TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

PASCHIMANCHAL CAMPUS
LAMACHAUR, POKHARA

SOFTWARE REQUIREMENT SPECIFICATIONS
on

FormalNet

BY

BIGYAN ARYAL [PAS079BCT009]

TO

Department of Computer and Electronics Engineering
POKHARA, NEPAL
July 19, 2025

Table of Contents

Chapter 1: Introduction 1
1.1 Background Theory 1
1.1.1 Natural Language Processing 1
1.1.2 Text Style Transfer 1
1.1.3 Transformer Architecture 1
1.1.4 T5 Model 1
1.1.5 Fine-Tuning Pretrained Models 1
1.1.6 Sequence-to-Sequence Learning 1
1.1.7 Dataset 2
1.2 Purpose 2
1.3 Scope 2
1.4 Overview 2
Chapter 2: Overall Description 3
2.1 Product Perspective 3
2.2 Product Functions 3
2.3 User Characteristics 3
2.4 Constraints 4
2.5 Assumptions and Dependencies 4
Chapter 3: Specific Requirements 5
3.1 Functional Requirements 5
3.2 Non-Functional Requirements 5
3.3 External Interface Requirements 5
3.3.1 User Interface 5
3.3.2 API Interface 6
Chapter 4: Software Architecture 7
4.1 Model Architecture 7
4.2 System Architecture 7
4.3 Training Pipeline 8
Chapter 5: Diagrams 9
5.1 ER Diagram 9
5.2 Dataflow Diagram 10
5.3 State Diagram 11
5.4 Use Case Diagram 12
Abbreviations 13

Chapter 1: Introduction

1.1 Background Theory

1.1.1 Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on the interaction between computers and human (natural) languages. Key tasks in NLP include tokenization, part-of-speech tagging, named entity recognition, machine translation, and text classification.

1.1.2 Text Style Transfer

Text style transfer is an NLP task that involves transforming text from one style or tone to another while preserving its original content. In the context of FormalNet, the goal is to convert informal or conversational text into a more formal version suitable for academic or professional communication.

1.1.3 Transformer Architecture

The Transformer architecture, introduced by Vaswani et al. (2017) [3], revolutionized NLP by eliminating recurrence and instead relying entirely on self-attention mechanisms to model relationships between words in a sequence. Transformers have become the foundation of many modern language models due to their scalability and performance on large-scale datasets.

1.1.4 T5 Model

The Text-to-Text Transfer Transformer (T5)[2] is a unified framework that treats every NLP problem as a text generation task. For example, instead of building separate models for translation, summarization, or classification, T5 reformulates all of them as feeding in text and generating target text.

1.1.5 Fine-Tuning Pretrained Models

Instead of training a model from scratch, Fine-Tuning involves taking a pretrained model (like T5-base) and training it further on a task-specific dataset. For FormalNet, the model was fine-tuned on a parallel corpus of informal and formal sentence pairs. This approach leverages learned linguistic patterns while specializing the model on domain-specific tasks.

1.1.6 Sequence-to-Sequence Learning

Sequence-to-sequence (Seq2Seq) learning is a framework for converting one sequence (e.g., informal sentence) into another (e.g., formal sentence). It typically involves an encoder to represent the input and a decoder to generate the output. The Transformer is a type of Seq2Seq model without recurrence, making it faster and more efficient than traditional RNN-based models.

1.1.7 Dataset

NUS Social Media Text Normalization and Translation Corpus [1] can be used for this case study. The corpus is created for social media text normalization and translation. It is built by randomly selecting 2,000 messages from the NUS English SMS corpus. The messages were first normalized into formal English and then translated into formal Chinese which we can ignore.

1.2 Purpose

The purpose of this project is to develop a system capable of transforming informal English text into a more formal and polished version, particularly suited for academic or professional settings.

1.3 Scope

FormalNet is designed as a web-based application that takes informal English text as input and returns a formalized version as output. The system incorporates a deep learning model based on the transformer architecture (T5) for text style transfer. The application also includes user authentication, prompt history tracking, and a clean user interface built with Django.

1.4 Overview

The development of FormalNet followed an iterative and incremental development cycle, blending elements of both traditional software engineering practices and modern machine learning workflows.

- **Requirements Analysis** The initiative commenced with the collection of requirements centered on the challenge of converting informal text to formal text.
- **Data Collection and Preprocessing** Created and cleaned a dataset of informal–formal sentence pairs, ensuring it was properly formatted for training a transformer model.
- **Model Training** Fine-tuned a T5-based transformer on the prepared dataset using supervised learning, optimizing for accuracy and generalization.
- **Web Application Development** Built the Django frontend for user interaction and a FastAPI backend to host the model, maintaining a modular and scalable design.
- **Integration and API Communication** Integrated the frontend and backend, enabling real-time requests from the web app to the ML model and returning formatted output.
- **Testing and Feedback** Conducted unit tests, model evaluations, and user trials to ensure correctness, usability, and linguistic quality of the system.

Chapter 2: Overall Description

2.1 Product Perspective

FormalNet is a standalone web-based system designed to provide intelligent text formalization through natural language processing (NLP). It operates as an end-to-end platform consisting of two decoupled services:

- A Django-based web interface for user interaction, input handling, and history management
- A FastAPI-based model backend that serves a fine-tuned T5 transformer model

The system is independent but modular, making it adaptable for integration with other writing tools or educational platforms in the future.

2.2 Product Functions

The core functionality of FormalNet is to accept informal English text and return a grammatically improved, more formal version. Major functions include:

- Accepting user input (informal text) through a web form.
- Processing and sending the input to a backend model API.
- Receiving and displaying the formalized output.
- Supporting user authentication and account management.
- Storing and displaying each user's text history with editing feature.

Each function is built to prioritize usability, speed, and linguistic accuracy.

2.3 User Characteristics

The target users of FormalNet are:

- Students looking to improve the formal quality of their assignments, emails, or research documents.
- Academic professionals drafting formal communication.
- General users seeking polished, professional wording.

Users are expected to have basic computer and internet literacy. No prior knowledge of NLP or machine learning is required to use the system.

2.4 Constraints

- The web application depends on a stable internet connection for communication between frontend and backend
- Response time should ideally remain under 3 seconds per conversion
- Security measures must be implemented to protect user data, especially in the prompt history feature

2.5 Assumptions and Dependencies

- The backend API and model service must be deployed and active for the Django app to function correctly
- The system assumes availability of a modern browser (Chrome, Firefox, etc.) for full compatibility

Chapter 3: Specific Requirements

3.1 Functional Requirements

The following are the primary functional requirements of the FormalNet system:

FR1. The system shall allow users to input an informal English sentence.

FR2. The system shall send the input sentence to the backend API for formalization.

FR3. The system shall return and display the formalized output to the user.

FR4. The system shall allow registered users to log in and log out securely.

FR5. The system shall store input-output history for logged-in users.

FR6. The system shall allow users to edit and re-submit previously entered prompts.

FR7. The system shall provide an admin panel for managing users and prompts.

3.2 Non-Functional Requirements

These are constraints on how the system performs:

NFR1. The system shall respond to input within 3 seconds under normal load.

NFR2. The backend shall be able to handle at least 10 concurrent users.

NFR3. The system shall be compatible with modern web browsers (Chrome, Firefox, Edge).

NFR4. The system shall maintain user data securely using hashed passwords.

NFR5. The model inference shall run on GPU if available, otherwise fallback to CPU.

NFR6. The application shall be deployed using a containerized architecture (e.g., Docker).

3.3 External Interface Requirements

3.3.1 User Interface

- The main page shall include a form to input informal text and display the formal output.
- The profile page shall show prompt history with options to edit or delete.
- The login/register pages shall include standard form fields with validation.

3.3.2 API Interface

- The Django app shall send a POST request to the FastAPI backend at `/predict` with JSON input.
- The backend shall respond with JSON containing the formalized text.
- API must return HTTP status codes for error handling (e.g., 200 OK, 400 Bad Request).

Chapter 4: Software Architecture

4.1 Model Architecture

FormalNet uses a T5 (Text-to-Text Transfer Transformer) model as its core engine for converting informal sentences into formal ones. T5 treats every NLP task as a text generation problem, making it suitable for style transfer tasks.
The model follows the typical encoder-decoder architecture:

- **Encoder:** Encodes the informal input sentence into a latent representation using self-attention.
- **Decoder:** Generates the formal version of the sentence, attending to the encoded input at each step.

We used the `t5-base` variant, which consists of:

- 12 Transformer layers each for encoder and decoder
- 768-dimensional hidden layers
- 12 attention heads
- 220 million parameters

The model was fine-tuned on a NUS Social Media Corpus[1] using supervised learning with teacher forcing.

4.2 System Architecture

FormalNet is implemented as a two-part modular system:

1. A **Django web application** that connects the frontend interface (using tailwind css), handles user authentication, and manages prompt history.
2. A **FastAPI model backend** that loads the T5 model and serves predictions via a RESTful API.

The system workflow is as follows:

1. User enters an informal sentence in the web form.
2. The Django server sends the input to the FastAPI backend via an HTTP POST request.
3. The FastAPI service runs inference using the finetuned T5 model and returns the formalized text.
4. The frontend displays the output and stores the input-output pair in the database (if the user is logged in).

4.3 Training Pipeline

The T5 model was fine-tuned on a custom dataset containing informal and formal sentence pairs. The training pipeline included:

- **Data preprocessing:** Text normalization, tokenization using T5 tokenizer, and input formatting.
- **Model loading:** Initialized from `t5-base` pretrained weights.
- **Training loop:** Used teacher forcing, AdamW optimizer, and early stopping to prevent overfitting.
- **Evaluation:** Model was evaluated using BLEU score and manual quality inspection.

Training was performed using PyTorch on a P100 GPU provided by Kaggle.

Chapter 5: Diagrams

5.1 ER Diagram

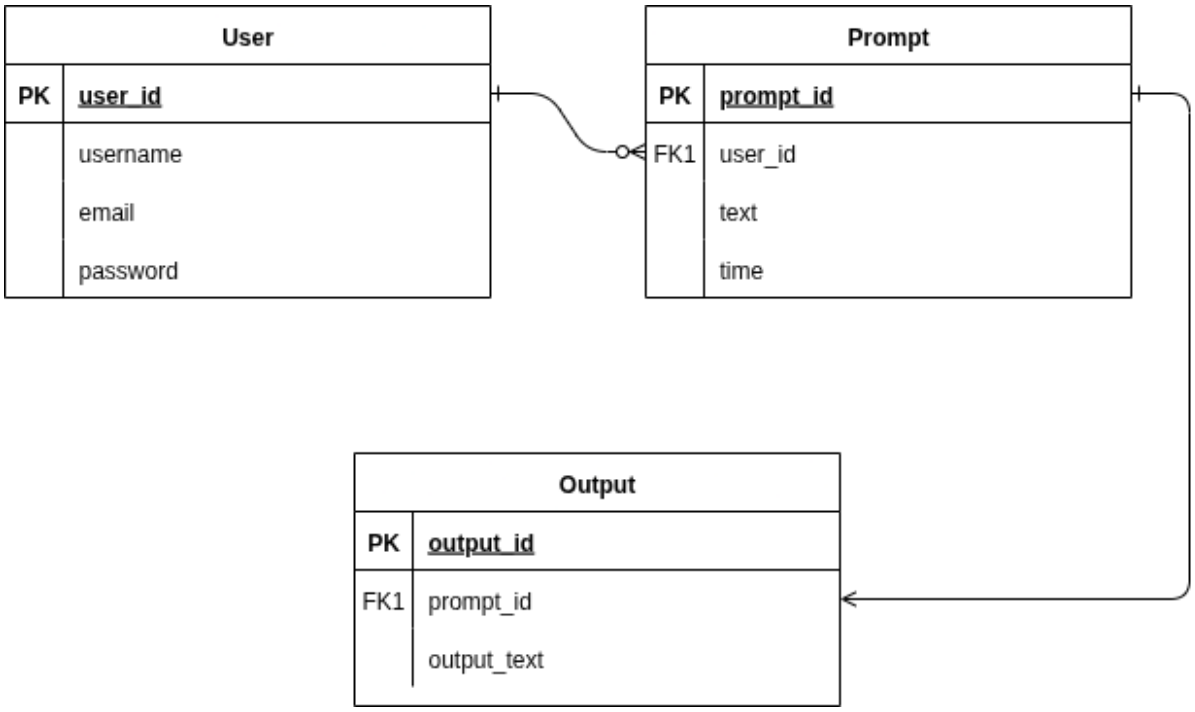

Figure 5.1: Entity-Relationship Diagram of FormalNet

The Entity Relationship Diagram represents the data model of the FormalNet system. It defines the structure of the database and the relationships between various entities.
In the FormalNet system:

- A User can submit multiple Prompts, forming a one-to-many relationship.
- Each Prompt is processed by the model, and the corresponding FormalOutput is generated and stored, establishing a one-to-one relationship between Prompt and Output.
- Foreign keys maintain referential integrity between the tables

This diagram helps in designing an efficient database schema and serves as a foundation for backend development and query optimization.

5.2 Dataflow Diagram

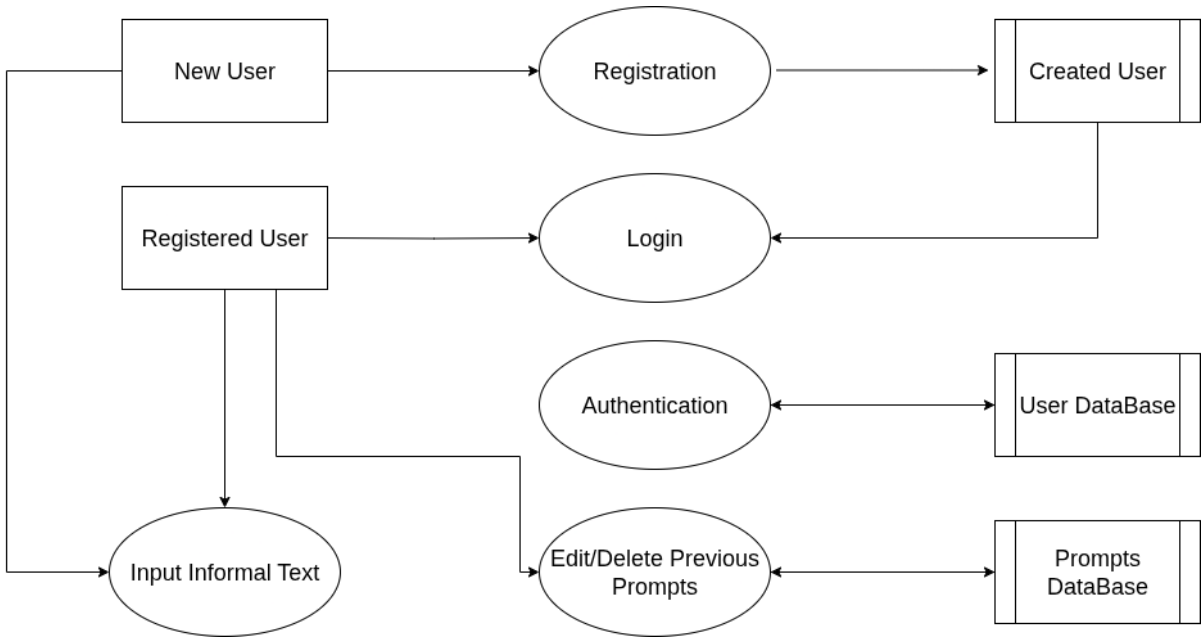

Figure 5.2: Dataflow Diagram of FormalNet

The data flow diagram (DFD) of the FormalNet system illustrates the movement of data between users, processes, and data stores. It shows how new users register and have their information stored, while registered users can log in and be authenticated using the user database. After authentication, users can input informal text for processing or manage their previous prompts. These actions interact with dedicated databases for storing user and prompt data. The diagram ensures a clear overview of how data flows through the system in a structured manner.

5.3 State Diagram

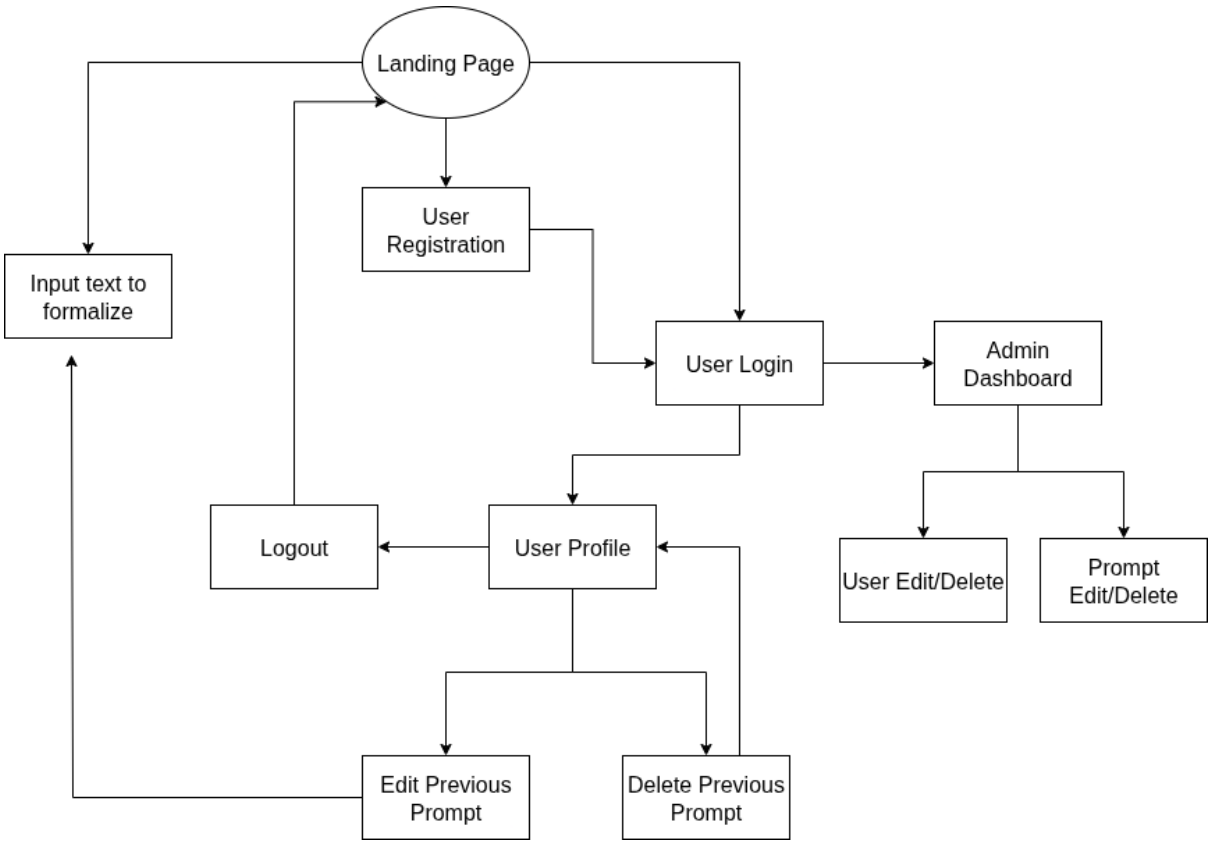

Figure 5.3: State Diagram of FormalNet

The state diagram of the FormalNet system represents the various states a user or the system can be in during interaction. It begins from the initial state where the user accesses the platform, followed by transitions through states like registration, login, authentication, and text input. Depending on user actions, the system can move to states such as prompt editing or logout. This diagram helps visualize the dynamic behavior of the system and how it responds to different user inputs and transitions.

5.4 Use Case Diagram

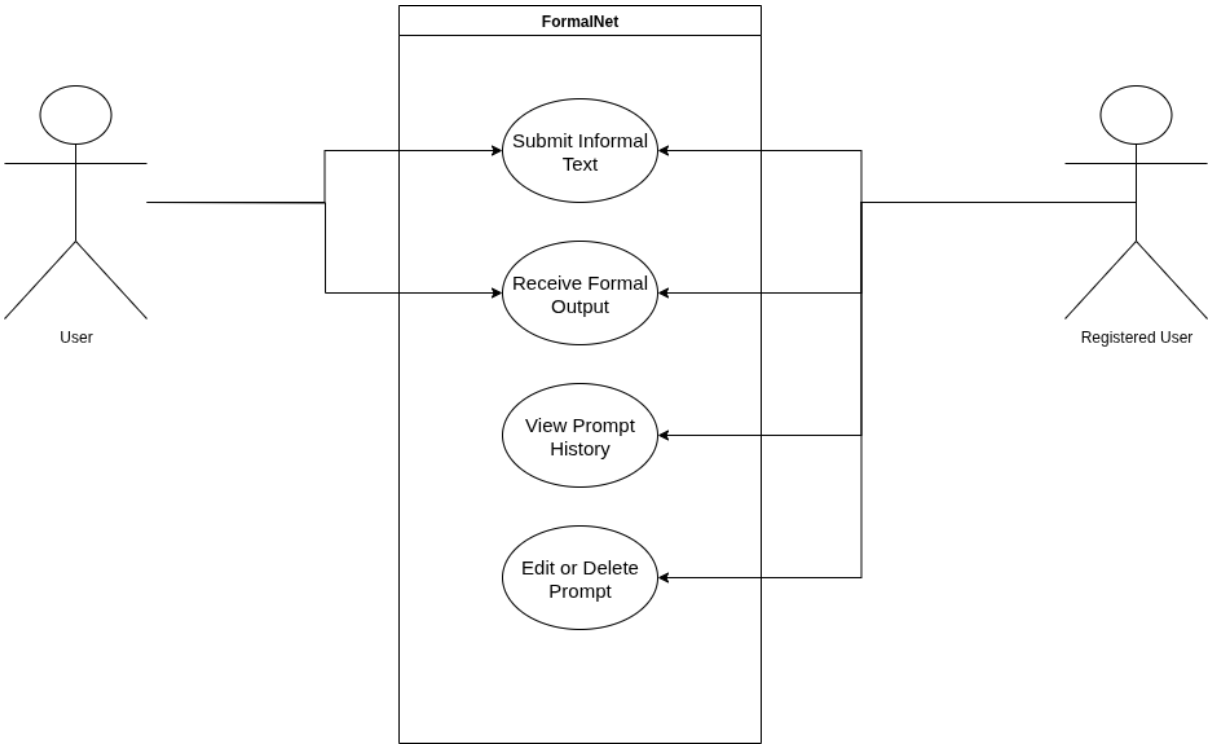

Figure 5.4: Use Case Diagram of FormalNet

The use case diagram provides a high-level visual representation of the interactions between users (actors) and the system (FormalNet). It outlines the core functionalities that the system offers and the roles that interact with them. This helps in identifying system boundaries and user goals in a clear and concise manner.
There are two main actors in the system:

- **New User**: A user who accesses the system without registration.
- **Registered User**: A user who has an account and can access extended functionalities.

The primary use cases shown in the diagram include:

- **Submit Informal Text**: Allows users to input informal text into the system.
- **Receive Formal Output**: The system processes the input and returns a formal version of the text.
- **View Prompt History**: Registered users can view their previous inputs and outputs.
- **Edit or Delete Prompt**: Enables users to manage their past prompts for corrections or updates.

This diagram helps in understanding the user interaction flow and serves as a blueprint for system behavior from the user's perspective.

Abbreviations

FormalNet The name of the application platform

SRS Software Requirements Specification

SDLC Software Development Life Cycle

NLP Natural Language Processing

Seq2Seq Sequence-to-sequence

RNN Recurrent Neural Network

NUS National University of Singapore

ML Machine Learning

GPU Graphical Processing Unit

CPU Central Processing Unit

T5 Text-to-Text Transfer Transformer

BLEU Bilingual Evaluation Understudy

DFD Data Flow Diagram

ER Entity Relation

API Application Programming Interface

Bibliography

[1] Prof. Siew Mei Wu Prof. Hwee Tou Ng. Nus social media text normalization and translation corpus, 2009. Accessed: 2025-07-14.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, volume 21, pages 1–67, 2020.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.