Language Learning Platform

A Real-Time / Field-Based Research Project (22DT284) report submitted to the Jawaharlal Nehru Technological University, Hyderabad

Submitted by

M.SAIBHARGAV	**23B81A7239**
D.SRINIDHI	**23B81A7250**
B.VARSHA	**23B81A7255**

Under the guidance of

Mr.V. RAMESH

Assistant Professor

CSE(Data Science)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CVR COLLEGE OF ENGINEERING

(***An Autonomous Institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)

Vastunagar, Mangalpalli(V),Ibrahimpatnam(M),
Rangareddy (D), Telangana- 501 510

APRIL 2025

CVR COLLEGE OF ENGINEERING

(*An Autonomous Institution , NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar, Mangalpalli(V), Ibrahimpatnam(M),
Rangareddy (D), Telangana- 501 510.

DEPARTMENT OF AID

CERTIFICATE

This is to certify that the Real time/ Field-Based research project(22DT284) report entitled **"Language Learning Platform"** is a record of work carried out by **M.Saibhargav , D.Srinidhi , B.Varsha** submitted to Department of **AID,** CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Project Guide	**Project Coordinator**	**Head of the Department**
Mr. V.Ramesh		**Dr.Lakshmi H N**
Assistant Professor		Professor & HOD
CSE(Data Science)		CSE(AI&ML,DS,CS)

DECLARATION

We hereby declare that the Real time/ Field-Based research project (22DT284) report entitled **"Language Learning Platform"** is an original work done and submitted to **CSE (DATA SCIENCE)** Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad and it is a record of bonafide project work carried out by us under the guidance of **MR.V.RAMESH , ASSISTANT PROFESSOR , CSE(DS).**

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

M SAIBHARGAV

23B81A7239

Signature of the Student

D SRINIDHI

23B81A7250

Signature of the Student

B VARSHA

23B81A7255

Date:

Place:

ACKNOWLEDGEMENT

We are thankful and fortunate enough to get constant encouragement, support, and guidance from all **Teaching staff of AID** Department who helped us in successfully completing this project work.

We thank **DR. A. SRINIVASA REDDY ,** Project Coordinator and **-------**Project Review Committee members for their valuable guidance and support which helped us to complete the project work successfully.

We respect and thank our internal guide, **MR. V. RAMESH,ASSISTANT PROFESSOR,CSE(DATA SCIENCE)**, for giving us all the support and guidance, which made us complete the project duly.

We would like to express heartfelt thanks to **Dr. H. N. Lakshmi,** Professor & Head of the Department, for providing us an opportunity to do this project and extending support and guidance.

We thank our Vice-Principal **Prof. L. C. Siva Reddy** for providing excellent computing facilities and a disciplined atmosphere for doing our work.

We wish a deep sense of gratitude and heartfelt thanks to Dr. Rama Mohan Reddy, Principal and the Management for providing excellent lab facilities and tools. Finally, we thank all those guidance helpful to us in this regard.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Language learning is an essential skill in today's globalized world, and the rise of digital platforms has made it more accessible than ever. This project focuses on the development of a web-based Language Learning Application that offers users an interactive environment to enhance their vocabulary, grammar, and comprehension skills in various languages.

The application provides features such as multiple-choice quizzes, fill-in-the-blank exercises, user authentication, and score tracking to create a personalized learning experience. It is built using modern web technologies for both frontend and backend development and deployed on a cloud platform for easy accessibility.

The primary objective of this project is to design a user-friendly and responsive platform that encourages consistent learning through engaging and interactive content. The application also supports future enhancements like adding audio pronunciation, speech recognition, and advanced analytics for tracking learning progress.

Overall, this project successfully demonstrates the application of web development concepts in creating a meaningful educational tool, with potential for further growth and integration of advanced language learning techniques.

CHAPTER 1
INTRODUCTION

1.1 PROBLEM STATEMENT

In today’s increasingly globalized world, the ability to learn and communicate in multiple languages has become essential for personal, academic, and professional growth. However, many existing language learning platforms are either expensive, lack personalization, or offer limited interactive features. There is a need for an accessible, user-friendly, and interactive web-based application that allows users to improve their vocabulary, grammar, and comprehension skills in a convenient and engaging environment.

This project aims to develop a **Language Learning Web Application** that provides features such as multiple-choice quizzes, fill-in-the-blank exercises, user authentication, score tracking, and progress monitoring. The goal is to create a modern, cloud-deployed educational platform that encourages consistent learning through interactive content, while also being scalable for future enhancements like audio pronunciation, speech recognition, and advanced analytics.

1.2 PROJECT OBJECTIVES:

The primary objectives of this project are:

- **To design and develop a web-based language learning application** that is user-friendly, interactive, and accessible to a wide range of learners.
- **To implement essential features such as multiple-choice , fill-in-the-blank exercises, and vocabulary tests** to assist users in improving their language skills.
- **To enable user registration and authentication functionality** for maintaining personalized profiles and tracking individual learning progress.
- **To develop a responsive and interactive user interface** that enhances the user experience and encourages regular usage.
- **To deploy the application on a cloud platform** for seamless, anytime-anywhere access through web browsers.
- **To establish a scalable system architecture** that allows for the integration of additional languages, exercises, and advanced features such as audio pronunciations, speech recognition, and progress analytics in the future.

1.3 SOFTWARE AND HARDWARE SPECIFICATIONS

1.3.1 SOFTWARE REQUIREMENTS

Category	Details
Operating System	Windows
Frontend Technologies	HTML5, CSS3, JavaScript
Backend Technologies	Python (Flask/Django)
Database	MongoDB (NoSQL) or MySQL (Relational)
Version Control	Git (GitHub/GitLab)
Deployment	Render.com
Development Tools	Web Browser (Chrome/Firefox/Safari)

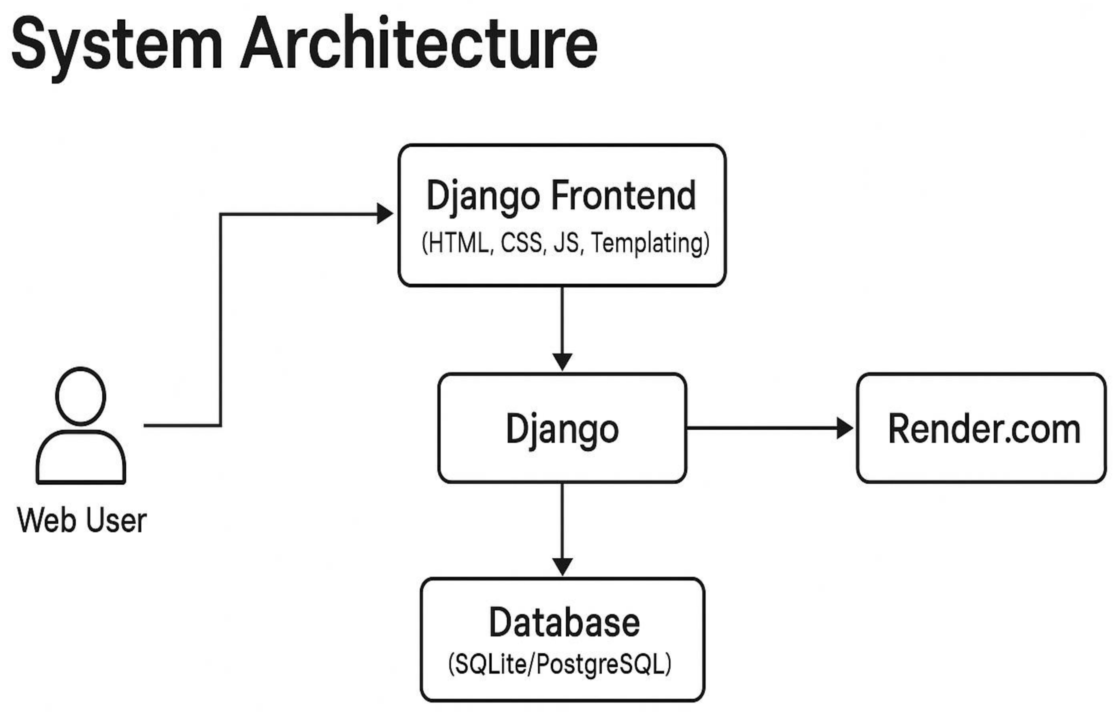
1.3.2 HARDWARE REQUIREMENTS

Processor	Intel Core i5 or higher (for development), 1 GHz processor (for end users)
Memory (RAM)	8GB RAM (development), 2GB RAM (end users)
Storage	50GB free disk space (development)
Internet Connection	Stable internet (1 Mbps or higher)
User Devices	Compatible with desktop, laptop, and mobile devices with Chrome, Firefox, or Safari browsers

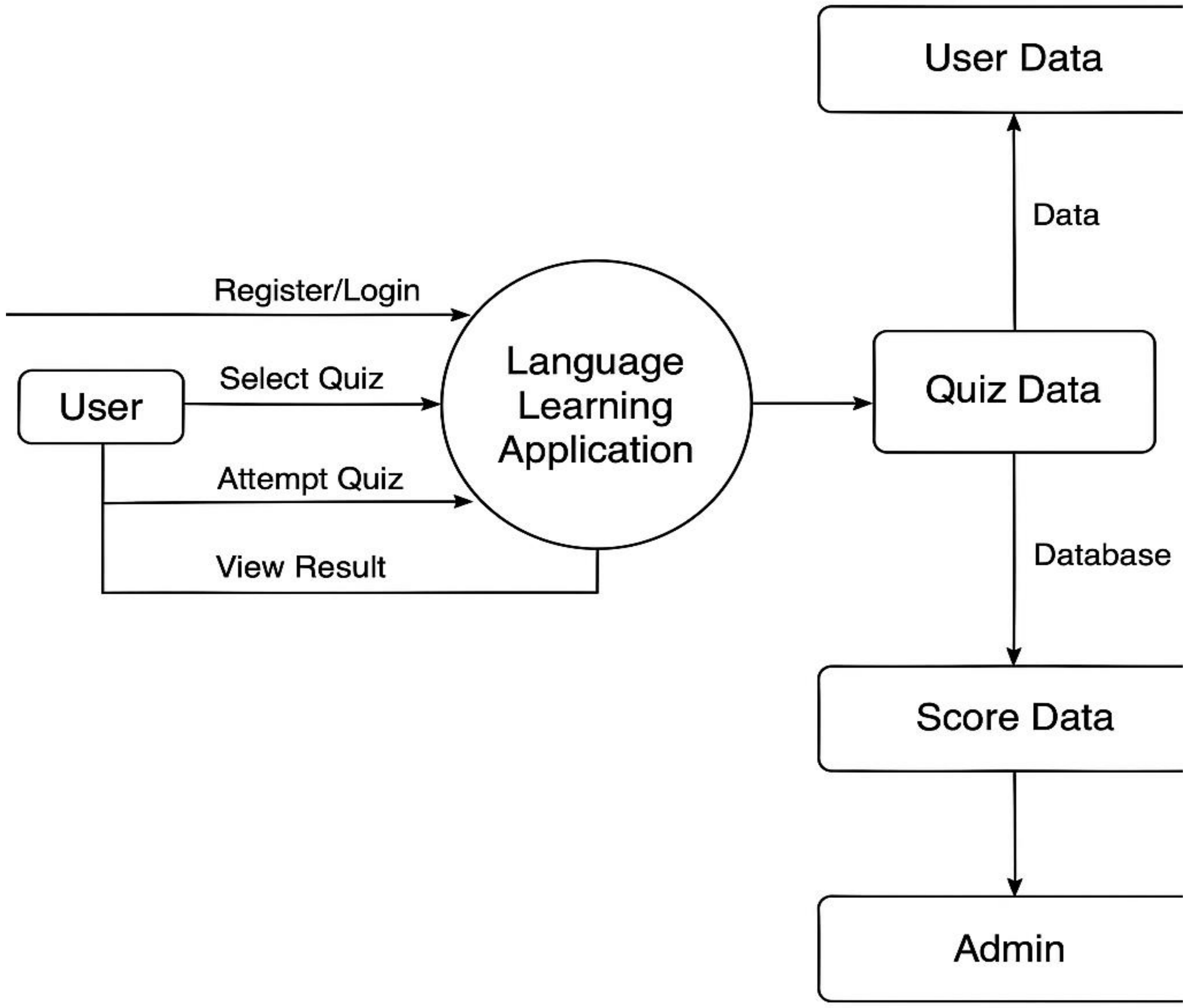
CHAPTER 2
DESIGN METHODOLOGY

2.1 SYSTEM ARCHITECTURE

- **Client Layer :** Web browsers render dynamic pages using HTML, CSS, JavaScript, and Django templates.
- **Application Layer :** Django (Python) handles user requests, quiz logic, authentication, and data management via views, models, and forms.
- **Database Layer :** Uses SQLite/PostgreSQL/MySQL to store user details, quiz content, and scores through Django's ORM.
- **Deployment :** Hosted on Render.com for online accessibility and scalability.



2.2 DATA FLOW DIAGRAM

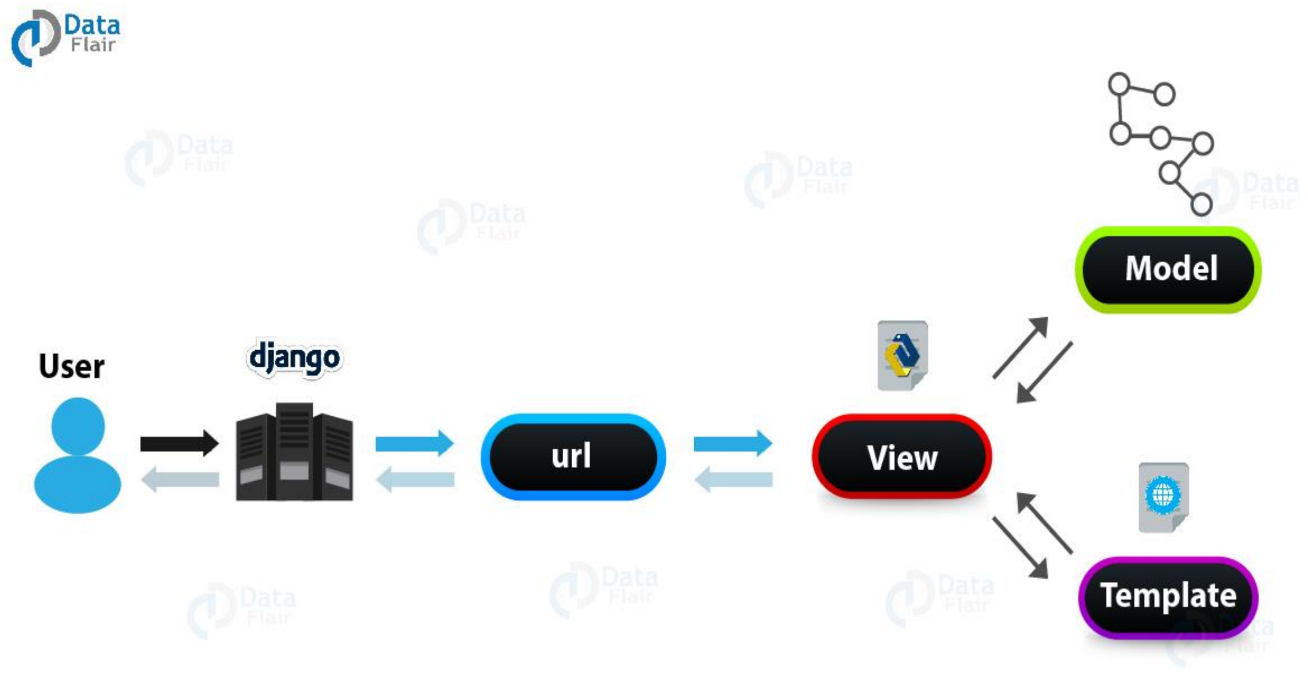


2.3 TECHNOLOGY DESCRIPTION

Technology	**Purpose / Description**
HTML5, CSS3, JavaScript	Used for creating interactive and responsive user interfaces for the web application.
Django (Python Framework)	A secure, high-level web framework for developing the backend, handling business logic, routing, user authentication, and integrating frontend templates.
SQLite / PostgreSQL / MySQL	Relational databases used to store user data, quiz questions, scores, and progress reports, managed via Django's ORM.
Render.com / Heroku	Cloud hosting platforms used to deploy the web application and make it accessible online.
Git & GitHub	Version control systems used for collaborative coding, tracking changes, and code management.
Postman	API development and testing tool used to check backend APIs and database interactions during development.
Visual Studio Code	The primary code editor with extensions for Django, Python, HTML, and version control integration.

CHAPTER 3

IMPLEMENTATION & TESTING



Directory Structure

courses/

- management/ – Custom Django management commands (optional).
- migrations/ – Database schema changes tracked by Django ORM.
- templates/ – HTML templates related to course views.
- templatetags/ – Custom template tags and filters for course templates**.**

Python Files

- __init__.py – Initializes the app package.
- admin.py – Registers models for Django Admin.
- apps.py – Configuration for the courses app.
- models.py – Defines the data models related to courses.

- tests.py – Unit tests for the courses app.
- urls.py – URL configuration specific to the courses app.
- views.py – Contains view functions or class-based views handling HTTP requests.

Sample Implementation Breakdown

models.py

Defines the structure of course-related data:

```
from django.db import models
class Course(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.title
```

admin.py

Registers the model to make it available in the admin interface:

```
from django.contrib import admin
from .models import Course
admin.site.register(Course)
```

views.py

Handles request logic:

```
from django.shortcuts import render
from .models import Course
def course_list(request):
    courses = Course.objects.all()
    return render(request, 'courses/course_list.html', {'courses': courses})
```

urls.py

Maps URLs to view functions:

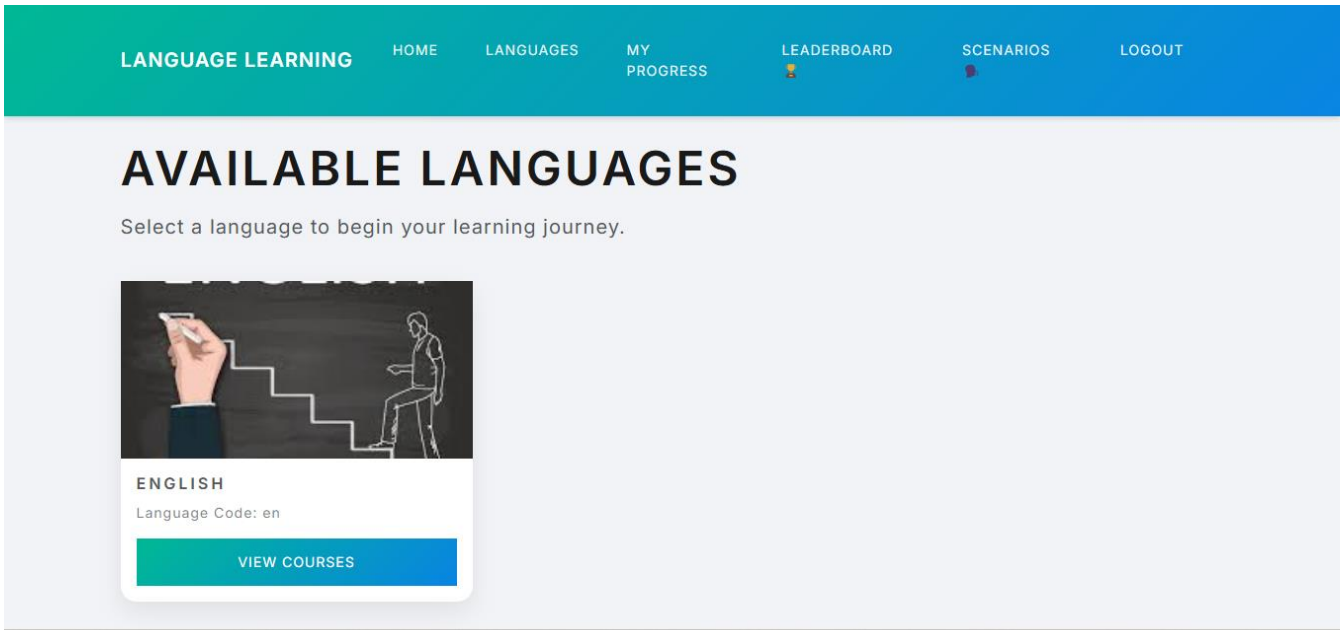
```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.course_list, name='course_list'),
]
```

templates/courses/course_list.html

Basic template to render the list of courses:

```
<h1>Courses</h1>
<ul>
  {% for course in courses %}
    <li>{{ course.title }}</li>
  {% endfor %}
</ul>
```

Result:



AVAILABLE LANGUAGES

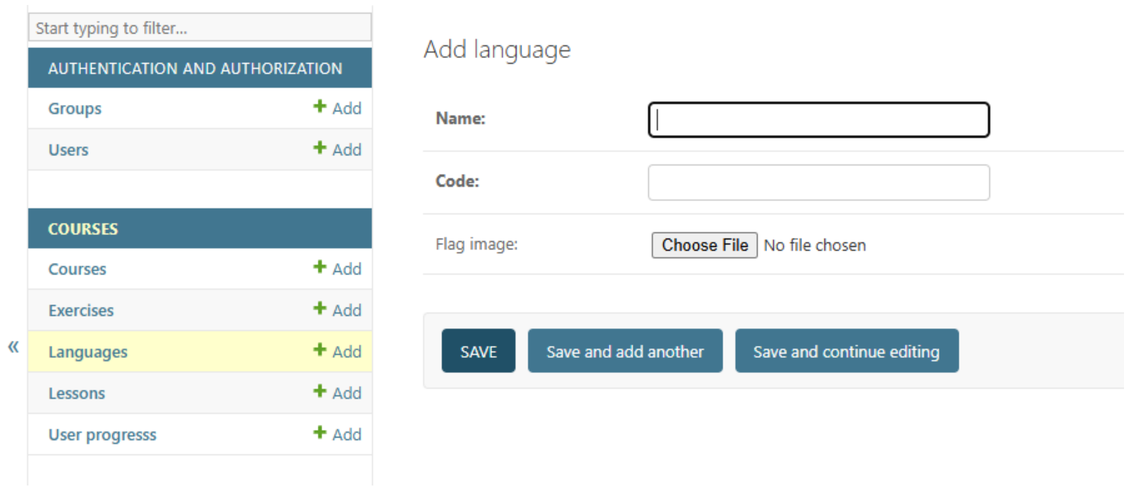
Select a language to begin your learning journey.

ENGLISH

Language Code: en

To add Language:

- Click the green + Add button next to "Languages".
- Fill in the form fields:
    a. Name: Enter the name of the language (e.g., *English*, *Spanish*).
    b. Code: Provide a short code for the language (e.g., *en*, *es*).



Add language

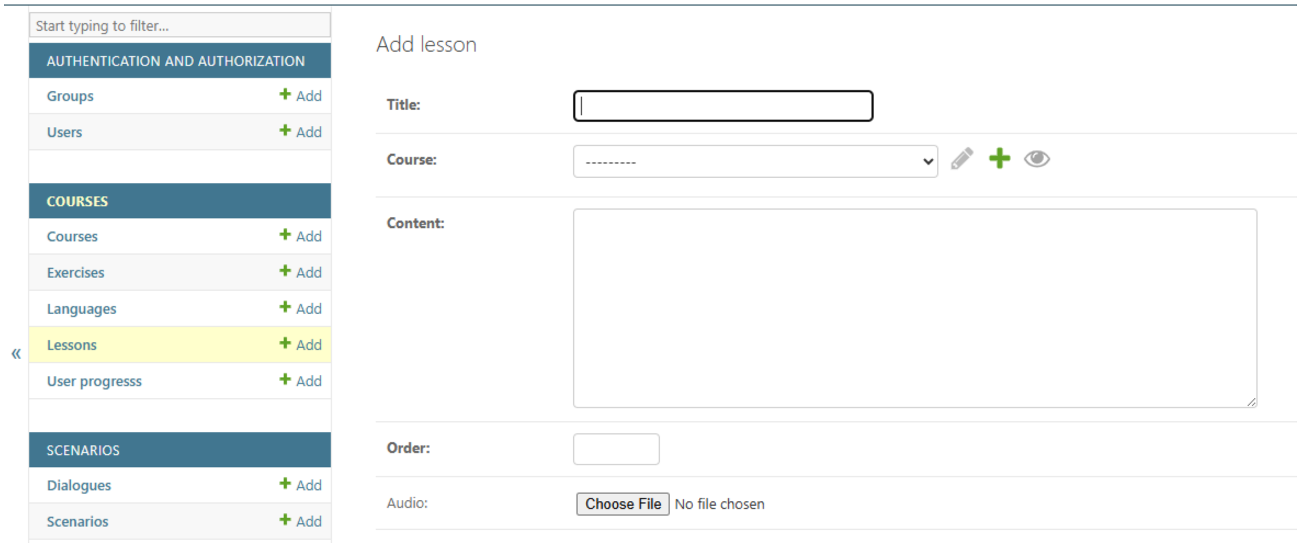
Name:	
Code:	
Flag image:	Choose File No file chosen

To Add lessons in a particular Language:

On the "Add lesson" page, you’ll typically see a form with fields like:

- Title – The name of the lesson (e.g., *Introduction to Grammar*).
- Course – A dropdown to select the course this lesson belongs to.
- Description / Content – The actual content or a summary of the lesson.
- Media (Optional) – Add any files, videos, or images related to the lesson.
- Order / Sequence – If applicable, specify the lesson's order within the course.

Result:



Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

COURSES
- Courses + Add
- Exercises + Add
- Languages + Add
- Lessons + Add
- User progresss + Add

SCENARIOS
- Dialogues + Add
- Scenarios + Add

Add lesson

Title:	
Course:	---------
Content:	
Order:	
Audio:	Choose File No file chosen

3.1 CODE SNIPPET

To Trigger Scenarios:

```
from django.shortcuts import render, get_object_or_404
from .models import Scenario

def scenario_list(request):
    scenarios = Scenario.objects.all()
    return render(request, 'scenarios/scenario_list.html', {'scenarios': scenarios})

def scenario_detail(request, scenario_id):
    scenario = get_object_or_404(Scenario, id=scenario_id)
    dialogues = scenario.dialogues.order_by('order')  # Ensure messages are in order
    return render(request, 'scenarios/scenario_detail.html', {'scenario': scenario, 'dialogues':
dialogues})
```

To access All urls:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('courses.urls')),
    path('users/', include('users.urls')),
    path('scenarios/', include('scenarios.urls')),



]
```

if settings.DEBUG:

```
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

3.2 TEST CASES

CHAPTER 4

CONCLUSION

The development of the Language Learning Web Application successfully demonstrates the effective use of modern web technologies to create an accessible, interactive, and educational platform for language enthusiasts. By integrating features like multiple-choice quizzes, fill-in-the-blank exercises, user authentication, and score tracking, the application provides users with a personalized and engaging learning environment.

The project also highlights the practical application of the Django framework for web development, along with cloud deployment for global accessibility. Through this system, users can consistently enhance their vocabulary, grammar, and comprehension skills while tracking their performance over time.

Additionally, the modular design of the application ensures that it can be easily extended in the future to include features like audio pronunciation, speech recognition, and advanced learning analytics. Overall, the project fulfills its objectives and serves as a scalable foundation for more comprehensive language learning solutions.

BIBILOGRAPHY

- Django Software Foundation. *Django Documentation*. Available at: https://docs.djangoproject.com
- MDN Web Docs. *HTML, CSS, and JavaScript Tutorials*. Available at: https://developer.mozilla.org
- Python Software Foundation. *Python Official Documentation*. Available at: https://docs.python.org
- Render Cloud Hosting. *Render Documentation*. Available at: https://render.com/docs
- W3Schools. *Web Development Tutorials*. Available at: https://www.w3schools.com
- GitHub. *Version Control and Code Repository*. Available at: https://github.com
- Ian Sommerville. *Software Engineering*, 10th Edition, Pearson Education.

APPENDIX

```
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from .models import Language, Course, Lesson, Exercise, UserProgress
import json
from django.core.files.storage import default_storage
from django.core.files.base import ContentFile
from users.models import Profile
def home(request):
    languages = Language.objects.all()
    return render(request, 'courses/home.html', {'languages': languages})
def language_list(request):
    languages = Language.objects.all()
    return render(request, 'courses/language_list.html', {'languages': languages})
```

```
def course_list(request, language_id):
    language = get_object_or_404(Language, id=language_id)
    courses = Course.objects.filter(language=language)
    return render(request, 'courses/course_list.html', {
        'language': language,
        'courses': courses
    })
def course_detail(request, course_id):
    course = get_object_or_404(Course, id=course_id)
    lessons = course.lessons.all()
        # Get user progress if logged in
    user_progress = {}
    if request.user.is_authenticated:
        progress_objects = UserProgress.objects.filter(
            user=request.user,
            lesson__course=course
        )
        for progress in progress_objects:
            user_progress[progress.lesson.id] = progress
     return render(request, 'courses/course_detail.html', {
       'course': course,
        'lessons': lessons,
        'user_progress': user_progress
    })
@login_required
def lesson_detail(request, lesson_id):
    lesson = get_object_or_404(Lesson, id=lesson_id)
        # Get or create user progress
    progress, created = UserProgress.objects.get_or_create(
        user=request.user,
        lesson=lesson,
        defaults={'completed': False, 'score': 0}
    )
```

```
   # Mark as accessed
    progress.save()
    return render(request, 'courses/lesson_detail.html', {
        'lesson': lesson,
        'progress': progress
    })
@login_required
def exercise_detail(request, exercise_id):
    exercise = get_object_or_404(Exercise, id=exercise_id)
    return render(request, 'courses/exercise_detail.html', {
        'exercise': exercise
    })
@login_required
def submit_exercise(request, exercise_id):
    if request.method != 'POST':
        return JsonResponse({'error': 'Only POST requests allowed'}, status=405)
    exercise = get_object_or_404(Exercise, id=exercise_id)
    user_answer = request.POST.get('answer', '')
     # Check if correct
    is_correct = False
    if exercise.type == 'multiple_choice':
        is_correct = user_answer == exercise.answer
    elif exercise.type == 'fill_blank' or exercise.type == 'translation':
        # Case insensitive and strip whitespace for text answers
        is_correct = user_answer.strip().lower() == exercise.answer.strip().lower()
    elif exercise.type == 'matching':
        # For matching, compare JSON objects
        try:
            user_matches = json.loads(user_answer)
            correct_matches = json.loads(exercise.answer)
            is_correct = user_matches == correct_matches
        except json.JSONDecodeError:
            is_correct = False
    # Update progress
    progress, created = UserProgress.objects.get_or_create(
        user=request.user,
        lesson=exercise.lesson,
        defaults={'completed': False, 'score': 0}
    )
    if is_correct:
        progress.score += exercise.points
        progress.save()
      # Check if this was the last exercise in the lesson
    last_exercise = exercise.lesson.exercises.order_by('-order').first()
    if exercise.id == last_exercise.id:
        progress.completed = True
        progress.save()
    return JsonResponse({
        'is_correct': is_correct,
        'correct_answer': exercise.answer,
        'score': progress.score,
        'completed': progress.completed
    })
@login_required
def user_profile(request):
    progress_list = UserProgress.objects.filter(user=request.user)
    # Group by course
    courses_progress = {}
    total_score = 0
    completed_lessons = 0
     for progress in progress_list:
        course_id = progress.lesson.course.id
        if course_id not in courses_progress:
            courses_progress[course_id] = {
                'course': progress.lesson.course,
                'lessons_completed': 0,
            'total_lessons': progress.lesson.course.lessons.count(),
            'score': 0
          }
      if progress.completed:
          courses_progress[course_id]['lessons_completed'] += 1
          completed_lessons += 1
    courses_progress[course_id]['score'] += progress.score
    total_score += progress.score
  return render(request, 'courses/user_profile.html', {
    'courses_progress': courses_progress.values(),
    'total_score': total_score,
    'completed_lessons': completed_lessons
  })
```