virtual or containerized environments.

**Solution:**

The required dependencies were manually installed using pip, ensuring compatibility with the selected Python version. Specific versions of critical modules were pinned (e.g., Cython==0.29.36) to maintain consistency. After manual installation, the build process ran successfully without missing module errors.

4. ***Attempts with Docker and Google Colab***

**Challenge**:

Docker containers and Google Colab environments were tested as alternative build environments to overcome local system limitations. However, both methods failed due to strict resource constraints and dependency mismatches. In Google Colab, timeouts and restricted file system access caused the build process to terminate prematurely, while in Docker, dependency installations failed because of missing Android SDK components and unlinked paths.

**Solution:**

The build process was moved entirely to a local Ubuntu 20.04.6 LTS setup with adequate storage and processing power. This environment provided full control over dependencies and system configuration, ensuring consistent and repeatable builds without internet restrictions or permission errors.

5. ***Slow or Failed APK Compilation***

**Challenge:**

APK compilation often stalled or failed midway especially during large dependency downloads. Limited system resources and occasional interruptions corrupted intermediate build files, leading to repeated rebuilds and wasted time.

**Solution:**

To address this, Buildozer’s cache was utilized to store downloaded SDK and NDK packages for reuse in subsequent builds. The commands buildozer android clean and buildozer android debug were used selectively to minimize unnecessary rebuilds. This optimization significantly reduced build time and improved stability.

**6. *Layout and Display Issues***

**Challenge:**

During UI design, layout inconsistencies were observed across different screen sizes and orientations. Some buttons appeared oversized while others overlapped, and the output display area lacked visual clarity, affecting user experience.

**Solution:**

The .kv file was revised to improve alignment and scalability using proportional layout properties such as size\_hint, padding, and spacing. The output display area was also redesigned to appear inside a distinct white box for better visibility, and the button font sizes were adjusted for readability and aesthetic balance.

**Conclusion**

We were able to come up with an android apk, while an Ios app is possible, we were not able to package it due to constraint of resources.

**References**

* Official kivy documentation.
* Google.
* Youtube.