CS1060 – HW2 Design Document

Numeric Converter – Bugfix Branch

Author: Mohamed Salam Moumie Ntieche
Repo: `https://github.com/GITWOCS/gitwocs-hw2.git`
Branch: bugfix

1. Overview

The project is a Python/Flask web application that converts a single number between different representations: English text (e.g., *"forty two"*), Binary, Octal, Decimal, Hexadecimal, Base64

The assignment required:

1. Creating a **pytest test suite** that covers all reasonable conversions and error cases.
2. Identifying at least one bug in the provided implementation.
3. Fixing the bug in `api/index.py`.
4. Deploying the application to **Vercel** and submitting the production link.

2. Initial Implementation

The starter `index.py` exposed Flask routes and helper functions for conversions. Major functions included:

- `text_to_number()` → convert English words to integer
- `number_to_text()` → convert integer to English words
- `base64_to_number()` / `number_to_base64()` → handle Base64 encoding/decoding
- A `/convert` endpoint that accepts JSON `{input, inputType, outputType}` and returns a result

The provided implementation worked for most flows but had several hidden issues.

3. Test Suite Design

The test suite is split into three parts:

1. **Conversions (`tests/test_conversions.py`)**
    - Matrix tests: every input type → every output type
    - Representative integers: 0, 1, 42, 255, 256, 65535, 1,048,576, $2^{31}-1$
    - Explicit Base64 endianness checks
    - Base64 round-trip identity (`base64 → base64`)
2. **Errors (`tests/test_errors.py`)**
    - Negative integers
    - Invalid hex (`0xG1`)
    - Invalid binary (`21010`)
    - Invalid base64 (`!!!`)
    - Unknown input/output types (e.g weird)
3. **README Examples (`tests/test_readme_examples.py`)**
    - Decimal 42 → Binary 101010
    - Text *"forty two"* → Decimal 42
    - Hexadecimal `2a` → Text *"forty two"*

The tests were designed to fail when bugs are present and pass (most) once they are fixed.

4. Bugs Identified

The test suite exposed three main bugs in the starter code:

1. **Base64 Endianness Bug (Critical)**
    - The code used **big-endian** encoding for Base64 conversions.
    - The assignment required **little-endian** (Windows/macOS default).
    - Example: integer 256 encoded to `"AQA="` instead of correct `"AAE="`.
2. **Zero Encodes to Empty String**
    - `number_to_base64(0)` returned `""` instead of `"AA=="`.
    - Cause: calculated byte length was 0, so no bytes were encoded.

3. **Invalid Base64 Not Rejected**
    - Input `"!!!"` decoded silently to 0 instead of erroring.
    - Cause: `base64.b64decode()` was called without `validate=True`.

4. **Text Parsing Too Limited**
    - `"forty two"` caused `"Unable to convert text to number"`.
    - Cause: `text_to_number` only handled very small hardcoded words.

5. Fixes Applied

For this assignment, I decided to focus on addressing the **Base64 Endianness** and the **Zero Handling** issues and applied the fixes in `api/index.py`.

- **Base64 Endianness:** I made sure that instead of using the "*big*" byteorder, it now correctly makes use of the *"little"* for both encoding and decoding.
- **Zero Handling:** I passed the initial result alongside 1 as parameters to max to ensure that if the number of bits is 0, from a number 0 entered, then it be considered as 1. ie length = max(1, (number.bit_length() + 7) // 8)

6. Results

- **Before fixes:**
    - 350 tests passed,
    - 35 failed (all base64 multi-byte cases, zero encoding, invalid base64, and "forty two").

- **After fixes:**
    - 383 tests passed,
    - 2 failed (invalid base64 i.e. '!!!' being converted to 0, and "forty two").

7. Deployment

- The application was deployed to **Vercel** from the `bugfix` branch.
- The production link is stored in `vercel-link.txt` at the repo root.

8. Lessons Learned

- Subtle issues like **endianness** can break conversions silently — tests must explicitly check these cases.
- Always ensure functions handle **edge cases**: 0, invalid inputs, and round-trip identity.
- Avoid naming conflicts (e.g., Flask route `convert` vs pure helper `convert`).
- A good test suite should both **demonstrate the bug** and **verify the fix**.

Deployment Link (For redundancy)
https://gitwocs-hw2-git-bugfix-gitwocs-projects.vercel.app/