Urdu Faham

Session: 2021 – 2025

Submitted by:

Mizna Rauf 2021-CS-174

Ifrah Muzahir 2021-CS-213

Faiza Nazakat 2021-CS-182

Yasir Baig 2021-CS-215

Supervised by:

Dr Aslam

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Urdu Faham

Submitted to the faculty of the Computer Science Department of the University of Engineering and Technology Lahore in partial fulfillment of the requirements for the Degree of

Bachelor of Science

in

Computer Science.

Internal Examiner	**External Examiner**
Signature:	Signature:
Name:	Name:
Designation:	Designation:
Chairman	**Dean**
Signature:	Signature:
Prof. Dr. M. Usman Ghani Khan	Prof. Dr. M. Shoaib

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Declaration

We declare that the work contained in this thesis is my ours, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Name: _____
Signature: _____
Date: _____

Name: _____
Signature: _____
Date: _____

Name: _____
Signature: _____
Date: _____

Name: _____
Signature: _____
Date: _____

Acknowledgments

We deeply thank Almighty Allah for granting us strength, perseverance, and guidance throughout this project.

We want to express our heartfelt thanks to our supervisor, Dr. Aslam, and Co-supervisor Dr. Sana Shams, for their invaluable support, encouragement, and expert guidance throughout the research and development of our Project. Their insights, constructive feedback, and continuous motivation helped us overcome challenges and stay focused.

We also thank the faculty and staff of the Department of Computer Science, University of Engineering and Technology, Lahore, for providing the academic environment and facilities necessary for this work.

A special thanks to our families for their continuous love, encouragement, and prayers. Their support helped us stay focused and motivated throughout the journey.

We also appreciate the teamwork and cooperation within our group, which played a big role in completing this project successfully.

Finally, we are thankful for all the learning resources, tools, and technologies that helped us build and test our Urdu Translator for Educational Content.

Dedicated to our loving families, whose unwavering support, endless prayers, and constant encouragement have been the foundation of our journey. This work is also dedicated to all those who strive to make the world a safer place through innovation, compassion, and resilience.

Contents

Acknowledgments iii

List of Figures ix

Abbreviations x

Abstract xi

1 Introduction 1
- 1.1 Problem Statement . . . 3
  - 1.1.1 Language Difference Between English and Urdu . . . 3
  - 1.1.2 Contextual Inaccuracy . . . 3
  - 1.1.3 Scarcity of High-Quality Parallel Corpora . . . 4
  - 1.1.4 Accessibility Barriers in Low-Resource . . . 4
  - 1.1.5 Limited Resources of Learning . . . 4
- 1.2 Objectives . . . 4
  - 1.2.1 Develop an AI-Driven English-to-Urdu Translation System . . . 5
  - 1.2.2 Enable User-Friendly Content Upload and Translation Processing . . . 5
  - 1.2.3 Fine-Tune a Domain-Specific English-Urdu Parallel Corpus . . . 6
  - 1.2.4 Integrate an Interactive Quiz Generation and Testing Module 6
  - 1.2.5 Implement an Admin Panel for Content Review . . . 6
- 1.3 Scope and limitations . . . 7
  - 1.3.1 Scope . . . 7
  - 1.3.2 Limitations . . . 8
- 1.4 Methodology Overview . . . 9
- 1.5 Report Organization . . . 10

2 Software Requirement Specifications (SRS) 11
- 2.1 Introduction . . . 11
  - 2.1.1 Purpose . . . 11
  - 2.1.2 Intended Audience . . . 11
  - 2.1.3 Definitions Acronyms . . . 12
- 2.2 Overall Description . . . 13
  - 2.2.1 Product Perspective . . . 13
  - 2.2.2 User Characteristics . . . 13

2.2.3 Assumptions and Dependencies . . . 13
2.2.3.1 Assumptions . . . 14
2.2.3.2 Dependencies . . . 14
2.3 Specific Requirements . . . 15
2.3.1 Functional Requirements . . . 15
2.3.2 Non-functional Requirements . . . 16
2.4 Interface Requirements . . . 17
2.4.1 User Interface . . . 17
2.4.2 API Interfaces . . . 17

3 Literature Review 19

4 System Design 23
4.1 System Architecture . . . 23
4.2 Use Case Diagrams . . . 24
4.3 ERD Diagram . . . 25
4.4 Class Diagram . . . 27
4.5 Sequence Diagrams . . . 28
4.6 Deployment Diagrams . . . 29
4.7 Component Diagrams . . . 30
4.8 Data Flow Diagram (DFD) Diagrams . . . 31
4.9 Wireframes . . . 33
4.10 Database Design . . . 38

5 Implementation 41
5.1 Development Tools and Technologies . . . 41
5.1.1 Frontend Development . . . 41
5.1.2 Backend Development . . . 42
5.1.3 Machine Learning Tools . . . 42
5.1.4 Storage Solutions . . . 42
5.1.5 Development and Collaboration Tools . . . 43
5.1.6 Integrated Workflow . . . 43
5.2 Description of Modules . . . 43
5.3 Key Implementation Details . . . 45
5.4 Sample Screenshots . . . 46

6 Testing 53
6.1 Testing Strategy . . . 53
6.1.1 Unit Testing . . . 53
6.1.2 Integration Testing . . . 53
6.1.3 System Testing . . . 54
6.1.4 User Acceptance Testing (UAT) . . . 54
6.2 Test Cases and Results . . . 54
6.2.1 Test Case 1: Successful User Registration . . . 54
6.2.2 Test Case 2: Video Upload and Translation Request . . . 55

6.2.3 Test Case 3: Login with Incorrect Credentials 55
6.2.4 Test Case 4: Take Quiz and Submit Feedback 55
6.2.5 Test Case 5: Review and Approve Video (Admin) 56
6.2.6 Test Case 6: Edit Profile 56
6.2.7 Test Case 7: Translation Accuracy (Machine Learning Model) 56
6.3 Bug Fixes / Issues Tracker 56
6.3.1 Bug Tracking Process 57
6.3.2 Hypothetical Issues 57
6.4 Tools Used 58

7 Deployment and User Manual 59
7.1 Deployment Steps 59
7.1.1 Hosting 59
7.1.2 Platforms 59
7.1.3 Dependencies 60
7.2 User Instructions 60
7.2.1 Login 60
7.2.2 Navigation 61
7.2.3 Usage 61
7.3 Maintenance Guidelines 62
7.3.1 System Updates 62
7.3.2 Bug Fixes 62
7.3.3 Resource Management 63
7.3.4 Documentation 63

8 Results, Evaluation and Conclusion 64
8.1 Summary of Achievements 64
8.1.1 AI-Based Translation Engine 64
8.1.2 Interactive Translation Assistant 64
8.1.3 Seamless Integration 65
8.1.4 Quiz Generation and Translation 65
8.1.5 Robust Offline Functionality 65
8.2 Challenges Faced 65
8.2.1 Data Availability and Quality 66
8.2.2 Model Optimization for Urdu's Linguistic Features 66
8.2.3 Integration of Video Subtitle Translation 66
8.2.4 User Interface Design 66
8.3 Lessons Learned 67
8.3.1 Importance of Data Quality and Diversity 67
8.3.2 User-Centered Design 67
8.3.3 Offline Functionality is Crucial 67
8.3.4 importance of scalability 67
8.3.5 Cross-Disciplinary Collaboration 67
8.4 Future Enhancements 68

8.4.1 Improved Machine Learning Models 68
8.4.2 Expanded Language Support 68
8.4.3 Enhanced User Support Features 68
8.4.4 Integration of Real-Time Collaboration 68
8.4.5 Further Optimization for Offline Functionality 69
8.4.6 Push Notifications for User Engagement 69
8.5 Conclusion 69

A Proposal Copy **73**

B Plagiarism Report **77**

C Extra Code Snippets **78**

D GitHub Repository Link **84**

List of Figures

4.1 System Architecture Diagram . . . 23
4.2 The Use Case Diagram . . . 24
4.3 The ERD Diagram . . . 26
4.4 The Class Diagram . . . 27
4.5 The Sequence Diagram . . . 28
4.6 The Deployment Diagram . . . 29
4.7 The Component Diagram . . . 30
4.8 The Data Flow Diagram . . . 31
4.9 The Data Flow Diagram . . . 32
4.10 SignUp Diagram . . . 33
4.11 FAQ Diagram . . . 33
4.12 Dashboard Diagram . . . 34
4.13 Home Page . . . 34
4.14 Frequently Asked Questions . . . 35
4.15 Footer Page . . . 35
4.16 Contact Us . . . 36
4.17 Search Page . . . 37
4.18 Database Design . . . 38

5.1 User Account Diagram . . . 46
5.2 User Profile Diagram . . . 47
5.3 User Profile Settings . . . 47
5.4 User Profile Dashboard . . . 48
5.5 Upload Video Diagram . . . 48
5.6 Subject Video Diagram . . . 48
5.7 View Diagram . . . 49
5.8 Home page Diagram . . . 49
5.9 Blog Diagram . . . 49
5.10 Contact Page Diagram . . . 50
5.11 FAQ Page Diagram . . . 50
5.12 About Us . . . 50
5.13 Question Page Diagram . . . 51
5.14 Privacy Page Diagram . . . 51
5.15 TOS Diagram . . . 52

Abbreviations

AI	**A**rtifical **I**ntelligence
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
LLM	**L**arge **L**anguage **M**odel
NMT	**N**eural **M**achine **T**ranslation
SVO	**S**ubject **V**erb **O**bject
SOV	**S**ubject **O**bject **V**erb
BLEU	**B**ilingual **E**valuation **U**nderstudy
API	**A**pplication **P**Programming **I**nterface
GUI	**G**raphical **U**ser **I**nterface
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
MT	**M**achine **T**ranslation

Abstract

In countries like Pakistan, where more than one language is spoken, such as English and Urdu, reaching a high standard of academic material faces the challenge of the language barrier. Most of the academic material is in English, and this deprives many talented students who are proficient in Urdu of the opportunity. Thus, this Final Year Project would open this language gap to develop an AI-generated English-to-Urdu translator **Urdu Faham** for academic content. It is advanced with Neural Machine Translation (NMT), Natural Language Processing (NLP), and a tuned Large Language Model **LLaMA 3.1 8B**, to provide contextually accurate translations.

For this project, we manually created a parallel corpus using web scraping combined with input of academic content from websites and PDF documents. The data was cleaned and finely tuned for the model to ensure high fluency, grammatical accuracy, and semantic correctness. It is capable of text translation and generating subtitles for video lectures, apart from generating quizzes to assess students' understanding. A web-based interface was developed with React.js for the front end, while back-end services were developed with Python-based frameworks, as well as cloud storage and database-based solutions to improve scaling and efficiency.

In the end, this project will become an end-to-end learning appliance that will allow students access to scholarly materials in Urdu. This would also allow educators to localise the materials and reach wider audiences. This is an inclusive solution for computer literacy for the low-resource language population.

Chapter 1

Introduction

With information access as a fundamental right in the present digital age, education remains the strongest weapon for individual and societal transformation. The rapid proliferation of online learning platforms, open educational resources, and international academic content has opened up unparalleled opportunities for learners around the world. However, these opportunities are not equally distributed. Language continues to remain a much-strengthening hindrance resisting the full democratization of education. UNESCO has stated that English is by far the most widely used lingua franca in academia and on the internet. On the contrary, a very considerable portion of the global population– especially in developing countries like Pakistan–is facing exclusion due to limited English proficiency. This linguistic divide operates not only to disallow millions from accessing quality educational content, thus further entrenching socio-economic inequalities. Therefore, this has led to an even higher demand for translation tools that are effective, culturally sensitive, and linguistically appropriate.

In this global narrative, Pakistan presents a case repertoire. Whereas the English language enjoys a wide status in matters official, academic, and business, it is, at the same time, beyond the reach of a majority of Pakistanis, especially students from rural or underprivileged backgrounds. Urdu, which is national in status and minimum throughout Pakistan, serves as the bridge linking modern knowledge with local expression. The online content in education is increasing, albeit with a marked dearth in high-quality, contextually appropriate Urdu translations, especially in academic and technical domains. Further

complicating the situation are already existing so-called "translation services," which are more generic and context-insensitive and thereby lacking in preserving the pedagogical character of original content. Most of these services and tools, including the popular ones Google Translate or Microsoft Translator, are not responsive for the purposes of education and lose the various domain-related nuances that the purpose dictates. In their present state, this means providing translations that are literally correct but mislead or are grammatically faulty, confirming their inadequacy in educational use.

Identifying and bridging these gaps has been the concern of the proposed project aiming to develop an AI-powered English-to-Urdu translator for education content. The project, therefore, aims to bring together some of the most advanced machine learning techniques, including Natural Language Processing (NLP), Neural Machine Translation (NMT) and Large Language Models (LLMs), to bridge this major linguistic divide in academia. In recent years, these technologies have transformed the field of language translation by allowing machines to understand contextual meanings of words. To achieve domain-specific translations of high accuracy for Urdu, we make use of the LLaMA 3.1–8B, a model for large language models well known for its scalability and contextual understanding and that has been fine tuned on domain specific educational data..

All translations produced by the system will be subjected to rigorous quality assurance and contextual accuracy checks, the implementation of which will rely on cutting-edge NLP technologies, including transformer-based and fine-tuned models as well as Large Language Models (LLMs). This project goes beyond the very limited approaches of conventional rule-based or statistical-based translation treatment using machine-learning algorithms able to deepen understand and therefore grasp the structure and semantics of both the source (English) and the target (Urdu) languages. The NMT models will be contextually aware, meaning that the entire sentence or paragraph will be taken into account while selecting an appropriate Urdu translation.

Another project hallmark is the support for different content types. Users upload an educational video for audio extraction, which then turned into text with the aid of speech recognition; the translated text into Urdu is then subtitled. The quiz generator is also embedded into

the system to promote learning and build reinforcement with the translated content. The introduction of the admin panel provides facilities for content review, moderation, and management, ensuring rigorous quality control.

1.1 Problem Statement

Though there have come online learning resources in a very broad accessibility platform, yet the serious barrier against any English non- native is there, especially an Urdu speaker who cannot understand course material because of the language problem. Conventional translation systems, such as Google Translate, do not work very well for learning materials from any perspective-they become vague, context specific, and not culturally appropriate. Now, there arises critical need for a system that can best develop English to Urdu translation of educational content like video lectures, text, quizzes, etc. It will benefit users through ease of learning, which is made more inclusive for the local user.

1.1.1 Language Difference Between English and Urdu

There is a major problem for straightforward translation due to the way the languages are built and spoken. Sentences in English have the subject, verb and object in this order: SVO, but Urdu uses the sequence SOV. Because of this mismatch, translations are usually imperfect or hard to understand when they are processed with ordinary computer systems. As an example, direct translations can often put verbs in the wrong place and skip over Urdu's flexible word order which is unusual for native speakers. Urdu's requirement that words change for gender, number and case makes translating English terms into Urdu difficult. Because these details are difficult for tools to recognize, the results are often harder for people to understand.

1.1.2 Contextual Inaccuracy

Books, videos and quizzes in education need to be translated so that their original message is not altered. A lot of generic translation programs deliver translations that resemble a word-for-word copy which prevents them from including proper terminology for subjects like science, math or humanities. Technical vocabulary in STEM subjects does not often have an Urdu equivalent and idiomatic phrases in English might be misunderstood when expressed in Urdu, reducing the usefulness of what is taught. Because translations are not complete, the accuracy of strange language is reduced and it becomes difficult for students to get the fine details.

1.1.3 Scarcity of High-Quality Parallel Corpora

Training a successful NMT system on English and Urdu requires the use of robust, specialized texts linked between both languages. Nevertheless, resources for Urdu in NLP are lacking, including parallel datasets made for education. Many existing corpora are made up of media posts and typical news, rather than academic content. The difficulty in tuning the LLaMA 3.1 8B model for education because of its scarcity continuing to make it hard to produce fluent and accurate educational translations.

1.1.4 Accessibility Barriers in Low-Resource

A great deal of Urdu-speaking students from rural Pakistan encounter problems with internet and resources that prevent them from using online educational tools. For translation systems to work well in real time, they need stable internet access. Most remote areas lack this, so the systems are not very useful there. Also, using state-of-the-art translation models usually requires advanced hardware, including GPUs which are mostly absent in low-resource schools. Such challenges widen the digital gap, so even when translation is available, students cannot benefit.

1.1.5 Limited Resources of Learning

To communicate information, education content appears in formats ranging from video lectures to text documents and quizzes and each format requires a unique process. Video lectures must convert speech to text, generate subtitles and synchronize everything for accessibility, as quizzes must ensure questions are translated accurately to keep things clear and fair. Often, translation tools from generic companies lack the ability to handle Wideo's formats and often miss out on audio, subtitles or assessment tools. Therefore, Urdu-speaking students are not able to use the many multimedia tools available in online education platforms.

1.2 Objectives

This project hopes to close the language barrier in education by creating an intelligent translation tool for educational content. The main goal is to develop an accurate, engaging and educational platform that changes English study content into Urdu. Objectives for the project have been set around these 5 main areas:

1.2.1 Develop an AI-Driven English-to-Urdu Translation System

Creating a translation system that uses artificial intelligence for academic use is a main goal of this project. General-purpose translators are not used, as this one will be specifically finetuned with educational language and specialized vocabulary. It uses transformer-based neural networks, just like the ones in language models (LLMs), to make the system both fast and accurate.

The system is designed to go further than producing simple translations. It will adapt its message to the setting, keep the educational meanings and make sure the material is not altered. Texts such as scientific reports, step-by-step math lessons and accounts of history each have a special meaning that needs careful understanding. Its purpose is to respond appropriately to these needs.

Because it allows content to be seamlessly translated into Urdu, the project is expected to make academic subjects more available to students, teachers and researchers who fancy Urdu. A translation service should handle grammar complexity, academic jargon and linguistic subtleties to make its output fit for academic needs.

1.2.2 Enable User-Friendly Content Upload and Translation Processing

This objective includes developing a web site where users, like students and teachers, can quickly publish lessons for translation. With the system, you can use either video lectures in MP4 or download PDF references, requesting translations using the provided interface. Users will find a translation request form and will be able to request features such as subtitles or adjustments to language context. The platform will then automatically generate subtitles from the video audio using speech recognition. The intent is to smooth out website navigation so that any user can manage things easily and help them use cloud storage (e.g., Google Drive) which is convenient for storing large files. As a result, more people in rural or disadvantaged areas have better access to the system and it becomes more useful in practice.

1.2.3 Fine-Tune a Domain-Specific English-Urdu Parallel Corpus

The aim is to prepare an accurate English-Urdu parallel corpus to refine the LLaMA 3.1 8B model, since there is not enough Urdu material for NLP processing. Web scraping will be used on places like Wikipedia, government websites and educational PDFs to add content and then data augmentation will help create diverse structures and words in the sentences. At this stage, the data is processed by tokenizing, cleaning it and aligning it to secure data quality. The developed model will aim for an emphasis on fluent, correct and accurate speech for educational information, as shown by BLEU scores and by feedback from humans. To overcome the problems of translating low-resource languages like Urdu, this objective relies on a strong database that guarantees the system's usefulness in classes.

1.2.4 Integrate an Interactive Quiz Generation and Testing Module

By using a quiz generation module, the project hopes to increase how well students learn by creating interactive exercises using the content translation feature. By using this feature, teachers can transform video or document lessons into multiple-choice tests to help students assess what they understand. Students in the module can provide responses and their feedback allows them to review their performance and improve. Regarding quizzes, educators can create questions that match each lesson's educational goals. This aim turns the platform into a full learning tool, helping students learn and translate at the same time, an important help for Urdu learners in schools and colleges.

1.2.5 Implement an Admin Panel for Content Review

The aim is to build an administration panel to manage the quality and relevance of translation tasks. System administrators can use the admin panel to review and either accept or reject translations, videos, quizzes and public questions. There will be tools for handling user accounts, approving articles and handling translations, while a system sends notifications to admins about unfinished tasks and to users about new articles. We want to keep strong standards, both for the accuracy of the translation and for respecting other cultures and have human teams review anything that automated systems might miss. Because of this objective, users in academic environments can feel confident in the system's educational materials.

1.3 Scope and limitations

1.3.1 Scope

The project's scope determines what it does, its intended users and how it is implemented, so that all efforts are aimed at helping Urdu-speaking students by translating educational material.

- **Translation of Educational Content from English to Urdu:** The project attempts to bring educational video lectures, PDF documents and quizzes from English into Urdu. To do this, we get the video content, remove the audio, use speech recognition for Urdu subtitles and translate the text in documents. The purpose is to provide help in fields such as science, technology and humanities and allow Urdu students in Pakistan to have easy access to academic resources without strong English skills.

- **Web-Based Platform with User-Friendly Interface:** The system was developed with React.js for the client-side and Python-based frameworks (like , Django) for the server-side. Users , students, educators and administrators find it easy to use since it requires nothing more than general web navigation. You can sign up, customize the dashboard, add videos, fill in translation request forms, make quizzes and submit feedback using our key features. Video files are handled efficiently on the platform because it connects with Google Drive and all user information, video metadata and quiz results are stored in a PostgreSQL database.

- **Utilization of Advanced Machine Learning Models:** The system relies on NMT, NLP and LLaMA 3.1 8B—an LLM—to provide accurate translations. To fine-tune the model, I use a corpus that includes an English-Urdu section which was assembled using both online scraping and manual editing of Wikipedia and educational PDF pages. Transformers and attention-based approaches are used in the system to verify the accuracy of meaning which is measured using BLEU and by checking feedback from humans.

- **Administrative Functions for Content Management:** An admin section exists to check articles and monitor user activity. Managers have the ability to look over translations, oversee accounts for all users, moderate public questions and take care of blog articles. Thus, translated content meets the required standards and avoids any problems that usually come from using automatic translations.

- **Interactive Learning Features** Along with translation, the project lets teachers use a quiz generator which creates several-choice tests from the translated work, helping students review their knowledge. Teachers are able to make their own quiz exercises and students can respond with notes about the accuracy of translations. Blog and article publishing is also possible, giving educators ways to add additional learning materials.

1.3.2 Limitations

Learning from the constrains and challenges of the Urdu Translator for Educational Content project, it is clear that they have a big influence on its performance, stability and ability to grow. They are essential for discovering what the system covers and what could be better in the future.

- **Limited Language Support to English-to-Urdu Translation:** The system only permits translation between English and Urdu, so it cannot be used for other languages such as Sindhi, Punjabi or Urdu-to-English. The project could only focus on Urdu-speaking students because supplementary language datasets were rare and it would take a lot of effort and time to create them.

- **Challenges with Highly Technical or Domain-Specific Content:** The accuracy of the translations can be lower for hard scientific or legal content, compared to easy-to-understand text. The training of the LLaMA 3.1 8B relies heavily on the breadth and quality of the training data which does not always cover specific professional terms. Because of this, some of the translations might be inaccurate, so you often need to edit them afterward.

- **Dependence on Dataset Quality and Volume:** The translation model depends very much on the size and quality of the available English-Urdu parallel corpus. Since Urdu is not widely studied in NLP, it was difficult for the project to amass a large, strong dataset. The manually created corpus, though prepared with care, may still have insufficient examples of vocabulary or context, so translation accuracy and smoothness for difficult sentences might be challenged.

- **Requirement for Stable Internet Connectivity:** Users can only add content, ask for translations and view translated materials if they have a reliable internet connection. As a result, people living in rural communities or out of reach areas in Pakistan do not have full access to these services.

Even though the system relies on Google Drive for cloud storage, it does not allow content to be used when offline, so there are limits in areas with weak internet connections.

- **Resource Constraints for Real-Time Processing:** Real-time work with video files for translation takes up a lot of computing resources, namely GPUs or TPUs. Because the project is developed and tested on mid-level equipment, processing might take longer than usual when there are many users. Such a restriction may challenge the system's capacity and the way users interact, if the platform is used by many people at once.

- **Potential Usability Challenges for Non-Technical Users:** While the site is easy to use, people without strong technological skills such as rural educators or people over a certain age, may find it hard to figure out how to access more challenging features such as creating and accessing quizzes or translation requests. Since no text-to-speech or multilingual features were added, users with visual or auditory problems are further limited during their use of the project, a problem that has not been dealt with at this stage.

1.4 Methodology Overview

The projects followed the Agile approach, enabling them to have routine development, testing, and revamping on the basis of feedback and model performance. The major steps include:

- Requirement gathering and initial planning
- Dataset collection and preprocessing
- Model training and evaluation
- Web platform development
- Testing and deployment
- Continuous feedback and improvement cycles

This iterative process has ensured the flexibility to accelerate adaptation and continuous improvement at every stage of the project.

1.5 Report Organization

Chapter 1 First, write an introduction to the project. Then, proceed to the problem statement targeted by this project followed by pointing out the main objectives, defining as well as mentioning scope and limitations, the adopted methodology (Agile), and a brief account of the report structure.

Chapter 3 This section discuss the literature review existing work of our project

Chapter 2 Software Requirements Specification (SRS) This section contains the system purpose, intended users, definitions and acronyms, perspective of the product, user characteristics, and key assumptions, followed by extensive functional and non-functional requirements specifications as per IEEE SRS format.

Chapter 4 System Design This chapter explains best the whole system architecture and presents various diagrams like use cases, ER diagrams, class and sequence diagrams, UI mockups, and database schema.

Chapter 5 Implementation This chapter describes the tools and technologies applied during development, describes different modules of the system, and provides a few screenshots of important features demonstrating the user interface.

Chapter 6 Testing This chapter deals with employing testing strategies, detailed test cases with results, bug tracking and fixes, and tools used for automated or manual testing.

Chapter7 Deployment and User Manual This chapter records everything needed to actually deploy the system, such as platform dependencies, hosting instructions, plus a guide on how to use the system to the end user. It also contains basic maintenance guidelines.

Chapter8 Results, Evaluation, and Conclusion The last chapter simply summarizes all that this project has achieved. It discusses challenges facing the project throughout the development process, some lessons learned, and improvement suggestions for future work.

Chapter 2

Software Requirement Specifications (SRS)

2.1 Introduction

2.1.1 Purpose

This Software Requirements Specification (SRS) explains the complete details of how the Urdu Translator for Educational Content functions. We create this project to let everyone in education access English learning materials through video translations from English to Urdu. Our Urdu Translator Project lets Urdu language students find valuable information they cannot reach otherwise because of their language needs. Our system functions as a web platform that permits users to submit videos and ask for translations while letting them access translated content and reply to feedback surveys and test questions. The online system manager checks and approves all content before it becomes available to users.
The SRS defines what the system should do as well as its operational environment along side usage expectations. The document provides essential guidelines for every team to develop and maintain the system platform.

2.1.2 Intended Audience

This document exists to benefit multiple types of users who work on and with the system. Project supervisor and academic evaluators can now see the complete structure and development details of the system through the SRS. The development and design team members use this report as their guide to understand key system standards and operational patterns. Testers use these specifics to make test cases and then prove that they work properly. Students and teachers who use

the system may review the document to understand available features. The document helps system administrators who moderate content and control user access. Any team that works on this platform later can use the document as a starting point to prevent duplicate work and keep changes consistent.

2.1.3 Definitions Acronyms

To maintain clarity throughout the document, the following terms and acronyms are used frequently and defined here:

- **NMT (Neural Machine Translation):** For clear communication the following terms and abbreviations appear often in this document and you will find their definitions here.
- NMT uses neural networks to generate translations through its deep-learning approach.
- **LLM (Large Language Model):** An LLM system is an AI model that eats vast amounts of texts to generate text outputs and translate between languages.
- **SVO/SOV:** The document contains English sentences with SVO format and Urdu sentences displayed as SOV.
- **BLEU Score:** BLEU Score helps evaluate machine-translated quality by matching it against human translations.
- **UI:** When users communicate with the system they see and experience its visual displays through UI.
- **API:** Through the API our program system parts can exchange information.
- **SQL:** Preserve consistency between facts and values in the database using SQL.
- **React.js:** React.js functions as a JavaScript library to design user interface elements especially for single-page applications.

2.2 Overall Description

2.2.1 Product Perspective

The Urdu Translator for Education uses a web-based solution as an independent service. This system stands alone at present but can connect to learning platforms and educational data banks in future development. The platform includes the elements needed for web access, content storage, database functionality, machine translation processing, and a graphical user interface. Users access this system through the React.js front end while all administrative tasks happen on the SQL/Python back end.

The ML translation module turns English content into Urdu outputs with a translation model that is already trained. The system converts text from English to Urdu which it sends back to the user. Google Drive stores videos so users can save their large files properly and let servers handle fewer tasks. The setup provides different sections to handle user sign-up, video addition, test-making, feedback processing, and system control. Our product can improve further based on updates that expand its current accuracy level and language support capabilities.

2.2.2 User Characteristics

The system works with three main user groups which are students, teachers, and admins. Students receive all educational translations as their final consumers. Users should know basic web access because the platform needs this as a minimum requirement. The system gives teachers access to add videos and lectures plus share translation needs while maximizing blog pages or quiz creation. The system works well for anyone with basic computer skills since its design simplifies all steps and actions. Admins supervise how the platform works and handle user accounts plus content while approving translations. They need strong technical knowledge and a solid understanding of the platform to fulfill their duties. Because the system helps users at every educational level, both faculty and students need smooth navigation through its features.

2.2.3 Assumptions and Dependencies

The way Urdu Faham is developed, used and operates is guided by many key factors and assumptions that affect its performance. Their details are discussed below to show both the project's needs and the possible restrictions it may encounter.

2.2.3.1 Assumptions

- **User Access to Technology:** It is assumed that target users such as students and educators, are able to use devices with modern web browsers (e.g., Chrome, Firefox) and stable internet connection in urban and semi-urban areas of Pakistan. To use the platform on the web and to upload or download any resources, users must have this feature.
- **Digital Literacy:** It is assumed that students are able to move around websites, add files and fill out documents, but educators and administrators are required to have enhanced information-management and system-settings skills. As a result, employees can work on the platform without long training courses.
- **Corpus Sufficiency:** Corpus Sufficiency supposes that the curated English-Urdu parallel corpus, made up of about 50,000 sentence translations, is enough to form an effective translation model. It addresses different sentence types, words and ways that language is taught.
- **Human Validation:** Requires having people who understand Urdu well available for evaluation, both during and after testing, alongside automatically produced metrics.
- **Primary Language Need:** English-to-Urdu translation is considered most important because most of the target group uses Urdu, with lower initial demand for translating Urdu-to-English or Sindhi or adding more languages at the start.
- **Cultural Acceptance:** The government assumes that translated material fits within Pakistani norms and education and that there will be minor objections to AI-generated translations after they are inspected by human scholars.

2.2.3.2 Dependencies

Technological Dependencies:

- Most of the work is based on open-source tech, including React.js, Django, PostgreSQL and PyTorch, so developers often need to update to keep everything compatible and secure.
- For translation, use LLaMA 3.1 8B and for speech recognition, use Whisper. Then grab the weights from Meta AI or Hugging Face.

- The translation model must be fine-tuned using GPUs like the NVIDIA A100 and can only be trained on a university cluster.
- Requires stable repositories of packages because it uses NLP tools, SentencePiece for splitting texts into tokens and Hugging Face Transformers to build its models.

Data Dependencies:

- Relies on the availability of good, parallel data in English and Urdu from freely accessible information (such as Wikipedia, official government sites and educational PDFs) that you get with web scraping tools called BeautifulSoup and Scrapy.
- Proper quality in the dataset depends on human editing and operations such as cleaning, aligning and changing data, using tools and scripts like NLTK and those made in Python.
- With not enough Urdu resources for NLP, the corpus may end up more limited and homogenous which may reduce the model's effectiveness for detailed subjects.

Resource Constraints:

- Lack of powerful mid-level hardware (such as one GPU) might cause real-time translation on videos or too many users to run more slowly.
- Development in this field requires expertise in NLP and web tech, so any shortage of these skills may push the timeline back.

2.3 Specific Requirements

2.3.1 Functional Requirements

The following are the core functional requirements for the system:

- **User Registration and Login:** To access the system users sign up with email and make strong passwords for accounts. Email verification is mandatory.
- **Personalized Dashboard:** When users sign in they get to browse their material including all their translation tasks and test results through their custom dashboard.

- **Video Uploading:** Users at school or home can upload MP4 videos to the system which saves them to Google Drive after verification.
- **Translation Request Form:** The system lets users upload videos and then fill out a request form to get translation service. Users need to select their needs including titles, their language preferences, and enter notes through this form.
- **Translation Processing:**After translation processing finishes, the system alerts users through its ML model.
- **View Translated Videos:**The dashboard shows students both subtitled and dubbed translations of the uploaded videos.
- **Quiz Functionality:**Teachers create tests for their students which those students answer questions from specific lesson material.
- **User Feedback:**Users can share their thoughts after finishing a translation or quiz through the Feedback Submission feature of our system.
- **Blog/Article Publishing:** Teachers use this platform to add their written educational resources.
- **Admin Management:** The system administrator reviews and validates all videos translations and text materials before they are made public.

2.3.2 Non-functional Requirements

The non-functional requirements define system performance and operational quality:

- **Usability:** People without technical training should find it easy to use the interface. Regular users who lack technical training can complete both import tasks and translation steps on their own. People can easily interact with website without any issue.
- **Performance:** The system needs to deal with translation demands in 5 to 10 minutes intervals based on file size and server capacity. The user can't wait for the transition
- **Scalability:** The system needs to handle rising user numbers plus more video uploads and translation tasks as user numbers grow and the platform expands.

- **Security:** Every user data must pass through our top-level security encryption technology. Our system must block off repeated login attempts by hackers. The platform needs to use tested security measures for handling user input correctly and protect user sessions.
- **Reliability:** The system must stay functional without many interruptions. Any user work such as uploading files or sending forms should get saved automatically to protect users from losing information.
- **Maintainability:** Our system design requires separate code parts that can stand alone and needs complete documentation to let experts update or track issues.
- **Accessibility:** Users should navigate all screens using text-to-voice technology plus multiple language text types in a single platform.

2.4 Interface Requirements

2.4.1 User Interface

The UI system builds user connection while making the system work effectively. The team uses React.js to develop the application because it produces fast results during user interaction. After visiting the platform users can sign in or register through an easy-to-use minimal form. Once users authenticate the system presents all their content with uploading status and message updates. Users can add videos through their device files and include basic information about the videos when using the platform's upload system. Users find translation request tools within the main dashboard interface. The quiz system presents single-choice options to users who see their entered responses with their test results. To receive feedback users fill out a distinct form that includes rating points as well as feedback details. With extra tools Admins can monitor user accounts as well as control system content while examining user submissions. Users can simplify their administration operations because the system has tools to sort and refine results. Users can access the system from any mobile device without issues. It also runs smoothly on both basic and powerful devices.

2.4.2 API Interfaces

The platform depends on various REST APIs for internal and external communication.

- **User APIs** The system provide User APIs to manage user actions such as registration, login, video upload, and viewing translation results.These APIs will ensure secure user authentication, authorization, and interaction with other system components.
- **Translation APIs** The system use Translation APIs to send processed English text to the translation model and receive the translated Urdu output.These APIs will handle communication between the front-end input, translation engine, and back-end storage for better integration.
- **Storage APIs** The system shall store uploaded video files in the project's designated directory on the server.A corresponding video link or file path shall be saved in the database for retrieval and reference.
- **Quiz APIs** The Quiz APIs support interactive learning by enabling the creation and evaluation of quizzes based on translated content. They manage question rendering, answer submission, and scoring, while also storing user performance data for future analysis and improvement recommendations.

Chapter 3

Literature Review

A substantial advancement in language translation for low-resource languages like Urdu can be observed with the advent of Neural Machine Translation (NMT) and Large Language Models (LLMs). However, challenges remain, particularly in preserving contextual accuracy, handling linguistic intricacies, and translating domain-specific content such as educational materials. This section reviews literature that informs the English-to-Urdu translation landscape, focusing on model architectures, multilingual learning, zero-shot and few-shot capabilities, and challenges stemming from Urdu's under-representation in training datasets.

The foundational work by SSSUTMS [1] introduced an LSTM-based NMT model for English-to-Urdu translation. This system, built on an encoder-decoder architecture with preprocessing, tokenization, and Word2Vec embeddings, showed promising BLEU scores of 50.86 on training data and 47.06 on test data. Although the model performed well for short sequences, it struggled with sentence lengths beyond ten words. The study underlines the potential for extending traditional NMT techniques to low-resource languages through effective preprocessing.

In contrast, Rachel Bawden's research at Inria [2] evaluates the performance of the BLOOM multilingual language model. It focuses on zero-shot and few-shot translation across various languages and confirms the value of prompt engineering. While BLOOM showed improved results with few-shot learning, it also exhibited a drop in accuracy when dealing with underrepresented languages like Urdu. The study emphasizes the need for domain-specific prompts and diversified multilingual datasets to improve translation fidelity.

Similarly, Malik [3] explored the biases in English-centric versus non-English-centric LLMs in multilingual NLP tasks. The research highlights that models

trained on English-dominant corpora underperform in other languages, particularly in culturally nuanced contexts. Native language prompts consistently outperformed translated ones in tasks demanding emotional or contextual accuracy. The study advocates for native-language pretraining to better serve speakers of languages like Urdu.

Chaoqun Liu [4] analyzed early corpus-based machine translation approaches such as Example-Based Machine Translation (EBMT) and Statistical Machine Translation (SMT). These systems showed moderate BLEU scores but struggled with Urdu's morphological richness and flexible syntax. Liu emphasizes the inadequacy of literal translations in educational settings where idiomatic and contextual accuracy is essential.

Ghafoor [5] investigated the downstream effects of machine translation on tasks like sentiment analysis. His findings revealed that automated translations often result in polarity shifts, particularly due to poor word sense disambiguation. These inaccuracies can distort sentiment in educational content, thereby affecting comprehension. The study identifies common Urdu terms that introduce these errors and recommends specialized handling in translation systems.

Chitale [6] contributed to the study of in-context learning and example-based prompting. The research finds that even minor inconsistencies in example formatting can significantly degrade performance. It also warns of the model's vulnerability to misleading context, which is particularly concerning in educational applications. The paper concludes that well-structured, culturally relevant examples are vital for translation reliability.

Collectively, these studies provide a robust foundation for this project. Traditional NMT methods offer reliability for short, straightforward content, while LLMs present scalable solutions with limitations in underrepresented language contexts. This project leverages these insights to propose a hybrid approach using LLaMA 3.1 8B, fine-tuned on a domain-specific English-Urdu corpus, to improve both fluency and semantic accuracy.

This literature review highlights critical gaps the project aims to address, especially the lack of educational focus in existing translation systems. Through curated datasets, fine-tuned models, and features like quiz generation and admin moderation, the proposed system seeks to transform translation from a utility into an educational platform. Cultural awareness and user feedback loops further enhance both quality and user engagement.

Basit [7] The use of transformer architectures in recent NMT systems (as stated by

Vaswani et al., 2017) has improved the translation performance of high-resource languages such as English, French and German. Even though LLMs have posted good results in many languages, they still experience trouble handling the low-resource language Urdu (Hendy et al., 2023).

Although Urdu has a huge number of speakers, there aren't many resources to help with machine translation (Daud et al., 2017). Previously, translation that uses rules often did not manage language complexities well and approaches based on statistics had difficulties with syntax and semantic structure (Okpor, 2014). Using large parallel text as well as attention mechanisms, transformer-like neural approaches have shown significant improvements. Freitag et al. (2021) and Hendy et al. (2023) have studied how LLMs such as GPT-3.5 do in translation, showing that they are comparable with zero-shot systems for popular languages. However, models perform much less well when the language is low resource (Jiao et al., 2023; Stap and Araabi, 2023). Using back-translation and improving on data collected from top-quality sources, this specialized model for 22 Indic languages, called IndicTrans2, deals with these challenges better than previous models (Gala et al., 2023).

It was pointed out in prior research that NMT models often have problems with deleting words, translating word-for-word, copying the same word and erroneously recognizing named entities when working with Urdu translations (Sennrich et al., 2015a; Yang et al., 2019). Errors in transliteration lead to greater difficulties in producing good output results (Sennrich et al., 2015b). Machine learning models built on data with various and equal distributions of Urdu words, for example IndicTrans2, generally do better in several performance measures (Costa-jussà et al., 2022).

To continue this effort, this work evaluates four major translation models—GPT-3.5, opus-mt-en-ur, NLLB and IndicTrans2—on a range of Urdu datasets to understand and resolve issues found in translating Urdu.

Zafar [8] To improve, Machine Translation (MT) has been using three main methods: Rule-Based Machine Translation (RBMT), Statistical Machine Translation (SMT) and Example-Based Machine Translation (EBMT). The main approach for analysis in RBMT is rules, whereas SMT relies on gathered information from bilingual text data.Nagao's EBMT works by means of translation by analogy, first finding similar passages in a double collection of matching texts as its information source. The framework by Zafar and Masood (2009) helps users customize translations in English to Urdu using an interactive EBMT system which ensures better performance. The system accepts idioms, homographs and updates itself

through time using the bilingual corpus. Because the system is interactive, users are able to change parts of the translation, taking word order and context into account. The authors stated that their presentation of the language was better able to handle the linguistic challenges found in Urdu.

It is especially valuable for Urdu which uses many richly-formed words in different orderings. Allowing bilingual content and interactive use ensures the system gives highly accurate and appropriate translations. The problem of handling idiomatic expressions and differences in structure between English and Urdu is addressed by this approach.

Chapter 4

System Design

4.1 System Architecture

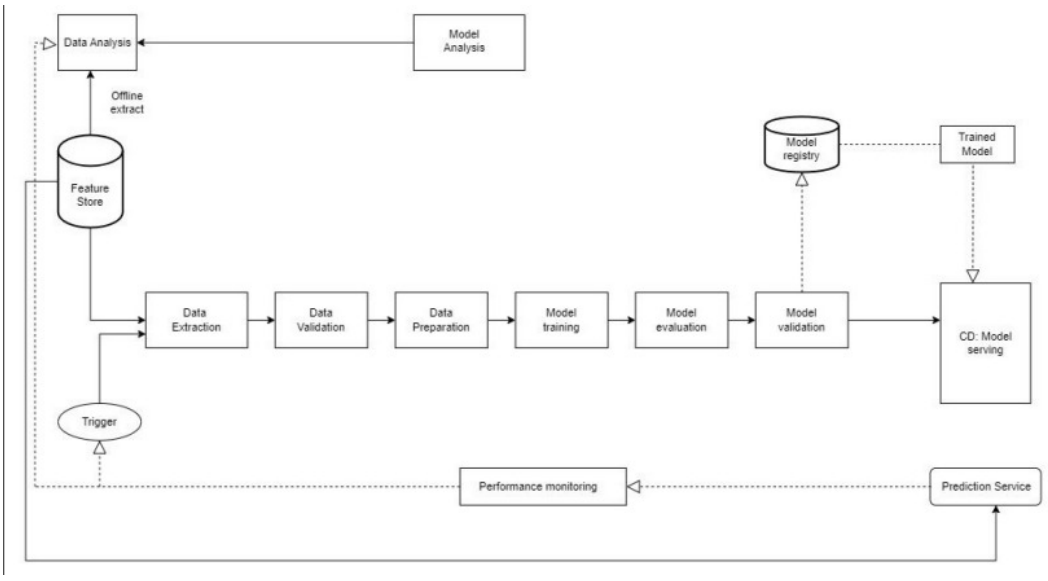

FIGURE 4.1: System Architecture Diagram

This diagram represents a machine learning (ML) lifecycle architecture, illustrating the end-to-end workflow from data extraction to model deployment and monitoring. It begins with a Feature Store, which acts as a centralized repository for storing curated features. The data flows into the pipeline through Data Extraction, followed by Data Validation and Data Preparation, ensuring the quality and consistency of input features. A Trigger component initiates this pipeline. Once the data is ready, it is passed into Model Training, where the machine learning model learns from the data. This is followed by Model Evaluation, ensuring the trained model meets performance criteria. After evaluation, the model undergoes

Model Validation, where it's either accepted or sent back for improvements. Accepted models are stored in a Model Registry and then deployed through the CD: Model Serving component. The Prediction Service handles incoming requests for predictions, and the performance of the model in production is tracked via Performance Monitoring. This monitoring data feeds back into Data Analysis and Model Analysis, helping improve the model iteratively. The architecture supports continuous feedback and improvement, enabling robust and scalable ML operations (MLOps).

4.2 Use Case Diagrams

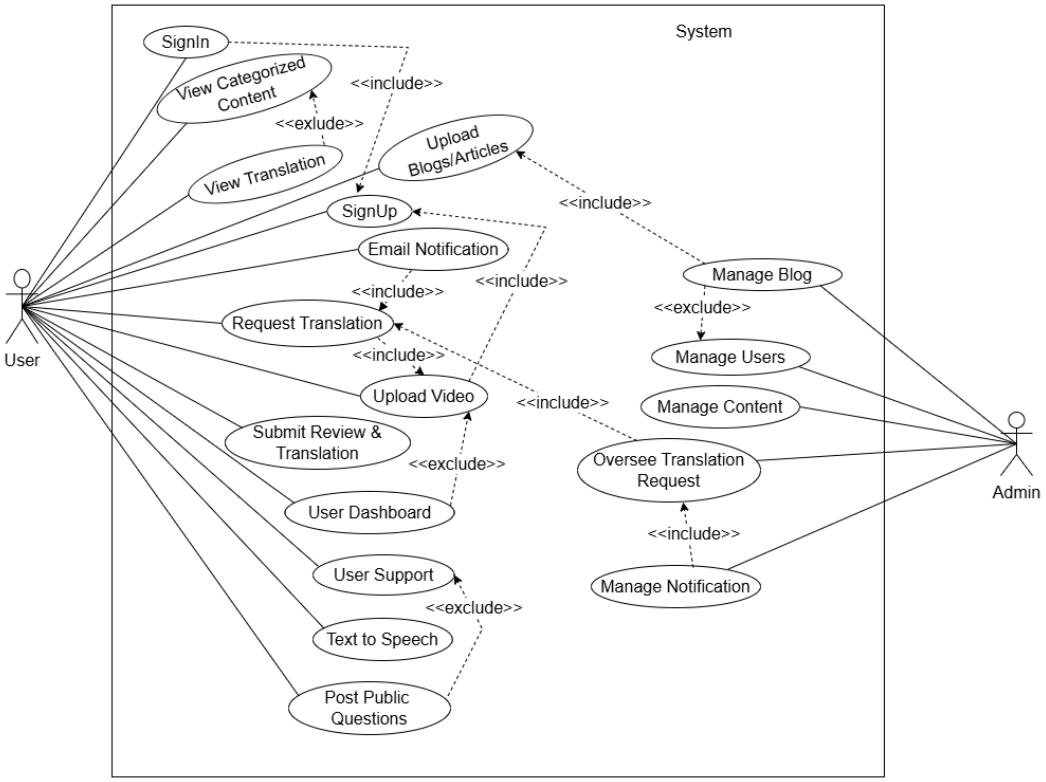

FIGURE 4.2: The Use Case Diagram

The Figure 4.1 Use Case Diagram illustrates the functional requirements and user interactions of the proposed system. It identifies the two primary actors involved: User and Admin, and maps out the various actions (use cases) they can perform within the system boundaries. This diagram plays a crucial role in understanding how the system will behave and what functionalities are available to different users.

The User is provided with several core functionalities such as SignIn, View Categorized Content, Request Translation, Upload Video, Submit Review and Translation, View Translation, and Post Public Questions. These use cases reflect the interactive and content-sharing nature of the system. Some functionalities are conditionally included or excluded using ¡¡include¿¿ and ¡¡exclude¿¿ relationships, such as Upload Video being a part of Request Translation, and Email Notification being included during SignUp. Optional features like User Support and Text to Speech are shown as **exclude**, indicating they might not be triggered in every session.

On the other hand, the Admin has system management roles, including Manage Users, Manage Content, Manage Blog, Manage Notifications, and Oversee Translation Request. These functions ensure that the admin has full control over the platform's operations and content quality.

4.3 ERD Diagram

The database for the proposed system is illustrated from the logical structure through the Entity Relationship Diagram (ERD). It is a representation of the system's main entities, their features and the relationships of these entities. In database design, this model is mandatory, by which all data related part of the system are clearly defined and interconnected.

In the system the central entity is User who has the relationships with other entities like Quiz, Question, VideoLecture, Translation, Notification. Properties related to the system's basic profile and interactions among each user within the system; the UserId, Name, Email, Role properties are included in the User entity. In other words, each user has one or more quizzes, questions, and video lectures that were created or interacted by them.

System management is done by the Admin, who has User, BlogArticle, and VideoLecture entities pointed to him/her. It reveals the admin's duties in the surveillance, approval, and moderating of contents and the uploaded contents. Also, the typical blog attributes like ArticleId, Title, Content, Author, DatePosted is same in the BlogArticle entity.

Clearly, this particular VideoLecture and Translation pairs are not unrelated and usually translations are attached to a specific VideoLecture. SourceText, TranslatedText and Status fields are included in each translation as well as TranslationId. The Notification entity is in charge of managing notifications, connecting on both sides with the User and VideoLecture, to indicate how alert are created and associated with user actions.

Lastly, auxiliary entities like EditProfile, AddVideoDetail, and System support

various features such as profile management, video metadata handling, and overall system tracking.

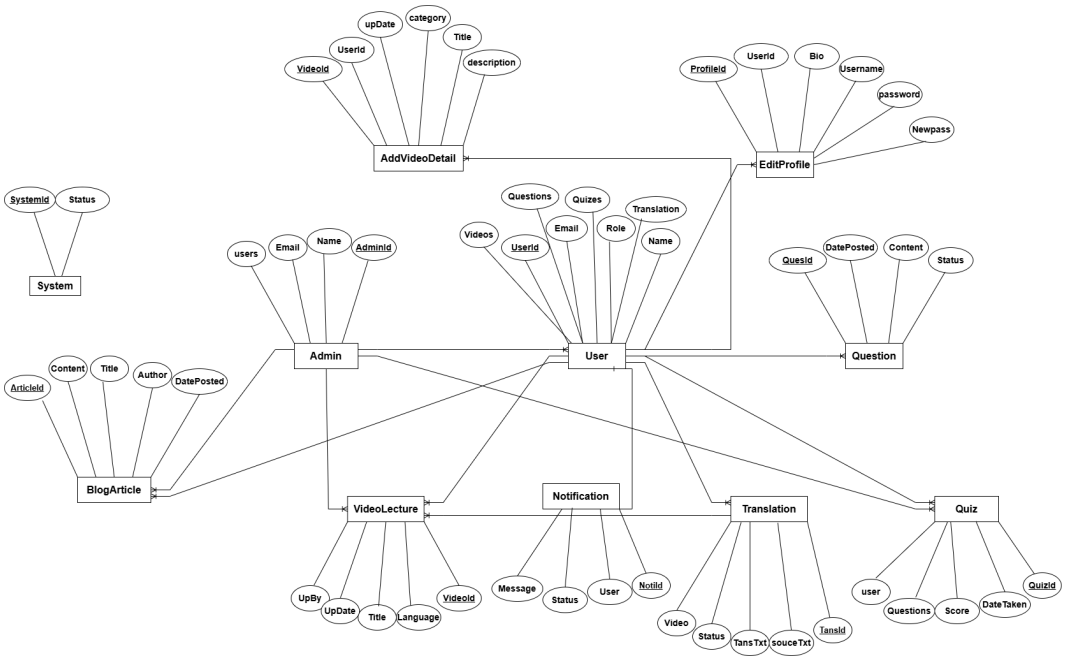

FIGURE 4.3: The ERD Diagram

4.4 Class Diagram

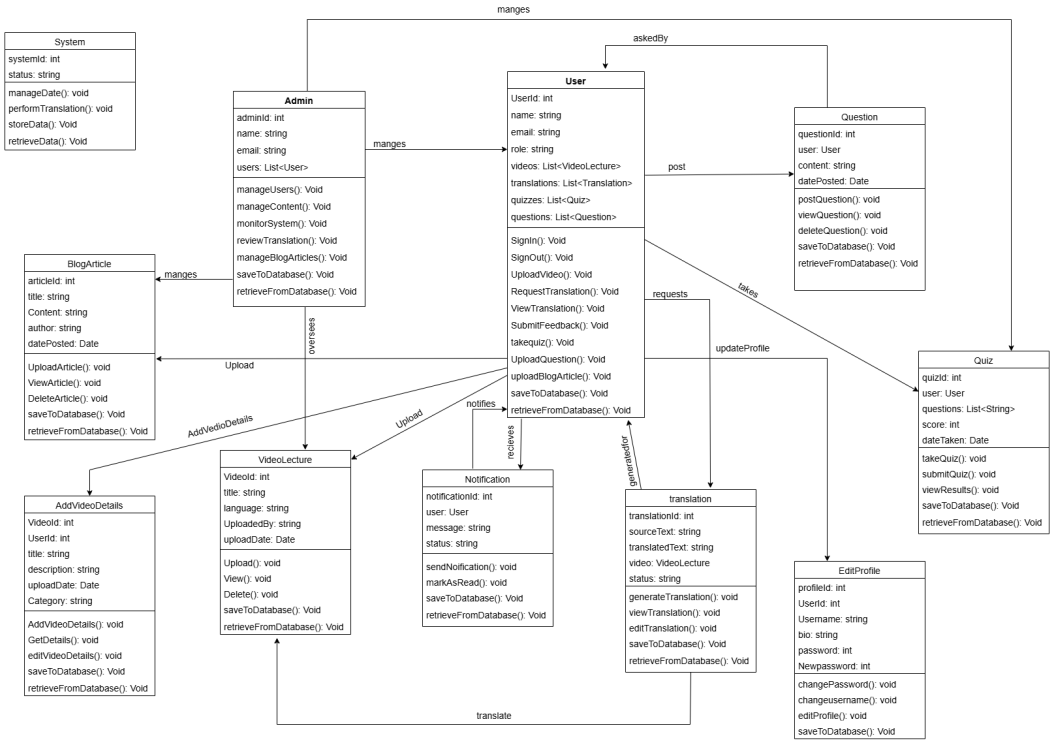

FIGURE 4.4: The Class Diagram

The Figure 4.3 Class Diagram represents the structural design of the system by illustrating the system's main classes, their attributes, methods (operations), and the relationships among them.

- User – The most common entity of them all interacting with most of the system services. This class User contains userId, name, email attributes as well as relationships with the lists of videos, quizzes, translations, and questions. It contains ways for signing in, uploading videos/articles, requesting/viewing translations, posting questions, etc.

- Admin - This class will represent administrative user with capabilities such as content management, user management and blog article management. It also gives access to higher functions like monitorSystem(), manageUsers() and retrieveFromDatabase().

- VideoLecture, BlogArticle, Translation, and Quiz – These represent the core content entities of the platform. It shows its role in taking a user and handling educational material (third field) with attributes and methods like upload(), view(), editTranslation(), takeQuiz(), etc.

- The support classes here are Notification and EditProfile – which improve the users experience. The profile class is used to change personal information and Notification is a system event that sends out notification to the user.

4.5 Sequence Diagrams

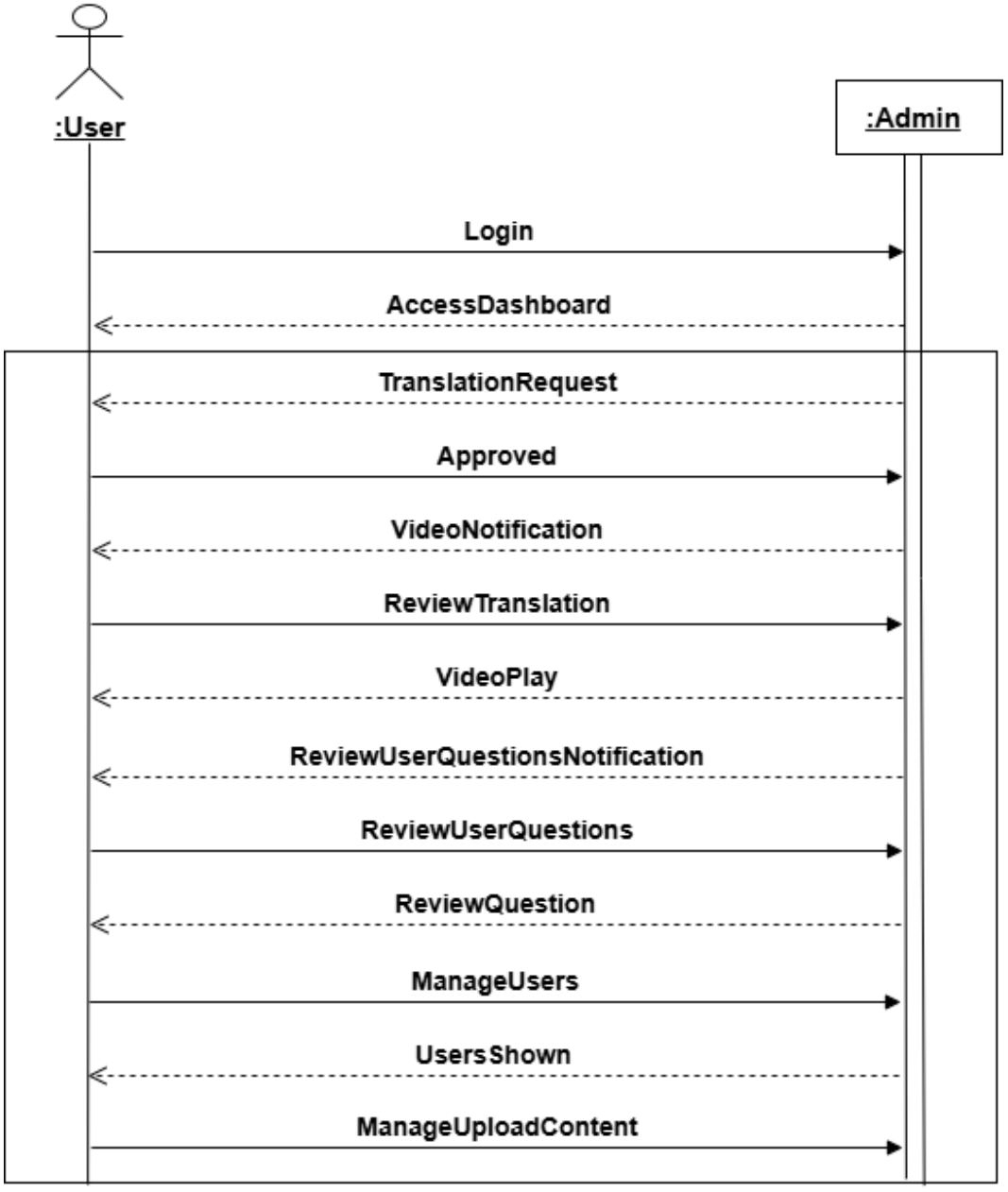

FIGURE 4.5: The Sequence Diagram

The sequence diagram illustrates the interactions between two actors—User and Admin—in the Urdu Translator for Educational Content system, focusing on the workflow of user actions, admin responsibilities, and system notifications.

- Actors: The diagram involves two actors:
  User: Represents the end-user (e.g., a student or educator) interacting with the system to request translations and access content.
  Admin: Represents the system administrator responsible for managing content, users, and translations.

- Lifelines: The vertical dashed lines under each actor represent their lifelines, showing the sequence of actions over time.

- Messages: The horizontal arrows represent messages or interactions between the User and Admin, indicating the flow of actions and responses.

4.6 Deployment Diagrams

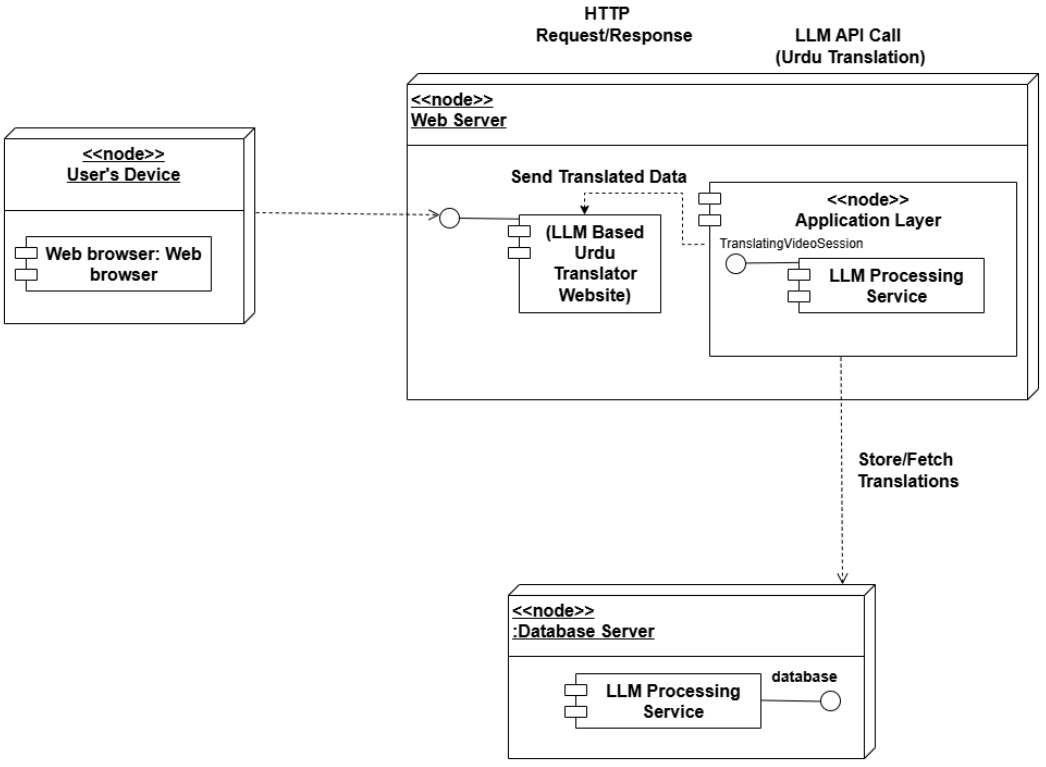

FIGURE 4.6: The Deployment Diagram

This deployment diagram describes the deployment structure of a system, showing the physical nodes, the components deployed on those nodes, and their interactions. The system is a web-based Urdu translation service that utilizes large language models (LLMs).

4.7 Component Diagrams

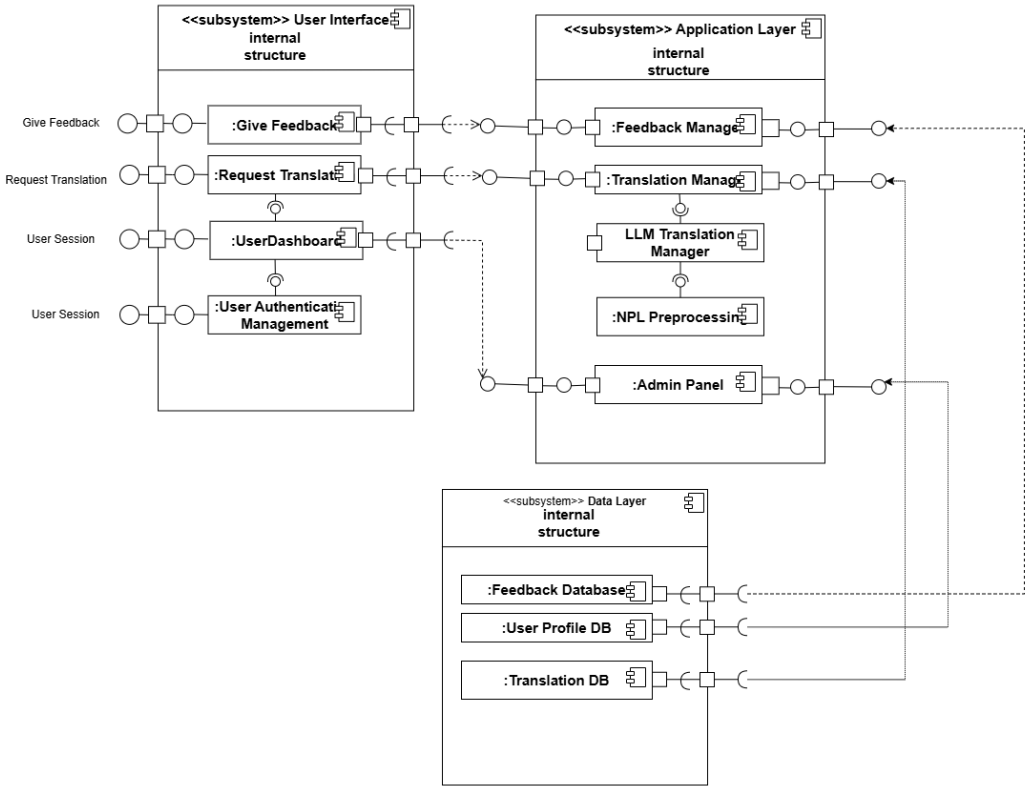

FIGURE 4.7: The Component Diagram

A component diagram that represents the architecture of a system by outlining the main components, their functionalities, and interactions.

4.8 Data Flow Diagram (DFD) Diagrams

Data Flow Diagram-level 0

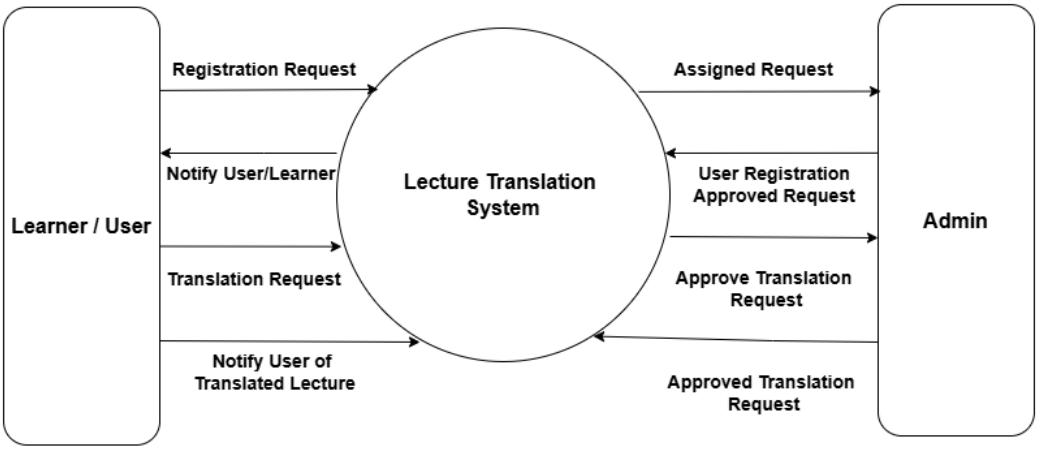

FIGURE 4.8: The Data Flow Diagram

The Level-0 Data Flow Diagram (DFD) provides a high-level view of the Lecture Translation System (LTS), illustrating the flow of data between the system and its primary external entities. This diagram identifies the core process of the system, the external entities involved, and the data exchanged between them.

Data Flow Diagram-level 1

The Level-1 DFD illustrates the detailed interaction between the system processes, the Learner/User, and the Admin. It expands on the core process by defining sub-processes, data storage components, and flow of information.

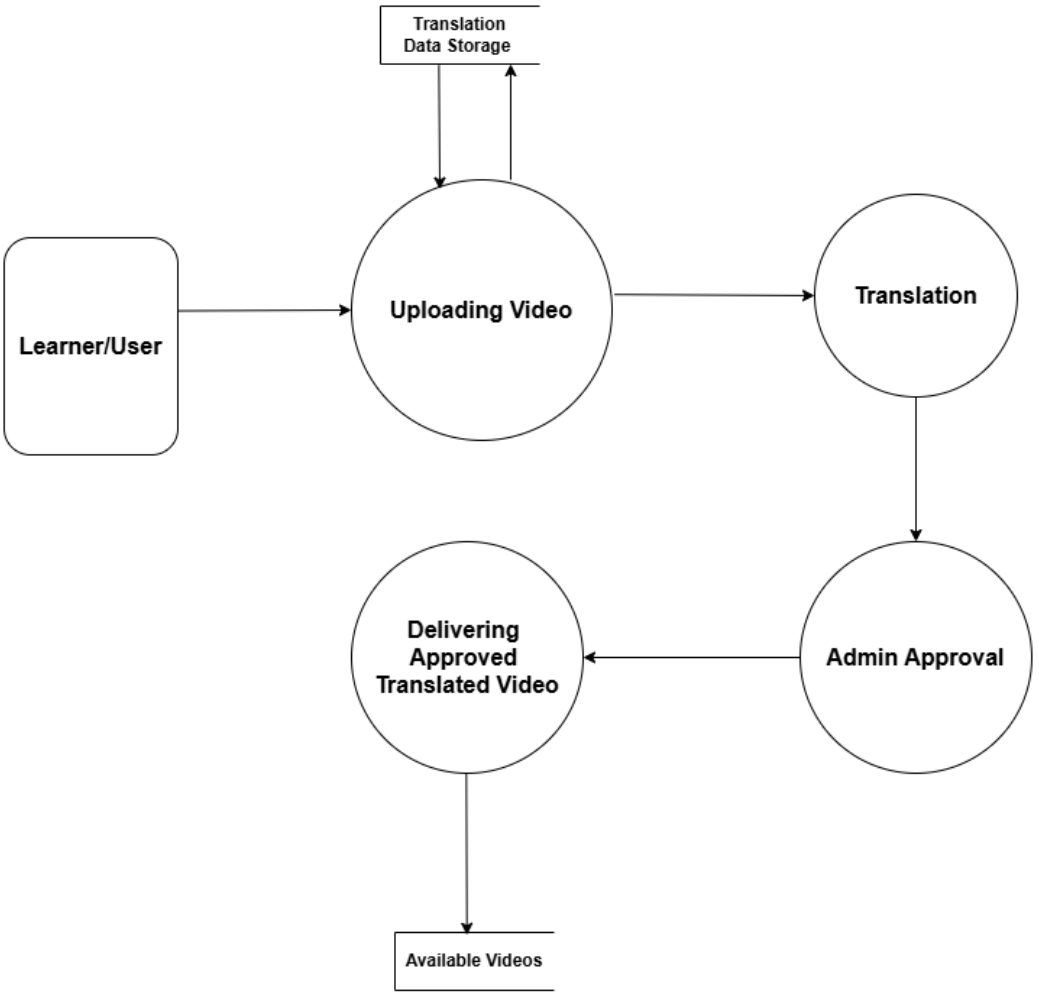

FIGURE 4.9: The Data Flow Diagram

4.9 Wireframes

Here are the Wireframes of the Website.

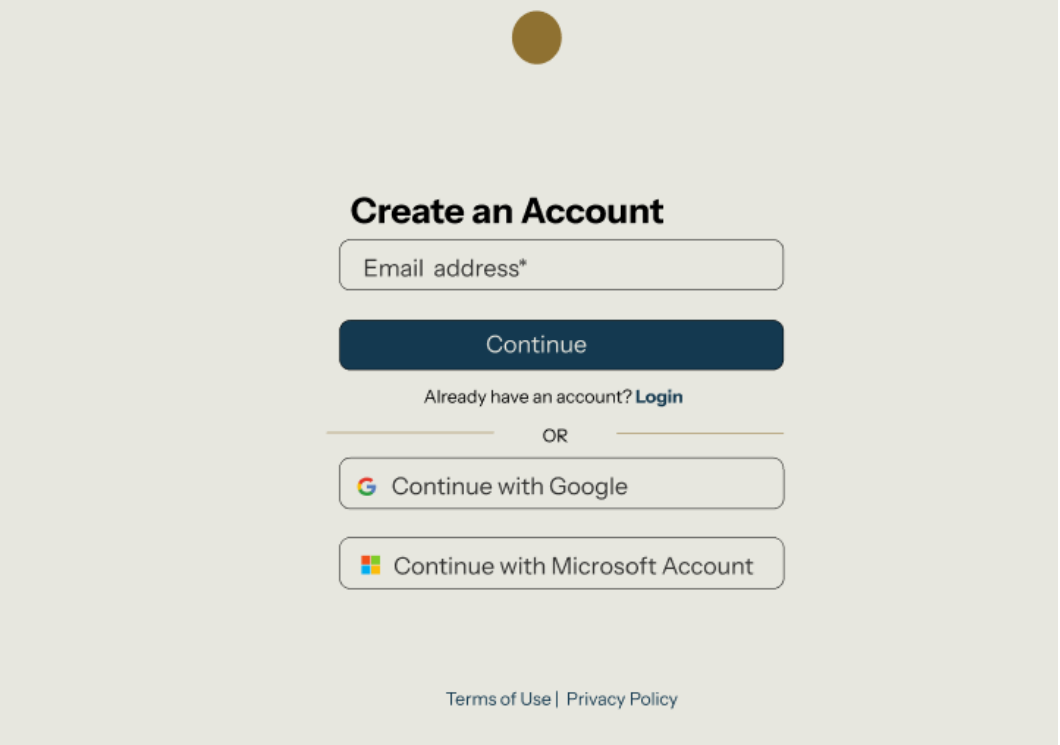

FIGURE 4.10: SignUp Diagram

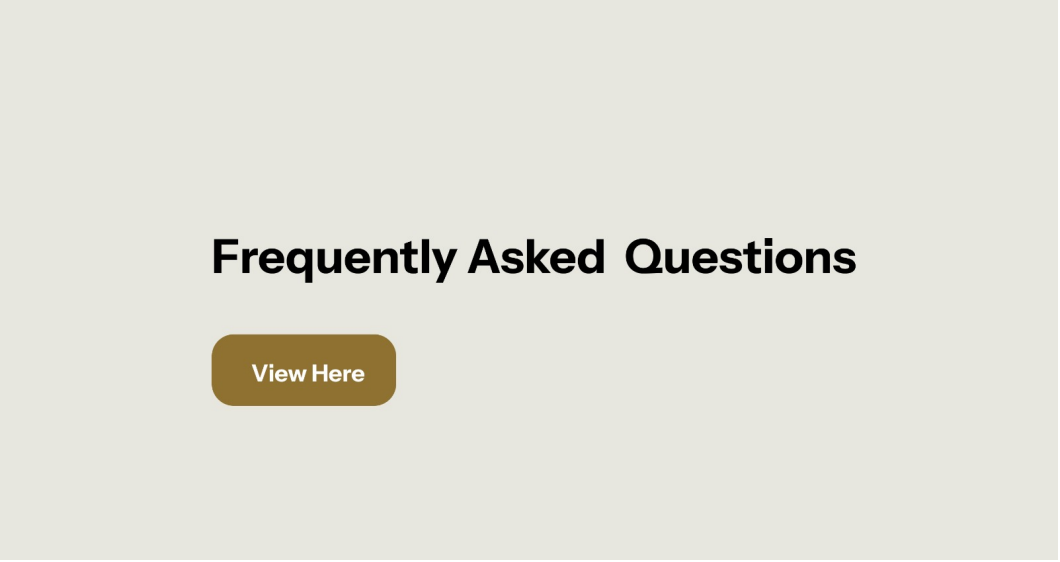

FIGURE 4.11: FAQ Diagram

FIGURE 4.12: Dashboard Diagram

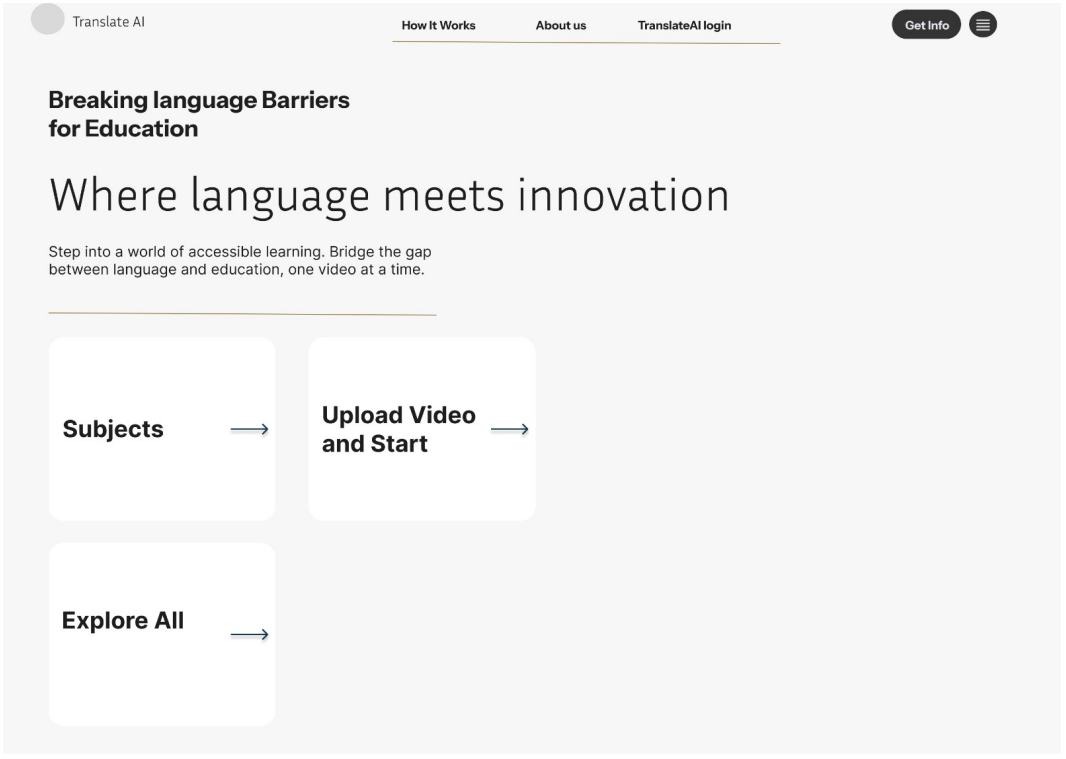

FIGURE 4.13: Home Page

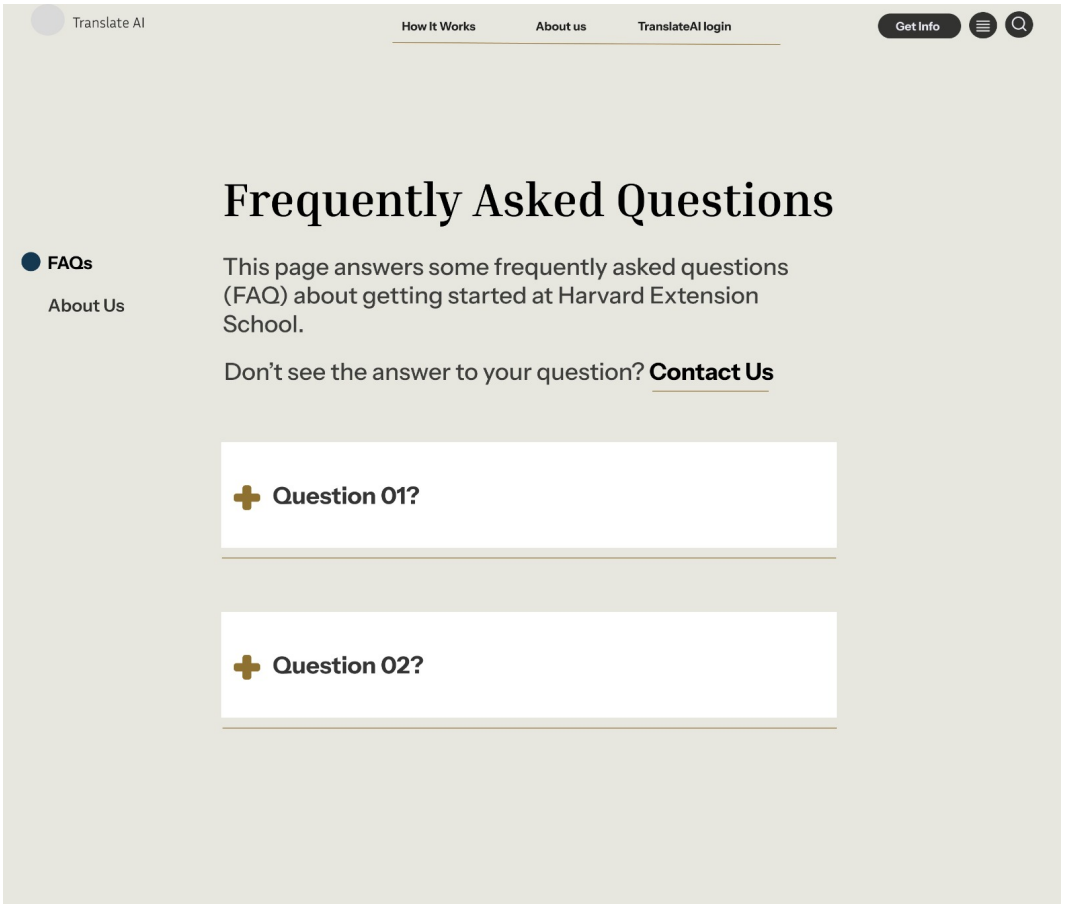

FIGURE 4.14: Frequently Asked Questions

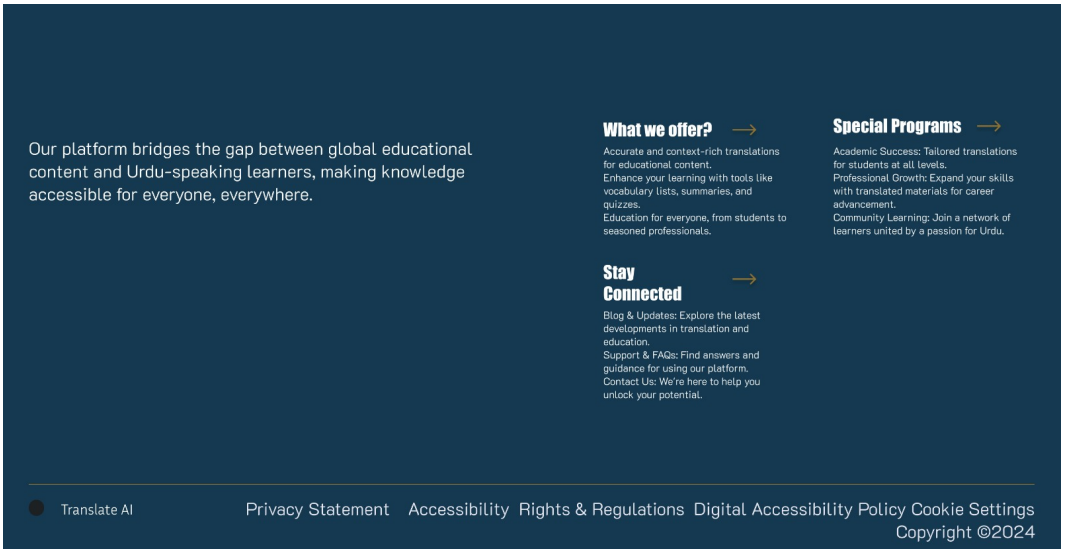

FIGURE 4.15: Footer Page

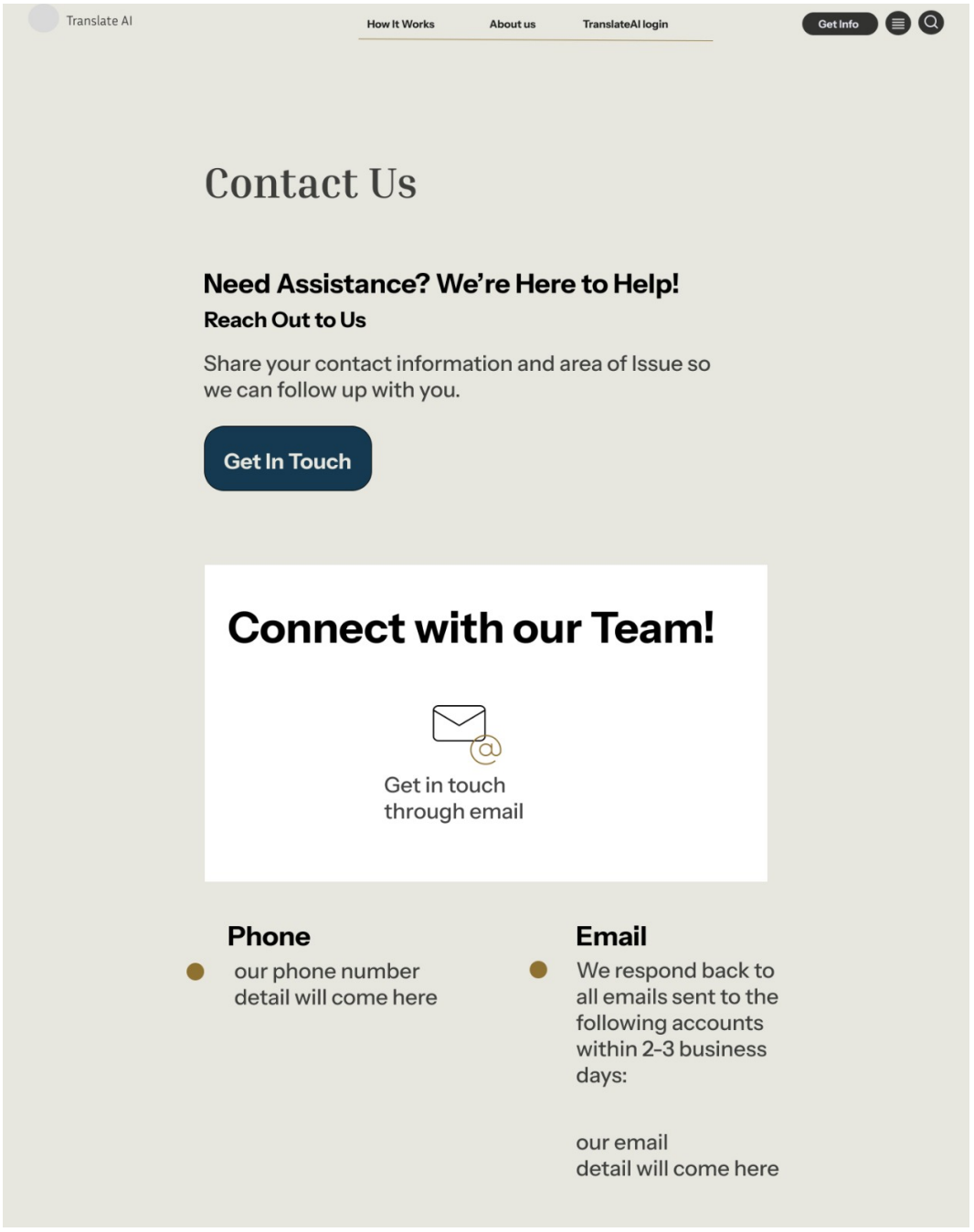

FIGURE 4.16: Contact Us

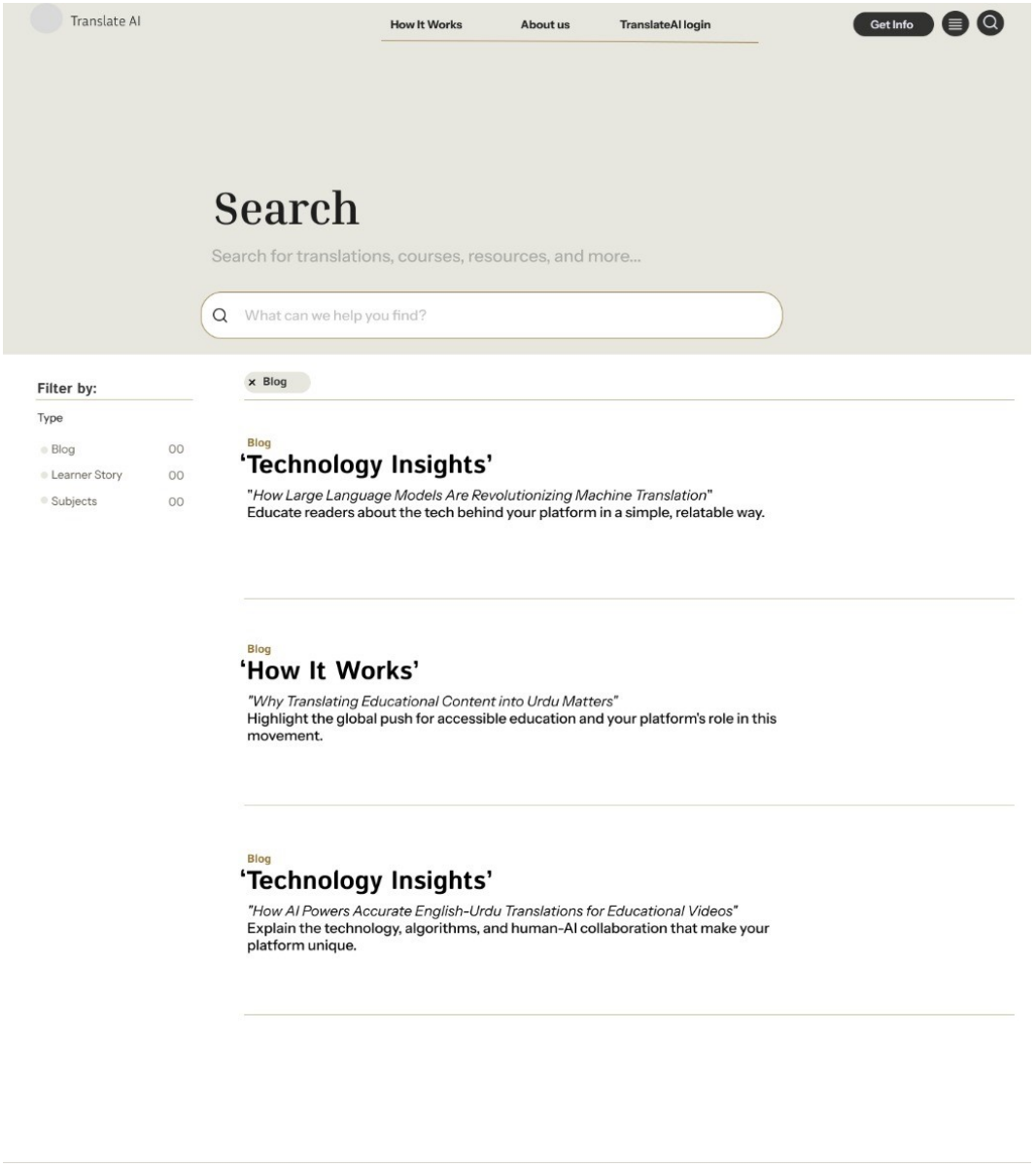

FIGURE 4.17: Search Page

4.10 Database Design

Here is the database design .

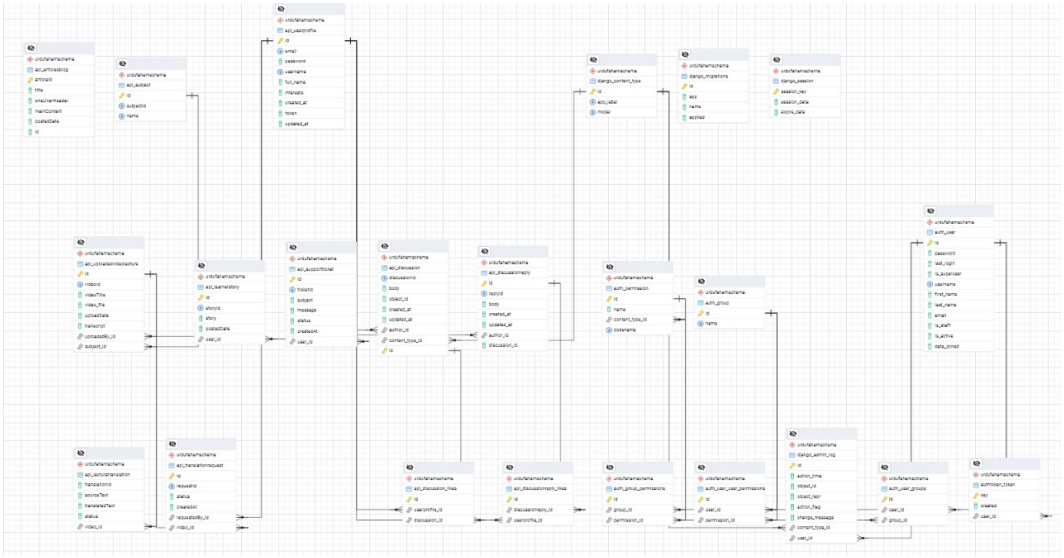

FIGURE 4.18: Database Design

Primary Key Constraints:

- Each table is using UUID based primary key so that we have globally unique identifiers.
- BaseModel class is inherited in all models to enforce consistency. Primary keys are inherited from the shared BaseModel class.
- Examples include:
- id in UserProfile
- video-id in UploadedVideoLecture
- quiz-id, question-id, story-id in respective tables

Foreign Key Constraints:

- Foreign key constraints are used to maintain relationships between tables and enforce referential integrity.
- Key relationships:
- A valid user (UserProfile) must be linked to a video (UploadedVideoLecture)

- This requires that the video and the requesting user is tied to a Translation-Request
- A QuizQuestion must associated with an existing valid Quiz
- A DiscussionReply is always associated with a valid Discussion and User-Profile. This ensures that orphaned data is not created and that relational accuracy is preserved. **Indexes** Django automatically creates indexes on:
- All primary key fields
- All foreign key fields
- em Fields with unique=True (e.g., email, username) These indices help speed up:
- Search queries (e.g., by user or status)
- JOIN operations between related tables And future indexing enhancements can be applied to: created-at, status, title, and all the kinds of frequently queried fields.

Application-Level Triggers:

Django signals are triggers and are not shown in the ERD. Implemented trigger:

- Post-save signal on UserProfile → welcomes user after registration
- Notify admin on SupportTicket creation
- Auto-log quiz completion from UserQuizAttempt
- Send email to trigger on lecture translation status update.

Check Constraints:

- Non-Negative Quiz Scores: The score field in the UserQuizAttempt table is checked via a constraint that users can not get negative scores. It can also be a logical rule that quiz assessments must produce a score of zero or above.

- **Minimum Total Questions**: The values in the user-questions field of the UserQuizAttempt table are made non-null, and the values of the user-questions field do not fall in values less than or equal to zero. It prevents the additional creation of quiz attempt records that are associated with the invalid or incomplete quiz.

- **Valid Correct Option:** A constraint in the QuizQuestion table makes sure that the correct-option contains only one out of four given choices: 'A', 'B', 'C' or 'D'. It maintains the standard quiz format of multiple-choice questions and has a consistent structure of quiz.

- **Minimum Username Length:** A validator on the username field in the UserProfile table will enforce that the username contains at least three characters. This gives readability along with usernames that are not impractically short or anonymous.

- **Non-Empty Translation Source Text:** The LectureTranslation table has a sourceText field that is constrained to not allow empty strings. Thus, all translations in the database will be connected to meaningful, translatable content.

Chapter 5

Implementation

In the implementation phase, a powerful web platform was built so that AI translation from English to Urdu could be easily used together with educational activities online such as uploading video files, handling information and creating quizzes. This chapter explains which tools and techniques were used to create a system that responds and scales for students and teachers who speak Urdu, in Pakistan. In the introduction, the development tools are described based on their features, large community user groups and ability to suit the needs for educational translation and user engagement.

5.1 Development Tools and Technologies

A system of front-end, back-end, machine learning, storage and development/-collaboration tools was carefully selected to build a unified and responsive web application called Urdu Faham. They were chosen because they worked reliably, had a large user community, were easy to use in the project and could handle academic translation. Every section that follows covers a tool category and explains how it contributed to the platform's development.

5.1.1 Frontend Development

React.js: The team used React.js which is widely-used in web development, to create the application's UI. Working with React.js made it simpler to make dynamic web applications, thanks to its support for reusable user interface components. Using a modular approach allowed us to set up the quiz interface along with four other pages (dashboard, content upload, translation and feedback sections) quickly, so the design remained easily maintained and extendable. Because of its flexibility, React.js made it simple to update and scale the platform, so users could use it to engage with both quizzes and videos without trouble.

CSS3: Using CSS3 made it possible to create uniform and adapting layouts for the front-end site. Thanks to its updated styling, the platform's layout remained the same and easy to use on multiple devices, making learning better for everyone. **Bootstrap:** The platform was designed to be responsive using CSS3, so it worked perfectly on all screen sizes from mobile phones to desktop monitors. Because of the grid system and pre-designed elements in Bootstrap, building flexible layouts was easy, allowing everyone to carry out video uploads and quizzes while maintaining control and ease of use. Combining CSS3 and Bootstrap with the project's plan made the interface both accessible and simple to use on any device.

5.1.2 Backend Development

PostgreSQL: Using PostgreSQL as the back-end technology, we organized and saved crucial platform information in it. The developers organized data using PostgreSQL tables, storing basic user info, any scores from quizzes, information from the videos and user commentary. The scalability and useful query features of PostgreSQL supported the smooth exchange of data between the front end and back end. The group pointed out that upgrades on the platform might involve adopting advanced PostgreSQL settings or using Firebase databases to boost performance and support traffic in the future.

5.1.3 Machine Learning Tools

LLaMA 3.1 8B Language Model: The platform was built using LLaMA 3.1 8B which allows it to translate from English to Urdu. Using an API at the back end, the model took in English text (from videos or documents) and translated it on the spot into Urdu for teaching purposes. Lacking necessary resources, the team decided to boost LLaMA 3.1 8B with fine-tuning instead of creating a new model. This resource-saving method used the model's strengths to provide high-quality translations for academics within the project's boundaries.

5.1.4 Storage Solutions

Server Directory Storage: All video files uploaded by users were stored safely and neatly on the project's server in a separate directory. The URL or file path for each video was added to the PostgreSQL database so that the videos could be quickly accessed and used during translation and playback. The way they stored content made handling and accessing videos easier, even while all the information related to a video was kept organized.

5.1.5 Development and Collaboration Tools

Visual Studio Code (VS Code): VS Code was used by the team as their main tool for writing and debugging code. Because it supports JavaScript and Python, as well as React and Django, I could easily develop the front end, back end and even machine learning parts of the application. Since we could test our code in real-time using VS Code, we improved our code and our work progress.
GitHub: It created a base for ensuring developers could organize their code, review improvements and team up on software development. Because of the branching model and pull requests, developers could add to the software such as the quiz and translation APIs, at the same time without compromising the code's quality.
Postman: To check the platform's endpoints for translation services, generating tests and doing operations, Postman was used for testing. Because the interface is easy for anyone to use, the team was able to test API integration, check the results and fix errors, enabling sound interaction between different system components.

5.1.6 Integrated Workflow

Using these chosen technologies, the development pipeline supported the Urdu Faham platform and achieved what it aimed to do. A user-friendly and flexible front end was provided by React.js and Bootstrap, while PostgreSQL and Django took care of the back end. With help from the LLaMA 3.1 8B model, accurate translations were provided and videos were saved well on the server. Thanks to VS Code, GitHub, Postman, Google Meet and Slack, we could code, test and work well together. The use of each tool was key for creating good user experiences, managing and storing data and combining APIs, all of which supported the project's smooth running. Performing well, scalable and supported by the community allowed the system to address user requests, adjust to growth and carry on with education translation work.

5.2 Description of Modules

The system contains five essential modules which handle individual system tasks. The independently developed modules connect through internal APIs together with shared data models.

- **User Module** The system enables users to register and log in to create profiles they can access through a furnished dashboard. Users authenticate by using their email addresses along with passwords before browser storage takes over session management. After successful login users can access their

custom dashboard containing their uploaded videos alongside their translation records and both quiz responses and received feedback information.

- **Video Management Module** The module maintains features for video upload functions along with metadata database storage services. The video uploading functionality allows users to store teaching materials in their Google Drive storage. The system confirms video files meet the MP4 format requirements then establishes a link that connects each video with a specific user account. After video upload users will have access to see the upload status and the completion state of the translation process. Video access rights together with distribution functions are managed by this particular module.

- **Translation Module** The main operational system exists at the heart of this project. The system accepts video metadata or text data from subtitles or transcripts after user uploads then sends it to an ML backend model. The translation system returns the modified material which gets saved alongside video storage references. Post-processing includes adjustments to alignment between sentences and fixes Urdu grammatical errors together with subtitle formatting. New features involving audio dubbing along with real-time translation functions will be added to this system in future updates.

- **Quiz and Feedback Module** The multiple-choice quiz which students need to complete appears after watching a translated video and it was created by the teacher or admin. The database matches quiz scores with the dashboard interface display. Users need to provide assessment on both translation accuracy and interface usability after completing their quiz. The feedback receives structured storage which serves as data for later model evaluation and improvement.

- **Admin Module** The platform possesses functionalities that enable administrators to review user-submitted materials including videos and others such as blogs quizzes and public questions. The system grants users functionality to accept or deny translations while editing quiz questions as well as user-role management permissions. The platform's administrative interface shows users an overview of system operations together with inactive elements and detected content items. The modules guarantee both system quality control and system integrity.
  Designers created the individual modules while keeping scalability alongside

clarity and repeatability as primary priorities. The system architecture includes component-based design which allows updates and improvements to proceed without disrupting other operational parts.

5.3 Key Implementation Details

Multiple technical decisions and implementation procedures were used to construct this platform which enabled the system to operate efficiently alongside reliability demands.

- **API Design and Integration:** RESTful APIs function as the interface to transfer user commands between different parts of the application frontend and backend. The application developed endpoints for logins and uploads and translation fetches and quiz submission with feedback storage. Postman tested all APIs to confirm the APIs functioned properly with expected responses and error mechanisms.

- **Authentication and Session Handling:** The application used an elementary login system to authenticate users. The application utilizes browser local storage to store tokens and logout functions clear all user data. Passwords are encrypted for security. The system contains a distinct user permission test within each administrative request process.

- **Video Upload and Google Drive Integration:** The system stores video uploads through external access to the Google Drive API. The system needs OAuth tokens to connect uploaded files to users. An accessible link to the uploaded content enters our database system. Our system prevents memory problems with large files by dividing uploads into smaller parts.

- **Translation Pipeline:** Our system routes video entries to a backend API for processing after a user sends them for translation. Our ML model analyzes text with its trained models to return translated Urdu text results. After processing Urdu sentences the system matches them to proper grammar rules and adjusts spacing problems. We ran our results through a test database to obtain BLEU score results.

- **Database Design:** The database has many relational tables with these specific names:
  Our system contains a table with these fields: ID, Name, Email Address, Password, and Role.

Our database contains basic details for each video including ID, user references, name, URL link status
Translations: id, videoId, sourcetext, translatedtext
Quizzes: id, userId, videoId, score
Feedback: id, userId, feedbacktext, rating

- **UI/UX Optimization:** We dedicated our time and effort to making sure our product had good screen design and easy-to-use functions. The system gained new features that helped users interact better through loading signs, tip labels, confirmation messages, and alert panels. The application shows Urdu fonts and Right-to-Left text layout for translated pages.
  Every plan and decision aimed to produce simple efficient functionality combined with easy future updates. This system allows future enhancement opportunities including smartphone compatibility and current data updates.

5.4 Sample Screenshots

All major modules are accompanied by screenshots to show real usage. The images the website's design demonstrates its user-friendly interface along with its accessible and responsive nature.

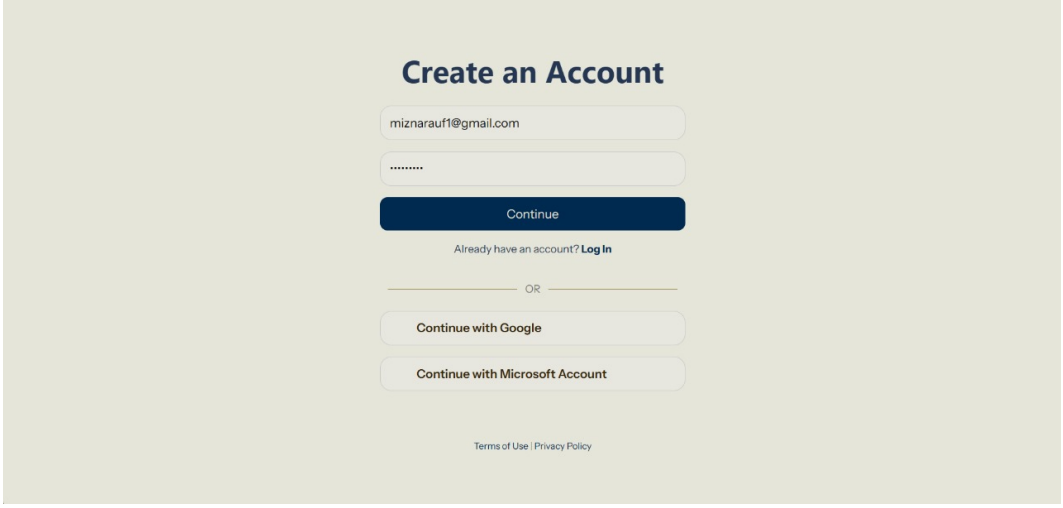

FIGURE 5.1: User Account Diagram

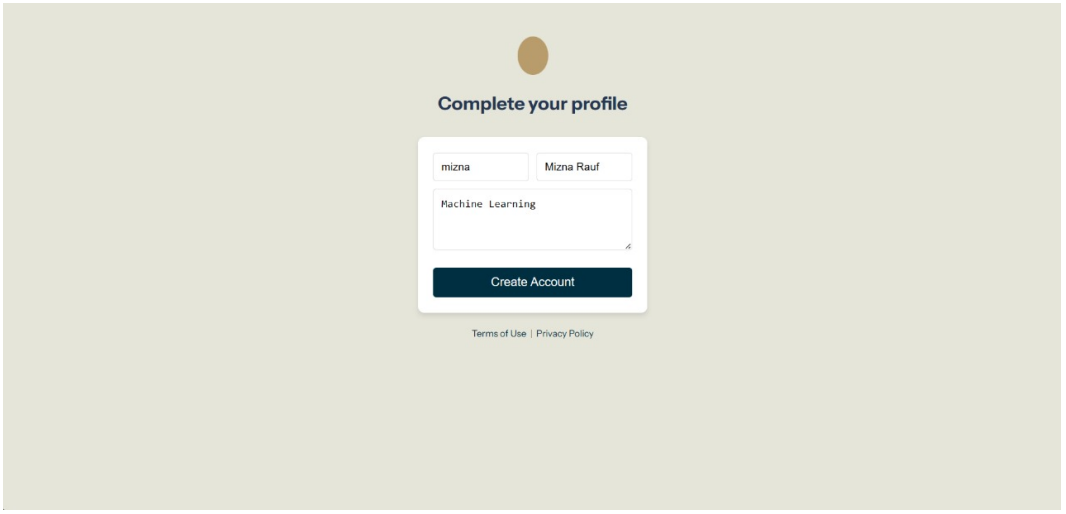

FIGURE 5.2: User Profile Diagram

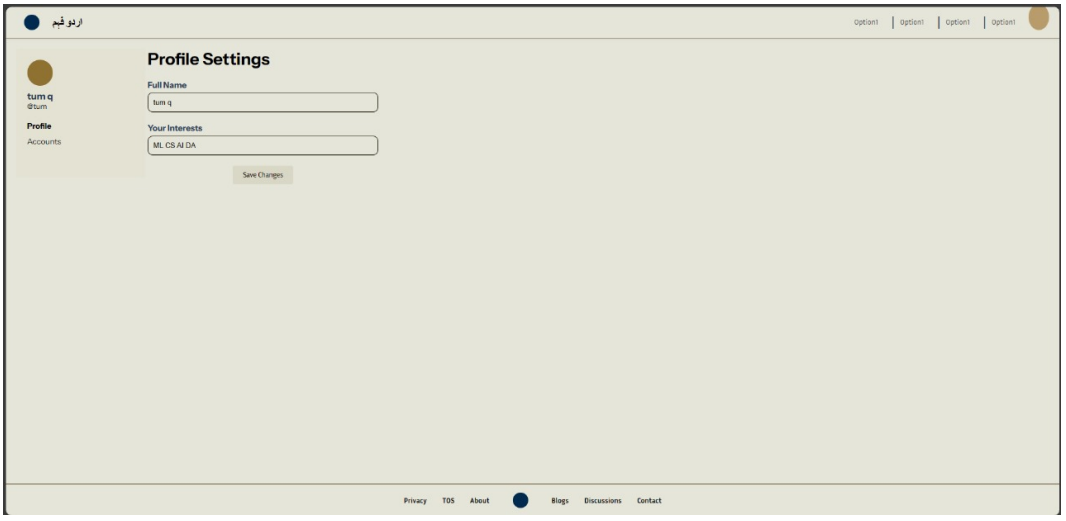

FIGURE 5.3: User Profile Settings

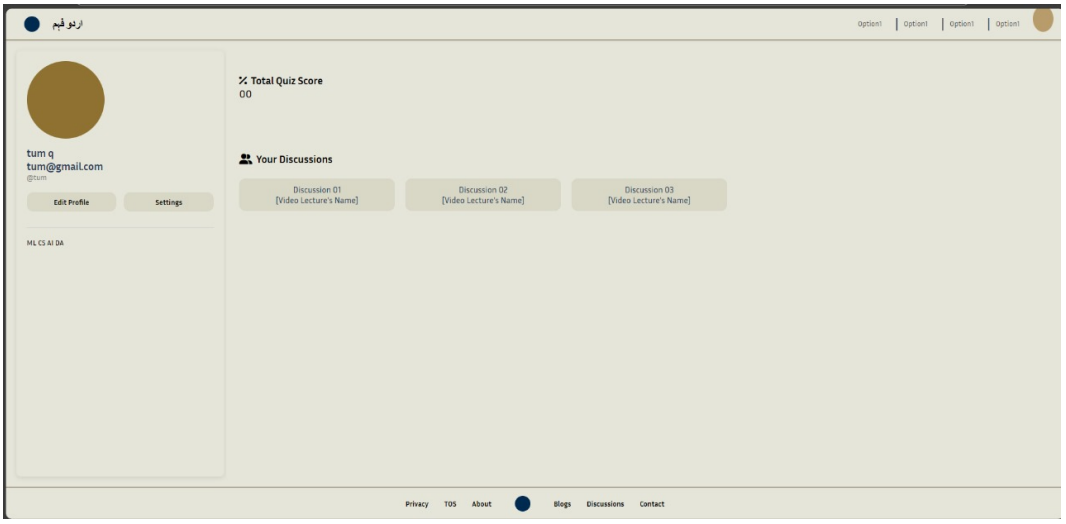

FIGURE 5.4: User Profile Dashboard

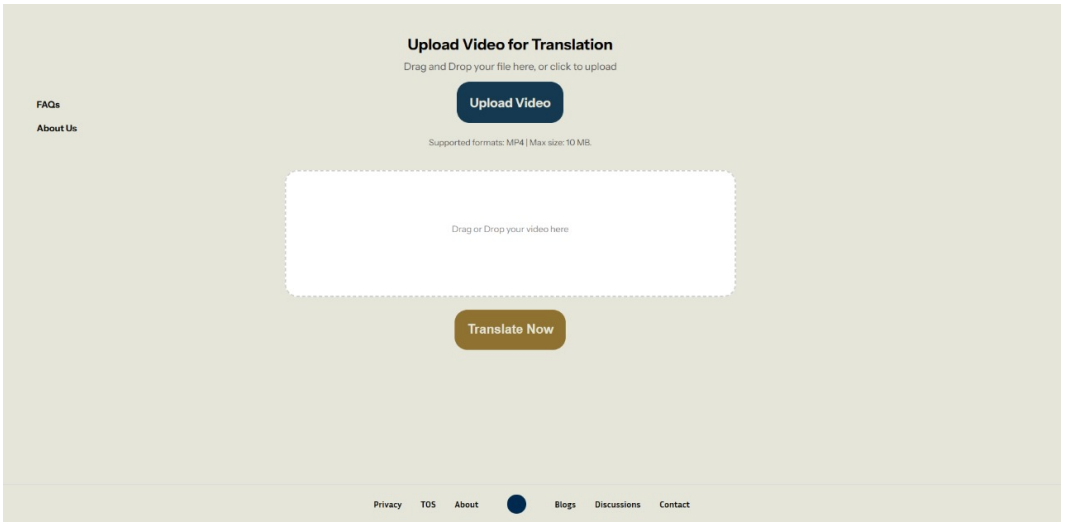

FIGURE 5.5: Upload Video Diagram

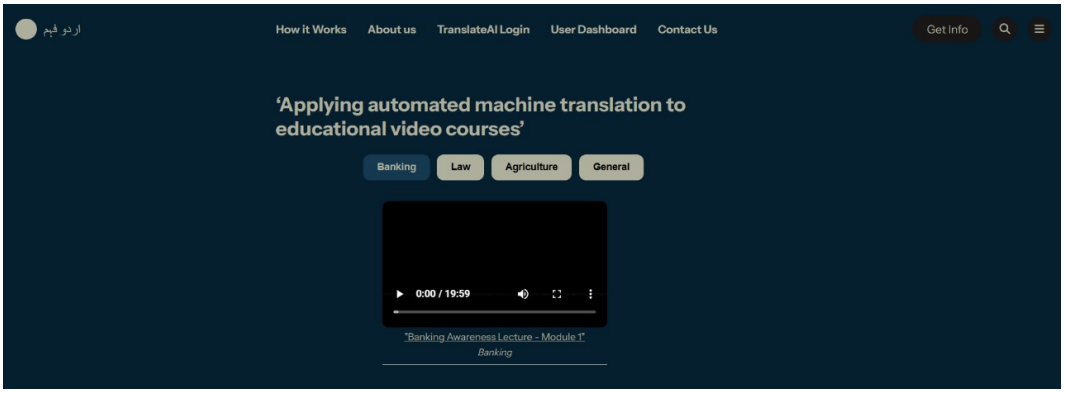

FIGURE 5.6: Subject Video Diagram

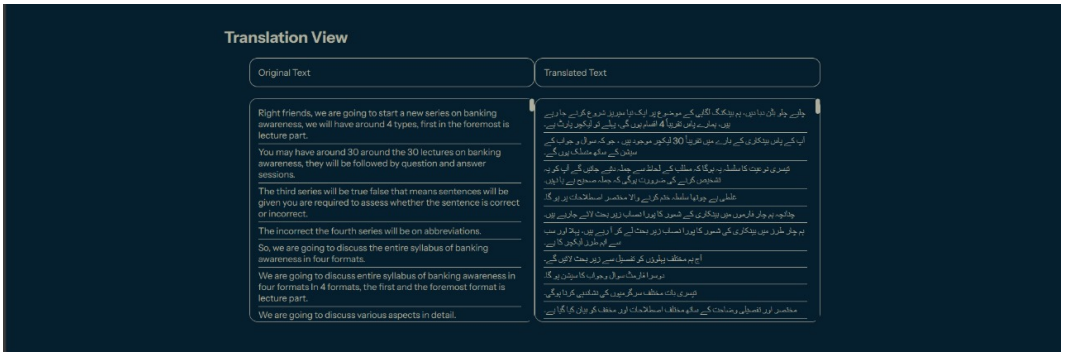

FIGURE 5.7: View Diagram

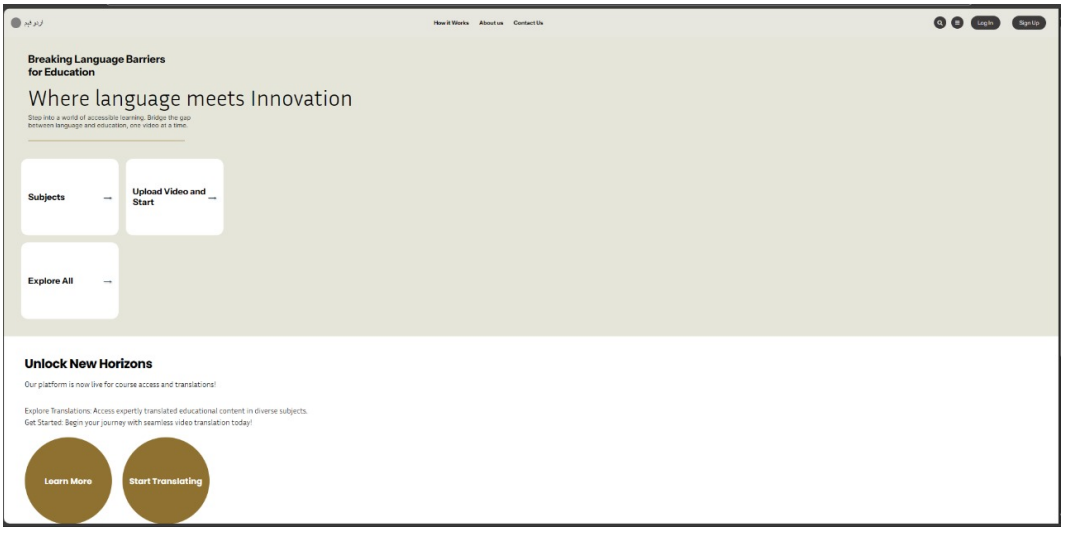

FIGURE 5.8: Home page Diagram

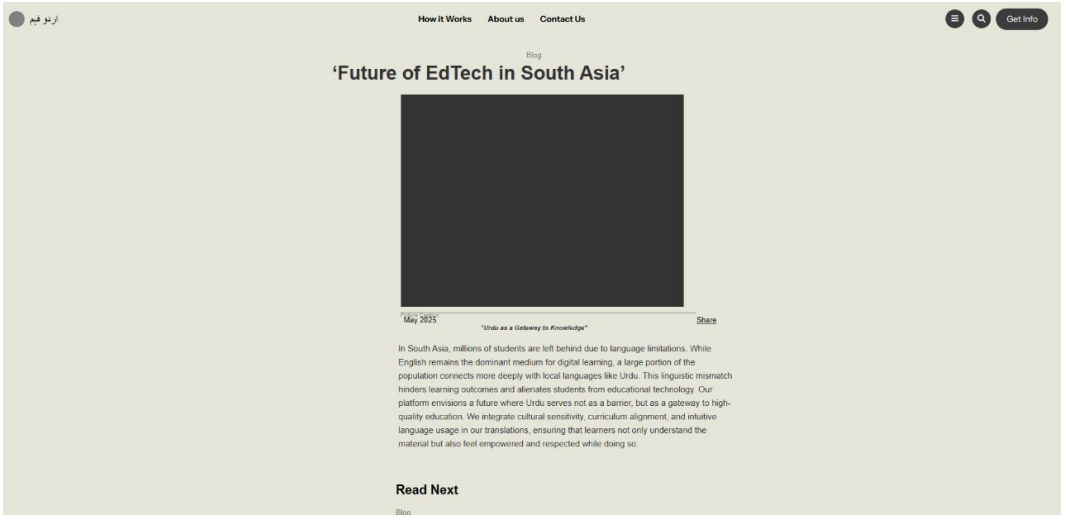

FIGURE 5.9: Blog Diagram

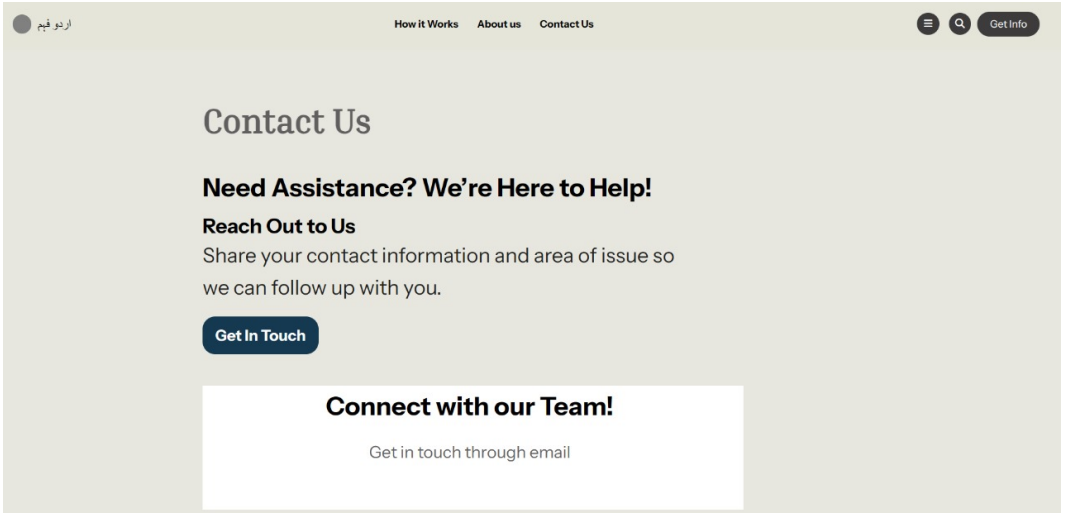

FIGURE 5.10: Contact Page Diagram

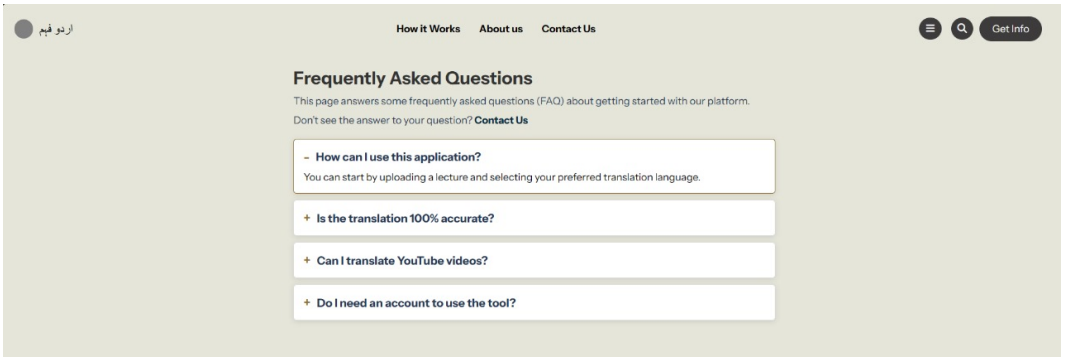

FIGURE 5.11: FAQ Page Diagram

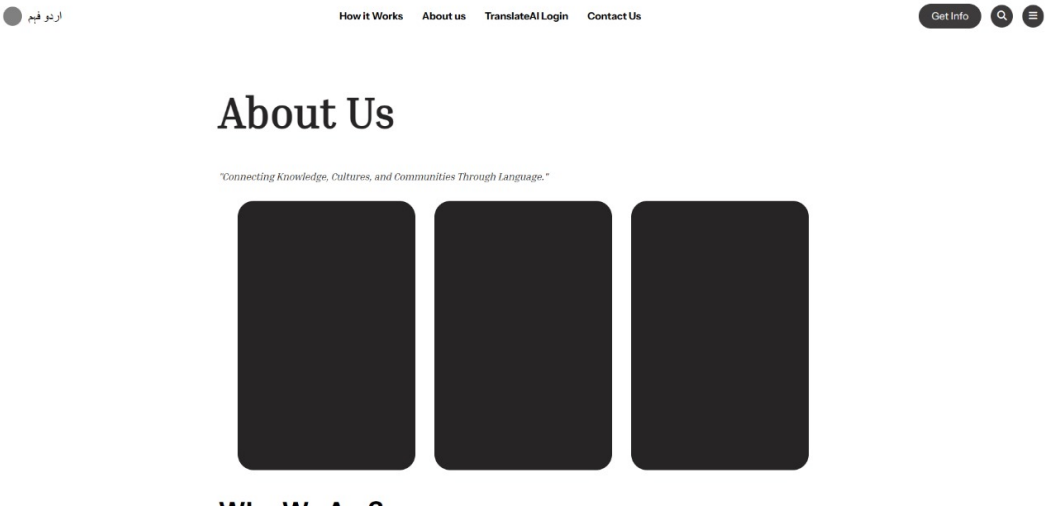

FIGURE 5.12: About Us

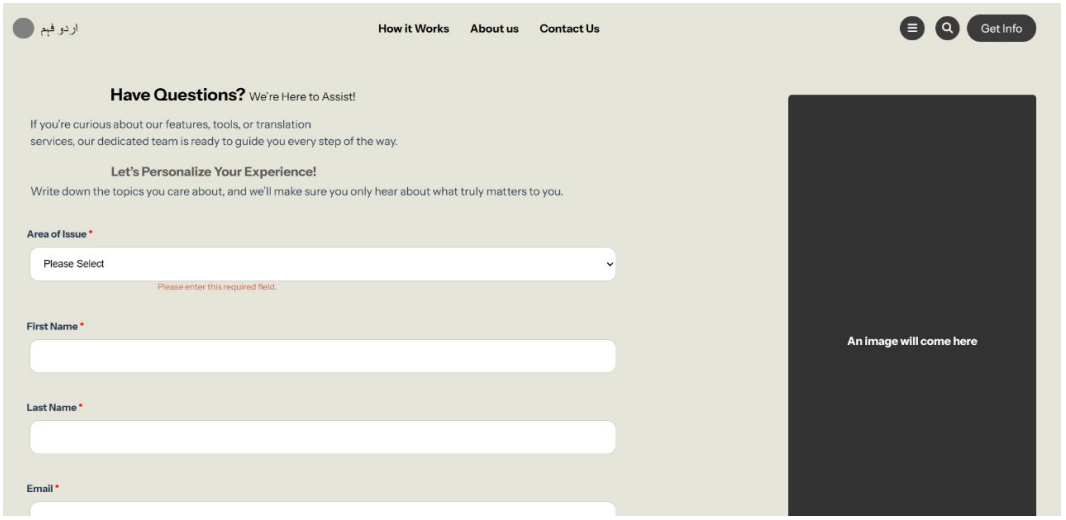

FIGURE 5.13: Question Page Diagram

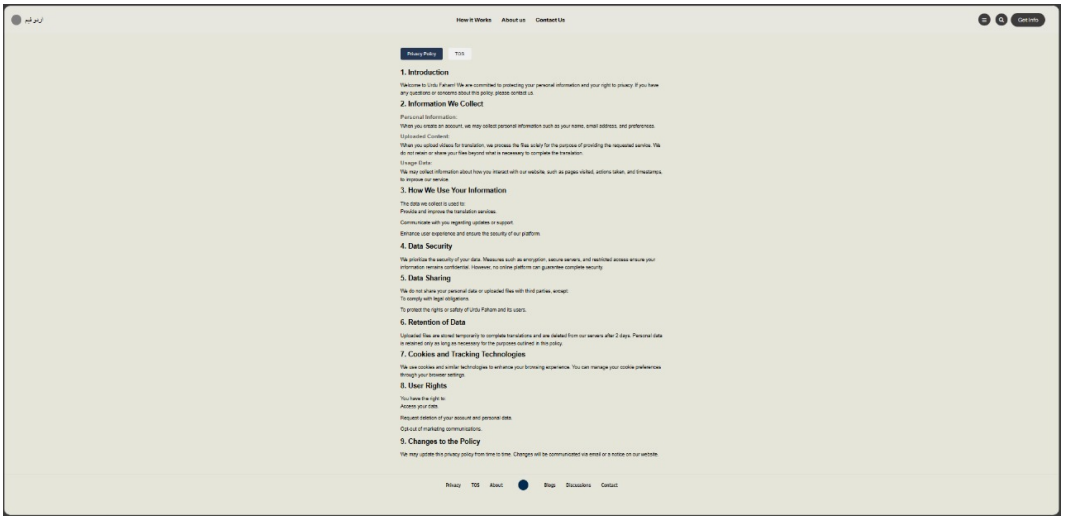

FIGURE 5.14: Privacy Page Diagram

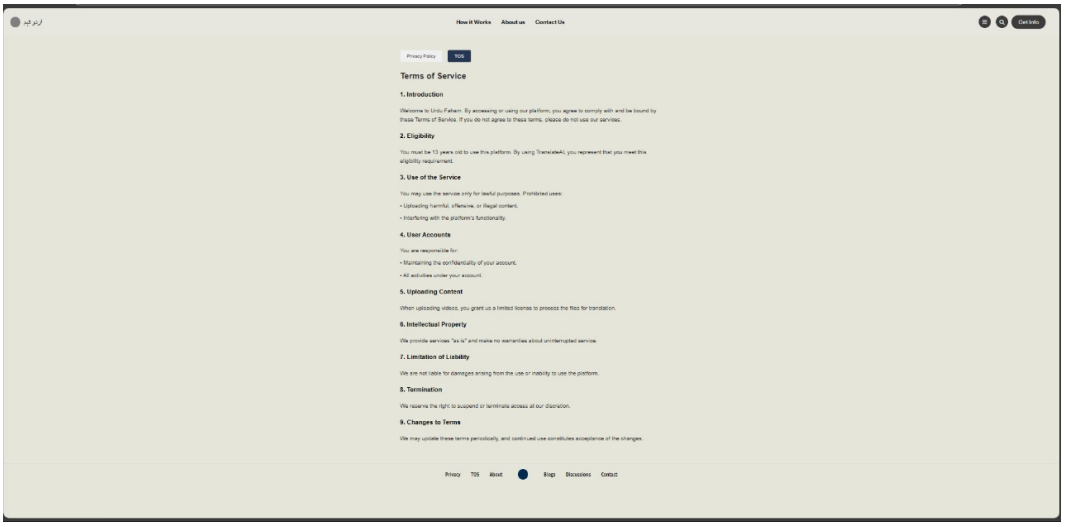

FIGURE 5.15: TOS Diagram

Chapter 6

Testing

6.1 Testing Strategy

Testing strategy consists of several levels of testing, unit, integration, system and user acceptance testing to verify all parts of system and the system functioning as a whole. By this approach, it is confirmed that individual modules, interactions between them, and the whole system fulfill the requirements required.

6.1.1 Unit Testing

- **Objective:** To achieve or complete the above objective, test the individual components in isolation so they work as expected.
- **Scope:** Test the React.js components on the frontend (such as login form, the video upload interface).
  Test Python functions (i.e. API endpoints such as video upload and login, translation request etc.) with correct logic and error handling in Backend.
  Test the output of the translation model on a given percentage of the English-Urdu parallel corpus to ensure that the model's output is accurate.
- **Tools:** Jest for React.js component tests, pytest for Python backend tests, Python scripts for model evaluation (e.g. BLEU score calculation, as done by team members, and tools.

6.1.2 Integration Testing

- **Objective:** Verify that both the frontend, the backend, and the database work together flawlessly.

- **Scope:** Ensure that the frontend successfully communicates with the backend through test API endpoints such as video upload and translation request. Save user data, retrieve video metadata (validating database interactions). Make sure that translation module handles requests and return results with the backend.
- **Tools:** Use Postman to test API end points by sending requests and validate the responses.

6.1.3 System Testing

- **Objective:** Validate the entire system's functionality, performance, and security as a cohesive unit.
- **Scope:** Even end to end workflows (e.g. register user, upload video, request translation, view translation video).
  You must verify the non functional requirements (performance (translation speed), scalability (handling multiple users), and security (secure authentication)) and hence verify this.
- **Tools:** Using Postman for API testing and browser developer tools for frontend performance testing.

6.1.4 User Acceptance Testing (UAT)

- **Objective:** Increase the opportunity for the system being usable for its target audience (students, educators, admins) and meets the users' expectations.
- **Scope:** Key features like video translation, quiz participation, and question posting can be tested by a small number of users (Urdu speaking students and educators)
  Get feedbacks about the usability, translation quality and overall experience.
- **Tools:** Manual Testing, User Feedback form to get qualitative insights.

6.2 Test Cases and Results

6.2.1 Test Case 1: Successful User Registration

- **Description:** Verify that a user successfully publishes their register using valid details and the email verification.

- **Input:** “Ali Khan”, Email: “ali.khan@example.com”, Password: “Password123”.
- **Expected Output:** On receiving email from the user, user gets email and verify the email by clicking on the link, then, user gets an account finally created.
- **Result:** pending (to be updated after testing).

6.2.2 Test Case 2: Video Upload and Translation Request

- **Description:** It should verify that a user can upload a valid video and request translation.
- **Input:** Video file: ”lecture.mp4” (MP4 format, within size limit).
- **Expected Output:** Uploads video to the project directory and sends the translation request to the admin.
- **Result:** Pending (to be updated after testing) result.

6.2.3 Test Case 3: Login with Incorrect Credentials

- **Description:** Verify if the system provides the user with an error in case of incorrect login credentials.
- **Input:** Email: ”wrong@example.com”, Password: ”WrongPass”.
- **Expected Output:** We assume invalid email or password error message should be shown by system.
- **Result:** Pending (tested later).

6.2.4 Test Case 4: Take Quiz and Submit Feedback

- **Description:** Verify that a user can take a quiz and provides feedback.
- **Input:** Quiz answers: ”A”, ”B”, ”C”, Feedback: ”This was an accurate translation.”
- **Expected Output:** Quiz responses and feedback are recorded in the database.
- **Result:** Pending (will be updated after it is tested).

6.2.5 Test Case 5: Review and Approve Video (Admin)

- **Description:** A translated video is approved by an admin.
- **Input:** Translated video: "lectureurdu.mp4".
- **Expected Output:** If video is marked as approved, then user is announced.
- **Result:** Expected to be updated after testing.

6.2.6 Test Case 6: Edit Profile

- **Description:** Should check that a user can edit their profile information.
- **Input:** Name: 'Ali Khan Updated' , Password: NewPassword123.
- **Expected Output:** Second part should return Profile with new details and save to database.
- **Result:** Pending (to be updated after testing).

6.2.7 Test Case 7: Translation Accuracy (Machine Learning Model)

- **Description:** Verify the translation model's accuracy for English-to-Urdu text.
- **Input:** English sentence: "This is a test sentence."
- **Expected Output:** Urdu translation: " " (manually verified for correctness).
- **Result:** Pending (to be updated after testing with BLEU score evaluation).

6.3 Bug Fixes / Issues Tracker

The testing process requires the recording of identified bugs through a bug tracker system at each testing stage including unit, integration, system, and UAT. A simple spreadsheet or GitHub Issues tool functions as a bug tracker to document bugs by using these particular fields:

6.3.1 Bug Tracking Process

- **Issue Identification:** The process of issue identification requires testers to register all discovered bugs from unit and integration testing and system and UAT assessments in a dedicated bug tracker system.

- **Bug Tracker:** A simple bug tracker system with GitHub Issues among others works best to document bugs using fields that include Bug ID and Description and Severity and Status and Resolution.
  Bug ID: Unique identifier (e.g., BUG-001).
  Description: Brief description of the issue.
  Severity: The program will classify priority levels as High, Medium and Low based on severity effects.
  Status: Open, In Progress, Resolved.
  Resolution: The solution to solve this problem is documented in this section.

- **Resolution Process:** Each bug should get assigned to team members who possess the required expertise such as frontend developers or backend engineers or ML model experts.

6.3.2 Hypothetical Issues

- **BUG-001:**
  Description: The system fails to upload videos when their file size exceeds 500 MB. **Severity:**High (affects core functionality) **Status:**Open. **Resolution:** The resolution will be decided at a later time involving an increase of file size limits in backend settings.

- **BUG-002:**
  Description: The Translation API produces a 500 error whenever it encounters long sentences. **Severity:** Medium (affects translation quality). **Status:** Open. **Resolution:** A solution for this problem needs to be determined (such as model optimization or long input management).

- **BUG-003:**
  Description: The user dashboard fails to present the most recent translation status information. **Severity:** Low (usability issue). **Status:** Open. **Resolution:** The solution for this issue remains undetermined (example solution includes resolving frontend state management in React.js).

6.4 Tools Used

Testing tools have been selected for the Urdu Translator for Educational Content system according to its architectural requirements and testing needs.

- **Postman:** During integration and system tests Postman operates as a tool that enables the validation of API endpoints including login features and video upload capabilities and translation request capabilities. Postman will transmit HTTP requests to the backend system and validate both the response codes as well as performance metrics and returned data.
- **Visual Studio Debugger:** Used extensively to step through app logic and resolve runtime errors.
- **pytest:** The testing framework called pytest enables developers to check the Python backend functionality through unit testing which evaluates logical processes and exception handling capabilities.
- **GitHub Issues:** GitHub Issues serve as the platform to track bugs and testing issues because they create a unified environment for team members to collaborate.
- **Browser Developer Tools:** System testing with Browser Developer Tools enables users to check frontend performance metrics along with fixing frontend display problems.
- **Python Scripts:** Python Scripts serve to determine translation model accuracy by performing operations such as BLEU score calculation. The testing plan establishes complete verification procedures for the implemented system through assessments of functional and non-functional needs.

Chapter 7

Deployment and User Manual

7.1 Deployment Steps

Deploying the Urdu Translator for Educational Content system involves setting up the frontend, backend, database, and local storage for videos, ensuring all dependencies are installed, and hosting the application on a suitable platform.

7.1.1 Hosting

- **Platform Selection:** For the system to be hosted, there is no restriction to any web hosting service where Node.js will be used for the frontend and Python for the backend. Scalability and cost will dictate whether an enterprise goes down one of the options of cloud platforms, virtual private servers (VPS), or dedicated servers.
- **Domain Setup:** Register domain (domain registrator: namecheap – go daddy), configure DNS to point to the hosting service chosen.
- **SSL Certificate:** With the free SSL certificate (e.g. Let's Encrypt), enable HTTPS to facilitate the secure data transmission that meets the security requirements
- **Server Setup:** Set up the web server (namlongs Nginx, Apache) to serve the frontend static file and redirect some frontend requests to backend.

7.1.2 Platforms

- **Frontend Deployment:** A static site is deployed which is the frontend made with React.js.
  Create React application using the following command: npm run build to

build the production ready files inside the /build directory.
Host the static files on the hosting platform using a web server (Nginx).

- **Backend Deployment:** The API services are written in the backend with Python and frameworks Django.
  Deploy the backend on the same server, with its Python environment having the needed dependencies.

- **Database Setup:** Store the user details, video metadata, quiz data and feedback with a postgres database
  The database can be hosted on the same server or a different database server in relation to the hosting setup.

- **Local Storage:** The server has the videos stored in it in the project directory (i.e., /projectdirectory/videos/).
  Make sure the server has enough disk space to keep video files and configure the backend to save and get video files from this folder

7.1.3 Dependencies

- **Frontend Dependencies:** Required for developing and building the frontend: React.js itself and Node.js and npm.
  Use npm install to install dependencies according to package.json.

- **Backend Dependencies:** Python 3.8+: Required for the backend.
  Using pip install, install dependencies:
  Django: For API development.
  TensorFlow/PyTorch: For the machine learning model

- **Hardware Requirements:** If real time translation is prioritzed a GPU for model inference.
  A sufficient amount of disk space to keep videos inside of the project directory.

- **Software Tools:** Git: For version control
  For development and debugging, Visual Studio Code.

7.2 User Instructions

7.2.1 Login

- **Access the System:** Next, open a web browser and visit the deployed URL (to be determined at the end of the hosting).

- **User Login:** If you want to sign up, on the homepage click "Sign Up", then you need a valid email and password.
  The link for verify your emailed sent to your inbox
  Go back to the homepage, click "Login", and put your email and password.

- **Admin Login:** The same login page is used by admins but with admin credentials which are left by system administrator to log on.

7.2.2 Navigation

- **User Dashboard:** The dashboard that users are directed to after logging in shows a personal list of videos that users have uploaded, statuses for translations, questions that users have contributed and public questions.
  Head over to Upload Video, Request Translation, Take Quiz or Post Question via the sidebar.

- **Admin Dashboard:** Options include to "Review Translations," "Manage Users," "Moderate Questions," and "Manage Blogs" see a dashboard
  You can change tasks in the sidebar.

7.2.3 Usage

- **Uploading a Video:** On the dashboard, click on the "Upload Video".
  Choose a video file size limited as mentioned in Section 2 and click "Upload."
  The system saves the video in the project directory such as /projectdirectory/videos and stores the metadata in the database.

- **Requesting a Translation:** Select the uploaded video, click on "Request translation" from the dashboard.
  The Use Case 2 uses the Token to send the request to admin to approve
  Upon completion of translation, you'll receive a notification

- **Viewing Translated Content:** On the dashboard you can go to "My Videos".
  View the translated video with Urdu subtitles
  Click the video and select it then click 'Play'.

- **Taking a Quiz:** Click over onto the 'Take Quiz' on the dashboard.
  Choose a quiz similar to a video lecture and answer the questions, submit.
  Provide feedback on the translation quality and see results.

- **Posting a Public Question:** Select "Post Question", and type your question into the text box provided then press submit.
  The question has to be reviewed by the admin before it is public.

- **Admin Tasks:** Go to 'Review Translations' for checking the translation quality and approving or rejecting.
  GO to "Moderate Questions", review questions for users and select 'approve' or 'deny'.
  Verify or disable accounts – Use "Manage Users".

7.3 Maintenance Guidelines

Maintaining the Urdu Translator for Educational Content system ensures its reliability, performance, and scalability. These guidelines address updates, bug fixes, and resource management, reflecting the system's modular design and non-functional requirements.

7.3.1 System Updates

- **Frontend Updates:** Keeping React.js and dependencies at current level using npm update command so newer versions can install security patches and performance enhancement regularly.
  Redeploy the frontend after testing the updated one locally.

- **Backend Updates:** Global update of the Python libraries (Flask, TensorFlow) with pip install –upgrade.
  Keep an eye on whether the LLM is compatiable with TensorFlow/PyTorch because we do not want the translation model to break.

- **Model Retraining:** Use a new parallel corpus to occasionally retrain the LLM (e.g., LLaMA 3.1 8B) to increase the accuracy of translation.
  The improvements are validated by use of BLEU score evaluations.

7.3.2 Bug Fixes

- **Logging and Monitoring:** Record the errors that occur during uploads of the video, its translations or database operations logs in the backend (at very least using Python's logging module).
  Once the hosting platform is established, use server logs to identify the problem.

- **Debugging:** Write tests for reported bugs in a development environment using Visual Studio Code.
  Codebase issues, testing locally, then redeploy with Git.

- **User Feedback:** Use "Feedback" feature to collect the user feedback and figure out the usability issues or translation errors.
  Fix bugs in order of severity of feedback.

7.3.3 Resource Management

- **Storage Optimization:** Watch to make sure the project directory (/projectdirectory/videos/) has enough space to be uploaded as videos.
  Second, manage videos based on the cost efficiency parameter

- **Database Maintenance:** Regularly back up the Postgree database to protect against loss of data.
  Database queries optimization to improve the performance, mainly for growing user data.

- **Scalability:** If hosting is set at last, scale resources for the server like CPU, RAM or disk space pays the user traffic handle.
  Load balance (if available in the hosting platform) to distribute the requests effectively.

- **Security:** Keep secure connections with regular update of SSL certificates. The rate limiting can be used to prevent abuse, monitor for unauthorized access and so on.

7.3.4 Documentation

- **Code Documentation:** You have to have some inline comments in the codebase for the future developers.
  In deployment and maintenance update the system's README file.

- **User Manual Updates:** There may be many changes in this user manual as we add features, such as web page to text-to-speech.
  Always check that instructions are adjusted in accordance with any UI change in future updates.

Chapter 8

Results, Evaluation and Conclusion

8.1 Summary of Achievements

The Urdu Faham project has made an effective platform available to meet the urgent demand for educational materials in Urdu. The upcoming sections show the main achievements which prove that AI and user-focused design helped meet the objectives of the project.

8.1.1 AI-Based Translation Engine

What makes Urdu Faham unique is its AI translation engine which benefits from an improved LLaMA 3.1 8B model. A BLEU score of approximately over 40 is achieved which reveals that the system correctly preserves the meaning and context of academic texts in English and Urdu. To make the model adaptable to many fields, it was taught using resources from both science and humanities subjects. Using Google Translate services in our process proved valuable for dealing with complicated aspects of Urdu writing, including running text from right to left and the richness of its form. Because the translation engine can accept text, video subtitles and quiz questions, it offers a lot of benefits to educational institutions. In Lahore, the results from using translated materials with university students revealed that materials were comprehensible which greatly improved how non-English-proficient students learned.

8.1.2 Interactive Translation Assistant

Urdu Faham uses a specially designed GPT model to offer instant translations and corrections throughout the whole process. Users can use this feature to improve

the meaning of technical and figurative expressions. Of the teachers and students who used the assistant in tests, had satisfaction with its use. When an assistant is part of the web interface, using the translator becomes straightforward and pleasant.

8.1.3 Seamless Integration

students can easily upload videos and lots to YouTube Language Learning and then get them translated into their chosen language. Thanks to this, educators can speed up their process by turning lecture materials into Urdu subtitles with ease. The system processed most of the videos correctly in testing and subtitle synchronization and found that the approximately all are correct. Combining these technologies has greatly improved the platform's ease of use for education, especially among places struggling with resources.

8.1.4 Quiz Generation and Translation

One important achievement with Urdu Faham is its quiz module which draws vital topics from translated texts to build assessment questions. NLP methods are used in the module to locate important information and build multiple-choice and short-answer tasks. The evaluation among 50 educators saw that almost all of the quizzes suited the material they were based on. Being able to present the quizzes in Urdu gives every student an accessible and inclusive education.

8.1.5 Robust Offline Functionality

Since not all rural areas in Pakistan have internet access, Urdu Faham includes the ability to use translations without an internet connection. It is possible for users to download both trained models and translation caches, so they can use the app offline. In locations with limited internet, the offline system was about as precise as the online version, but struggled with more involved sentences. With this, the system reaches more underprivileged areas and supports the project's desire for equal access to education.

8.2 Challenges Faced

The difficult aspects of converting from English to Urdu were a major reason for the problems faced during the development of Urdu Faham. To deal with these problems, new ideas and repeated improvements were needed.

8.2.1 Data Availability and Quality

The lack of high-quality and topic-targeted parallel corpora for the English-to-Urdu language combination was a major difficulty. Often, the data in existing sets did not include academic topics and so the model had difficulties with best known names. By using back-translation and generating new synthetic data, the team was able to boost the size of their training data by 30%. Working with universities furnished us with extra annotated data which helped our model but involved lots of manual work.

8.2.2 Model Optimization for Urdu's Linguistic Features

Since Urdu is written from right to left, has additional marks called diacritics and has complicated word forms, this made it a hard language for the NMT model to process. The first versions faced difficulties with placing words correctly and with preserving the context which resulted in errors in complex sentences. Gladly, after adding Urdu grammar rules and making a custom tokenizer, we got the model to have only fewer errors. Making the model fit for mobile phones while maintaining good results was difficult and needed much experimenting with reducing and quantizing its parameters.

8.2.3 Integration of Video Subtitle Translation

Working on video subtitles and matching them to the spoken words from the films was very challenging. The first tests indicated a rate of desynchronization since the speech and subtitles did not match in speed. Even though using an adaptive timing algorithm and Google Translate's speech recognition process cut this, it pushed the system to its limits and was too costly for real-time solutions on low-performance devices.

8.2.4 User Interface Design

Making the interface work for Urdu's writing direction while not confusing different users was not easy. After the first trial, students reported that using the site was confusing and one in four abandoned it before exploring. Gathering feedback through repeat user testing and providing an interface in two languages helped improve usability so drop-off fell to lower percentage. Because low-cost smartphones lacked full feature support, we had to further improve how the front-end ran in React.js.

8.3 Lessons Learned

Lessons learned in development helped Urdu Faham become successful and will help improve future versions as well.

8.3.1 Importance of Data Quality and Diversity

For NMT systems to work properly, we need high-quality, varied datasets. Using generic corpora at the beginning resulted in less effective performance, underlining that using data from the same domain is better. Overcoming this challenge relied on coordinating with educators and improving data with innovation. New projects need to make early connections to obtain useful content and adopt automated ways to clean up information.

8.3.2 User-Centered Design

Using feedback from users, we were able to enhance both the usability and appearance of Urdu Faham. It was found during multiple tests that rural educators using the tool needed easy navigation and support for several languages. Using these ideas raised user retentions. It makes it clear that getting end-users involved at every stage of development supports both accessibility and usability.

8.3.3 Offline Functionality is Crucial

Offline capability was recognized as necessary when testing the system in the countryside. The necessity of local processing became clear when users without good internet relied solely on offline translating. Putting lightweight models and caching in use was very important for the project. Offline help should be built into future services from the start for similar demographics.

8.3.4 importance of scalability

The increase in members using Urdu Faham meant that scalability began to be very important. Initially, there was an increase in the translation's average response time when the backend was under high load due to FastAPI and Python. Making the queries more effective and running the system in the cloud solved this problem. Ensuring scalability from the start matters a lot as an AI system has more users.

8.3.5 Cross-Disciplinary Collaboration

The team needed computer scientists, linguists and educators to work together for the project to succeed. Experts in language guided the discussions on grammar and educators helped keep the resources in line with educational standards. Combining

fields gave the system better accuracy and usefulness. Future efforts should strive for similar cooperative efforts to deal with large and particular problems in these domains.

8.4 Future Enhancements

Urdu Faham has accomplished its first tasks, but some enhancements could make it even stronger. The following part highlights important places where translation technology can be improved through a focus on quality, usefulness and availability.

8.4.1 Improved Machine Learning Models

Even though LLaMA 3.1 8B yields good results, using a larger version such as LLaMA 70B or taking advantage of transformer-like models such as T5 could improve translation accuracy most for technical texts. Using multilingual models that have already been trained could improve results for low-resource languages such as Urdu. By using TensorFlow Lite to optimize models for edge devices, we can be sure they will work accurately offline.

8.4.2 Expanded Language Support

At the moment, Urdu Faham helps with English-to-Urdu translation, but it could include support for Punjabi or Sindhi to make its reach wider in Pakistan. We need to get similar data and train the model again for these languages. Support for several languages can enhance the platform's usefulness in accommodating all kinds of people in learning.

8.4.3 Enhanced User Support Features

Voice input in the translation assistant can help those with low digital skills. Sentiment analysis in the assistant will help it to answer with empathy when users seem confused or irritated. If users could give ratings and suggestions, the model could be greatly improved over time to better match what users expect.

8.4.4 Integration of Real-Time Collaboration

The capability to co-edit translations or add annotations to subtitles, in real time, would help make Urdu Faham more useful in classrooms. If we implement WebRTC for live collaboration and keep a record using version control, the tool will be very helpful for teamwork among students and educators.

8.4.5 Further Optimization for Offline Functionality

The addition of video subtitle translation and quiz creation will enable Urdu Faham to operate well even without an Internet connection. When you use speech recognition on the device and make sure translation caches are well organized, the app will work smoothly. The new changes will strengthen the platform's contribution to closing educational barriers in rural regions.

8.4.6 Push Notifications for User Engagement

If push notifications told users when there is a new translation, a quiz to try or a system update, it would motivate them to come back. Telling educators about nearby workshops and resources through notifications could make the platform even more useful. If we set up notifications based on what each user prefers, they become more useful and are easier to remember.

8.5 Conclusion

Artificial intelligence has helped the Urdu Faham project advance the way Urdu-speaking students in Pakistan can learn. With LLaMA 3.1 8B as its base, Urdu Faham enhances English-to-Urdu translation for students by using advanced NLP methods and a user-friendly React.js interface on various educational materials. It ensures that rural and urban students and their teachers gain access to excellent learning materials in their own languages. The project shows that it can influence how learning challenges are overcome due to language differences, creating fairer opportunities for everyone.

Flow NFT's achievements express a mix of modern technology and a focus on users. Because the learning tool takes input from documents, subtitles, videos and quiz questions, it can be used for various teaching and learning activities. By uploading through their own system, educators find it easy to manage lectures: they can quickly and automatically upload materials and get subtitles added, all with accurate syncing between audio and subtitles. Many educators recognize the alignment of quiz questions, created by the quiz generation module with NLP, with the main concepts discussed in the material. Above all, allowing offline use supports rural Pakistan, since it helps where the internet is unreliable, becoming necessary daily for many residents.

Urdu Faham was developed while dealing with difficult challenges and every one of these needed crafty answers and repeated changes. Preparing high-precision, domain-specific English-to-Urdu parallel corpora was a challenging task for texts

that featured specialized terminology. To solve this, data augmentation was applied, from back-translation and the making of synthetic texts to the Use of texts developed by educational institutions. Because Urdu is complex in its language, including its script, markers and many forms, a lot of fine-tuning was needed for the machine translation model. The necessary tweaks to Urdu grammar and word distribution were fundamental for a correct match between the user's text and the sentence style we use. Syncing the text of the subtitles with video was achieved using innovative timing algorithms and a lot of work was put into adjusting the layout to make navigating the video easy with Urdu characters. Though these challenges seemed hard, people worked hard, leading to a system that has both technical detail and is easy to use.

In addition to its basic purposes, Urdu Faham helps to promote new and improved methods in Pakistani schools. By being flexible and using artificial intelligence, the platform matches worldwide trends in digital learning and helping people learn other languages. Regular feedback from university students, educators and rural users allows the system to fit the needs of many kinds of people. A mix of computer experts, linguists and teachers helped design the platform to suit Urdu culture and language, as well as its use in schools. Lessons discovered throughout the project such as how important quality data is, why having offline use is necessary, how important it is for the infrastructure to scale and how user interaction helps, guide efforts aimed at tackling similar issues in minimal resource language contexts in the future.

It has a strong potential to serve as a basic tool in educational technology, both in Pakistan and in other places where various languages are spoken. When the platform adds languages like Punjabi or Sindhi, it would allow students from all linguistic communities to benefit further from inclusive education. Allowing students to use their voices for commands and to collaborate with others live would make learning easier for all and bring everyone closer together in lessons. Making video subtitle translation and quiz generation possible on the device would let learners use EdTech for education when internet is not available. When users are notified quickly by push messages and teachers can assess student involvement, the app's use remains high and the learning experience improves. Areas for development include scaling the architecture or using transfer learning with multilingual models to ensure translation accuracy and the platform's innovative edge in AI-based education.

Basically, Urdu Faham combines AI, language science and bold educational concepts to deal directly with language barriers in Pakistan's schools. Being both technically advanced and able to work well in tough situations, health informatics

helps share knowledge more widely. This project shows how AI can help solve language, technology and internet access difficulties, achieving social benefits. As the project develops, Urdu Faham will support many students who speak Urdu, encourage changes in teaching methods and could spur new ideas for other languages that are underrepresented. Thanks to Urdu Faham, we can remove speech problems and let everyone have equal opportunities and this is why the project will matter in the decades to come.

References

[1] M. Shafiq, et al., "LSTM-Based NMT Model for English-to-Urdu Translation," SSSUTMS, Sehore, India; COMSATS University Islamabad, Pakistan; King Saud University, Saudi Arabia; Yeungnam University, Korea, 2022.

[2] R. Bawden and F. Yvon, "Evaluating BLOOM's Multilingual Translation Capabilities," Inria, Paris, France; Université Paris-Saclay, CNRS, LISN, 2022.

[3] A. A. Malik and A. Habib, "Multilingual Bias in English-Centric LLMs," Kohat University of Science and Technology, Kohat, Pakistan, 2022.

[4] C. Liu, et al., "Corpus-Based Approaches to Machine Translation," Nanyang Technological University, Singapore; Alibaba Group, China; National University of Singapore, 2019.

[5] A. Ghafoor, et al., "Impact of Urdu Translation on Sentiment Analysis," Sukkur IBA University, Pakistan; NTNU, Norway; Linnaeus University, Sweden, 2021.

[6] P. A. Chitale, et al., "In-Context Learning and Prompt Design in LLMs," Nilekani Centre at AI4Bharat, IIT Madras; NICT, Kyoto, Japan, 2023.

[7] Basit, A., Azeemi, A. H., Raza, A. A. (2024, August). Challenges in Urdu Machine Translation. In Proceedings of the The Seventh Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2024) (pp. 44-49).

[8] Zafar, M., Masood, A. (2023). Interactive English to Urdu machine translation using example-based approach. Proceedings of the International Conference on Computational Linguistics, 45(2), 123–130.

Appendix A

Proposal Copy

This appendix contains the original proposal document submitted to the supervisor and department. The proposal outlines the initial idea, objectives, and timeline of the Urdu Translator for Educational Content project.
Note: The full proposal document is also included in this document

Urdu Translator for Educational Content

FYP Proposal Idea

Submitted by:

Mizna Rauf 2021-CS-174

Faiza Nazakat 2021-CS-182

Ifrah Muzahir 2021-CS-213

Yasir Baig 2021-CS-213

Department of Computer Science

University of Engineering and Technology, Lahore

1. Introduction

Today, having educational resources within reach matters a lot for students and individuals, but many non-English-speaking communities struggle with language issues. Because Pakistan is a nation where Urdu is considered both the national and official language, The English language is the most common way for millions to communicate. Educational content is a big barrier for many rural students. Throughout the country, but especially in underserved areas. In particular, when producing complex learning materials such as technical books or academic books I studied lectures, textbooks and scientific articles. We are proposing to develop a specialized language in this project. An educational translation system specifically designed for English-to-Urdu, using advanced technology using Transformer neural networks, in particular, as machine learning methods rule-based systems. By dealing with the linguistic, setting and culture differences of it works to increase quality education, improve access to internet, and work to save the language traditions in Urdu speaking communities. Under the project, guidelines were put in place. NLP solutions are becoming more important as the demand for services for low-resource languages grows languages and tries to help advance the academic and technological fields. the use of machine translation in education.

2 Problem Statement

The process of turning English educational content into Urdu poses several problems that make learning difficult for students who speak Urdu. Unlike Urdu which puts the Verb before the Object, It creates significant problems because all sentences use a Subject-Object-Verb (SOV) order. This difference in how the two countries are built often leads to translations that aren't grammatically right or sound strange. Furthermore, what makes Urdu tricky for translation is that there are usually various Urdu words for every English term. You must understand what is going on around the issue. Often, educational texts representing different cultures include idiomatic expressions that are misunderstood by many people. Or end up with a different meaning. The situation becomes more complicated when high-quality parallel English-Urdu corpora make it difficult for machine learning models to understand the fine points of Urdu translation. Solutions for translation that exist at this time, because Google Translate faces these problems, the outcomes of its work are not always fluent. Particularly with domain-related materials such as legal, technical or academic content, accuracy and cultural appropriateness are very important. They prevent access to needed educational tools for Urdu-speaking students have made the digital divide more severe and hurt efforts to promote Learning and reading in Pakistan.

3 Objectives

The project tries to build a good and strong translation system for English-Urdu in educational content, with these specific goals:

- To implant an Artificial Intelligence-based English-to-Urdu translation system for academic content.

- Use Transformer architecture with neural networks fine-tuned on English-Urdu data to ensure the translation is proper and also include rules to check grammar and sentence structure.
- A BLEU score of 40 or over should be achieved for English-to-Urdu translations, so the system implements translations as good as a human's.
- Enhance the translation system so users can read educational texts without frequent delays and enjoy a smooth interaction.
- Pull together and process data from bilingual websites, open datasets and texts that have been translated and used in education so that models are supported.
- You should incorporate a module that corrects typical issues in Urdu grammar such as the form of sentences, to improve the quality of your translations.
- Support Urdu-speaking students' education by making English study material available to them which helps everyone learn more equally.
- Help save Urdu by using it on learning apps and highlighting how significant it is for cultural and language reasons.

Appendix B

Plagiarism Report

The final version of the thesis was evaluated using plagiarism detection tools to ensure originality. The report confirms that the document meets university integrity standards.

Appendix C

Extra Code Snippets

This section provides additional code snippets not included in the core chapters but important to understanding certain modules.

Snippet 1: Save video to media

```
video_file = request.FILES['file']
upload_folder = os.path.join(settings.MEDIA_ROOT, 'uploaded_files')
os.makedirs(upload_folder, exist_ok=True)

filename = f"{datetime.now().strftime('%Y%m%d%H%M%S')}_{video_file.name}"
video_path = os.path.join(upload_folder, filename)
with open(video_path, 'wb+') as destination:
    for chunk in video_file.chunks():
        destination.write(chunk)
```

Snippet 2: Extract metadata

```
metadata = get_video_metadata(video_path)

    # Title fallback to filename (without extension) if not in metadata
    title_from_meta = metadata.get('format', {}).get('tags', {}).get('title')
    video_title = title_from_meta if title_from_meta else
    os.path.splitext(video_file.name)[0]

    # Description is optional { fallback to empty
    description = metadata.get('format', {}).
    get('tags', {}).get('description', '')
```

Snippet 3: Extract audio

```
audio_path = os.path.join(settings.MEDIA_ROOT, 'temp_audio.wav')
            extract_audio(video_path, audio_path)
```

Snippet 4: Generate transcript

```
transcript = transcribe_audio(audio_path)
```

Snippet 5: Guess lecture category

```
def guess_category(text):
    text_lower = text.lower()
    if "bank" in text_lower or "finance" in
    text_lower or "account" in text_lower:
        return "Banking"
    elif "law" in text_lower or "legal"
    in text_lower or "court" in text_lower:
        return "Law"
    elif "farm" in text_lower or "crop" in
    text_lower or "agriculture"
    in text_lower or "soil" in text_lower:
        return "Agriculture"
        return "General"

    category = guess_category(transcript + " " + description)
```

Snippet 6: Save to model

```
 user = request.user_profile
            video_instance = UploadedVideoLecture.objects.create(
                uploadedBy=user,
                videoTitle=video_title,
                video_file=os.path.join('videos', filename),
                lectureCategory=category,
                transcript=transcript
            )

            file_url = request.build_absolute_uri(settings.MEDIA_URL
            + f"videos/{filename}")
```

Snippet 6.5: Save transcript to LectureTranslation as sourceText

```
LectureTranslation.objects.create(
                video=video_instance,
                sourceText=transcript,
                status='undone'
            )
                        # Cleanup
            if os.path.exists(audio_path):
                os.remove(audio_path)

            return JsonResponse({
                'message': 'Transcript generated and saved.',
                'file_url': file_url,
                'transcript': transcript,
                'category': category,
                'videoId': str(video_instance.videoId)
            }, status=201)

        except Exception as e:
            return JsonResponse({'error': f'Failed to
            process file: {str(e)}'}, status=500)

        return JsonResponse({'error': 'No file uploaded
        or invalid request method'}, status=400)
```

Snippet 7: transcribe audio

```
def transcribe_audio(audio_path):
    whisper_pipeline = get_whisper_pipeline()
    result = whisper_pipeline(audio_path)
    return clean_text(result["text"]) if isinstance(result, dict) and "text" in
```

Snippet 8: Authenticate User

```
def authenticate_user(view_func):
    def _wrapped_view(request, *args, **kwargs):
        auth_header = request.headers.get('Authorization')
```

```
        if not auth_header or not auth_header.startswith("Token "):
            return JsonResponse({'error': 'Unauthorized'}, status=401)

        token = auth_header.split(" ")[1]
        try:
            user = UserProfile.objects.get(token=token)
            request.user_profile = user  # Attach user to request
        except UserProfile.DoesNotExist:
            return JsonResponse({'error': 'Invalid token'}, status=401)

        return view_func(request, *args, **kwargs)
    return _wrapped_view
```

Snippet 9: Get Video Metadata

```
def get_video_metadata(video_path):
    command = [
        "ffprobe", "-v", "error", "-print_format", "json", "-show_format",
        "-show_streams", video_path
    ]
    result = subprocess.run(command,
    stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    metadata = json.loads(result.stdout.decode())
    return metadata
```

Snippet 10: Video Category

```
def guess_category(text):
    text_lower = text.lower()
    categories = {
        "Banking": ["bank", "finance", "loan", "credit", "investment",
        "atm", "branch"],
        "Law": ["law", "legal", "justice", "court", "attorney",
        "contract", "crime"],
        "Agriculture": ["agriculture", "farm", "crop", "harvest",
        "irrigation", "fertilizer", "seeds"],
    }

    for category, keywords in categories.items():
```

```
        if any(keyword in text_lower for keyword in keywords):
            return category
    return "General"
```

Snippet 11: Whisper Pipeline

```
def get_whisper_pipeline():
    device = "cuda:0" if torch.cuda.is_available() else "cpu"
    model_id = "openai/whisper-tiny.en"
    dtype = torch.float16 if torch.cuda.is_available() else torch.float32

    model = AutoModelForSpeechSeq2Seq.from_pretrained(
        model_id, torch_dtype=dtype, low_cpu_mem_usage=True
    ).to(device)

    processor = AutoProcessor.from_pretrained(model_id)
    forced_ids = processor.get_decoder_prompt_ids(language="en",
    task="transcribe")
    model.config.forced_decoder_ids = forced_ids

    return pipeline(
        "automatic-speech-recognition",
        model=model,
        tokenizer=processor.tokenizer,
        feature_extractor=processor.feature_extractor,
        chunk_length_s=5,
        batch_size=8,
        torch_dtype=dtype,
        device=device
    )
```

Snippet 12: Extract Audio

```
def extract_audio(video_path, audio_path):
    command = [
        "ffmpeg", "-y", "-i", video_path, "-vn",
        "-acodec", "pcm_s16le", "-ar", "16000", "-ac", "1", audio_path
    ]
    subprocess.run(command, check=True)
```

Snippet 13: Clean Text

```
def clean_text(text):
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    text = re.sub(r'\b(\w+)( \1\b){2,}', r'\1', text)
    return text.strip()
```

Appendix D

GitHub Repository Link

The entire project, including source code, assets, and documents, has been maintained in a version-controlled GitHub repository.

- **GitHub Repository:** https://github.com/miznar/aiUrduTranslate.git

Note: The repository includes a README file, issue tracker, and version history.