# Project Proposal

## Project Title

Wordle Clone in Assembly Language (Irvine32)

## Group Members

Ahmed Raza (24K-1010)  
Simal Hassan (24K-0688)  
Laiba Jamil (24K-0812)

## 1. Introduction

Background: This project focuses on low-level programming using Assembly Language to build a text-based version of the popular game Wordle. The aim is to apply concepts of computer organization, string manipulation, and control flow in assembly to demonstrate how modern applications can be recreated at the hardware-near level.

Problem Statement: Many programming projects are created in high-level languages, which makes it easy to overlook the complexity of basic operations. This project addresses the challenge of implementing a word-based game purely in assembly language, where every input, comparison, and output must be handled manually.

Objectives:  
- To build a simplified Wordle clone using Assembly Language and Irvine32 library.  
- To practice and demonstrate string handling, loops, and conditional logic in assembly.  
- To enhance understanding of how low-level programming connects software logic to underlying hardware operations.

## 2. Scope of the Project

Inclusions:  
- A 5-letter word guessing game with up to 6 attempts.  
- Input validation (only 5 characters allowed per guess).  
- Feedback system:  
  
 - Correct letter in correct position (Green → represented with []).  
 - Correct letter in wrong position (Yellow → represented with ()).  
 - Incorrect letter (Grey → printed normally).  
Note: If time allows, we may try implementing colors to indicate the above feedback system.  
  
- A win/lose message at the end of the game.

Exclusions:  
- No large dictionary support (only a fixed word or a small set of hidden words).  
- No advanced graphics; output will be text-based.  
- No multiplayer or online functionality.

## 3. Project Description

Overview: The project will recreate Wordle’s core logic in Assembly Language using Irvine32. The user will attempt to guess a hidden 5-letter word within 6 tries, receiving feedback for each guess. The program will highlight correct and misplaced letters using symbols to simulate the original game’s green/yellow/grey indicators.

Technical Requirements:  
- Microsoft Visual Studio (with MASM and Irvine32 library).  
- Windows OS (for compatibility with Irvine32).  
- MS Word/PowerPoint (for documentation and presentation).

Project Phases:  
1. Research – Understanding Wordle rules and translating them into assembly logic.  
2. Planning – Dividing roles and designing the program structure.  
3. Design – Writing pseudocode/flowcharts for input handling, comparison, and output.  
4. Implementation – Writing and testing the code in iterative steps.  
5. Finalization – Debugging, documentation, and preparing the presentation.

## 4. Methodology

Approach: The group will work in iterative steps, starting with basic input and output, then gradually adding logic for word comparison and feedback. Testing will be done after each phase to ensure correctness.

Team Responsibilities:  
- Ahmed Raza: Input handling and validation logic.  
- Simal Hassan: Word comparison logic (checking correct letters/positions).  
- Laiba Jamil: Output formatting, feedback display, and game loop control.

## 5. Expected Outcomes

Deliverables:  
- A working Assembly program that plays a simplified Wordle game.  
- A short report detailing the design, logic, and challenges.  
- User instructions for running and playing the game.

Relevance: The project connects directly with Computer Organization and Assembly Language (COAL) by showcasing low-level string manipulation, loops, and conditional execution. It highlights how simple games and applications can be implemented close to machine-level programming.

## 6. Resources Needed

Software:  
- Microsoft Visual Studio with MASM and Irvine32 library.  
- MS Word/PowerPoint for documentation and presentation.

Other Resources:  
- Online tutorials/documentation for Irvine32 functions.  
- Guidance from the course instructor when needed.