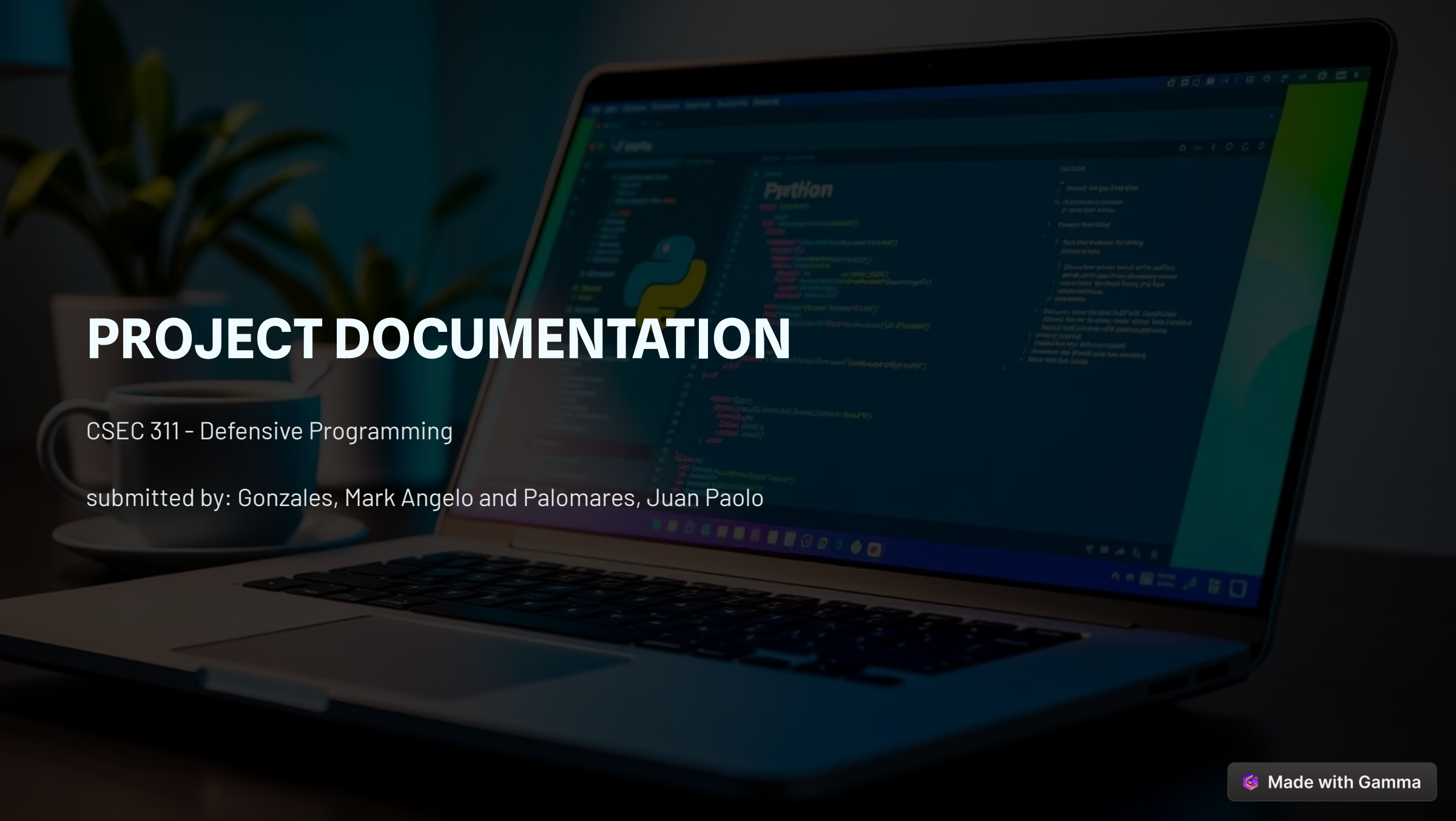


PROJECT DOCUMENTATION

CSEC 311 - Defensive Programming

submitted by: Gonzales, Mark Angelo and Palomares, Juan Paolo

Project Overview

Description

The "All-in-One Converter" is a Python-based GUI application created using Tkinter. It integrates multiple types of converters (currency, temperature, weight, length, and area) in a single, user-friendly interface.

Rationale

This project was chosen to demonstrate skills in Python, GUI programming with Tkinter, and modular application design. The purpose is to build a practical, everyday tool that simplifies conversions.

Installation and Setup Guide

1

Install Python

Download and install Python 3.x from the official website: https://www.python.org/downloads/.

2

Install Tkinter

Tkinter is the standard GUI library for Python and is usually pre-installed with Python. If not, install it using: pip install tk.

User Manual

1 Launch the Application

Execute the Python script to open the application window.

2 Choose a Converter

Select one of the conversion types by clicking the corresponding buttons (e.g., Currency, Temperature, Length, Weight, Area).

3 Enter Input Value

Input a numerical value in the designated input field.

4 Select Units for Conversion

Choose the appropriate units for both the "from" and "to" fields.

5 Perform Conversion

Click the "Convert" button to see the result.

Technical Documentation

Architectural Changes

The architecture follows a modular design pattern. Each converter (Currency, Temperature, Length, Weight, and Area) is implemented as an individual function.

New Classes, Methods, and Functions

CurrencyConverter(), TemperatureConverter(), LengthConverter(), AreaConverter(), WeightConverter()

Defensive Programming Techniques

Assertion and Invariants Error Handling: The program uses try-except blocks to catch any errors. Input Validation: Only numerical inputs are allowed for conversion.

Code Review and Refactoring

Refactoring the Currency Converter

The original currency conversion function was refactored to utilize the API for real-time exchange rates.

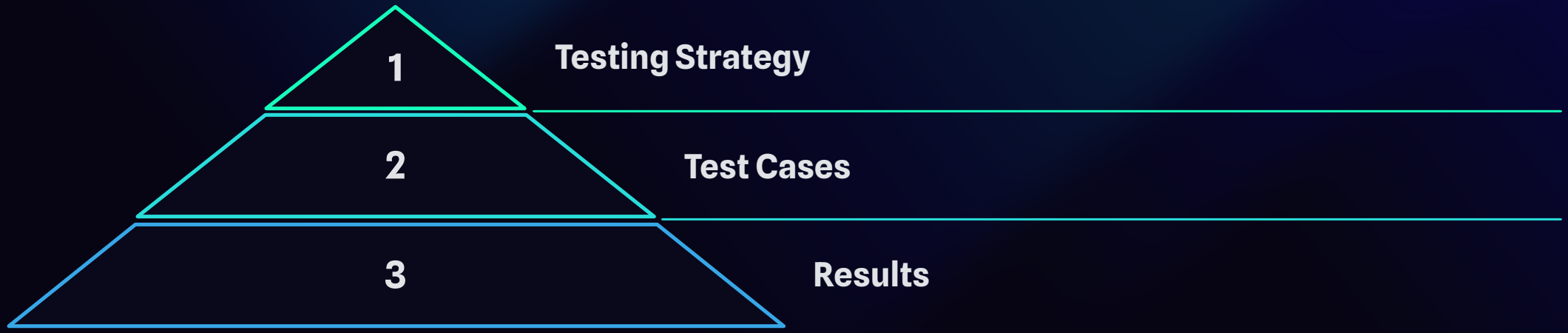
Optimizing the Layout

The widgets were rearranged to improve readability and the flow of the application.

Modularization

The conversion functions were separated into their respective modules, improving the maintainability of the code.

Testing Documentation



Testing was performed manually by inputting different values and verifying the correctness of the output for each converter. Functional testing was used to ensure the converters performed as expected.

Challenges and Solutions

1 **Handling Multiple Conversions**

2 **Currency Conversion Accuracy**

3 **Solutions**

Initially, the application struggled with managing multiple conversion types in a single interface. Fetching real-time exchange rates from an external API introduced challenges related to API limits and error handling.

Future Improvements

1

More Conversions

Future versions of the application could support additional conversions, such as volume, speed, and energy.

2

Unit Testing

Implement automated unit tests to validate the functionality of each converter.

3

UI Enhancements

Enhance the user interface with additional features, such as real-time updates and graphical representations of conversion data.