Here's an outline for building a Flask site for language learning:

Project Setup

a. Create a new Flask project

b. Set up a virtual environment

c. Install required packages (Flask, Flask-SQLAlchemy, Flask-Login, Flask-WTF, requests, etc.)

1. d. Initialize version control (Git)

Database Models

a. User: stores user information (username, email, password, etc.)

b. TextResource: stores uploaded texts (title, content, user\_id, etc.)

c. Sentence: stores individual sentences from texts (content, text\_resource\_id, etc.)

d. Word: stores individual words and their translations (content, translation, sentence\_id, etc.)

e. UserWord: stores user's knowledge of words (user\_id, word\_id, proficiency\_level, etc.)

1. f. Quiz: stores quizzes taken by the user (user\_id, text\_resource\_id, timestamp, score, etc.)

Authentication

a. User registration

b. User login

1. c. User logout

Text Resource Management

a. Upload a new text resource

b. List all uploaded text resources

c. View a specific text resource

1. d. Delete a text resource

Text Processing

a. Split text into sentences

b. Split sentences into words

1. c. Translate words into English using an external API (e.g., Google Translate API)

Learning Mode

a. Choose a text resource or study words from the entire set of works

b. Present a sentence and its translation

c. Teach word by word for each sentence

d. Review the whole sentence

e. Quiz the user using multiple choice options

1. f. Update the user's knowledge of words based on their answers

Quizzing

a. Create quizzes based on user's weak words or from specific text resources

b. Provide multiple choice questions

c. Keep track of quiz results

1. d. Update user's word proficiency based on quiz performance

Progress Statistics

a. Display user's overall vocabulary size

b. Show user's progress in learning words from specific text resources

1. c. Visualize user's improvement over time

Testing

a. Write unit tests for critical functionality

1. b. Test the application with sample texts and users

Deployment

a. Deploy the application to a web server (e.g., Heroku, AWS, etc.)

1. b. Set up domain and SSL certificate

File Organization:  
dictionLang/

-App/

--static/

---custom.css

--templates/

base.html

index.html

login.html

new\_text\_resource.html

register.html

text\_resources.html

view\_text\_resource.html

\_\_init\_\_.py  
 forms.py  
 models.py  
 routes.py

-instance/

-migrations/

config.py

requirements.txt

run.py

Code:

config.py :

class Config:

SECRET\_KEY = 'your\_secret\_key'

SQLALCHEMY\_DATABASE\_URI = 'sqlite:///language\_learning.db'

SQLALCHEMY\_TRACK\_MODIFICATIONS = False

run.py :

from app import app

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

\_\_init\_\_.py :

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager

from flask\_migrate import Migrate

app = Flask(\_\_name\_\_)

app.config.from\_object('config.Config')

db = SQLAlchemy(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = "login"

migrate = Migrate(app, db)

from app.models import User

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

from app import routes  
  
forms.py :

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, SubmitField, TextAreaField

from wtforms.validators import DataRequired, Email, EqualTo, ValidationError

from app.models import User

class RegistrationForm(FlaskForm):

username = StringField('Username', validators=[DataRequired()])

email = StringField('Email', validators=[DataRequired(), Email()])

password = PasswordField('Password', validators=[DataRequired()])

confirm\_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])

submit = SubmitField('Register')

def validate\_username(self, username):

user = User.query.filter\_by(username=username.data).first()

if user:

raise ValidationError('That username is already taken.')

def validate\_email(self, email):

user = User.query.filter\_by(email=email.data).first()

if user:

raise ValidationError('That email is already in use.')

class LoginForm(FlaskForm):

email = StringField('Email', validators=[DataRequired(), Email()])

password = PasswordField('Password', validators=[DataRequired()])

submit = SubmitField('Login')

class TextResourceForm(FlaskForm):

title = StringField('Title', validators=[DataRequired()])

content = TextAreaField('Content', validators=[DataRequired()])

submit = SubmitField('Upload')

models.py :

from app import db

from flask\_login import UserMixin

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

password = db.Column(db.String(128), nullable=False)

def \_\_repr\_\_(self):

return f'<User {self.username}>'

@property

def is\_active(self):

return True

@property

def is\_authenticated(self):

return True

@property

def is\_anonymous(self):

return False

def get\_id(self):

return str(self.id)

class TextResource(db.Model):

id = db.Column(db.Integer, primary\_key=True)

title = db.Column(db.String(255), nullable=False)

content = db.Column(db.Text, nullable=False)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

user = db.relationship('User', backref=db.backref('text\_resources', lazy=True))

def \_\_repr\_\_(self):

return f'<TextResource {self.title}>'

class Sentence(db.Model):

id = db.Column(db.Integer, primary\_key=True)

content = db.Column(db.Text, nullable=False)

text\_resource\_id = db.Column(db.Integer, db.ForeignKey('text\_resource.id'), nullable=False)

text\_resource = db.relationship('TextResource', backref=db.backref('sentences', lazy=True))

def \_\_repr\_\_(self):

return f'<Sentence {self.id}>'

class Word(db.Model):

id = db.Column(db.Integer, primary\_key=True)

content = db.Column(db.String(255), nullable=False)

translation = db.Column(db.String(255), nullable=False)

sentence\_id = db.Column(db.Integer, db.ForeignKey('sentence.id'), nullable=False)

sentence = db.relationship('Sentence', backref=db.backref('words', lazy=True))

def \_\_repr\_\_(self):

return f'<Word {self.content}>'

class UserWord(db.Model):

id = db.Column(db.Integer, primary\_key=True)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

word\_id = db.Column(db.Integer, db.ForeignKey('word.id'), nullable=False)

proficiency\_level = db.Column(db.Integer, nullable=False, default=0)

user = db.relationship('User', backref=db.backref('user\_words', lazy=True))

word = db.relationship('Word', backref=db.backref('user\_words', lazy=True))

def \_\_repr\_\_(self):

return f'<UserWord {self.id}>'

class Quiz(db.Model):

id = db.Column(db.Integer, primary\_key=True)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

text\_resource\_id = db.Column(db.Integer, db.ForeignKey('text\_resource.id'))

def \_\_repr\_\_(self):

return f'<Quiz {self.id}>'

routes.py :

from flask import render\_template, redirect, url\_for, flash, request

from flask\_login import login\_user, login\_required, logout\_user, current\_user

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from app import app, db, login\_manager

from app.models import User, TextResource

from app.forms import RegistrationForm, LoginForm, TextResourceForm

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/register', methods=['GET', 'POST'])

def register():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = RegistrationForm()

if form.validate\_on\_submit():

hashed\_password = generate\_password\_hash(form.password.data, method='sha256')

user = User(username=form.username.data, email=form.email.data, password=hashed\_password)

db.session.add(user)

db.session.commit()

flash('Registration successful! You can now log in.', 'success')

return redirect(url\_for('login'))

return render\_template('register.html', form=form)

@app.route('/login', methods=['GET', 'POST'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(email=form.email.data).first()

if user and check\_password\_hash(user.password, form.password.data):

login\_user(user)

flash('Login successful!', 'success')

next\_page = request.args.get('next')

return redirect(next\_page) if next\_page else redirect(url\_for('index'))

else:

flash('Login failed. Please check your email and password.', 'danger')

return render\_template('login.html', form=form)

@app.route('/logout')

@login\_required

def logout():

logout\_user()

flash('You have been logged out.', 'info')

return redirect(url\_for('index'))

@app.route('/text\_resources/new', methods=['GET', 'POST'])

@login\_required

def new\_text\_resource():

form = TextResourceForm()

if form.validate\_on\_submit():

text\_resource = TextResource(title=form.title.data, content=form.content.data, user\_id=current\_user.id)

db.session.add(text\_resource)

db.session.commit()

flash('Text resource uploaded successfully!', 'success')

return redirect(url\_for('text\_resources'))

return render\_template('new\_text\_resource.html', form=form)

@app.route('/text\_resources')

@login\_required

def text\_resources():

resources = TextResource.query.filter\_by(user\_id=current\_user.id).all()

return render\_template('text\_resources.html', resources=resources)

@app.route('/text\_resources/<int:text\_resource\_id>')

@login\_required

def view\_text\_resource(text\_resource\_id):

text\_resource = TextResource.query.get\_or\_404(text\_resource\_id)

if text\_resource.user\_id != current\_user.id:

abort(403)

return render\_template('view\_text\_resource.html', text\_resource=text\_resource)

custom.css :

body {

font-family: 'Roboto', sans-serif;

}

.navbar-custom {

background-color: #5a5a5a;

}

.navbar-custom .navbar-nav .nav-link {

color: white;

}

.navbar-custom .navbar-nav .nav-link:hover {

color: rgba(255, 255, 255, 0.75);

}

.navbar-custom .navbar-nav .nav-button {

display: inline-block;

padding: 0.5rem 1rem;

border: none;

background-color: transparent;

color: white;

text-align: center;

text-decoration: none;

cursor: pointer;

}

.navbar-custom .navbar-nav .nav-button:hover {

background-color: rgba(255, 255, 255, 0.1);

}

.flash-message {

padding: 10px;

margin-bottom: 10px;

border: 1px solid #e0e0e0;

border-radius: 5px;

}

.navbar-toggler {

display: none;

}

.navbar-nav {

display: flex;

flex-direction: row;

list-style-type: none;

justify-content: center;

align-items: center;

}

.navbar-nav .nav-item {

margin-left: 10px;

}

.navbar-nav .nav-link {

padding: 0;

}

.navbar-nav .nav-link:last-child {

margin-left: auto;

}

.navbar-nav .nav-button {

margin-left: 10px;

}

.navbar-nav .nav-button:last-child {

margin-right: 10px;

}

.navbar-nav.ml-auto {

display: flex;

align-items: center;

}

base.html:  
<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<title>{% block title %} {% endblock %}</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-pzjw8f+ua7Kw1TIq0v8FqFjcJ6pajs/rfdfs3SO+kD4Ck5BdPtF+to8xM6CU5vX8" crossorigin="anonymous">

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="{{ url\_for('static', filename='custom.css') }}">

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSGFpoO/elps3DEujpr20rvzL1UYPjBmNhAMYCrprI9umgrw5f7" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js" integrity="sha384-eMNCOe7tC1doHpGoJtKh7z7lGz7fuP4F8nfdFvAOA6Gg/z6Y5J6XqqyGXYM2ntX1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8sh+Wy4Ck4SOF4y4Ck4C2DgHfViXydVeLm+JDM" crossorigin="anonymous"></script>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-custom bg-custom">

<a class="navbar-brand" href="{{ url\_for('index') }}">dictionLang</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse justify-content-between" id="navbarSupportedContent">

<ul class="navbar-nav">

<li class="nav-item">

<button class="nav-button" onclick="location.href='#'">Submit</button>

</li>

<li class="nav-item">

<button class="nav-button" onclick="location.href='#'">My Collections</button>

</li>

<li class="nav-item">

<button class="nav-button" onclick="location.href='#'">Text Bank</button>

</li>

<li class="nav-item">

<button class="nav-button" onclick="location.href='#'">Learn</button>

</li>

</ul>

<ul class="navbar-nav ml-auto">

{% if current\_user.is\_authenticated %}

<li class="nav-item">

<span class="navbar-text">user: {{ current\_user.username }}</span>

</li>

<li class="nav-item logout-btn">

<button class="nav-button" onclick="location.href='{{ url\_for('logout') }}'">Logout</button>

</li>

{% else %}

<li class="nav-item">

<button class="nav-button" onclick="location.href='{{ url\_for('login') }}'">Login</button>

</li>

<li class="nav-item">

<button class="nav-button" onclick="location.href='{{ url\_for('register') }}'">Register</button>

</li>

{% endif %}

</ul>

</div>

</nav>

<main role="main" class="container">

<div class="row">

<div class="col-md-12">

{% for message in get\_flashed\_messages(with\_categories=true) %}

<div class="alert alert-{{ message[0] }} flash-message">{{ message[1] }}</div>

{% endfor %}

{% block content %}{% endblock %}

</div>

</div>

</main>

<footer class="footer mt-auto py-3 bg-dark">

<div class="container">

<span class="text-muted">&copy; dictionLang {{ year }}. All Rights Reserved.</span>

</div>

</footer>

<script>

document.addEventListener('DOMContentLoaded', function() {

let currentYear = new Date().getFullYear();

let yearElement = document.querySelector('.footer .text-muted');

yearElement.innerHTML = yearElement.innerHTML.replace('{{ year }}', currentYear);

});

</script>

</body>

</html>

Index.html:  
{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}

<h1>Welcome to the Language Learning App!</h1>

{% endblock %}

Login.html:  
{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}

<h1>Login</h1>

<form method="POST" action="{{ url\_for('login') }}">

{{ form.hidden\_tag() }}

{{ form.email.label }} {{ form.email() }}<br>

{{ form.password.label }} {{ form.password() }}<br>

<input type="submit" value="Log In">

</form>

{% endblock %}

New\_text\_resource.html  
{% extends 'base.html' %}

{% block title %}Upload Text Resource{% endblock %}

{% block content %}

<h1>Upload a New Text Resource</h1>

<form method="POST" action="{{ url\_for('new\_text\_resource') }}">

{{ form.hidden\_tag() }}

{{ form.title.label }} {{ form.title() }}<br>

{{ form.content.label }} {{ form.content() }}<br>

{{ form.submit() }}

</form>

{% endblock %}

Register.html:  
{% extends 'base.html' %}

{% block title %}Register{% endblock %}

{% block content %}

<h1>Register</h1>

<form method="POST" action="{{ url\_for('register') }}">

{{ form.hidden\_tag() }}

{{ form.username.label }} {{ form.username() }}<br>

{{ form.email.label }} {{ form.email() }}<br>

{{ form.password.label }} {{ form.password() }}<br>

{{ form.confirm\_password.label }} {{ form.confirm\_password() }}<br>

<input type="submit" value="Register">

</form>

{% endblock %}

Text\_resources.html:  
{% extends 'base.html' %}

{% block title %}Text Resources{% endblock %}

{% block content %}

<h1>Your Text Resources</h1>

<a href="{{ url\_for('new\_text\_resource') }}">Upload a New Text Resource</a>

<ul>

{% for resource in resources %}

<li><a href="{{ url\_for('view\_text\_resource', text\_resource\_id=resource.id) }}">{{ resource.title }}</a></li>

{% endfor %}

</ul>

{% endblock %}

View\_text\_resource.html:  
{% extends 'base.html' %}

{% block content %}

<h2>{{ text\_resource.title }}</h2>

<pre>{{ text\_resource.content }}</pre>

<a href="{{ url\_for('text\_resources') }}">Back to Text Resources</a>

{% endblock %}

Instructions for you (which apply for all of our conversations):   
When changing, deleting, or adding code to any file, do not give the contents of the entire file unless explicitly asked for. This is to avoid your character limit. If you need to display long code files, you will warn me with “This is a big file. I will give you the first part now. Please ask me for the second part when you are ready”.

If you understand the purpose of my project and are ready to help me, simply reply with: yallah habibi!