**AI-Powered Automated Braille Translation System.**

**Batch No.:** 7CSE-B-BNO-5.

**Team Members:** Sharon A Dobbin (1AT22CS120), P Shreya (1AT22CS069), Sandhya S (1AT22CS085), Pooja Vijay Bijapur (1AT23CS404).

**Guide Name:** Prof. Y Mary Angel.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **CONTENT** | **PAGE NO.** |
| CHAPTER 1 | Software Requirements Specification. | 2 – 14 |
| CHAPTER 2 | Architectural Design and Diagrams | 15 –22 |

**Chapter 1: Software Requirements Specification.**

**Table of Contents**

**1.1.0 Introduction**

1.1.1 Purpose

1.1.2 Scope of the Project

1.1.3 Glossary

1.1.4 References

1.1.5 Overview of the Document

**1.2.0 Overall Description**

1.2.1 System Environment

1.2.2 Functional Requirements Specification

1.2.2.1 Core Use Cases

1.2.3 User Characteristics

1.2.4 User Needs

1.2.5 Non-Functional Requirements Specification

**1.3.0 Detailed Requirement Specification**

1.3.1 Detailed External Interface Requirements Specifications

1.3.1.1 GUI

1.3.1.2 Hardware Interfaces

1.3.1.3 Software Interfaces

1.3.1.4 Communications Interfaces

1.3.2 Detailed Functional Requirements Specifications

1.3.2.1 User Authentication & Profile Management

1.3.2.2 Braille Translation & Document Handling

1.3.2.3 AI-Powered Input Translation

1.3.2.4 Accessibility & Output Features

1.3.3 Detailed Non-Functional Requirements Specifications

1.3.3.1 Performance Requirements

1.3.3.2 Usability Requirements

1.3.3.3 Security Requirements

1.3.3.4 Maintainability Requirements

1.3.3.5 Portability Requirements

**List of Figures**

**Figure 1.1:** Demo of Grade 1 Translation using Python logic.

**Figure 1.2:** Primary operational flow of the Braille Translation

**1.1.0 Introduction.**

**1.1.1 Purpose.**

Visually impaired individuals face a significant obstacle in accessing written material due to the limited availability of Braille content and the high cost and inefficiency of existing solutions. To address this, our project leverages **Machine Learning** and **Natural Language Processing (NLP)** to create an accessible and efficient AI-powered software platform. The system will seamlessly translate both digital and spoken English into Grade 2 Braille and will also feature the ability to recognize Braille from images. The primary goal is to enhance digital inclusivity and empower visually impaired users with greater independence by providing a fast, reliable, and user-friendly tool.

**1.1.2 Scope of the Project**

This project will deliver a standalone AI-powered software application for English Braille translation. The initial version will focus on core capabilities, with plans to expand to include multilingual support.

The software's core capabilities will include:

1. **Multi-Modal Input:**

* **Digital Text:** The system will accept digital text from files such as Word, PDF, and TXT, as well as from web content.
* **Printed/Handwritten Text:** An AI-driven Optical Character Recognition (OCR) model will be used to translate text captured from images of printed or handwritten documents.
* **Spoken English:** The application will use voice recognition to transcribe spoken English into Braille.

1. **Core Translation Functionality:**

* **Text-to-Braille:** The software will convert English text into both Grade 1 and Grade 2 Braille.
* **Braille-to-Text:** It will also be capable of converting Braille back into readable English text.
* The system will use **machine learning (ML)** and **natural language processing (NLP)** to accurately handle the complexities of Grade 2 Braille, including contractions and grammatical rules.

1. **Output Formats:**

* **On-screen Display:** Users will see both the original text and the translated Braille, with a visual Braille cell preview for confirmation.
* **Digital Files:** Translated Braille can be exported as standard digital files, such as .brf and .brl.
* **Voice Output:** The system will use voice assistance to read the translated text aloud.

1. **User Interface & Key Features:**

The application will feature a **user-friendly and accessible interface**.

Key features will include:

* **AI-driven OCR** for high-accuracy document recognition.
* **Extensible support** for future multilingual functionality.
* **Accessibility features** like simple interface navigation and voice assistance.
* **Adjustable settings** for customization, including contraction levels and formatting rules.

**1.1.3. Glossary**

|  |  |  |
| --- | --- | --- |
| 1. | AI | Artificial Intelligence |
| 2. | API | Application Programming Interface |
| 3. | CNN | Convolutional Neural Network |
| 4. | GUI | Graphical User Interface |
| 5. | HTTPs | Hyper-Text Transfer Protocol secure |
| 6. | ML | Machine Learning |
| 7. | NLP | Natural Language Processing |
| 8. | OCR | Optical Character Recognition |
| 9. | SPA | Single Page Application |
| 10. | PyTorch | An open-source machine learning framework used for deep learning models. |
| 11. | spaCy | An open-source library for advanced Natural Language Processing. |
| 12. | Braille | A tactile writing system used by people who are blind or visually impaired, consisting of raised dots that can be read with the fingers. |
| 13. | Grade 1 Braille | A form of Braille that provides a letter-for-letter transcription of the alphabet without contractions. |
| 14. | Grade 2 Braille | A more advanced form of Braille that uses contractions and abbreviations to save space and increase reading speed. |

**1.1.4. References**

1) A. Moustafa et al., "Empowering Accessibility: A Braille Translator App with Enhanced User Experience and Integration of Machine Learning Technologies*.*”[**Base Research Paper]**

2) J. Aswini et al., "Translation System for the Visually Impaired from English to Braille," in *2024 2nd World Conference on Communication & Computing (WCONF)*, Raipur, India, Jul. 12-14, 2024, pp. 1-6. doi: 10.1109/WCONF61366.2024.10692175.

3) F. S. Apu et al., "Text and Voice to Braille Translator for Blind People," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, Jul. 8-9, 2021, pp. 1-6. doi: 10.1109/ACMI53472.2021.9485899.

5) Braille Blaster: A downloadable software used to translate text into American Braille Script. – <https://www.brailleblaster.org/>

6) Braille Translator: An online translator used as a reference to identify areas for improvement in available translation tools. — <https://www.brailletranslator.org/>

7) India's Braille literacy crisis: A Deccan Herald article used to support the project's problem statement.-<https://www.deccanherald.com/opinion/indias-braille-literacy-crisis-3339748?utm_source=whatsapp&utm_medium=referral&utm_campaign=socialshare>

**1.1.5. Overview of Document**

This Software Requirements Specification (SRS) is organized into three main sections to provide a comprehensive and detailed description of the system's requirements:

* **Section 1: Introduction** provides the purpose, scope, glossary, references, and an overview of the document itself.
* **Section 2: Overall Description** details the system's environment, user characteristics, and a high-level overview of functional and non-functional requirements.
* **Section 3: Requirements Specification** delves into detailed external interfaces, specific functional requirements, and detailed non-functional requirements.

**1.2.0. Overall Description**

The Braille Translation and Accessibility Application is a web-based platform designed to enhance digital inclusivity for the visually impaired. It leverages AI and NLP to provide seamless, bidirectional translation between digital text, spoken English, and Braille.

**1.2.1 System Environment**

The **Braille Translation** and **Accessibility Application** will operate within a **web-based, cross-platform** environment, accessible via modern desktop browsers.

* **Frontend:** The frontend will be a web-based, intuitive, and accessible graphical user interface (GUI) designed to be cross-platform compatible on desktop browsers (Windows, macOS, Linux).
* **Backend:** The backend will consist of Python services hosted on a remote server. It will manage the core functionalities, including Braille translation, optical character recognition (OCR), and speech services.
* **Database:** A database is not strictly necessary for the core functionality but could be included in the future for storing user profiles, saved documents, or a library of accessible content.
* **Programming Language:** Python.
* **AI/ML Fields:** The application will utilize Artificial Intelligence (AI) and Machine Learning (ML) techniques, specifically in the areas of Optical Character Recognition (OCR) using deep learning models (CNN) with libraries like TensorFlow or PyTorch.
* **NLP:** Natural Language Processing (NLP) will be a key component, with libraries such as NLTK or spaCy used for translating English text to Grade 2 English Braille.
* **Deployment:** The current deployment is a local development environment. Future deployment is planned for a remote server, potentially on a cloud platform, to host the web application and its backend Python services.

**Key Libraries and APIs**

The following libraries and APIs will be used to implement the system's core features:

* **NLP:** NLTK or spaCy for Grade 2 English Braille translation.
* **OCR:** TensorFlow or PyTorch for training a deep learning model (CNN) to recognize text.
* **Speech Recognition:** Google's Speech-to-Text API or OpenAI's Whisper API.
* **Text-to-Speech:** Google's Text-to-Speech API or the gTTS Python library.

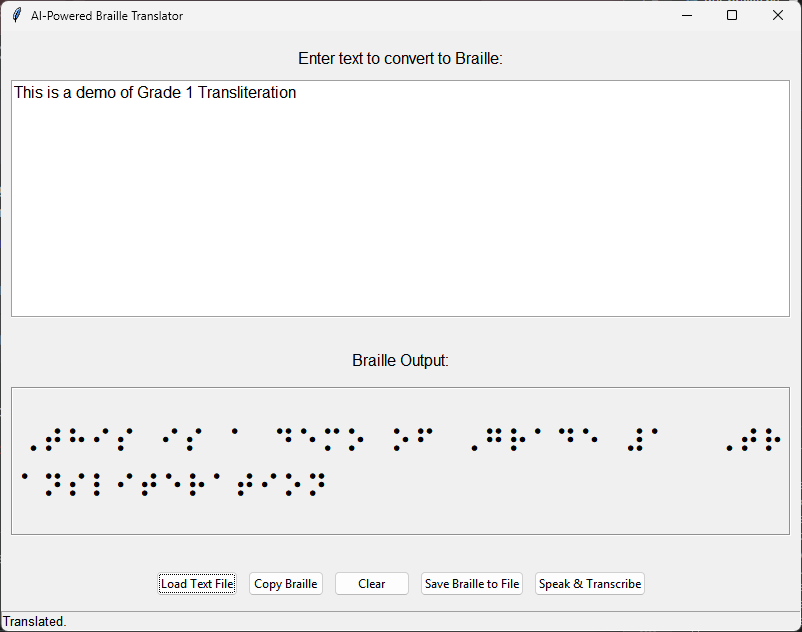


Figure 1.1. Demo of Grade 1 Translation using Python logic.

**1.2.2 Functional Requirements Specification**

The system performs bidirectional translation between text, speech, and braille in English.

**1.2.2.1 Core User Use Cases**

***Use Case: Convert Digital Text to Braille***

The system converts user-provided text from various sources into the selected Braille format.

A user begins by **selecting the Braille grade** and **inputting text** (via paste, file upload, or voice). The system then **processes the text**, applying the correct conversion rules (including contractions for Grade 2) → **displays the digital Braille output** → finally allows the user to **download the output** as a Braille-Ready Format (.brf) file.

***Use Case: Convert Braille Image to Text***

The system uses Optical Character Recognition (OCR) to analyze an image of Braille and convert it back into standard English text.

A user selects the **"Image-to-Text"** function → **uploads** or **captures** a Braille image → the system's **AI/ML model analyzes the image** to **identify** and **map** the **Braille dots to digital characters** → **converts** the digital Braille to standard English text → **displays the translated text** for review.

***Use Case: Convert Spoken Language to Braille***

This use case describes a system that listens to spoken words and automatically converts them to Braille, providing an easy way to create tactile text from voice.

A user selects the **"Voice-to-Braille"** option → **speaks** into a microphone → a **speech recognition API** captures the audio and **converts** it to a digital text string → this text is automatically passed to the **Text-to-Braille module** → the system **generates** and **displays** the corresponding Braille output.

***Use Case: Convert Braille-to-Speech***

An application converts an image of Braille into audible speech for the user.

The flow of events is as follows: The **user selects "Image-to-Speech"** and **uploads an image** containing Braille. This image is then processed by an **OCR model**, which converts the Braille to text. The digital text is sent to a **text-to-speech API** for audio conversion, and the system then **plays the audio** for the user.

***Use Case: Manage and Export Translations***

An application allows the user to save translated Braille content in various file formats and send it to external hardware.

After a successful translation, the **user selects a "Download" option** → a **dialog box appears** to choose the output format (.txt or .brf) → the **system generates and downloads the file** to the user’s device. The system also offers an option to **send the output directly to a connected braille display or embosser.**

***Use Case: Adjust Settings***

The user can customize their preferred language and Braille grade for future translations.

The **user navigates to the settings menu** → they **select their preferred language** (e.g., "English") → they **select the default braille grade** ("Grade 1" or "Grade 2") → the **system saves and applies these settings** to all subsequent translations until they are changed.

**1.2.3 User Characteristics**

The Braille and Speech Translation System is designed for a diverse user base with varying levels of technical proficiency and specific accessibility needs.

***Primary Users: Visually Impaired or Blind Individuals***

* **Description:** Individuals who have low vision.
* **Goals:** Independently read and write Braille, access content via speech, and create Braille documents.

***Secondary Users: Educators, Students, and Transcriptionists***

* **Description:** Sighted individuals and students who work with Braille content.
* **Goals:**
  + **Educators:** Efficiently generate accurate Braille materials for students.
  + **Students:** Practice and learn Braille in an interactive way.
  + **Transcriptionists:** Process large volumes of text into Grade 2 Braille with high accuracy and speed.

**1.2.4 User Needs:**

* **Accessible UI:** The interface must be fully compatible with screen readers and keyboard navigation.
* **Accurate Translation:** Users need high-accuracy translation between standard text and Grade 2 English Braille.
* **Efficient Input/Output:** The system should support multiple input methods (text, OCR, speech) and output options (Braille text, speech, file download).
* **Intuitive Design:** The application must be easy to use for individuals with varying levels of technical skill.
* **Educational Features:** The system should offer tools to support learning and practicing Braille for both students and educators.

**1.2.5 Non-Functional Requirements Specification**

This section details the quality attributes that the Braille and Speech Translation System must possess to meet user expectations and operational needs. These requirements are crucial for the system's success, focusing on how well it performs, rather than what it does.

**Accuracy:** The system's translation accuracy is critical. The AI/ML model for Grade 2 braille and the OCR model for braille-to-text conversion must maintain an accuracy rate greater than 95% to ensure reliable and useful translations.

**Performance:** All translation, recognition, and conversion tasks must be completed with minimal latency. The system should aim for near real-time performance to provide a smooth user experience, especially for voice-to-braille and braille-to-voice translations.

**Reliability:** The system must function consistently across supported devices and gracefully handle invalid inputs without crashing. It should have a high uptime to minimize disruption for users who depend on it.

**Usability:** The user interface will be intuitive and accessible to a wide range of users, including those who are visually impaired. The design will prioritize clear buttons, simple layouts, and support for voice commands.

**Portability:** The application architecture will be designed to allow for deployment on multiple platforms, including web, desktop, and mobile devices, without significant code changes. This maximizes the system's reach and accessibility.

**Scalability:** The system will be built with a modular architecture that allows for the addition of new languages or braille grades in the future. The core translation and recognition modules should be able to integrate new data sets with minimal modifications.

## **1.3.0. Requirements Specification**

This section defines the hardware, software, and communication interfaces that the **Braille Translation and Accessibility Application** will interact with.

### **1.3.1 External Interface Requirements**

**1.3.1.1 GUI**

The application will feature a web-based, accessible, and intuitive GUI designed for cross-platform compatibility on modern desktop browsers (Windows, macOS, Linux). The UI will be a single-page application (SPA) to ensure a fluid user experience without page reloads. It will include:

* + An input area for text and/or a file upload button.
  + A display area for the translated Braille text.
  + Buttons for key functionalities (e.g., "Translate to Braille," "Translate to Text," "Listen," "Download").
  + A status bar or message box to provide feedback on the translation process, errors, or warnings.

**Accessibility Features:** The UI will be designed to include:

* + Full keyboard navigation support for all interactive elements.
  + High color contrast and customizable font sizes for users with low vision.
  + Clear, descriptive labels and alt text for all controls and images.

**1.3.1.2. Hardware Interfaces**

The application will not require any specific hardware beyond a standard desktop computer with a keyboard, mouse, or other pointing device. It will also be compatible with:

* **Braille Displays:** The system will support interaction with standard refreshable Braille displays, although direct API integration may be limited by browser capabilities. The primary method of interaction will be for the user's screen reader to read the translated output from the web page.
* **Microphones:** A standard microphone will be required for the speech-to-text input functionality.
* **Speakers/Headphones:** Standard audio output devices are necessary for the text-to-speech functionality.

#### **1.3.1.3. Software Interfaces**

* **Operating Systems:** The web application will be accessible on modern desktop operating systems, including Windows, macOS, and Linux, without requiring a separate installation.
* **Web Browsers:** The application will be optimized for the latest versions of major web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
* **External APIs:** The backend services will integrate with the following third-party APIs:
  + **Speech Recognition:** Google's Speech-to-Text API or OpenAI's Whisper API.
  + **Text-to-Speech:** Google's Text-to-Speech API or the gTTS Python library.
* **Internal Libraries:** The backend will use standard Python libraries for core functionality:
  + **NLP:** NLTK or spaCy for Braille translation logic.
  + **OCR:** TensorFlow or PyTorch for training and running the CNN model.

#### **1.3.1.4. Communications Interfaces**

* **Client-Server Communication:** The frontend will communicate with the backend services using standard **HTTP/S protocols**. This will involve sending user requests and receiving data, such as translated text or status updates, typically in **JSON** format.
* **Internet Connectivity:** A stable and reliable internet connection is required for all functionalities that rely on the remote backend server and external APIs. The system will handle connection errors gracefully and provide appropriate user feedback.

### **1.3.2. Functional Requirements**

This section details the specific functions and behaviours the **Braille Translation and Accessibility Application** must perform to meet the needs of its users.

**1.3.2.1. User Authentication and Profile Management**

* **FR-AUTH-1:** The system shall provide a user preference page to display and manage user preferences, such as default Braille grade (Grade 1 or 2) for the current session.
* **FR-AUTH-2:** The system shall maintain user session state to allow users to save their translated documents or settings only for the current session.

#### **1.3.2.2. Braille Translation & Document Handling**

* **FR-TRANS-1:** The system shall translate standard English text into **Grade 1 and Grade 2 English Braille** upon user request.
* **FR-TRANS-2:** The system shall provide a text input area where users can manually type or paste text for translation.
* **FR-TRANS-3:** The system shall allow users to upload text-based files (e.g., .txt, .docx) for batch translation.
* **FR-TRANS-4:** The system shall allow users to download the translated Braille output as a standard text file or a Braille-Ready Format (.brf) file.

**1.3.2.3. AI-Powered Input Translation**

* **FR-AI-1 (OCR):** The system shall allow users to upload image files (.jpg, .png) and PDFs containing text.
* **FR-AI-2 (OCR):** Upon upload, the system shall use an Optical Character Recognition (OCR) model to accurately extract text from the uploaded documents.
* **FR-AI-3 (Speech-to-Text):** The system shall provide a button to activate microphone input, capturing a user's speech.
* **FR-AI-4 (Speech-to-Text):** The system shall convert the captured speech into a text string using an external Speech-to-Text API.
* **FR-AI-5:** The system shall display the extracted or converted text to the user for verification before performing the Braille translation.

#### **1.3.2.4. Accessibility and Output Features**

* **FR-ACC-1:** The system shall provide a text-to-speech function that reads the translated Braille output aloud to the user.
* **FR-ACC-2:** The system shall use an external Text-to-Speech API to convert the Braille output to natural-sounding audio.
* **FR-ACC-3:** The system shall provide visual feedback (e.g., loading indicators, progress bars) during all translation and API-driven processes.

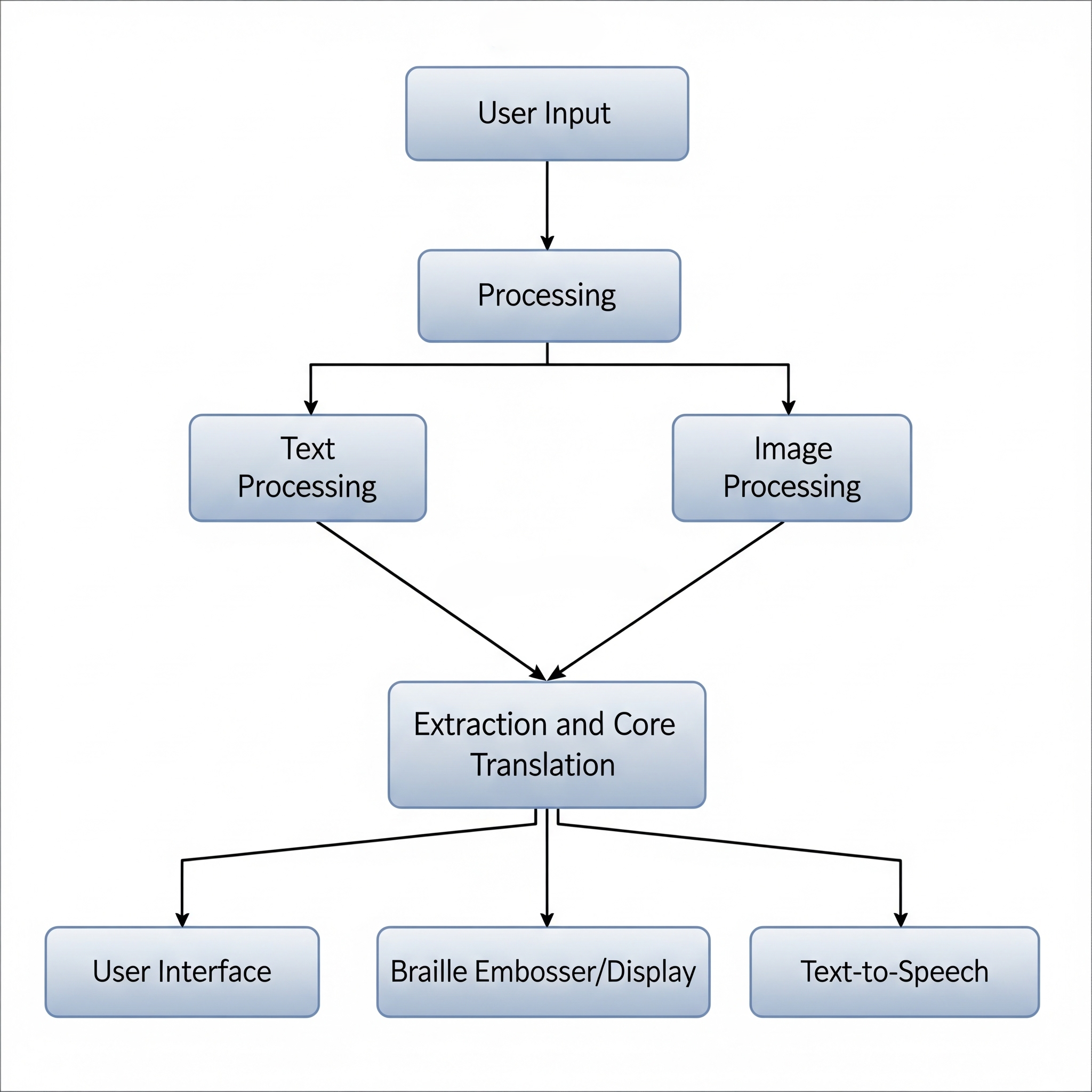


Figure 1.2: Primary operational flow of the Braille Translation

### **1.3.3. Detailed Non-Functional Requirements**

This section defines the quality attributes and constraints that the **Braille Translation and Accessibility Application** must meet, beyond its core functionality. These requirements are crucial for ensuring the system's overall performance, security, and usability.

#### **1.3.3.1. Performance Requirements**

* **NFR-PERF-1 (Latency):** The system shall translate and display text-to-Braille output within **minimal time** for a typical input of 500 characters.
* **NFR-PERF-2 (API Response):** The backend shall respond to API calls for OCR, Speech-to-Text, and Text-to-Speech for a file size up to 10MB.
* **NFR-PERF-3 (Scalability):** The system's architecture shall be scalable to support a **large number of users concurrently** without a noticeable degradation in performance.
* **NFR-PERF-4 (File Processing):** The system shall process and translate a 2+ page .docx or .txt file within a minimal period of time.

#### **1.3.3.2. Usability Requirements**

* **NFR-USABILITY-1 (Learnability):** A new user shall be able to perform a basic text-to-Braille translation with **no prior training**, using only the on-screen instructions.
* **NFR-USABILITY-2 (Efficiency):** An experienced user shall be able to complete a translation task (text input, Braille grade selection, and output).
* **NFR-USABILITY-3 (Accessibility):** As per the user interface requirements, the application must be fully accessible to users with disabilities, particularly those relying on screen readers and alternative navigation methods.

#### **1.3.3.3. Security Requirements**

* **NFR-SEC-1 (Authentication):** User passwords shall be securely stored using a one-way hashing algorithm with a strong salt (e.g., bcrypt) to prevent unauthorized access.
* **NFR-SEC-2 (Data Privacy):** The system shall not store or retain user-uploaded documents or translated text for longer than is necessary to complete the requested task, unless the user explicitly saves the content to their profile.
* **NFR-SEC-3 (Data Transmission):** All communication between the frontend and the backend shall be encrypted using **SSL/TLS (HTTPS)** to prevent eavesdropping and data tampering.
* **NFR-SEC-4 (API Security):** All third-party API calls shall be made from the secure backend to prevent the exposure of API keys on the client-side.

#### **1.3.3.4. Maintainability Requirements**

* **NFR-MAINT-1 (Codebase):** The codebase shall be well-documented and follow clean coding principles to facilitate future maintenance and feature additions.
* **NFR-MAINT-2 (Updates):** The system shall be designed to allow for seamless updates and patches to the Braille translation logic, OCR models, and other libraries without requiring a full system redeployment.

#### **1.3.3.5. Portability Requirements**

* **NFR-PORT-1 (Browser Compatibility):** The web application shall be cross-browser compatible and function correctly on the latest stable versions of Chrome, Firefox, Safari, and Edge.
* **NFR-PORT-2 (Platform Compatibility):** The application shall be accessible and fully functional on major desktop operating systems, including Windows, macOS, and Linux.

**Chapter 2: Architectural Design and Diagrams**

**Table of Contents**

2.1 Activity Diagram

2.2 Use Case Diagrams

2.2.1 Use Case: Convert Digital Text to Braille

2.2.2 Use Case: Convert Braille Image to Text

2.2.3 Use Case: Convert Spoken Language to Braille

2.2.4 Use Case: Convert Braille to Speech

2.2.5 Use Case: Manage and Export Translations

2.3 Sequence Diagram

2.4 Collaboration Diagram

**List of Figures**

2.1 Activity Diagram

2.2 Convert Digital Text to Braille

2.3 Convert Braille Image to Text

2.4 Convert Voice-to-Braille

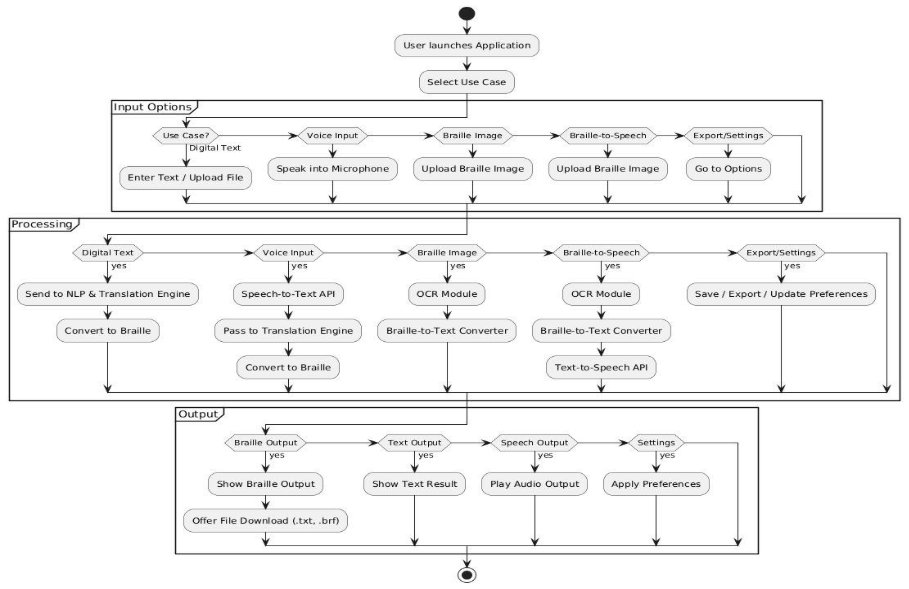
2.5 Convert Braille-to-Speech

2.6 Manage and Export Translations as (.txt) or (.brf) files.

2.7 Sequence Diagram

2.8 Collaboration Diagram

**2.1 Activity Diagram**



**Fig 2.1 Activity Diagram**

**Description:**

This activity diagram illustrates the complete workflow of the Braille Translation and Accessibility System, covering user interaction, processing, and output delivery. The process is divided into three sections: Input Options, Processing, and Output.

* **Initialization Phase**

The workflow begins when the *user launches the application* and selects a use case, which defines the type of input and desired operation.

* **Input Options**

The system supports multiple input modes:

* **Digital Text –** enter text or upload a file.
* **Voice Input –** speak into the microphone for audio capture.
* **Braille Image** – upload a Braille script for recognition.
* **Braille-to-Speech –** upload Braille for speech conversion.
* **Export/Settings –** manage preferences, save, or export.
* **Processing**

**Each input is directed to specific modules:**

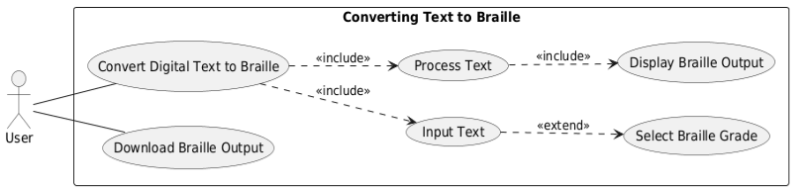
* **Digital Text** → *NLP & Translation Engine → Braille*.
* **Voice Input** → *Speech-to-Text API → Translation Engine → Braille*.
* **Braille Image** → *OCR Module → Braille-to-Text Converter*.
* **Braille-to-Speech** → *OCR → Braille-to-Text → Text-to-Speech API*.
* **Export/Settings** → *Save or update preferences*.

* **Output Options**

**The system provides:**

* **Braille Output** – view/download results (.txt, .brf).
* **Text Output** – display converted text.
* **Speech Output** – play audio results.
* **Settings** – apply preferences**.**
  1. **Use case diagram**

**2.2.1  Use Case : Convert Digital Text to Braille**



**Fig 2.2 Convert Digital Text to Braille**

**Description:**

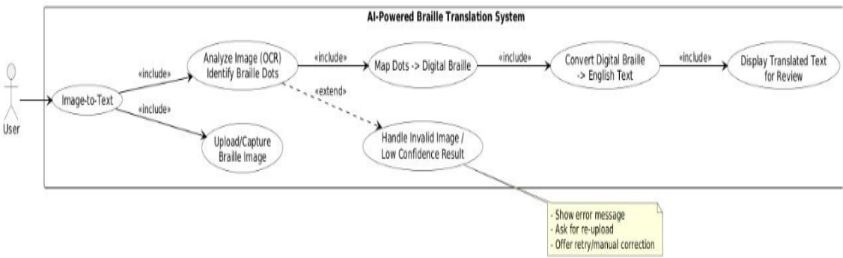
The Use Case Diagram illustrates how a user interacts with the AI-Powered Braille Translation System for the **Convert Digital Text to Braille** functionality.

The primary actor is the **User**, who initiates the process by selecting the **Convert Digital Text to    Braille** option.

**The system then performs a series of included use cases:**

1. **Input Text** – The user provides digital text that they want to convert into Braille.
2. **Select Braille Grade** – (Extended Use Case) The user can optionally select the Braille grade (Grade 1 or Grade 2) to customize the output format.
3. **Process Text** – The system processes the input text based on the selected Braille grade, converting it into Braille cells.
4. **Display Braille Output** – The resulting Braille output is displayed on the system interface for the user to review.
5. **Download Braille Output** – The user has the option to download the Braille output for offline use or printing.

**2.2.2  Use Case: Convert Braille Image to Text**



**Fig 2.3 Convert Braille Image to Tex**t

**Description:**

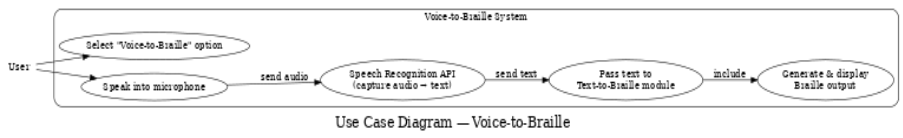
The Use Case Diagram illustrates how a user interacts with the AI-Powered Braille Translation System for the *Convert Braille Image to Text* functionality. The primary actor is the *User*, who initiates the process by selecting the Image-to-Text option.

**The system then performs a series of included use cases:**

1. **Upload/Capture Braille Image** – The user provides an image of Braille text.
2. **Analyse Image (OCR)** – The system applies OCR techniques to detect Braille dots.
3. **Map Dots** → Digital Braille – The identified dots are converted into digital Braille cells.
4. **Convert Digital Braille** → English Text – The system translates Braille into standard English text.
5. **Display Translated Text for Review** – The output is shown to the user for validation.

An extended use case is Handle Invalid Image / Low Confidence Result, which activates when the OCR model is unable to process the image with sufficient accuracy. In such cases, the system displays an error, asks the user to re-upload, or offers retry/manual correction options.

**2.2.3 Use Case: Convert Spoken Language to Braille**



**Fig 2.4 Convert Voice-to-Braille**

**Description:**

The Use Case Diagram presented above represents the functional interaction between the User and the Voice-to-Braille System. It demonstrates how the system processes user input (voice) and converts it into Braille output.  The *User* is the primary actor who interacts with the system by providing voice input and receiving Braille output.

**The system provides the following use cases:**

* 1. **Select "Voice-to-Braille" Option**

The User initiates the process by selecting the functionality from the system interface.

* 1. **Speak into Microphone**

The User provides audio input by speaking into the microphone.

* 1. **Speech Recognition API (Capture Audio → Text)**

The system converts the audio signal into digital text using speech recognition technology.

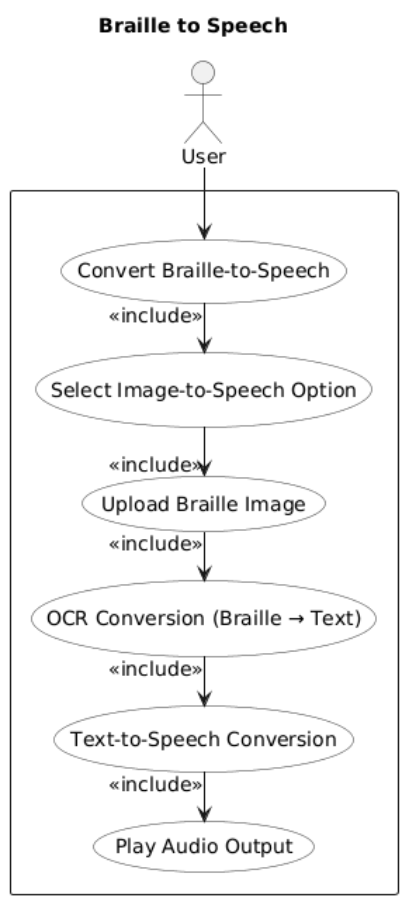
* 1. **Pass Text to Text-to-Braille Module**

The recognized text is sent to the Braille conversion module for further processing.

* 1. **Generate and Display Braille Output**

The final Braille representation of the spoken input is generated and displayed to the User.

**2.2.4 Use Case: Convert Braille-to-Speech**



**Fig. 2.5 Convert Braille-to-Speech**

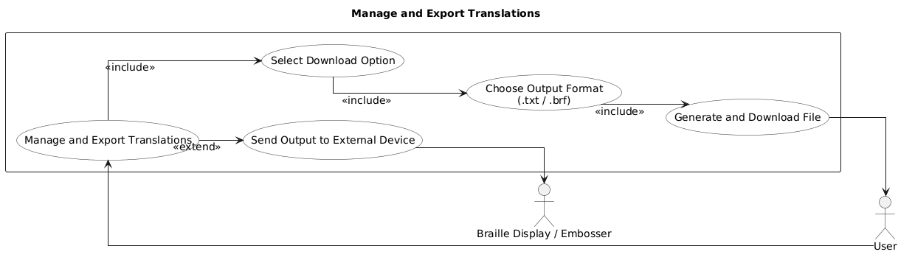
**Description:**

The *Braille-to-Speech* use case enables users to convert Braille text from an image into audible speech. The primary user interacts with the system by providing an image input, after which the system automatically processes and delivers spoken output. This use case is particularly useful for individuals who rely on auditory feedback to access Braille-based content.

**The system provides the following use cases:**

* 1. **Convert Braille-to-Speech** – The main use case that drives the process of transforming Braille into spoken language.
  2. **Select Image-to-Speech Option** – The user begins by choosing the feature to convert Braille images into speech.
  3. **Upload Braille Image** – The user uploads an image containing Braille text for processing.
  4. **OCR Conversion (Braille → Text)** – The system uses Optical Character Recognition to detect and translate Braille patterns into digital text.
  5. **Text-to-Speech Conversion** – The extracted text is converted into audible speech using a text-to-speech engine.
  6. **Play Audio Output** – The final speech output is delivered to the user as audio.

**2.2.5 Use Case: Manage and Export Translations**



**Fig. 2.6 Manage and Export Translations as (.txt) or (.brf) files.**

**Description:** 

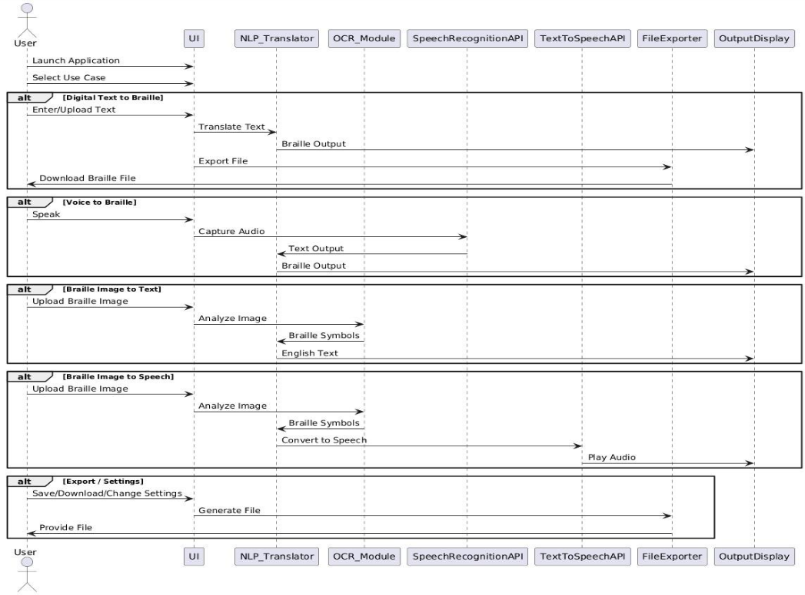
The *Manage and Export Translations* use case enables users to store and share translated Braille content in digital formats or through external devices. After completing a translation, the user can download the result as a file for offline use or send it directly to supported Braille hardware. This ensures flexibility for both digital archiving and real-world tactile reading.

User is the primary actor who manages and exports the Braille translation. The Braille Display / Embosser is an external device that optionally receives the exported Braille output.

**The system provides the following use cases:**

* 1. **Manage and Export Translations** – the overall process of saving or sharing the translated Braille content.
  2. **Select Download Option** – the user chooses to download the translated content.
  3. **Choose Output Format (.txt / .brf)** – the system presents available formats for the download.
  4. **Generate and Download File** – the system produces the file in the chosen format and saves it to the user’s device.
  5. **Send Output to External Device** – an optional step where the system transmits the output to a connected Braille display or embosser.

* 1. **Sequence diagram**



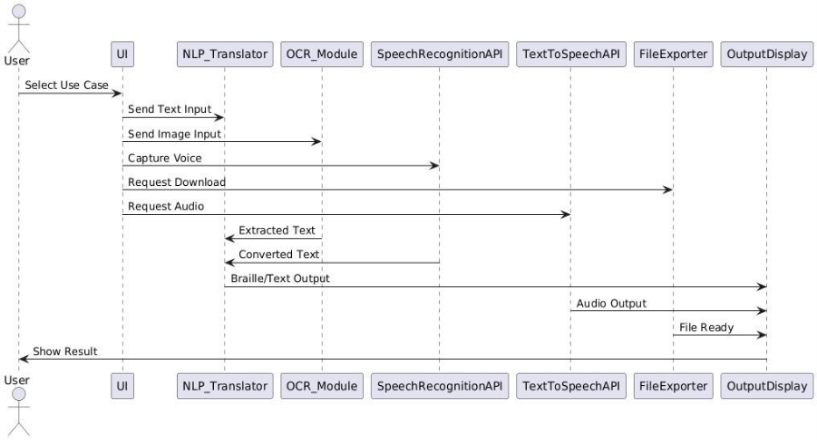
**Fig 2.7 Sequence Diagram**

**Description:**

1. Demonstrates the time-ordered sequence of interactions for each use case.
2. Covers all major system functions:
3. Digital text to Braille.
4. Voice input to Braille.
5. Braille image to text.
6. Braille image to speech.
7. Exporting translated files.
8. Adjusting user settings.
9. Shows how requests flow from the User → UI → Core Modules → Output Display.
10. Useful for visualizing the dynamic behaviour of the system and validating that implementation steps match the functional requirements.

Ensures that each user action is supported by a clear series of system responses.

* 1. **Collaboration diagram**



**Fig 2.8 Collaboration Diagram**

**Description:**

Depicts the interaction between system components and the user:- 

1. Highlights the role of the UI as the central point of communication with NLP Translator, OCR Module, Speech Recognition API, Text-to-Speech API, File Exporter, and Output Display.
2. Shows how different modules exchange data (e.g., OCR sending extracted text to NLP, Speech API sending converted text).
3. Helps identify dependencies and communication paths among modules.
4. Ensures modular design by clarifying how different components collaborate to deliver functionality.