Python Design Document (UML/Architecture, etc.): Translator Application

This document contains UML class diagrams for each module in the Translator Application, presented as Mermaid code blocks and ASCII-style diagrams for inclusion in the CS50 final project submission.
Note: Auto-generated by a third-party application while analyzing code….

main.py

Program entry point. Initializes DB and launches TranslatorApp.

Mermaid UML Code:

```
classDiagram
    class Main {
        +main()
    }
    Main : +init_db()
    Main : +run TranslatorApp()
```

ASCII-style Diagram:

```
+---------------------+
|       main.py       |
+---------------------+
| - none              |
+---------------------+
| + main()            |
+---------------------+
```

gui.py (TranslatorApp)

Tkinter GUI class that manages windows, input, translation flow, and callbacks.

Mermaid UML Code:

```
classDiagram
    class TranslatorApp {
        - word_input: Text
        - lang_var: StringVar
        - result_label: Label
        + __init__(test_mode=False)
        + translate()
        + _online_translate_thread(word, lang_code, lang_name)
        + clear()
        + open_admin()
        + resize_to_fit_text()
    }
```

ASCII-style Diagram:

```
+-------------------------------------------+
|           TranslatorApp (gui.py)          |
+-------------------------------------------+
| - word_input: Text                         |
| - lang_var: StringVar                      |
| - result_label: Label                      |
+-------------------------------------------+
| + __init__(test_mode=False)               |
| + translate()                             |
| + _online_translate_thread(...)           |
| + clear()                                 |
| + open_admin()                            |
| + resize_to_fit_text()                    |
+------------------------------------------+
```

admin_panel.py (AdminPanel, EditDialog)

Admin UI with tabs for Dictionary, History, and Settings. Includes import/export and edit dialogs.

Mermaid UML Code:

```
classDiagram
    class AdminPanel {
        - dict_tree
        - hist_tree
        - dict_page
        - hist_page
        + __init__(master)
        + load_dictionary_page()
        + load_history_page()
        + import_dictionary_csv()
        + export_dictionary_csv()
        + export_history_csv()
        + export_history_pdf()
        + email_history_dialog()
    }

    class EditDialog {
        - row_id
        - table
        + __init__(master,row_id,field1,field2,field3,table,refresh_callback)
        + save()
        + delete_row()
        + delete_history_row()
    }

    AdminPanel --> EditDialog : uses
```

ASCII-style Diagram:

```
+-------------------------------------------------------+
|                 AdminPanel (admin_panel.py)           |
+-------------------------------------------------------+
| - dict_tree                                           |
| - hist_tree                                           |
| - dict_page, hist_page                                |
+-------------------------------------------------------+
| + __init__(master)                                    |
| + load_dictionary_page()                              |
| + load_history_page()                                 |
| + import_dictionary_csv()                             |
| + export_dictionary_csv()                             |
| + export_history_csv()                               |
| + export_history_pdf()                               |
| + email_history_dialog()                             |
+------------------------------------------------------+

+------------------------------+
|        EditDialog            |
+------------------------------+
| - row_id                     |
| - table                      |
+------------------------------+
| + save()                     |
| + delete_row()               |
| + delete_history_row()       |
+------------------------------+
```

database.py

SQLite helpers: init_db, CRUD operations for dictionary and history, pagination helpers.

Mermaid UML Code:

```
classDiagram
    class Database {
        + init_db()
        + insert_dictionary_row(word, translation, language)
        + update_dictionary_row(row_id, word, translation, language)
        + delete_dictionary_row(row_id)
        + query_dictionary(filter_text=None, order_by='id DESC', limit=None,
offset=None)
        + query_history(filter_text=None, order_by='id DESC', limit=None,
offset=None)
        + save_history(input_word, output_word, language, used_online=False)
    }
```

ASCII-style Diagram:

```
+------------------------------------------+
|              database.py                 |
+------------------------------------------+
| - DB_NAME (from settings)                |
+------------------------------------------+
| + init_db()                              |
| + insert_dictionary_row(...)             |
| + update_dictionary_row(...)             |
| + delete_dictionary_row(row_id)          |
| + query_dictionary(...)                  |
| + query_history(...)                     |
| + save_history(...)                      |
+------------------------------------------+
```

translation.py (sql_translate, online_translate)

Translation logic: local SQL lookup and online translation wrapper.

Mermaid UML Code:

```
classDiagram
    class Translator {
        + sql_translate(word, lang_code)
        + online_translate(word, lang_code)
    }
```

ASCII-style Diagram:

```
+---------------------------------------+
|         translation.py                |
+---------------------------------------+
| - (uses external API libs optionally) |
+---------------------------------------+
| + sql_translate(word, lang_code)      |
| + online_translate(word, lang_code)   |
+---------------------------------------+
```

exports.py (export_history_json, export_history_pdf)

Export utilities: JSON and PDF (ReportLab) with UTF-8 font fallback to text.

Mermaid UML Code:

```
classDiagram
    class ExportUtils {
        + export_history_json(path, rows)
        + export_history_pdf(path, rows, title='Translation History')
    }
```

ASCII-style Diagram:

```
+-----------------------------------------------+
|                 exports.py                    |
+-----------------------------------------------+
| + export_history_json(path, rows)             |
| + export_history_pdf(path, rows, title)       |
+-----------------------------------------------+
```

emailer.py (send_history_email)

Email helper: Outlook COM draft or mailto fallback.

Mermaid UML Code:

```
classDiagram
    class EmailUtils {
        + send_history_email(to_addr, subject, body, attachment_path=None)
    }
```

ASCII-style Diagram:

```
+--------------------------------------------+
|               emailer.py                   |
+--------------------------------------------+
| + send_history_email(to_addr,subject,body, |
|                       attachment_path=None) |
+--------------------------------------------+
```

algorithms.py

Educational algorithm implementations: selection_sort and binary_search used in admin search.

Mermaid UML Code:

```
classDiagram
    class Algorithms {
        + selection_sort(arr)
        + binary_search(sorted_list, target)
    }
```

ASCII-style Diagram:

```
+------------------------------------+
|           algorithms.py            |
+------------------------------------+
| + selection_sort(arr)              |
| + binary_search(sorted_list,target)|
+------------------------------------+
```

settings.py

Configuration constants: DB_NAME, LANG_OPTIONS, PAGE_SIZE, TXT_FILES, DEEP_TRANSLATOR_AVAILABLE, font paths.

Mermaid UML Code:

```
classDiagram
    class Settings {
        - DB_NAME
        - LANG_OPTIONS
        - PAGE_SIZE
        - TXT_FILES
        - DEEP_TRANSLATOR_AVAILABLE
        - FONT_PATHS
    }
```

ASCII-style Diagram:

```
+------------------------------+
|         settings.py          |
+------------------------------+
| - DB_NAME                    |
| - LANG_OPTIONS               |
| - PAGE_SIZE                  |
| - TXT_FILES                  |
| - DEEP_TRANSLATOR_AVAILABLE  |
+------------------------------+
```

System Architecture (Mermaid)

```
flowchart TD
    User((User)) -->|Input| GUI[TranslatorApp (Tkinter)]
    GUI --> Translator[translation.py]
    Translator -->|Local lookup| DB[(SQLite Database)]
    Translator -->|Online| OnlineAPI((MyMemory API / Deep Translator))
    GUI --> Admin[AdminPanel]
    Admin --> Database
    GUI --> Exporter[exports.py]
    Exporter --> Files[(JSON / PDF)]
    GUI --> Emailer[emailer.py]
    Emailer --> EmailClient((Outlook / mailto))
```

---

1) Full UML (Mermaid) + ASCII diagrams for each module

admin_panel.py

```
classDiagram
    class AdminPanel {
        - dict_tree
        - hist_tree
        - dict_page
        - hist_page
        + __init__(master)
        + load_dictionary_page()
        + load_history_page()
        + import_dictionary_csv()
        + export_dictionary_csv()
        + export_history_csv()
        + export_history_pdf()
        + email_history_dialog()
    }
    class EditDialog {
        - row_id
        - table
        + __init__(master,row_id,field1,field2,field3,table,refresh_callback)
        + save()
        + delete_row()
        + delete_history_row()
    }
    AdminPanel --> EditDialog : uses
ASCII:
+-------------------------------------------------------+
|                AdminPanel (admin_panel.py)            |
+-------------------------------------------------------+
| - dict_tree                                           |
| - hist_tree                                           |
| - dict_page, hist_page                                |
+-------------------------------------------------------+
| + __init__(master)                                    |
| + load_dictionary_page()                              |
| + load_history_page()                                 |
| + import_dictionary_csv()                             |
| + export_dictionary_csv()                             |
| + export_history_csv()                                |
| + export_history_pdf()                                |
| + email_history_dialog()                              |
+-------------------------------------------------------+
```

---

algorithms.py

```
classDiagram
    class Algorithms {
        + selection_sort(arr)
        + binary_search(sorted_list, target)
    }
ASCII:
+------------------------------------+
|           algorithms.py            |
+------------------------------------+
| + selection_sort(arr)              |
| + binary_search(sorted_list,target)|
+------------------------------------+
```

---

database.py

```
classDiagram
    class Database {
        + init_db()
        + insert_dictionary_row(word, translation, language)
        + update_dictionary_row(row_id, word, translation, language)
        + delete_dictionary_row(row_id)
        + query_dictionary(filter_text=None, order_by='id DESC', limit=None,
offset=None)
        + query_history(filter_text=None, order_by='id DESC', limit=None,
offset=None)
        + save_history(input_word, output_word, language, used_online=False)
    }
```

ASCII:

```
+-----------------------------------------+
|              database.py                |
+-----------------------------------------+
| - DB_NAME (from settings)               |
+-----------------------------------------+
| + init_db()                             |
| + insert_dictionary_row(...)            |
| + update_dictionary_row(...)            |
| + delete_dictionary_row(row_id)         |
| + query_dictionary(...)                 |
| + query_history(...)                    |
| + save_history(...)                     |
+-----------------------------------------+
```

---

emailer.py

```
classDiagram
    class Emailer {
        + send_history_email(to_addr, subject, body, attachment_path=None)
    }
ASCII:
+--------------------------------------------+
|               emailer.py                   |
+--------------------------------------------+
| + send_history_email(to_addr,subject,body, |
|                      attachment_path=None) |
+--------------------------------------------+
```

---

exports.py

```
classDiagram
    class ExportUtils {
        + export_history_json(path, rows)
        + export_history_pdf(path, rows, title='Translation History')
    }
ASCII:
+-----------------------------------------------+
|                 exports.py                     |
+-----------------------------------------------+
| + export_history_json(path, rows)              |
| + export_history_pdf(path, rows, title)        |
+-----------------------------------------------+
```

---

gui.py (TranslatorApp)

```
classDiagram
    class TranslatorApp {
        - word_input: Text
        - lang_var: StringVar
        - result_label: Label
        + __init__(test_mode=False)
        + translate()
        + _online_translate_thread(word, lang_code, lang_name)
        + clear()
        + open_admin()
        + resize_to_fit_text()
    }
```

ASCII:

```
+------------------------------------------+
|           TranslatorApp (gui.py)          |
+------------------------------------------+
| - word_input: Text                         |
| - lang_var: StringVar                      |
| - result_label: Label                      |
+------------------------------------------+
| + __init__(test_mode=False)                |
| + translate()                              |
| + _online_translate_thread(...)            |
| + clear()                                  |
| + open_admin()                             |
| + resize_to_fit_text()                     |
+------------------------------------------+
```

---

main.py

```
classDiagram
    class Main {
        + main()
    }
    Main : +init_db()
    Main : +run TranslatorApp()
```

ASCII:

```
+---------------------+
|       main.py       |
+---------------------+
| - none              |
+---------------------+
| + main()            |
+---------------------+
```

---

requirements.txt / requirements-dev.txt

```
(plain text; list of packages)
requirements.txt — contains runtime dependencies (e.g., requests, reportlab,
deep-translator (optional), python-docx, python-pptx)
requirements-dev.txt — contains development deps (pytest, flake8, etc.)
```

---

settings.py

```
classDiagram
    class Settings {
        - DB_NAME
        - LANG_OPTIONS
        - PAGE_SIZE
        - TXT_FILES
        - DEEP_TRANSLATOR_AVAILABLE
        - FONT_PATHS
    }
```

ASCII:

```
+-----------------------------+
|         settings.py         |
+-----------------------------+
| - DB_NAME                   |
| - LANG_OPTIONS              |
| - PAGE_SIZE                 |
| - TXT_FILES                 |
| - DEEP_TRANSLATOR_AVAILABLE |
+-----------------------------+
```

---

spelling.py

```
classDiagram
    class Spelling {
        + auto_correct(word)
        + load_cache()
        + save_cache()
    }
```

ASCII:

```
+-----------------------------------+
|            spelling.py            |
+-----------------------------------+
| + auto_correct(word)              |
| + load_cache()                    |
| + save_cache()                    |
+-----------------------------------+
```

---

translation.py

```
classDiagram
    class Translator {
        + sql_translate(word, lang_code)
        + online_translate(word, lang_code)
    }
```

ASCII:

```
+---------------------------------------+
|         translation.py                |
+---------------------------------------+
| - (uses external API libs optionally) |
+---------------------------------------+
| + sql_translate(word, lang_code)      |
| + online_translate(word, lang_code)   |
+---------------------------------------+
```

---

validation.py

```
classDiagram
    class Validation {
        + validate_input_word(word)
    }
```

ASCII:

```
+--------------------------------+
|         validation.py          |
+--------------------------------+
| + validate_input_word(word)    |
+--------------------------------+
```