Project Dard

Complete Documentation & Source Code

Generated on: July 03, 2025 at 04:20 PM

Project Overview

Project Dard is a bilingual community discussion platform built with Flask (Python) that facilitates academic discussions and knowledge sharing. The platform supports English and Persian (Farsi) languages with RTL text support, featuring weekly topics, stories, community-driven content moderation, and an administrative dashboard.

Key Features

• Bilingual support (English & Persian/Farsi) with RTL text direction
• Four main pages: Home, Archive, Info, and Appeal
• Community-driven comment system with like/dislike voting
• Automatic comment hiding (5 dislikes) with appeal process
• Admin dashboard for content management with delete functionality
• PDF export functionality for discussions
• Automatic Excel export for all database data
• PostgreSQL database with full CRUD operations
• Dark theme with modern, minimalistic design
• Visitor tracking and statistics
• Secure admin authentication system

Technical Stack

Component	Technology
Backend Framework	Flask (Python)
Database ORM	SQLAlchemy with Flask-SQLAlchemy
Authentication	Flask-Login
Form Handling	Flask-WTF with WTForms
Template Engine	Jinja2
CSS Framework	Bootstrap 5
Icons	Font Awesome 6
PDF Generation	ReportLab
Excel Export	openpyxl, pandas
Database	PostgreSQL (production)
Fonts	Vazirmatn (Persian), Google Fonts
Server	Gunicorn WSGI server

Project Structure

```
Project Dard/
├── app.py                 # Flask application setup and configuration
├── main.py                # Application entry point
├── models.py              # Database models (Admin, WeeklyTopic, etc.)
├── routes.py              # URL routes and view functions
├── forms.py               # WTForms form definitions
├── utils.py               # Utility functions
├── translations.py        # Internationalization support
├── excel_manager.py       # Excel export functionality
├── generate_documentation.py # PDF documentation generator
├── templates/             # Jinja2 templates
│   ├── base.html          # Base template
│   ├── home.html          # Home page template
│   ├── archive.html       # Archive page template
│   ├── info.html          # Info page template
│   ├── appeal.html        # Appeal page template
│   ├── admin/             # Admin templates
│       ├── login.html     # Admin login
│       ├── dashboard.html # Admin dashboard
│       ├── comments.html # Comment management
├── static/                # Static assets
│   ├── css/
│   │   ├── style.css      # Custom CSS styles
│   ├── js/
│   │   ├── main.js        # JavaScript functionality
│   ├── images/
│       ├── logo.jpg       # Project logo
├── excel_exports/         # Excel export files directory
├── pyproject.toml         # Project dependencies
├── replit.md             # Project documentation
```

Flask Application Setup

File: **app.py**

```
import os
import logging
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from sqlalchemy.orm import DeclarativeBase
from werkzeug.middleware.proxy_fix import ProxyFix

# Configure logging
logging.basicConfig(level=logging.DEBUG)

class Base(DeclarativeBase):
    pass

db = SQLAlchemy(model_class=Base)

# Create the app
app = Flask(__name__)
app.secret_key = os.environ.get("SESSION_SECRET", "dev-secret-key-change-in-production")
app.wsgi_app = ProxyFix(app.wsgi_app, x_proto=1, x_host=1)

# Configure the database
app.config["SQLALCHEMY_DATABASE_URI"] = os.environ.get("DATABASE_URL", "sqlite:///project_dard.db")
app.config["SQLALCHEMY_ENGINE_OPTIONS"] = {
    "pool_recycle": 300,
    "pool_pre_ping": True,
}

# Initialize extensions
db.init_app(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'admin_login'
login_manager.login_message = 'Please log in to access the admin area.'

@login_manager.user_loader
def load_user(user_id):
    from models import Admin
    return Admin.query.get(int(user_id))

# Add nl2br filter for templates
@app.template_filter('nl2br')
def nl2br_filter(text):
    """Convert newlines to HTML line breaks"""
    if text is None:
        return ''
    return text.replace('\n', '<br>')

# Import routes after app initialization
from routes import *

with app.app_context():
    # Import models to ensure tables are created
    import models
    db.create_all()

    # Create default admin if not exists
    from models import Admin
    from werkzeug.security import generate_password_hash

    if not Admin.query.first():
        admin = Admin(
            username='admin',
            email='admin@projectdard.com',
            password_hash=generate_password_hash('admin123')
        )
        db.session.add(admin)
        db.session.commit()
        logging.info("Default admin created: username=admin, password=admin123")
```

Application Entry Point

File: **main.py**

```
from app import app

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Database Models

File: **models.py**

```
from datetime import datetime
from app import db
from flask_login import UserMixin

class Admin(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(256), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

class WeeklyTopic(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title_en = db.Column(db.Text, nullable=False)
    title_fa = db.Column(db.Text)
    content_en = db.Column(db.Text, nullable=False)
    content_fa = db.Column(db.Text)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    is_active = db.Column(db.Boolean, default=True)
    admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'), nullable=False)

    comments = db.relationship('Comment', backref='topic', lazy=True)

class WeeklyStory(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title_en = db.Column(db.Text, nullable=False)
    title_fa = db.Column(db.Text)
    content_en = db.Column(db.Text, nullable=False)
    content_fa = db.Column(db.Text)
    author_en = db.Column(db.String(100))
    author_fa = db.Column(db.String(100))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    is_active = db.Column(db.Boolean, default=True)
    admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'), nullable=False)

class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.Text, nullable=False)
    language = db.Column(db.String(2), default='en')  # 'en' or 'fa'
    author_name = db.Column(db.String(100), nullable=False)
    author_email = db.Column(db.String(120))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    is_hidden = db.Column(db.Boolean, default=False)
    likes = db.Column(db.Integer, default=0)
    dislikes = db.Column(db.Integer, default=0)
    topic_id = db.Column(db.Integer, db.ForeignKey('weekly_topic.id'), nullable=False)

    votes = db.relationship('CommentVote', backref='comment', lazy=True)
    appeals = db.relationship('Appeal', backref='comment', lazy=True)

class CommentVote(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    comment_id = db.Column(db.Integer, db.ForeignKey('comment.id'), nullable=False)
    ip_address = db.Column(db.String(45), nullable=False)  # Support IPv6
    vote_type = db.Column(db.String(10), nullable=False)  # 'like' or 'dislike'
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    __table_args__ = (db.UniqueConstraint('comment_id', 'ip_address'),)

class Appeal(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    comment_id = db.Column(db.Integer, db.ForeignKey('comment.id'), nullable=False)
    appellant_name = db.Column(db.String(100), nullable=False)
    appellant_email = db.Column(db.String(120), nullable=False)
    reason = db.Column(db.Text, nullable=False)
    status = db.Column(db.String(20), default='pending')  # 'pending', 'approved', 'rejected'
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    reviewed_at = db.Column(db.DateTime)
    admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'))

class Visitor(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    ip_address = db.Column(db.String(45), nullable=False)
    user_agent = db.Column(db.Text)
    page_visited = db.Column(db.String(100))
    visited_at = db.Column(db.DateTime, default=datetime.utcnow)
```

URL Routes and Views

File: **routes.py**

```
import os
from datetime import datetime
from flask import render_template, request, redirect, url_for, flash, session, jsonify, make_response
from flask_login import login_user, logout_user, login_required, current_user
from werkzeug.security import check_password_hash, generate_password_hash
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet
from io import BytesIO

from app import app, db
from models import Admin, WeeklyTopic, WeeklyStory, Comment, CommentVote, Appeal, Visitor
from forms import LoginForm, CommentForm, WeeklyTopicForm, WeeklyStoryForm, AppealForm
from utils import get_client_ip, track_visitor
from translations import get_text, get_supported_languages

@app.before_request
def before_request():
    # Set default language
    if 'language' not in session:
        session['language'] = 'en'

    # Track visitor
    track_visitor()

@app.context_processor
def inject_globals():
    return {
        'get_text': get_text,
        'current_language': session.get('language', 'en'),
        'supported_languages': get_supported_languages()
    }

@app.route('/', methods=['GET', 'POST'])
def home():
    page = request.args.get('page', 1, type=int)
    per_page = 10

    # Get active topic
    active_topic = WeeklyTopic.query.filter_by(is_active=True).order_by(WeeklyTopic.created_at.desc()).f

    # Get active story
    active_story = WeeklyStory.query.filter_by(is_active=True).order_by(WeeklyStory.created_at.desc()).f

    # Get comments for active topic
    comments = None
    if active_topic:
        comments = Comment.query.filter_by(topic_id=active_topic.id, is_hidden=False)\
                                .order_by(Comment.created_at.desc())\
                                .paginate(page=page, per_page=per_page, error_out=False)

    # Comment form
    form = CommentForm()
    if form.validate_on_submit() and active_topic:
        comment = Comment(
            content=form.content.data,
            language=form.language.data,
            author_name=form.author_name.data,
            author_email=form.author_email.data,
            topic_id=active_topic.id
        )
        db.session.add(comment)
        db.session.commit()

        # Update Excel with latest data
        try:
            from excel_manager import excel_manager
            excel_manager.export_all_data()
        except Exception as e:
            print(f"Excel export error: {e}")

        flash(get_text('comment_posted'), 'success')
        return redirect(url_for('home'))

    return render_template('home.html',
                           active_topic=active_topic,
                           active_story=active_story,
                           comments=comments,
```

```
                          form=form)

@app.route('/vote_comment/<int:comment_id>/<vote_type>')
def vote_comment(comment_id, vote_type):
    if vote_type not in ['like', 'dislike']:
        return jsonify({'error': 'Invalid vote type'}), 400

    comment = Comment.query.get_or_404(comment_id)
    ip_address = get_client_ip()

    # Check if user already voted
    existing_vote = CommentVote.query.filter_by(comment_id=comment_id, ip_address=ip_address).first()

    if existing_vote:
        if existing_vote.vote_type == vote_type:
            # Remove vote
            db.session.delete(existing_vote)
            if vote_type == 'like':
                comment.likes -= 1
            else:
                comment.dislikes -= 1
        else:

            # Change vote
            existing_vote.vote_type = vote_type
            if vote_type == 'like':
                comment.likes += 1
                comment.dislikes -= 1
            else:
                comment.dislikes += 1
                comment.likes -= 1
    else:
        # New vote
        vote = CommentVote(comment_id=comment_id, ip_address=ip_address, vote_type=vote_type)
        db.session.add(vote)
        if vote_type == 'like':
            comment.likes += 1
        else:
            comment.dislikes += 1

    # Auto-hide comment if it has 5 or more dislikes
    if comment.dislikes >= 5:
        comment.is_hidden = True

    db.session.commit()

    return jsonify({
        'likes': comment.likes,
        'dislikes': comment.dislikes,
        'hidden': comment.is_hidden
    })

@app.route('/archive')
def archive():
    page = request.args.get('page', 1, type=int)
    per_page = 10

    topics = WeeklyTopic.query.order_by(WeeklyTopic.created_at.desc())\
                              .paginate(page=page, per_page=per_page, error_out=False)

    return render_template('archive.html', topics=topics)

@app.route('/download_discussion/<int:topic_id>')
def download_discussion(topic_id):
    topic = WeeklyTopic.query.get_or_404(topic_id)
    comments = Comment.query.filter_by(topic_id=topic_id, is_hidden=False)\
                            .order_by(Comment.created_at.asc()).all()

    # Create PDF
    buffer = BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=letter)
    styles = getSampleStyleSheet()
    story = []

    # Title
    title = topic.title_en if session.get('language') == 'en' else (topic.title_fa or topic.title_en)
    story.append(Paragraph(title, styles['Title']))
    story.append(Spacer(1, 12))

    # Content
    content = topic.content_en if session.get('language') == 'en' else (topic.content_fa or topic.conter
    story.append(Paragraph(content, styles['Normal']))
    story.append(Spacer(1, 12))
    # Comments
```

```
    story.append(Paragraph("Comments", styles['Heading1']))
    story.append(Spacer(1, 12))

    for comment in comments:
        story.append(Paragraph(f"<b>{comment.author_name}</b> - {comment.created_at.strftime('%Y-%m-%d %
        story.append(Paragraph(comment.content, styles['Normal']))
        story.append(Spacer(1, 6))

    doc.build(story)

    buffer.seek(0)
    response = make_response(buffer.getvalue())
    response.headers['Content-Type'] = 'application/pdf'
    response.headers['Content-Disposition'] = f'attachment; filename=discussion_{topic_id}.pdf'

    return response

@app.route('/info')
def info():
    return render_template('info.html')

@app.route('/appeal', methods=['GET', 'POST'])
def appeal():
    form = AppealForm()

    if form.validate_on_submit():
        # Get comment ID from form or session
        comment_id = request.form.get('comment_id')
        if not comment_id:
            flash(get_text('invalid_comment'), 'error')
            return redirect(url_for('appeal'))

        comment = Comment.query.get(comment_id)
        if not comment or not comment.is_hidden:
            flash(get_text('comment_not_hidden'), 'error')
            return redirect(url_for('appeal'))

        appeal = Appeal(

            comment_id=comment_id,
            appellant_name=form.appellant_name.data,
            appellant_email=form.appellant_email.data,
            reason=form.reason.data
        )
        db.session.add(appeal)
        db.session.commit()

        flash(get_text('appeal_submitted'), 'success')
        return redirect(url_for('appeal'))

    return render_template('appeal.html', form=form)

@app.route('/set_language/<language>')
def set_language(language):
    if language in get_supported_languages():
        session['language'] = language
    return redirect(request.referrer or url_for('home'))

# Admin routes
@app.route('/admin/login', methods=['GET', 'POST'])
def admin_login():
    if current_user.is_authenticated:
        return redirect(url_for('admin_dashboard'))

    form = LoginForm()
    if form.validate_on_submit():
        admin = Admin.query.filter_by(username=form.username.data).first()
        if admin and check_password_hash(admin.password_hash, form.password.data):
            login_user(admin, remember=form.remember_me.data)
            return redirect(url_for('admin_dashboard'))
        flash(get_text('invalid_credentials'), 'error')

    return render_template('admin/login.html', form=form)

@app.route('/admin/logout')
@login_required
def admin_logout():
    logout_user()
    return redirect(url_for('home'))

@app.route('/admin')
@login_required
def admin_dashboard():
    # Statistics
```

```
    total_topics = WeeklyTopic.query.count()
    total_comments = Comment.query.count()
    hidden_comments = Comment.query.filter_by(is_hidden=True).count()
    pending_appeals = Appeal.query.filter_by(status='pending').count()

    recent_comments = Comment.query.order_by(Comment.created_at.desc()).limit(5).all()

    # Get current topics and stories for management
    current_topics = WeeklyTopic.query.order_by(WeeklyTopic.created_at.desc()).limit(5).all()
    current_stories = WeeklyStory.query.order_by(WeeklyStory.created_at.desc()).limit(5).all()

    return render_template('admin/dashboard.html',
                           total_topics=total_topics,
                           total_comments=total_comments,
                           hidden_comments=hidden_comments,
                           pending_appeals=pending_appeals,
                           recent_comments=recent_comments,
                           current_topics=current_topics,
                           current_stories=current_stories)

@app.route('/admin/post_topic', methods=['GET', 'POST'])
@login_required
def admin_post_topic():
    form = WeeklyTopicForm()

    if form.validate_on_submit():
        # Deactivate current active topic
        current_active = WeeklyTopic.query.filter_by(is_active=True).first()
        if current_active:
            current_active.is_active = False

        topic = WeeklyTopic(
            title_en=form.title_en.data,
            title_fa=form.title_fa.data,
            content_en=form.content_en.data,
            content_fa=form.content_fa.data,
            admin_id=current_user.id
        )
        db.session.add(topic)
        db.session.commit()

        # Update Excel with latest data
        try:
            from excel_manager import excel_manager
            excel_manager.export_all_data()
        except Exception as e:
            print(f"Excel export error: {e}")

        flash(get_text('topic_posted'), 'success')
        return redirect(url_for('admin_dashboard'))

    return render_template('admin/dashboard.html', topic_form=form)

@app.route('/admin/post_story', methods=['GET', 'POST'])
@login_required
def admin_post_story():
    form = WeeklyStoryForm()

    if form.validate_on_submit():
        # Deactivate current active story
        current_active = WeeklyStory.query.filter_by(is_active=True).first()
        if current_active:
            current_active.is_active = False

        story = WeeklyStory(
            title_en=form.title_en.data,
            title_fa=form.title_fa.data,
            content_en=form.content_en.data,
            content_fa=form.content_fa.data,
            author_en=form.author_en.data,
            author_fa=form.author_fa.data,
            admin_id=current_user.id
        )
        db.session.add(story)
        db.session.commit()

        # Update Excel with latest data
        try:
            from excel_manager import excel_manager
            excel_manager.export_all_data()
        except Exception as e:
            print(f"Excel export error: {e}")
        flash(get_text('story_posted'), 'success')
```

```
        return redirect(url_for('admin_dashboard'))

    return render_template('admin/dashboard.html', story_form=form)

@app.route('/admin/comments')
@login_required
def admin_comments():
    page = request.args.get('page', 1, type=int)
    show_hidden = request.args.get('hidden', 'false') == 'true'

    query = Comment.query
    if show_hidden:
        query = query.filter_by(is_hidden=True)

    comments = query.order_by(Comment.created_at.desc())\
                    .paginate(page=page, per_page=20, error_out=False)

    appeals = Appeal.query.filter_by(status='pending').all()

    return render_template('admin/comments.html',
                           comments=comments,
                           appeals=appeals,
                           show_hidden=show_hidden)

@app.route('/admin/toggle_comment/<int:comment_id>')
@login_required
def toggle_comment(comment_id):
    comment = Comment.query.get_or_404(comment_id)
    comment.is_hidden = not comment.is_hidden
    db.session.commit()

    action = 'hidden' if comment.is_hidden else 'unhidden'
    flash(f'Comment {action} successfully', 'success')

    return redirect(url_for('admin_comments'))

@app.route('/admin/delete_comment/<int:comment_id>')
@login_required
def delete_comment(comment_id):
    comment = Comment.query.get_or_404(comment_id)
    db.session.delete(comment)
    db.session.commit()

    flash('Comment deleted successfully', 'success')
    return redirect(url_for('admin_comments'))

@app.route('/admin/review_appeal/<int:appeal_id>/<action>')
@login_required
def review_appeal(appeal_id, action):
    if action not in ['approve', 'reject']:
        flash('Invalid action', 'error')
        return redirect(url_for('admin_comments'))

    appeal = Appeal.query.get_or_404(appeal_id)
    appeal.status = 'approved' if action == 'approve' else 'rejected'
    appeal.reviewed_at = datetime.utcnow()
    appeal.admin_id = current_user.id

    if action == 'approve':
        appeal.comment.is_hidden = False

    db.session.commit()

    # Update Excel with latest data
    try:
        from excel_manager import excel_manager
        excel_manager.export_all_data()
    except Exception as e:
        print(f"Excel export error: {e}")

    flash(f'Appeal {action}d successfully', 'success')

    return redirect(url_for('admin_comments'))

@app.route('/admin/topic/<int:topic_id>/delete', methods=['POST'])
@login_required
def delete_topic(topic_id):
    topic = WeeklyTopic.query.get_or_404(topic_id)

    # Delete all associated comments and their votes/appeals first
    comments = Comment.query.filter_by(topic_id=topic.id).all()
    for comment in comments:
        # Delete votes for this comment
```

```
            CommentVote.query.filter_by(comment_id=comment.id).delete()
            # Delete appeals for this comment
            Appeal.query.filter_by(comment_id=comment.id).delete()

        # Delete all comments for this topic
        Comment.query.filter_by(topic_id=topic.id).delete()

        # Delete the topic
        db.session.delete(topic)
        db.session.commit()

        flash('Topic and all associated data deleted successfully!', 'success')

        # Update Excel with latest data
        try:
            from excel_manager import excel_manager
            excel_manager.export_all_data()
        except Exception as e:
            print(f"Excel export error: {e}")

        return redirect(url_for('admin_dashboard'))

    @app.route('/admin/story/<int:story_id>/delete', methods=['POST'])
    @login_required
    def delete_story(story_id):
        story = WeeklyStory.query.get_or_404(story_id)

        # Delete the story
        db.session.delete(story)
        db.session.commit()

        flash('Story deleted successfully!', 'success')

        # Update Excel with latest data
        try:
            from excel_manager import excel_manager
            excel_manager.export_all_data()
        except Exception as e:
            print(f"Excel export error: {e}")

        return redirect(url_for('admin_dashboard'))

    @app.route('/admin/export-excel')
    @login_required
    def export_excel():
        """Export all data to Excel files"""
        try:
            from excel_manager import excel_manager
            exported_files = excel_manager.export_all_data()
            flash(f'Excel files exported successfully! {len(exported_files)} files created.', 'success')
        except Exception as e:
            flash(f'Error exporting Excel files: {str(e)}', 'danger')

        return redirect(url_for('admin_dashboard'))
```

Form Definitions

File: **forms.py**

```
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, PasswordField, SelectField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Email, Length, Optional

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Sign In')

class CommentForm(FlaskForm):
    author_name = StringField('Name', validators=[DataRequired(), Length(min=2, max=100)])
    author_email = StringField('Email', validators=[Optional(), Email()])
    content = TextAreaField('Comment', validators=[DataRequired(), Length(min=10, max=700)])
    language = SelectField('Language', choices=[('en', 'English'), ('fa', '[redacted]')], default='en')
    submit = SubmitField('Post Comment')

class WeeklyTopicForm(FlaskForm):
    title_en = StringField('Title (English)', validators=[DataRequired()])
    title_fa = StringField('Title (Persian)', validators=[Optional()])
    content_en = TextAreaField('Content (English)', validators=[DataRequired()])
    content_fa = TextAreaField('Content (Persian)', validators=[Optional()])
    submit = SubmitField('Post Topic')

class WeeklyStoryForm(FlaskForm):
    title_en = StringField('Title (English)', validators=[DataRequired()])
    title_fa = StringField('Title (Persian)', validators=[Optional()])
    content_en = TextAreaField('Content (English)', validators=[DataRequired()])
    content_fa = TextAreaField('Content (Persian)', validators=[Optional()])
    author_en = StringField('Author (English)', validators=[Optional()])
    author_fa = StringField('Author (Persian)', validators=[Optional()])
    submit = SubmitField('Post Story')

class AppealForm(FlaskForm):
    appellant_name = StringField('Your Name', validators=[DataRequired(), Length(min=2, max=100)])
    appellant_email = StringField('Your Email', validators=[DataRequired(), Email()])
    reason = TextAreaField('Reason for Appeal', validators=[DataRequired(), Length(min=20, max=1000)])
    submit = SubmitField('Submit Appeal')
```

Utility Functions

File: **utils.py**

```
    from flask import request, session
    from app import db
    from models import Visitor

    def get_client_ip():
        """Get the client's IP address"""
        if request.environ.get('HTTP_X_FORWARDED_FOR'):
            return request.environ['HTTP_X_FORWARDED_FOR'].split(',')[0].strip()
        elif request.environ.get('HTTP_X_REAL_IP'):
            return request.environ['HTTP_X_REAL_IP']
        else:
            return request.environ.get('REMOTE_ADDR', 'unknown')

    def track_visitor():
        """Track visitor statistics"""
        ip_address = get_client_ip()
        user_agent = request.headers.get('User-Agent', '')
        page_visited = request.endpoint or 'unknown'

        visitor = Visitor(
            ip_address=ip_address,
            user_agent=user_agent,
            page_visited=page_visited
        )

        try:
            db.session.add(visitor)
            db.session.commit()
        except Exception:
            db.session.rollback()

    def get_language_direction(language):
        """Get text direction for language"""
        rtl_languages = ['fa', 'ar', 'he', 'ur']
        return 'rtl' if language in rtl_languages else 'ltr'

    def format_date(date, language='en'):
        """Format date based on language"""
        if language == 'fa':
            # Persian date formatting would go here
            # For now, return English format
            return date.strftime('%Y/%m/%d')
        else:
            return date.strftime('%B %d, %Y')
```

Internationalization

File: **translations.py**

```
from flask import session

# Translation dictionary
translations = {
    'en': {
        # Navigation
        'home': 'Home',
        'archive': 'Archive',
        'info': 'Info',
        'appeal': 'Appeal',
        'admin': 'Admin',
        'language': 'Language',

        # Home page
        'weekly_topic': 'Weekly Topic',
        'weekly_story': 'Weekly Story',
        'comments': 'Comments',
        'post_comment': 'Post Comment',
        'name': 'Name',
        'email': 'Email (optional)',
        'comment': 'Comment',
        'comment_language': 'Comment Language',
        'english': 'English',
        'persian': '[redacted]',
        'like': 'Like',
        'dislike': 'Dislike',
        'reply': 'Reply',
        'no_comments': 'No comments yet. Be the first to comment!',
        'comment_posted': 'Comment posted successfully!',
        'comment_hidden': 'This comment has been hidden due to community feedback.',

        # Archive page
        'discussion_archive': 'Discussion Archive',
        'download_pdf': 'Download PDF',
        'no_topics': 'No topics available.',

        # Info page
        'about_platform': 'About Project Dard',
        'platform_description': 'Project Dard is a community platform for academic discussions and knowl
        'founder': 'Founder',
        'team': 'Team',
        'featured_author': 'Featured Author',

        # Appeal page
        'appeal_form': 'Appeal Form',
        'appeal_description': 'If your comment was hidden, you can submit an appeal here.',
        'your_name': 'Your Name',
        'your_email': 'Your Email',
        'reason_for_appeal': 'Reason for Appeal',
        'submit_appeal': 'Submit Appeal',
        'appeal_submitted': 'Appeal submitted successfully!',
        'contact_info': 'Contact Information',
        'social_media': 'Social Media',

        # Admin
        'login': 'Login',
        'username': 'Username',
        'password': 'Password',
        'remember_me': 'Remember Me',
        'sign_in': 'Sign In',
        'dashboard': 'Dashboard',
        'statistics': 'Statistics',
        'recent_activity': 'Recent Activity',
        'manage_comments': 'Manage Comments',
        'post_topic': 'Post Topic',
        'post_story': 'Post Story',
        'logout': 'Logout',

        # Messages
        'invalid_credentials': 'Invalid username or password',
        'access_denied': 'Access denied',
        'invalid_comment': 'Invalid comment reference',
        'comment_not_hidden': 'Comment is not hidden',
        'topic_posted': 'Topic posted successfully!',
        'story_posted': 'Story posted successfully!',

        # Form labels
```

```
    'title_english': 'Title (English)',
    'title_persian': 'Title (Persian)',
    'content_english': 'Content (English)',
    'content_persian': 'Content (Persian)',
    'author_english': 'Author (English)',
    'author_persian': 'Author (Persian)',
},
'fa': {
    # Navigation
    'home': '[redacted]',
    'archive': '[redacted]',
    'info': '[redacted]',
    'appeal': '[redacted]',
    'admin': '[redacted]',
    'language': '[redacted]',

    # Home page
    'weekly_topic': '[redacted]',
    'weekly_story': '[redacted]',
    'comments': '[redacted]',
    'post_comment': '[redacted]',
    'name': '[redacted]',
    'email': '[redacted]',
    'comment': '[redacted]',
    'comment_language': '[redacted]',
    'english': 'English',
    'persian': '[redacted]',
    'like': '[redacted]',
    'dislike': '[redacted]',
    'reply': '[redacted]',
    'no_comments': '[redacted]',
    'comment_posted': '[redacted]',
    'comment_hidden': '[redacted]',

    # Archive page
    'discussion_archive': '[redacted]',
    'download_pdf': '[redacted]',
    'no_topics': '[redacted]',

    # Info page
    'about_platform': '[redacted]',
    'platform_description': '[redacted]',
    'founder': '[redacted]',
    'team': '[redacted]',
    'featured_author': '[redacted]',

    # Appeal page
    'appeal_form': '[redacted]',
    'appeal_description': '[redacted]',
    'your_name': '[redacted]',
    'your_email': '[redacted]',
    'reason_for_appeal': '[redacted]',
    'submit_appeal': '[redacted]',
    'appeal_submitted': '[redacted]',
    'contact_info': '[redacted]',
    'social_media': '[redacted]',

    # Admin
    'login': '[redacted]',
    'username': '[redacted]',
    'password': '[redacted]',
    'remember_me': '[redacted]',
    'sign_in': '[redacted]',
    'dashboard': '[redacted]',
    'statistics': '[redacted]',
    'recent_activity': '[redacted]',
    'manage_comments': '[redacted]',
    'post_topic': '[redacted]',
    'post_story': '[redacted]',
    'logout': '[redacted]',

    # Messages
    'invalid_credentials': '[redacted]',
    'access_denied': '[redacted]',
    'invalid_comment': '[redacted]',
    'comment_not_hidden': '[redacted]',
    'topic_posted': '[redacted]',
    'story_posted': '[redacted]',

    # Form labels
    'title_english': '[redacted]',
    'title_persian': '[redacted]',
```

```
        'content_english': '[redacted] ([redacted])',
        'content_persian': '[redacted] ([redacted])',
        'author_english': '[redacted] ([redacted])',
        'author_persian': '[redacted] ([redacted])',
    }
}

def get_text(key, language=None):
    """Get translated text for a key"""
    if language is None:
        language = session.get('language', 'en')

    return translations.get(language, {}).get(key, translations['en'].get(key, key))

def get_supported_languages():
    """Get list of supported languages"""
    return list(translations.keys())
```

Excel Export Manager

File: **excel_manager.py**

```
"""
Excel Manager for Project Dard
Handles automatic Excel file creation and data export
"""

import os
import pandas as pd
from datetime import datetime
from openpyxl import Workbook, load_workbook
from openpyxl.styles import Font, PatternFill, Alignment
from openpyxl.utils.dataframe import dataframe_to_rows

class ExcelManager:
    def __init__(self, excel_dir='excel_exports'):
        self.excel_dir = excel_dir
        self.ensure_directory_exists()

    def ensure_directory_exists(self):
        """Create excel exports directory if it doesn't exist"""
        if not os.path.exists(self.excel_dir):
            os.makedirs(self.excel_dir)

    def get_excel_path(self, filename):
        """Get full path for Excel file"""
        return os.path.join(self.excel_dir, filename)

    def create_or_update_topics_excel(self, topics_data):
        """Create or update topics Excel file"""
        filename = 'weekly_topics.xlsx'
        filepath = self.get_excel_path(filename)

        # Convert to DataFrame
        df = pd.DataFrame(topics_data)

        if os.path.exists(filepath):
            # Load existing workbook
            with pd.ExcelWriter(filepath, engine='openpyxl', mode='a', if_sheet_exists='replace') as wri
                df.to_excel(writer, sheet_name='Topics', index=False)
        else:
            # Create new workbook
            with pd.ExcelWriter(filepath, engine='openpyxl') as writer:
                df.to_excel(writer, sheet_name='Topics', index=False)

        self.format_excel_file(filepath, 'Topics')
        return filepath

    def create_or_update_stories_excel(self, stories_data):
        """Create or update stories Excel file"""
        filename = 'weekly_stories.xlsx'
        filepath = self.get_excel_path(filename)

        # Convert to DataFrame
        df = pd.DataFrame(stories_data)

        if os.path.exists(filepath):
            # Load existing workbook
            with pd.ExcelWriter(filepath, engine='openpyxl', mode='a', if_sheet_exists='replace') as wri
                df.to_excel(writer, sheet_name='Stories', index=False)
        else:
            # Create new workbook
            with pd.ExcelWriter(filepath, engine='openpyxl') as writer:
                df.to_excel(writer, sheet_name='Stories', index=False)

        self.format_excel_file(filepath, 'Stories')
        return filepath

    def create_or_update_comments_excel(self, comments_data):
        """Create or update comments Excel file"""
        filename = 'comments.xlsx'
        filepath = self.get_excel_path(filename)

        # Convert to DataFrame
        df = pd.DataFrame(comments_data)

        if os.path.exists(filepath):
            # Load existing workbook
            with pd.ExcelWriter(filepath, engine='openpyxl', mode='a', if_sheet_exists='replace') as wri
                df.to_excel(writer, sheet_name='Comments', index=False)
```

```
        else:
            # Create new workbook
            with pd.ExcelWriter(filepath, engine='openpyxl') as writer:
                df.to_excel(writer, sheet_name='Comments', index=False)

        self.format_excel_file(filepath, 'Comments')
        return filepath

    def create_or_update_appeals_excel(self, appeals_data):
        """Create or update appeals Excel file"""
        filename = 'appeals.xlsx'
        filepath = self.get_excel_path(filename)

        # Convert to DataFrame
        df = pd.DataFrame(appeals_data)

        if os.path.exists(filepath):
            # Load existing workbook
            with pd.ExcelWriter(filepath, engine='openpyxl', mode='a', if_sheet_exists='replace') as wri
                df.to_excel(writer, sheet_name='Appeals', index=False)
        else:
            # Create new workbook

            with pd.ExcelWriter(filepath, engine='openpyxl') as writer:
                df.to_excel(writer, sheet_name='Appeals', index=False)

        self.format_excel_file(filepath, 'Appeals')
        return filepath

    def create_or_update_visitors_excel(self, visitors_data):
        """Create or update visitors Excel file"""
        filename = 'visitors.xlsx'
        filepath = self.get_excel_path(filename)

        # Convert to DataFrame
        df = pd.DataFrame(visitors_data)

        if os.path.exists(filepath):
            # Load existing workbook
            with pd.ExcelWriter(filepath, engine='openpyxl', mode='a', if_sheet_exists='replace') as wri
                df.to_excel(writer, sheet_name='Visitors', index=False)
        else:
            # Create new workbook
            with pd.ExcelWriter(filepath, engine='openpyxl') as writer:
                df.to_excel(writer, sheet_name='Visitors', index=False)

        self.format_excel_file(filepath, 'Visitors')
        return filepath

    def format_excel_file(self, filepath, sheet_name):
        """Format Excel file with proper styling"""
        try:
            wb = load_workbook(filepath)
            ws = wb[sheet_name]

            # Header formatting
            header_font = Font(bold=True, color="FFFFFF")
            header_fill = PatternFill(start_color="366092", end_color="366092", fill_type="solid")
            header_alignment = Alignment(horizontal="center", vertical="center")

            # Apply header formatting
            for cell in ws[1]:
                cell.font = header_font
                cell.fill = header_fill
                cell.alignment = header_alignment

            # Auto-adjust column widths
            for column in ws.columns:
                max_length = 0
                column_letter = column[0].column_letter
                for cell in column:
                    try:
                        if len(str(cell.value)) > max_length:

                            max_length = len(str(cell.value))
                    except:
                        pass
                adjusted_width = min(max_length + 2, 50)
                ws.column_dimensions[column_letter].width = adjusted_width

            wb.save(filepath)
        except Exception as e:
            print(f"Error formatting Excel file {filepath}: {str(e)}")
    def export_all_data(self):
```

```
"""Export all database data to Excel files"""
from app import app, db

with app.app_context():
    from models import WeeklyTopic, WeeklyStory, Comment, Appeal, Visitor

    # Topics
    topics = WeeklyTopic.query.all()
    topics_data = []
    for topic in topics:
        topics_data.append({
            'ID': topic.id,
            'Title (English)': topic.title_en,
            'Title (Persian)': topic.title_fa,
            'Content (English)': topic.content_en[:500] + '...' if len(topic.content_en) > 500 e
            'Content (Persian)': topic.content_fa[:500] + '...' if topic.content_fa and len(topi
            'Created At': topic.created_at.strftime('%Y-%m-%d %H:%M:%S'),
            'Is Active': topic.is_active,
            'Admin ID': topic.admin_id
        })

    # Stories
    stories = WeeklyStory.query.all()
    stories_data = []
    for story in stories:
        stories_data.append({
            'ID': story.id,
            'Title (English)': story.title_en,
            'Title (Persian)': story.title_fa,
            'Content (English)': story.content_en[:500] + '...' if len(story.content_en) > 500 e
            'Content (Persian)': story.content_fa[:500] + '...' if story.content_fa and len(stor
            'Author (English)': story.author_en,
            'Author (Persian)': story.author_fa,
            'Created At': story.created_at.strftime('%Y-%m-%d %H:%M:%S'),
            'Is Active': story.is_active,
            'Admin ID': story.admin_id
        })

    # Comments

    comments = Comment.query.all()
    comments_data = []
    for comment in comments:
        comments_data.append({
            'ID': comment.id,
            'Content': comment.content[:300] + '...' if len(comment.content) > 300 else comment.
            'Language': comment.language,
            'Author Name': comment.author_name,
            'Author Email': comment.author_email,
            'Created At': comment.created_at.strftime('%Y-%m-%d %H:%M:%S'),
            'Is Hidden': comment.is_hidden,
            'Likes': comment.likes,
            'Dislikes': comment.dislikes,
            'Topic ID': comment.topic_id
        })

    # Appeals
    appeals = Appeal.query.all()
    appeals_data = []
    for appeal in appeals:
        appeals_data.append({
            'ID': appeal.id,
            'Comment ID': appeal.comment_id,
            'Appellant Name': appeal.appellant_name,
            'Appellant Email': appeal.appellant_email,
            'Reason': appeal.reason[:300] + '...' if len(appeal.reason) > 300 else appeal.reason
            'Status': appeal.status,
            'Created At': appeal.created_at.strftime('%Y-%m-%d %H:%M:%S'),
            'Reviewed At': appeal.reviewed_at.strftime('%Y-%m-%d %H:%M:%S') if appeal.reviewed_a
            'Admin ID': appeal.admin_id
        })

    # Visitors
    visitors = Visitor.query.all()
    visitors_data = []
    for visitor in visitors:
        visitors_data.append({
            'ID': visitor.id,
            'IP Address': visitor.ip_address,
            'User Agent': visitor.user_agent[:100] + '...' if visitor.user_agent and len(visitor
            'Page Visited': visitor.page_visited,
            'Visited At': visitor.visited_at.strftime('%Y-%m-%d %H:%M:%S')
        })
    # Create Excel files
```

```
            exported_files = []
            if topics_data:
                exported_files.append(self.create_or_update_topics_excel(topics_data))
            if stories_data:
                exported_files.append(self.create_or_update_stories_excel(stories_data))
            if comments_data:
                exported_files.append(self.create_or_update_comments_excel(comments_data))
            if appeals_data:
                exported_files.append(self.create_or_update_appeals_excel(appeals_data))
            if visitors_data:
                exported_files.append(self.create_or_update_visitors_excel(visitors_data))

            return exported_files

# Global instance
excel_manager = ExcelManager()
```

Base Template

File: **templates/base.html**

```
<!DOCTYPE html>
<html lang="{{ current_language }}" dir="{{ 'rtl' if current_language == 'fa' else 'ltr' }}">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Project Dard{% endblock %}</title>

    <!-- Bootstrap 5 CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet
    <!-- Font Awesome Icons -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="styleshe
    <!-- Persian Font -->
    <link href="https://fonts.googleapis.com/css2?family=Vazirmatn:wght@300;400;500;700&display=swap" re
    <!-- Custom CSS -->
    <link href="{{ url_for('static', filename='css/style.css') }}" rel="stylesheet">
</head>
<body class="bg-dark text-light">
    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <div class="container">
            <a class="navbar-brand d-flex align-items-center" href="{{ url_for('home') }}">
                <img src="{{ url_for('static', filename='images/logo.jpg') }}" alt="Project Dard" width=
                <span class="fw-bold">Project Dard</span>
            </a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navb
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('home') }}">
                            <i class="fas fa-home me-1"></i>{{ get_text('home') }}
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('archive') }}">
                            <i class="fas fa-archive me-1"></i>{{ get_text('archive') }}
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('info') }}">
                            <i class="fas fa-info-circle me-1"></i>{{ get_text('info') }}
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('appeal') }}">
                            <i class="fas fa-gavel me-1"></i>{{ get_text('appeal') }}
                        </a>

                    </li>
                </ul>

                <!-- Language Switcher -->
                <div class="navbar-nav me-3">
                    <div class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropd
                            <i class="fas fa-globe me-1"></i>{{ current_language.upper() }}
                        </a>
                        <ul class="dropdown-menu dropdown-menu-dark">
                            <li><a class="dropdown-item" href="{{ url_for('set_language', language='en')
                            <li><a class="dropdown-item" href="{{ url_for('set_language', language='fa')
                        </ul>
                    </div>
                </div>

                <!-- Admin Link -->
                <div class="navbar-nav">
                    {% if current_user.is_authenticated %}
                        <a class="nav-link" href="{{ url_for('admin_dashboard') }}">
                            <i class="fas fa-cog me-1"></i>{{ get_text('dashboard') }}
                        </a>
                        <a class="nav-link" href="{{ url_for('admin_logout') }}">
                            <i class="fas fa-sign-out-alt me-1"></i>{{ get_text('logout') }}
                        </a>
                    {% else %}
                        <a class="nav-link" href="{{ url_for('admin_login') }}">
```

```
                            <i class="fas fa-sign-in-alt me-1"></i>{{ get_text('admin') }}
                        </a>
                    {% endif %}
                </div>
            </div>
        </div>
    </nav>

    <!-- Main Content -->
    <main class="container mt-5 pt-4">
        <!-- Flash Messages -->
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for category, message in messages %}
                    <div class="alert alert-{{ 'danger' if category == 'error' else 'success' if categor
                        {{ message }}
                        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
                    </div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        {% block content %}{% endblock %}

    </main>

    <!-- Footer -->
    <footer class="bg-dark text-light py-4 mt-5">
        <div class="container">
            <div class="row">
                <div class="col-md-6">
                    <h5>Project Dard</h5>
                    <p class="mb-0">{{ get_text('platform_description') }}</p>
                </div>
                <div class="col-md-6 text-md-end">
                    <div class="social-links">
                        <a href="#" class="text-light me-3"><i class="fab fa-twitter"></i></a>
                        <a href="#" class="text-light me-3"><i class="fab fa-facebook"></i></a>
                        <a href="#" class="text-light me-3"><i class="fab fa-instagram"></i></a>
                        <a href="#" class="text-light"><i class="fab fa-linkedin"></i></a>
                    </div>
                </div>
            </div>
            <hr class="my-3">
            <div class="text-center">
                <small>&copy; 2025 Project Dard. All rights reserved.</small>
            </div>
        </div>
    </footer>

    <!-- Bootstrap 5 JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
    <!-- Custom JS -->
    <script src="{{ url_for('static', filename='js/main.js') }}"></script>
</body>
</html>
```

Home Page Template

File: **templates/home.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('home') }} - Project Dard{% endblock %}

{% block content %}
<div class="row">
    <!-- Main Content -->
    <div class="col-lg-8">
        <!-- Weekly Topic -->
        {% if active_topic %}
        <div class="card bg-secondary mb-4">
            <div class="card-header">
                <h2 class="card-title mb-0">
                    <i class="fas fa-comments me-2"></i>{{ get_text('weekly_topic') }}
                </h2>
            </div>
            <div class="card-body">
                <h3>{{ active_topic.title_fa if current_language == 'fa' and active_topic.title_fa else
                <div class="mt-3">
                    {{ (active_topic.content_fa if current_language == 'fa' and active_topic.content_fa
                </div>
                <small class="text-muted">
                    <i class="fas fa-clock me-1"></i>
                    {{ active_topic.created_at.strftime('%B %d, %Y') }}
                </small>
            </div>
        </div>
        {% endif %}

        <!-- Comments Section -->
        <div class="card bg-secondary">
            <div class="card-header d-flex justify-content-between align-items-center">
                <h4 class="mb-0">
                    <i class="fas fa-comments me-2"></i>{{ get_text('comments') }}
                </h4>
                {% if comments %}
                <span class="badge bg-primary">{{ comments.total }}</span>
                {% endif %}
            </div>
            <div class="card-body">
                <!-- Comment Form -->
                {% if active_topic %}
                <form method="POST" class="mb-4">
                    {{ form.hidden_tag() }}
                    <div class="row">
                        <div class="col-md-6 mb-3">
                            {{ form.author_name.label(class="form-label") }}
                            {{ form.author_name(class="form-control bg-dark text-light border-secondary"
                        </div>
                        <div class="col-md-6 mb-3">

                            {{ form.author_email.label(class="form-label") }}
                            {{ form.author_email(class="form-control bg-dark text-light border-secondary
                        </div>
                    </div>
                    <div class="mb-3">
                        {{ form.language.label(class="form-label") }}
                        {{ form.language(class="form-select bg-dark text-light border-secondary") }}
                    </div>
                    <div class="mb-3">
                        {{ form.content.label(class="form-label") }}
                        {{ form.content(class="form-control bg-dark text-light border-secondary", rows="
                        <div class="form-text">Maximum 700 characters</div>
                    </div>
                    <button type="submit" class="btn btn-primary">
                        <i class="fas fa-paper-plane me-1"></i>{{ get_text('post_comment') }}
                    </button>
                </form>
                {% endif %}

                <!-- Comments List -->
                {% if comments and comments.items %}
                    {% for comment in comments.items %}
                    <div class="comment-item border-bottom border-secondary pb-3 mb-3" data-comment-id="
                        <div class="d-flex justify-content-between align-items-start mb-2">
                            <div>
                                <strong>{{ comment.author_name }}</strong>
                                <small class="text-muted ms-2">
```

```
                                    <i class="fas fa-clock me-1"></i>
                                    {{ comment.created_at.strftime('%B %d, %Y at %I:%M %p') }}
                                </small>
                                {% if comment.language == 'fa' %}
                                <span class="badge bg-info ms-2">■■■■■</span>
                                {% endif %}
                            </div>
                        </div>

                        <div class="comment-content mb-2" dir="{{ 'rtl' if comment.language == 'fa' else
                            {{ comment.content | nl2br }}
                        </div>

                        <div class="comment-actions d-flex align-items-center">
                            <button class="btn btn-sm btn-outline-success me-2 vote-btn"
                                    data-comment-id="{{ comment.id }}"
                                    data-vote-type="like">
                                <i class="fas fa-thumbs-up me-1"></i>
                                <span class="like-count">{{ comment.likes }}</span>
                            </button>
                            <button class="btn btn-sm btn-outline-danger vote-btn"
                                    data-comment-id="{{ comment.id }}"
                                    data-vote-type="dislike">
                                <i class="fas fa-thumbs-down me-1"></i>
                                <span class="dislike-count">{{ comment.dislikes }}</span>
                            </button>
                        </div>
                    </div>
                    {% endfor %}

                    <!-- Pagination -->
                    {% if comments.pages > 1 %}
                    <nav>
                        <ul class="pagination justify-content-center">
                            {% if comments.has_prev %}
                            <li class="page-item">
                                <a class="page-link bg-dark text-light border-secondary" href="{{ url_fo
                            </li>
                            {% endif %}

                            {% for page_num in comments.iter_pages() %}
                                {% if page_num %}
                                    {% if page_num != comments.page %}
                                    <li class="page-item">
                                        <a class="page-link bg-dark text-light border-secondary" href="{
                                    </li>
                                    {% else %}
                                    <li class="page-item active">
                                        <span class="page-link bg-primary border-primary">{{ page_num }}
                                    </li>
                                    {% endif %}
                                {% else %}
                                <li class="page-item disabled">
                                    <span class="page-link bg-dark text-light border-secondary">...</spa
                                </li>
                                {% endif %}
                            {% endfor %}

                            {% if comments.has_next %}
                            <li class="page-item">
                                <a class="page-link bg-dark text-light border-secondary" href="{{ url_fo
                            </li>
                            {% endif %}
                        </ul>
                    </nav>
                    {% endif %}
                {% else %}
                <div class="text-center py-4">
                    <i class="fas fa-comments fa-3x text-muted mb-3"></i>
                    <p class="text-muted">{{ get_text('no_comments') }}</p>
                </div>
                {% endif %}
            </div>

        </div>
    </div>

    <!-- Sidebar -->
    <div class="col-lg-4">
        <!-- Weekly Story -->
        {% if active_story %}
        <div class="card bg-secondary mb-4">
            <div class="card-header">
```

```
                <h4 class="card-title mb-0">
                    <i class="fas fa-book me-2"></i>{{ get_text('weekly_story') }}
                </h4>
            </div>
            <div class="card-body">
                <h5>{{ active_story.title_fa if current_language == 'fa' and active_story.title_fa else
                {% set author_name = active_story.author_fa if current_language == 'fa' and active_story
                {% if author_name %}
                <p class="text-muted mb-2">
                    <i class="fas fa-user me-1"></i>
                    {{ author_name }}
                </p>
                {% endif %}
                <div class="story-content">
                    {{ (active_story.content_fa if current_language == 'fa' and active_story.content_fa
                </div>
                <small class="text-muted">
                    <i class="fas fa-clock me-1"></i>
                    {{ active_story.created_at.strftime('%B %d, %Y') }}
                </small>
            </div>
        </div>
        {% endif %}

        <!-- Quick Links -->
        <div class="card bg-secondary">
            <div class="card-header">
                <h4 class="card-title mb-0">
                    <i class="fas fa-link me-2"></i>Quick Links
                </h4>
            </div>
            <div class="card-body">
                <div class="d-grid gap-2">
                    <a href="{{ url_for('archive') }}" class="btn btn-outline-primary">
                        <i class="fas fa-archive me-1"></i>{{ get_text('archive') }}
                    </a>
                    <a href="{{ url_for('info') }}" class="btn btn-outline-info">
                        <i class="fas fa-info-circle me-1"></i>{{ get_text('info') }}
                    </a>
                    <a href="{{ url_for('appeal') }}" class="btn btn-outline-warning">
                        <i class="fas fa-gavel me-1"></i>{{ get_text('appeal') }}

                    </a>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Archive Page Template

File: **templates/archive.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('archive') }} - Project Dard{% endblock %}

{% block content %}
<div class="row">
    <div class="col-12">
        <div class="d-flex justify-content-between align-items-center mb-4">
            <h1>
                <i class="fas fa-archive me-2"></i>{{ get_text('discussion_archive') }}
            </h1>
        </div>

        {% if topics.items %}
        <div class="row">
            {% for topic in topics.items %}
            <div class="col-md-6 mb-4">
                <div class="card bg-secondary h-100">
                    <div class="card-body">
                        <h5 class="card-title">
                            {{ topic.title_fa if current_language == 'fa' and topic.title_fa else topic.
                        </h5>
                        <div class="card-text mb-3">
                            {{ (topic.content_fa if current_language == 'fa' and topic.content_fa else t
                        </div>

                        <div class="d-flex justify-content-between align-items-center">
                            <small class="text-muted">
                                <i class="fas fa-clock me-1"></i>
                                {{ topic.created_at.strftime('%B %d, %Y') }}
                            </small>
                            <div>
                                <span class="badge bg-primary me-2">
                                    <i class="fas fa-comments me-1"></i>{{ topic.comments | length }}
                                </span>
                                {% if topic.is_active %}
                                <span class="badge bg-success">Active</span>
                                {% endif %}
                            </div>
                        </div>
                    </div>
                    <div class="card-footer bg-dark">
                        <div class="d-flex justify-content-between">
                            <a href="{{ url_for('home') }}#topic-{{ topic.id }}" class="btn btn-sm btn-o
                                <i class="fas fa-eye me-1"></i>View Discussion
                            </a>
                            <a href="{{ url_for('download_discussion', topic_id=topic.id) }}" class="btn
                                <i class="fas fa-download me-1"></i>{{ get_text('download_pdf') }}
                            </a>
                        </div>

                    </div>
                </div>
            </div>
            {% endfor %}
        </div>

        <!-- Pagination -->
        {% if topics.pages > 1 %}
        <nav>
            <ul class="pagination justify-content-center">
                {% if topics.has_prev %}
                <li class="page-item">
                    <a class="page-link bg-dark text-light border-secondary" href="{{ url_for('archive',
                </li>
                {% endif %}

                {% for page_num in topics.iter_pages() %}
                    {% if page_num %}
                        {% if page_num != topics.page %}
                        <li class="page-item">
                            <a class="page-link bg-dark text-light border-secondary" href="{{ url_for('a
                        </li>
                        {% else %}
                        <li class="page-item active">
                            <span class="page-link bg-primary border-primary">{{ page_num }}</span>
                        </li>
                        {% endif %}
```

```
                    {% else %}
                    <li class="page-item disabled">
                        <span class="page-link bg-dark text-light border-secondary">...</span>
                    </li>
                    {% endif %}
                {% endfor %}

                {% if topics.has_next %}
                <li class="page-item">
                    <a class="page-link bg-dark text-light border-secondary" href="{{ url_for('archive',
                </li>
                {% endif %}
            </ul>
        </nav>
        {% endif %}

        {% else %}
        <div class="text-center py-5">
            <i class="fas fa-archive fa-5x text-muted mb-4"></i>
            <h3 class="text-muted">{{ get_text('no_topics') }}</h3>
            <p class="text-muted">Check back later for archived discussions.</p>
        </div>
        {% endif %}

    </div>
</div>
{% endblock %}
```

Info Page Template

File: **templates/info.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('info') }} - Project Dard{% endblock %}

{% block content %}
<div class="row">
    <div class="col-12">
        <h1 class="mb-4">
            <i class="fas fa-info-circle me-2"></i>{{ get_text('about_platform') }}
        </h1>

        <!-- About Section -->
        <div class="card bg-secondary mb-4">
            <div class="card-body">
                <h3>{{ get_text('about_platform') }}</h3>
                <p class="lead">{{ get_text('platform_description') }}</p>
                <p>
                    Project Dard is designed to foster meaningful academic discussions and create a spac
                    intellectual exchange. Our platform supports multiple languages and encourages diver
                    perspectives on various topics.
                </p>
                <p>
                    We believe in the power of community-driven content moderation and democratic discou
                    Through our like/dislike system and appeal process, we maintain a balance between fr
                    expression and community standards.
                </p>
            </div>
        </div>

        <!-- Founder Section -->
        <div class="row mb-4">
            <div class="col-lg-6">
                <div class="card bg-secondary h-100">
                    <div class="card-header">
                        <h4 class="mb-0">
                            <i class="fas fa-user-tie me-2"></i>{{ get_text('founder') }}
                        </h4>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-md-4 text-center mb-3">
                                <div class="profile-image-placeholder bg-primary rounded-circle mx-auto"
                                    <i class="fas fa-user fa-3x text-white"></i>
                                </div>
                            </div>
                            <div class="col-md-8">
                                <h5>Dr. [Founder Name]</h5>
                                <p class="text-muted">Founder & Visionary</p>
                                <p>
                                    [Founder bio would go here. This section would contain information a

                                    founder's background, education, and vision for the platform.]
                                </p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Featured Author -->
            <div class="col-lg-6">
                <div class="card bg-secondary h-100">
                    <div class="card-header">
                        <h4 class="mb-0">
                            <i class="fas fa-star me-2"></i>{{ get_text('featured_author') }}
                        </h4>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-md-4 text-center mb-3">
                                <div class="profile-image-placeholder bg-warning rounded-circle mx-auto"
                                    <i class="fas fa-pen fa-3x text-white"></i>
                                </div>
                            </div>
                            <div class="col-md-8">
                                <h5>[Featured Author Name]</h5>
                                <p class="text-muted">This Week's Featured Author</p>
                                <p>
```

```
                                    [Featured author bio would go here. This section would highlight the
                                    week's featured author and their contributions.]
                                </p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Team Section -->
            <div class="card bg-secondary mb-4">
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-users me-2"></i>{{ get_text('team') }}
                    </h4>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col-md-4 text-center mb-3">
                            <div class="profile-image-placeholder bg-info rounded-circle mx-auto mb-2" style
                                <i class="fas fa-user fa-2x text-white"></i>
                            </div>

                            <h6>[Team Member 1]</h6>
                            <p class="text-muted small">Community Manager</p>
                        </div>
                        <div class="col-md-4 text-center mb-3">
                            <div class="profile-image-placeholder bg-success rounded-circle mx-auto mb-2" st
                                <i class="fas fa-user fa-2x text-white"></i>
                            </div>
                            <h6>[Team Member 2]</h6>
                            <p class="text-muted small">Content Moderator</p>
                        </div>
                        <div class="col-md-4 text-center mb-3">
                            <div class="profile-image-placeholder bg-danger rounded-circle mx-auto mb-2" sty
                                <i class="fas fa-user fa-2x text-white"></i>
                            </div>
                            <h6>[Team Member 3]</h6>
                            <p class="text-muted small">Technical Lead</p>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Platform Features -->
            <div class="card bg-secondary">
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-cogs me-2"></i>Platform Features
                    </h4>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col-md-6">
                            <h6><i class="fas fa-globe me-2"></i>Multi-language Support</h6>
                            <p class="small text-muted">Support for English and Persian languages with RTL t

                            <h6><i class="fas fa-comments me-2"></i>Community Discussions</h6>
                            <p class="small text-muted">Weekly topics and open discussions with community en

                            <h6><i class="fas fa-thumbs-up me-2"></i>Democratic Moderation</h6>
                            <p class="small text-muted">Community-driven content moderation through likes an
                        </div>
                        <div class="col-md-6">
                            <h6><i class="fas fa-archive me-2"></i>Discussion Archive</h6>
                            <p class="small text-muted">Complete archive of past discussions with PDF downlo

                            <h6><i class="fas fa-gavel me-2"></i>Appeal Process</h6>
                            <p class="small text-muted">Fair appeal system for content moderation decisions.

                            <h6><i class="fas fa-book me-2"></i>Weekly Stories</h6>
                            <p class="small text-muted">Featured stories and content from community members.
                        </div>

                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Appeal Page Template

File: **templates/appeal.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('appeal') }} - Project Dard{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-lg-8">
        <h1 class="mb-4">
            <i class="fas fa-gavel me-2"></i>{{ get_text('appeal_form') }}
        </h1>

        <!-- Appeal Information -->
        <div class="card bg-secondary mb-4">
            <div class="card-body">
                <h5>{{ get_text('appeal_description') }}</h5>
                <p>
                    If your comment was automatically hidden due to receiving 5 or more dislikes,
                    you can submit an appeal here. Our moderation team will review your appeal
                    and make a decision within 24-48 hours.
                </p>
                <div class="alert alert-info">
                    <i class="fas fa-info-circle me-2"></i>
                    <strong>Please note:</strong> Appeals should include a valid reason for why
                    your comment should be restored. Generic appeals may be rejected.
                </div>
            </div>
        </div>

        <!-- Appeal Form -->
        <div class="card bg-secondary">
            <div class="card-header">
                <h4 class="mb-0">
                    <i class="fas fa-edit me-2"></i>Submit Appeal
                </h4>
            </div>
            <div class="card-body">
                <form method="POST">
                    {{ form.hidden_tag() }}

                    <div class="mb-3">
                        {{ form.appellant_name.label(class="form-label") }}
                        {{ form.appellant_name(class="form-control bg-dark text-light border-secondary")
                        {% if form.appellant_name.errors %}
                            <div class="text-danger small">
                                {% for error in form.appellant_name.errors %}
                                    {{ error }}
                                {% endfor %}
                            </div>
                        {% endif %}
                    </div>

                    <div class="mb-3">
                        {{ form.appellant_email.label(class="form-label") }}
                        {{ form.appellant_email(class="form-control bg-dark text-light border-secondary"
                        {% if form.appellant_email.errors %}
                            <div class="text-danger small">
                                {% for error in form.appellant_email.errors %}
                                    {{ error }}
                                {% endfor %}
                            </div>
                        {% endif %}
                    </div>

                    <div class="mb-3">
                        <label for="comment_id" class="form-label">Comment ID (if known)</label>
                        <input type="number" id="comment_id" name="comment_id" class="form-control bg-da
                        <div class="form-text">
                            If you don't know the comment ID, please provide as much detail as possible
                        </div>
                    </div>

                    <div class="mb-3">
                        {{ form.reason.label(class="form-label") }}
                        {{ form.reason(class="form-control bg-dark text-light border-secondary", rows="6
                        {% if form.reason.errors %}
                            <div class="text-danger small">
                                {% for error in form.reason.errors %}
                                    {{ error }}
```

```
                                {% endfor %}
                            </div>
                        {% endif %}
                        <div class="form-text">
                            Please provide a detailed explanation of why your comment should be restored
                            Include the original comment content if possible.
                        </div>
                    </div>

                    <div class="d-grid">
                        {{ form.submit(class="btn btn-primary btn-lg") }}
                    </div>
                </form>
            </div>
        </div>

        <!-- Contact Information -->
        <div class="card bg-secondary mt-4">
            <div class="card-header">
                <h4 class="mb-0">
                    <i class="fas fa-phone me-2"></i>{{ get_text('contact_info') }}
                </h4>

            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6">
                        <h6>Email</h6>
                        <p><a href="mailto:contact@projectdard.com" class="text-light">contact@projectda

                        <h6>Response Time</h6>
                        <p class="text-muted">24-48 hours for appeal reviews</p>
                    </div>
                    <div class="col-md-6">
                        <h6>{{ get_text('social_media') }}</h6>
                        <div class="social-links">
                            <a href="#" class="btn btn-outline-primary btn-sm me-2">
                                <i class="fab fa-twitter"></i> Twitter
                            </a>
                            <a href="#" class="btn btn-outline-primary btn-sm me-2">
                                <i class="fab fa-facebook"></i> Facebook
                            </a>
                            <a href="#" class="btn btn-outline-primary btn-sm">
                                <i class="fab fa-instagram"></i> Instagram
                            </a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Admin Login Template

File: **templates/admin/login.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('admin') }} {{ get_text('login') }} - Project Dard{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6 col-lg-4">
        <div class="card bg-secondary">
            <div class="card-header text-center">
                <h3 class="mb-0">
                    <i class="fas fa-sign-in-alt me-2"></i>{{ get_text('admin') }} {{ get_text('login')
                </h3>
            </div>
            <div class="card-body">
                <form method="POST">
                    {{ form.hidden_tag() }}

                    <div class="mb-3">
                        {{ form.username.label(class="form-label") }}
                        {{ form.username(class="form-control bg-dark text-light border-secondary") }}
                        {% if form.username.errors %}
                            <div class="text-danger small">
                                {% for error in form.username.errors %}
                                    {{ error }}
                                {% endfor %}
                            </div>
                        {% endif %}
                    </div>

                    <div class="mb-3">
                        {{ form.password.label(class="form-label") }}
                        {{ form.password(class="form-control bg-dark text-light border-secondary") }}
                        {% if form.password.errors %}
                            <div class="text-danger small">
                                {% for error in form.password.errors %}
                                    {{ error }}
                                {% endfor %}
                            </div>
                        {% endif %}
                    </div>

                    <div class="mb-3 form-check">
                        {{ form.remember_me(class="form-check-input") }}
                        {{ form.remember_me.label(class="form-check-label") }}
                    </div>

                    <div class="d-grid">
                        {{ form.submit(class="btn btn-primary") }}
                    </div>
                </form>

            </div>
            <div class="card-footer text-center text-muted">
                <small>
                    <i class="fas fa-shield-alt me-1"></i>
                    Secure Admin Access
                </small>
            </div>
        </div>

        <!-- Default Credentials Notice (Remove in production) -->
        <div class="alert alert-warning mt-3">
            <i class="fas fa-exclamation-triangle me-2"></i>
            <strong>Development Mode:</strong> Default credentials are username: <code>admin</code>, pas
        </div>
    </div>
</div>
{% endblock %}
```

Admin Dashboard Template

File: **templates/admin/dashboard.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('dashboard') }} - Project Dard{% endblock %}

{% block content %}
<div class="row">
    <div class="col-12">
        <div class="d-flex justify-content-between align-items-center mb-4">
            <h1>
                <i class="fas fa-tachometer-alt me-2"></i>{{ get_text('dashboard') }}
            </h1>
            <div>
                <span class="text-muted">Welcome, {{ current_user.username }}!</span>
            </div>
        </div>

        <!-- Statistics Cards -->
        <div class="row mb-4">
            <div class="col-md-3">
                <div class="card bg-primary">
                    <div class="card-body text-center">
                        <i class="fas fa-comments fa-2x mb-2"></i>
                        <h3>{{ total_topics }}</h3>
                        <p class="mb-0">Total Topics</p>
                    </div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="card bg-success">
                    <div class="card-body text-center">
                        <i class="fas fa-comment fa-2x mb-2"></i>
                        <h3>{{ total_comments }}</h3>
                        <p class="mb-0">Total Comments</p>
                    </div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="card bg-warning">
                    <div class="card-body text-center">
                        <i class="fas fa-eye-slash fa-2x mb-2"></i>
                        <h3>{{ hidden_comments }}</h3>
                        <p class="mb-0">Hidden Comments</p>
                    </div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="card bg-danger">
                    <div class="card-body text-center">
                        <i class="fas fa-gavel fa-2x mb-2"></i>
                        <h3>{{ pending_appeals }}</h3>
                        <p class="mb-0">Pending Appeals</p>
                    </div>
                </div>
            </div>
        </div>

        <!-- Quick Actions -->
        <div class="row mb-4">
            <div class="col-md-6">
                <div class="card bg-secondary">
                    <div class="card-header">
                        <h4 class="mb-0">
                            <i class="fas fa-plus me-2"></i>Post New Topic
                        </h4>
                    </div>
                    <div class="card-body">
                        {% if topic_form %}
                        <form method="POST" action="{{ url_for('admin_post_topic') }}">
                            {{ topic_form.hidden_tag() }}

                            <div class="mb-3">
                                {{ topic_form.title_en.label(class="form-label") }}
                                {{ topic_form.title_en(class="form-control bg-dark text-light border-sec
                            </div>

                            <div class="mb-3">
                                {{ topic_form.title_fa.label(class="form-label") }}
```

```
                        {{ topic_form.title_fa(class="form-control bg-dark text-light border-sec
                    </div>

                    <div class="mb-3">
                        {{ topic_form.content_en.label(class="form-label") }}
                        {{ topic_form.content_en(class="form-control bg-dark text-light border-s
                    </div>

                    <div class="mb-3">
                        {{ topic_form.content_fa.label(class="form-label") }}
                        {{ topic_form.content_fa(class="form-control bg-dark text-light border-s
                    </div>

                    {{ topic_form.submit(class="btn btn-primary") }}
                </form>
                {% else %}
                <a href="{{ url_for('admin_post_topic') }}" class="btn btn-primary">
                    <i class="fas fa-plus me-1"></i>Create New Topic
                </a>
                {% endif %}
            </div>
        </div>
    </div>

    <div class="col-md-6">
        <div class="card bg-secondary">
            <div class="card-header">
                <h4 class="mb-0">
                    <i class="fas fa-book me-2"></i>Post New Story
                </h4>
            </div>
            <div class="card-body">
                {% if story_form %}
                <form method="POST" action="{{ url_for('admin_post_story') }}">
                    {{ story_form.hidden_tag() }}

                    <div class="mb-3">
                        {{ story_form.title_en.label(class="form-label") }}
                        {{ story_form.title_en(class="form-control bg-dark text-light border-sec
                    </div>

                    <div class="mb-3">
                        {{ story_form.title_fa.label(class="form-label") }}
                        {{ story_form.title_fa(class="form-control bg-dark text-light border-sec
                    </div>

                    <div class="row">
                        <div class="col-md-6 mb-3">
                            {{ story_form.author_en.label(class="form-label") }}
                            {{ story_form.author_en(class="form-control bg-dark text-light borde
                        </div>
                        <div class="col-md-6 mb-3">
                            {{ story_form.author_fa.label(class="form-label") }}
                            {{ story_form.author_fa(class="form-control bg-dark text-light borde
                        </div>
                    </div>

                    <div class="mb-3">
                        {{ story_form.content_en.label(class="form-label") }}
                        {{ story_form.content_en(class="form-control bg-dark text-light border-s
                    </div>

                    <div class="mb-3">
                        {{ story_form.content_fa.label(class="form-label") }}
                        {{ story_form.content_fa(class="form-control bg-dark text-light border-s
                    </div>

                    {{ story_form.submit(class="btn btn-success") }}
                </form>
                {% else %}
                <a href="{{ url_for('admin_post_story') }}" class="btn btn-success">
                    <i class="fas fa-plus me-1"></i>Create New Story
                </a>

                {% endif %}
            </div>
        </div>
    </div>
</div>

<!-- Current Content Management -->
<div class="row mb-4">
    <div class="col-md-6">
        <div class="card bg-secondary">
```

```
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-list me-2"></i>Current Topics
                    </h4>
                </div>
                <div class="card-body">
                    {% for topic in current_topics %}
                    <div class="d-flex justify-content-between align-items-start border-bottom borde
                        <div>
                            <strong>{{ topic.title_en }}</strong>
                            {% if topic.title_fa %}
                            <br><small class="text-muted">{{ topic.title_fa }}</small>
                            {% endif %}
                            <br><small class="text-muted">{{ topic.created_at.strftime('%m/%d/%Y') }
                            {% if topic.is_active %}
                            <span class="badge bg-success ms-2">Active</span>
                            {% endif %}
                        </div>
                        <div>
                            <form method="POST" action="{{ url_for('delete_topic', topic_id=topic.id
                                  onsubmit="return confirm('Are you sure you want to delete this top
                                  style="display: inline;">
                                <button type="submit" class="btn btn-outline-danger btn-sm">
                                    <i class="fas fa-trash"></i>
                                </button>
                            </form>
                        </div>
                    </div>
                    {% else %}
                    <p class="text-muted">No topics found.</p>
                    {% endfor %}
                </div>
            </div>
        </div>

        <div class="col-md-6">
            <div class="card bg-secondary">
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-book me-2"></i>Current Stories

                    </h4>
                </div>
                <div class="card-body">
                    {% for story in current_stories %}
                    <div class="d-flex justify-content-between align-items-start border-bottom borde
                        <div>
                            <strong>{{ story.title_en }}</strong>
                            {% if story.title_fa %}
                            <br><small class="text-muted">{{ story.title_fa }}</small>
                            {% endif %}
                            {% if story.author_en %}
                            <br><small class="text-muted">by {{ story.author_en }}</small>
                            {% endif %}
                            <br><small class="text-muted">{{ story.created_at.strftime('%m/%d/%Y') }
                            {% if story.is_active %}
                            <span class="badge bg-success ms-2">Active</span>
                            {% endif %}
                        </div>
                        <div>
                            <form method="POST" action="{{ url_for('delete_story', story_id=story.id
                                  onsubmit="return confirm('Are you sure you want to delete this sto
                                  style="display: inline;">
                                <button type="submit" class="btn btn-outline-danger btn-sm">
                                    <i class="fas fa-trash"></i>
                                </button>
                            </form>
                        </div>
                    </div>
                    {% else %}
                    <p class="text-muted">No stories found.</p>
                    {% endfor %}
                </div>
            </div>
        </div>
    </div>

    <!-- Management Links -->
    <div class="row mb-4">
        <div class="col-12">
            <div class="card bg-secondary">
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-cogs me-2"></i>Management
```

```
                        </h4>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <div class="col-md-3">
                                <a href="{{ url_for('admin_comments') }}" class="btn btn-outline-warning
                                    <i class="fas fa-comments me-1"></i>Manage Comments

                                </a>
                            </div>
                            <div class="col-md-3">
                                <a href="{{ url_for('admin_comments', hidden='true') }}" class="btn btn-
                                    <i class="fas fa-eye-slash me-1"></i>Hidden Comments
                                </a>
                            </div>
                            <div class="col-md-3">
                                <a href="{{ url_for('export_excel') }}" class="btn btn-outline-success w
                                    <i class="fas fa-file-excel me-1"></i>Export to Excel
                                </a>
                            </div>
                            <div class="col-md-3">
                                <a href="{{ url_for('archive') }}" class="btn btn-outline-info w-100 mb-
                                    <i class="fas fa-archive me-1"></i>View Archive
                                </a>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Recent Activity -->
            {% if recent_comments %}
            <div class="card bg-secondary">
                <div class="card-header">
                    <h4 class="mb-0">
                        <i class="fas fa-clock me-2"></i>{{ get_text('recent_activity') }}
                    </h4>
                </div>
                <div class="card-body">
                    {% for comment in recent_comments %}
                    <div class="d-flex justify-content-between align-items-start border-bottom border-second
                        <div>
                            <strong>{{ comment.author_name }}</strong>
                            <small class="text-muted ms-2">{{ comment.created_at.strftime('%m/%d/%Y %H:%M')
                            {% if comment.is_hidden %}
                            <span class="badge bg-danger ms-2">Hidden</span>
                            {% endif %}
                            <div class="text-truncate mt-1" style="max-width: 400px;">
                                {{ comment.content | truncate(100) }}
                            </div>
                        </div>
                        <div class="text-end">
                            <small class="text-muted">
                                <i class="fas fa-thumbs-up me-1"></i>{{ comment.likes }}
                                <i class="fas fa-thumbs-down ms-2 me-1"></i>{{ comment.dislikes }}
                            </small>
                        </div>

                    </div>
                    {% endfor %}
                </div>
            </div>
            {% endif %}
        </div>
    </div>
    {% endblock %}
```

Comment Management Template

File: **templates/admin/comments.html**

```
{% extends "base.html" %}

{% block title %}{{ get_text('manage_comments') }} - Project Dard{% endblock %}

{% block content %}
<div class="row">
    <div class="col-12">
        <div class="d-flex justify-content-between align-items-center mb-4">
            <h1>
                <i class="fas fa-comments me-2"></i>{{ get_text('manage_comments') }}
            </h1>
            <div>
                <a href="{{ url_for('admin_comments') }}" class="btn btn-outline-primary {% if not show_
                    All Comments
                </a>
                <a href="{{ url_for('admin_comments', hidden='true') }}" class="btn btn-outline-warning
                    Hidden Comments
                </a>
            </div>
        </div>

        <!-- Pending Appeals -->
        {% if appeals %}
        <div class="card bg-danger mb-4">
            <div class="card-header">
                <h4 class="mb-0">
                    <i class="fas fa-gavel me-2"></i>Pending Appeals ({{ appeals | length }})
                </h4>
            </div>
            <div class="card-body">
                {% for appeal in appeals %}
                <div class="border-bottom border-light pb-3 mb-3">
                    <div class="row">
                        <div class="col-md-8">
                            <h6>{{ appeal.appellant_name }} ({{ appeal.appellant_email }})</h6>
                            <p class="mb-1"><strong>Reason:</strong> {{ appeal.reason | truncate(200) }}
                            <small class="text-muted">
                                Submitted: {{ appeal.created_at.strftime('%B %d, %Y at %I:%M %p') }}
                            </small>
                        </div>
                        <div class="col-md-4 text-end">
                            <div class="btn-group">
                                <a href="{{ url_for('review_appeal', appeal_id=appeal.id, action='approv
                                   class="btn btn-sm btn-success"
                                   onclick="return confirm('Approve this appeal and unhide the comment?'
                                    <i class="fas fa-check me-1"></i>Approve
                                </a>
                                <a href="{{ url_for('review_appeal', appeal_id=appeal.id, action='reject
                                   class="btn btn-sm btn-danger"
                                   onclick="return confirm('Reject this appeal?')">
                                    <i class="fas fa-times me-1"></i>Reject
                                </a>
                            </div>
                        </div>
                    </div>
                </div>
                {% endfor %}
            </div>
        </div>
        {% endif %}

        <!-- Comments List -->
        {% if comments.items %}
        <div class="card bg-secondary">
            <div class="card-header">
                <h4 class="mb-0">
                    Comments ({{ comments.total }})
                </h4>
            </div>
            <div class="card-body">
                {% for comment in comments.items %}
                <div class="comment-admin-item border-bottom border-secondary pb-3 mb-3">
                    <div class="row">
                        <div class="col-md-8">
                            <div class="d-flex align-items-start mb-2">
                                <div>
                                    <strong>{{ comment.author_name }}</strong>
```

```
                                {% if comment.author_email %}
                                <small class="text-muted">({{ comment.author_email }})</small>
                                {% endif %}
                                <br>
                                <small class="text-muted">
                                    <i class="fas fa-clock me-1"></i>
                                    {{ comment.created_at.strftime('%B %d, %Y at %I:%M %p') }}
                                </small>
                                {% if comment.language == 'fa' %}
                                <span class="badge bg-info ms-2">■■■■■</span>
                                {% endif %}
                                {% if comment.is_hidden %}
                                <span class="badge bg-danger ms-2">Hidden</span>
                                {% endif %}
                            </div>
                        </div>

                        <div class="comment-content mb-2" dir="{{ 'rtl' if comment.language == 'fa'
                            {{ comment.content | nl2br }}
                        </div>

                        <div class="comment-stats">
                            <span class="badge bg-success me-2">

                                <i class="fas fa-thumbs-up me-1"></i>{{ comment.likes }}
                            </span>
                            <span class="badge bg-danger me-2">
                                <i class="fas fa-thumbs-down me-1"></i>{{ comment.dislikes }}
                            </span>
                            <small class="text-muted">
                                Topic: {{ comment.topic.title_en | truncate(50) }}
                            </small>
                        </div>
                    </div>

                    <div class="col-md-4 text-end">
                        <div class="btn-group-vertical">
                            <a href="{{ url_for('toggle_comment', comment_id=comment.id) }}"
                               class="btn btn-sm {% if comment.is_hidden %}btn-outline-success{% els
                               onclick="return confirm('{% if comment.is_hidden %}Unhide{% else %}Hi
                                <i class="fas fa-{% if comment.is_hidden %}eye{% else %}eye-slash{%
                                {% if comment.is_hidden %}Unhide{% else %}Hide{% endif %}
                            </a>
                            <a href="{{ url_for('delete_comment', comment_id=comment.id) }}"
                               class="btn btn-sm btn-outline-danger"
                               onclick="return confirm('Permanently delete this comment? This action
                                <i class="fas fa-trash me-1"></i>Delete
                            </a>
                        </div>
                    </div>
                </div>
            </div>
            {% endfor %}
        </div>
    </div>

    <!-- Pagination -->
    {% if comments.pages > 1 %}
    <nav class="mt-4">
        <ul class="pagination justify-content-center">
            {% if comments.has_prev %}
            <li class="page-item">
                <a class="page-link bg-dark text-light border-secondary"
                   href="{{ url_for('admin_comments', page=comments.prev_num, hidden=show_hidden|str
            </li>
            {% endif %}

            {% for page_num in comments.iter_pages() %}
                {% if page_num %}
                    {% if page_num != comments.page %}
                    <li class="page-item">
                        <a class="page-link bg-dark text-light border-secondary"
                           href="{{ url_for('admin_comments', page=page_num, hidden=show_hidden|stri
                    </li>

                    {% else %}
                    <li class="page-item active">
                        <span class="page-link bg-primary border-primary">{{ page_num }}</span>
                    </li>
                    {% endif %}
                {% else %}
                <li class="page-item disabled">
                    <span class="page-link bg-dark text-light border-secondary">...</span>
                </li>
```

```
                    {% endif %}
                {% endfor %}

                {% if comments.has_next %}
                <li class="page-item">
                    <a class="page-link bg-dark text-light border-secondary"
                       href="{{ url_for('admin_comments', page=comments.next_num, hidden=show_hidden|str
                </li>
                {% endif %}
            </ul>
        </nav>
        {% endif %}

        {% else %}
        <div class="text-center py-5">
            <i class="fas fa-comments fa-5x text-muted mb-4"></i>
            <h3 class="text-muted">No comments found</h3>
            <p class="text-muted">
                {% if show_hidden %}
                No hidden comments at this time.
                {% else %}
                No comments have been posted yet.
                {% endif %}
            </p>
        </div>
        {% endif %}
    </div>
</div>
{% endblock %}
```

CSS Styles

File: **static/css/style.css**

```
/* Custom CSS for Project Dard */

/* Root Variables */
:root {
    --primary-color: #007bff;
    --secondary-color: #6c757d;
    --dark-color: #212529;
    --light-color: #f8f9fa;
    --border-color: #495057;
}

/* Global Styles */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #121212;
    color: #ffffff !important;
    line-height: 1.6;
}

/* Persian Font Support */
body[dir="rtl"],
.fa-text,
[dir="rtl"] {
    font-family: 'Vazirmatn', 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

/* Dark Theme Overrides */
.bg-dark {
    background-color: #1a1a1a !important;
}

.bg-secondary {
    background-color: #2d3748 !important;
}

.text-light {
    color: #ffffff !important;
}

.border-secondary {
    border-color: #4a5568 !important;
}

/* Force white text for all content */
h1, h2, h3, h4, h5, h6, p, span, div, a, label, small, strong, li, td, th {
    color: #ffffff !important;
}

/* Links */
a {

    color: #ffffff !important;
    text-decoration: none;
}

a:hover {
    color: #007bff !important;
    text-decoration: underline;
}

/* Specific overrides for muted text */
.text-muted {
    color: #c7c7c7 !important;
}

/* Card text */
.card-title, .card-text, .card-body {
    color: #ffffff !important;
}

/* Footer text */
footer, footer * {
    color: #ffffff !important;
}

/* Social links */
.social-links a {
    color: #ffffff !important;
```

```
}

.social-links a:hover {
    color: #ffffff !important;
}

/* Navigation */
.navbar {
    border-bottom: 1px solid var(--border-color);
    backdrop-filter: blur(10px);
}

.navbar-brand {
    font-weight: 700;
    font-size: 1.5rem;
    color: #ffffff !important;
}

.navbar-nav .nav-link {
    color: #ffffff !important;
}

.navbar-nav .nav-link:hover {
    color: #007bff !important;
}

.dropdown-menu {
    background-color: #2d3748 !important;
}

.dropdown-item {
    color: #ffffff !important;
}

.dropdown-item:hover {
    background-color: #007bff !important;
    color: #ffffff !important;
}

/* Cards */
.card {
    border: 1px solid var(--border-color);
    border-radius: 12px;
    transition: transform 0.2s ease-in-out, box-shadow 0.2s ease-in-out;
}

.card:hover {
    transform: translateY(-2px);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.3);
}

.card-header {
    border-bottom: 1px solid var(--border-color);
    background-color: rgba(255, 255, 255, 0.05);
}

/* Forms */
.form-control, .form-select {
    background-color: #2d3748 !important;
    border-color: #4a5568 !important;
    color: #ffffff !important;
}

.form-control:focus,
.form-select:focus {
    background-color: #2d3748 !important;
    border-color: var(--primary-color) !important;
    color: #ffffff !important;
    box-shadow: 0 0 0 0.2rem rgba(0, 123, 255, 0.25);
}

.form-control::placeholder {
    color: #a0aec0 !important;
}

.form-label {
    color: #ffffff !important;
}

.form-text {
    color: #c7c7c7 !important;
}
```

```css
/* Buttons */
.btn {
    border-radius: 8px;
    font-weight: 500;
    transition: all 0.2s ease-in-out;
    color: #ffffff !important;
}

.btn:hover {
    transform: translateY(-1px);
}

.btn-outline-primary,
.btn-outline-success,
.btn-outline-warning,
.btn-outline-danger,
.btn-outline-info {
    color: #ffffff !important;
    border-color: currentColor !important;
}

.btn-outline-primary:hover,
.btn-outline-success:hover,
.btn-outline-warning:hover,
.btn-outline-danger:hover,
.btn-outline-info:hover {
    color: #ffffff !important;
}

/* Specific button styles */
.btn-primary {
    background-color: #007bff !important;
    border-color: #007bff !important;
    color: #ffffff !important;
}

.btn-success {
    background-color: #28a745 !important;
    border-color: #28a745 !important;
    color: #ffffff !important;

}

.btn-warning {
    background-color: #ffc107 !important;
    border-color: #ffc107 !important;
    color: #000000 !important;
}

.btn-danger {
    background-color: #dc3545 !important;
    border-color: #dc3545 !important;
    color: #ffffff !important;
}

/* Comments */
.comment-item {
    transition: background-color 0.2s ease-in-out;
    border-radius: 8px;
    padding: 1rem;
    margin-bottom: 1rem;
}

.comment-item:hover {
    background-color: rgba(255, 255, 255, 0.05);
}

.comment-content {
    background-color: rgba(255, 255, 255, 0.05);
    padding: 1rem;
    border-radius: 8px;
    border-left: 4px solid var(--primary-color);
    margin: 0.5rem 0;
}

.comment-content[dir="rtl"] {
    border-left: none;
    border-right: 4px solid var(--primary-color);
}

.vote-btn {
    min-width: 70px;
    transition: all 0.2s ease-in-out;
}
```

```
.vote-btn:hover {
    transform: scale(1.05);
}

/* RTL Support */
[dir="rtl"] {
    text-align: right;
}

[dir="rtl"] .navbar-nav {
    flex-direction: row-reverse;
}

[dir="rtl"] .btn-group {
    flex-direction: row-reverse;
}

[dir="rtl"] .pagination {
    flex-direction: row-reverse;
}

/* Language Switcher */
.language-switcher {
    border: 1px solid var(--border-color);
    border-radius: 20px;
    padding: 0.25rem;
}

.language-switcher .btn {
    border-radius: 16px;
    padding: 0.375rem 1rem;
    font-size: 0.875rem;
}

/* Profile Images */
.profile-image-placeholder {
    border: 3px solid rgba(255, 255, 255, 0.1);
    transition: transform 0.2s ease-in-out;
}

.profile-image-placeholder:hover {
    transform: scale(1.05);
}

/* Statistics Cards */
.card.bg-primary,
.card.bg-success,
.card.bg-warning,
.card.bg-danger,
.card.bg-info {
    border: none;
}

.card.bg-primary:hover,
.card.bg-success:hover,
.card.bg-warning:hover,
.card.bg-danger:hover,
.card.bg-info:hover {
    transform: translateY(-4px);
    box-shadow: 0 12px 30px rgba(0, 0, 0, 0.4);
}

/* Alerts */
.alert {
    border-radius: 8px;
    border: none;
    color: #ffffff !important;
}

.alert-info {
    background-color: rgba(23, 162, 184, 0.3);
    color: #ffffff !important;
    border-left: 4px solid #17a2b8;
}

.alert-success {
    background-color: rgba(40, 167, 69, 0.3);
    color: #ffffff !important;
    border-left: 4px solid #28a745;
}

.alert-danger {
```

```
    background-color: rgba(220, 53, 69, 0.3);
    color: #ffffff !important;
    border-left: 4px solid #dc3545;
}

.alert-warning {
    background-color: rgba(255, 193, 7, 0.3);
    color: #000000 !important;
    border-left: 4px solid #ffc107;
}

/* Badges */
.badge {
    color: #ffffff !important;
}

.badge.bg-primary {
    background-color: #007bff !important;
}

.badge.bg-success {
    background-color: #28a745 !important;
}

.badge.bg-warning {
    background-color: #ffc107 !important;
    color: #000000 !important;
}

.badge.bg-danger {
    background-color: #dc3545 !important;
}

.badge.bg-info {
    background-color: #17a2b8 !important;
}

/* Pagination */
.pagination .page-link {
    background-color: #2d3748 !important;
    border-color: var(--border-color) !important;
    color: #ffffff !important;
}

.pagination .page-link:hover {
    background-color: var(--primary-color) !important;
    border-color: var(--primary-color) !important;
    color: #ffffff !important;
}

.pagination .page-item.active .page-link {
    background-color: var(--primary-color) !important;
    border-color: var(--primary-color) !important;
    color: #ffffff !important;
}

/* Footer */
footer {
    border-top: 1px solid var(--border-color);
    margin-top: auto;
}

.social-links a {
    display: inline-block;
    width: 40px;
    height: 40px;
    line-height: 40px;
    text-align: center;
    border-radius: 50%;
    background-color: rgba(255, 255, 255, 0.1);
    transition: all 0.2s ease-in-out;
}

.social-links a:hover {
    background-color: var(--primary-color);
    transform: translateY(-2px);
    text-decoration: none;
    color: #ffffff;
}

/* Admin Dashboard */
.comment-admin-item {
    background-color: rgba(255, 255, 255, 0.02);
```

```
    border-radius: 8px;
    padding: 1rem;
    margin-bottom: 1rem;
    transition: background-color 0.2s ease-in-out;
}

.comment-admin-item:hover {
    background-color: rgba(255, 255, 255, 0.05);
}

/* Responsive Design */
@media (max-width: 768px) {
    .navbar-brand {
        font-size: 1.2rem;
    }

    .card {
        margin-bottom: 1rem;
    }

    .btn-group-vertical {
        width: 100%;
    }

    .comment-actions {
        flex-wrap: wrap;
        gap: 0.5rem;
    }

    .vote-btn {
        flex: 1;
        min-width: auto;
    }
}

@media (max-width: 576px) {
    .container {
        padding-left: 1rem;
        padding-right: 1rem;
    }

    .card-body {
        padding: 1rem;
    }

    .pagination .page-link {
        padding: 0.375rem 0.75rem;
        font-size: 0.875rem;
    }
}

/* Animations */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

.fade-in {
    animation: fadeIn 0.5s ease-in-out;
}

/* Loading States */
.loading {
    position: relative;
    pointer-events: none;
}

.loading::after {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    border-radius: inherit;
    display: flex;
    align-items: center;
```

```
    justify-content: center;
}

/* Scroll Behavior */
html {
    scroll-behavior: smooth;
}

/* Custom Scrollbar */
::-webkit-scrollbar {
    width: 8px;
}

::-webkit-scrollbar-track {
    background: #2d3748;
}

::-webkit-scrollbar-thumb {
    background: var(--primary-color);
    border-radius: 4px;
}

::-webkit-scrollbar-thumb:hover {
    background: #0056b3;
}

/* Print Styles */
@media print {
    .navbar,
    .btn,
    .pagination,
    footer {
        display: none !important;
    }

    .container {
        width: 100% !important;
        max-width: none !important;
    }

    .card {
        border: 1px solid #000 !important;
        box-shadow: none !important;
    }
}
```

JavaScript Code

File: **static/js/main.js**

```
// Main JavaScript for Project Dard

// DOM Content Loaded
document.addEventListener('DOMContentLoaded', function() {
    initializeVoting();
    initializeFormValidation();
    initializeTooltips();
    initializeLanguageSwitcher();
    initializeCharacterCount();
    initializeAutoResize();
});

// Initialize voting functionality
function initializeVoting() {
    const voteButtons = document.querySelectorAll('.vote-btn');

    voteButtons.forEach(button => {
        button.addEventListener('click', function(e) {
            e.preventDefault();

            const commentId = this.dataset.commentId;
            const voteType = this.dataset.voteType;

            // Disable button temporarily
            this.disabled = true;
            this.classList.add('loading');

            // Make AJAX request
            fetch(`/vote_comment/${commentId}/${voteType}`)
                .then(response => response.json())
                .then(data => {
                    if (data.error) {
                        showAlert('Error: ' + data.error, 'danger');
                        return;
                    }

                    // Update vote counts
                    const commentItem = document.querySelector(`[data-comment-id="${commentId}"]`);
                    const likeCountSpan = commentItem.querySelector('.like-count');
                    const dislikeCountSpan = commentItem.querySelector('.dislike-count');

                    if (likeCountSpan) likeCountSpan.textContent = data.likes;
                    if (dislikeCountSpan) dislikeCountSpan.textContent = data.dislikes;

                    // Handle hidden comment
                    if (data.hidden) {
                        commentItem.style.opacity = '0.5';
                        commentItem.innerHTML += '<div class="alert alert-warning mt-2"><i class="fas fa
                    }

                    // Visual feedback
                    this.classList.add('animate__animated', 'animate__pulse');
                    setTimeout(() => {
                        this.classList.remove('animate__animated', 'animate__pulse');
                    }, 1000);
                })
                .catch(error => {
                    console.error('Error:', error);
                    showAlert('An error occurred while voting.', 'danger');
                })
                .finally(() => {
                    // Re-enable button
                    this.disabled = false;
                    this.classList.remove('loading');
                });
        });
    });
}

// Initialize form validation
function initializeFormValidation() {
    const forms = document.querySelectorAll('form');

    forms.forEach(form => {
        form.addEventListener('submit', function(e) {
            if (!this.checkValidity()) {
                e.preventDefault();
                e.stopPropagation();
```

```
            }

            this.classList.add('was-validated');
        });
    });
}

// Initialize tooltips
function initializeTooltips() {
    const tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'));
    tooltipTriggerList.map(function(tooltipTriggerEl) {
        return new bootstrap.Tooltip(tooltipTriggerEl);
    });
}

// Initialize language switcher
function initializeLanguageSwitcher() {
    const languageLinks = document.querySelectorAll('a[href*="/set_language/"]');

    languageLinks.forEach(link => {
        link.addEventListener('click', function(e) {
            // Show loading state

            const currentLang = document.querySelector('.navbar-nav .dropdown-toggle');
            if (currentLang) {
                currentLang.innerHTML = '<i class="fas fa-spinner fa-spin me-1"></i>Loading...';
            }
        });
    });
}

// Initialize character count for textareas
function initializeCharacterCount() {
    const textareas = document.querySelectorAll('textarea[maxlength]');

    textareas.forEach(textarea => {
        const maxLength = parseInt(textarea.getAttribute('maxlength'));

        // Create counter element
        const counter = document.createElement('div');
        counter.className = 'form-text text-end';
        counter.innerHTML = `<span class="char-count">0</span>/${maxLength} characters`;

        // Insert after textarea
        textarea.parentNode.insertBefore(counter, textarea.nextSibling);

        // Update counter on input
        textarea.addEventListener('input', function() {
            const currentLength = this.value.length;
            const charCountSpan = counter.querySelector('.char-count');

            charCountSpan.textContent = currentLength;

            // Change color based on limit
            if (currentLength > maxLength * 0.9) {
                charCountSpan.style.color = '#dc3545';
            } else if (currentLength > maxLength * 0.8) {
                charCountSpan.style.color = '#ffc107';
            } else {
                charCountSpan.style.color = '#28a745';
            }
        });

        // Trigger initial update
        textarea.dispatchEvent(new Event('input'));
    });
}

// Initialize auto-resize for textareas
function initializeAutoResize() {
    const textareas = document.querySelectorAll('textarea');

    textareas.forEach(textarea => {

        // Set initial height
        textarea.style.height = 'auto';
        textarea.style.height = textarea.scrollHeight + 'px';

        // Auto-resize on input
        textarea.addEventListener('input', function() {
            this.style.height = 'auto';
            this.style.height = this.scrollHeight + 'px';
        });
    });
```

```
}

// Show alert message
function showAlert(message, type = 'info') {
    const alertContainer = document.querySelector('.container.mt-5.pt-4');

    const alert = document.createElement('div');
    alert.className = `alert alert-${type} alert-dismissible fade show`;
    alert.setAttribute('role', 'alert');
    alert.innerHTML = `
        ${message}
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
    `;

    // Insert at the beginning of the container
    alertContainer.insertBefore(alert, alertContainer.firstChild);

    // Auto-remove after 5 seconds
    setTimeout(() => {
        if (alert.parentNode) {
            alert.remove();
        }
    }, 5000);
}

// Smooth scrolling for anchor links
document.querySelectorAll('a[href^="#"]').forEach(anchor => {
    anchor.addEventListener('click', function(e) {
        e.preventDefault();

        const target = document.querySelector(this.getAttribute('href'));
        if (target) {
            target.scrollIntoView({
                behavior: 'smooth',
                block: 'start'
            });
        }
    });
});

// Handle form submissions with loading states
document.querySelectorAll('form').forEach(form => {
    form.addEventListener('submit', function() {
        const submitBtn = this.querySelector('button[type="submit"]');
        if (submitBtn) {
            submitBtn.disabled = true;
            submitBtn.innerHTML = '<i class="fas fa-spinner fa-spin me-1"></i>Please wait...';
        }
    });
});

// Handle back button
window.addEventListener('popstate', function(event) {
    // Reload page on back button to ensure fresh content
    location.reload();
});

// Add fade-in animation to cards
const observerOptions = {
    threshold: 0.1,
    rootMargin: '0px 0px -50px 0px'
};

const observer = new IntersectionObserver(function(entries) {
    entries.forEach(entry => {
        if (entry.isIntersecting) {
            entry.target.classList.add('fade-in');
            observer.unobserve(entry.target);
        }
    });
}, observerOptions);

// Observe all cards
document.querySelectorAll('.card').forEach(card => {
    observer.observe(card);
});

// Handle language direction changes
function updateTextDirection() {
    const html = document.documentElement;
    const currentLang = html.getAttribute('lang');

    if (currentLang === 'fa') {
        html.setAttribute('dir', 'rtl');
```

```
        document.body.classList.add('rtl');
    } else {
        html.setAttribute('dir', 'ltr');
        document.body.classList.remove('rtl');
    }
}

// Call on page load
updateTextDirection();

// Keyboard shortcuts
document.addEventListener('keydown', function(e) {
    // Ctrl/Cmd + / for search (if implemented)
    if ((e.ctrlKey || e.metaKey) && e.key === '/') {
        e.preventDefault();
        const searchInput = document.querySelector('input[type="search"]');
        if (searchInput) {
            searchInput.focus();
        }
    }

    // Escape to close modals/dropdowns
    if (e.key === 'Escape') {
        // Close any open dropdowns
        document.querySelectorAll('.dropdown-menu.show').forEach(menu => {
            const dropdown = bootstrap.Dropdown.getInstance(menu.previousElementSibling);
            if (dropdown) {
                dropdown.hide();
            }
        });
    }
});

// Print functionality
function printPage() {
    window.print();
}

// Export functionality (if needed)
function exportData(format) {
    // This would be implemented based on specific requirements
    console.log(`Exporting data in ${format} format`);
}

// Theme toggle (if dark/light mode toggle is added)
function toggleTheme() {
    document.body.classList.toggle('light-theme');

    // Save preference
    localStorage.setItem('theme',
        document.body.classList.contains('light-theme') ? 'light' : 'dark'
    );
}

// Load saved theme preference
const savedTheme = localStorage.getItem('theme');

if (savedTheme === 'light') {
    document.body.classList.add('light-theme');
}

// Utility functions
const utils = {
    // Format date
    formatDate: function(date, language = 'en') {
        const options = {
            year: 'numeric',
            month: 'long',
            day: 'numeric',
            hour: '2-digit',
            minute: '2-digit'
        };

        return new Date(date).toLocaleDateString(language === 'fa' ? 'fa-IR' : 'en-US', options);
    },

    // Truncate text
    truncateText: function(text, maxLength) {
        if (text.length <= maxLength) return text;
        return text.substr(0, maxLength) + '...';
    },

    // Escape HTML
    escapeHtml: function(text) {
```

```
        const div = document.createElement('div');
        div.textContent = text;
        return div.innerHTML;
    },

    // Get URL parameter
    getUrlParameter: function(name) {
        const urlParams = new URLSearchParams(window.location.search);
        return urlParams.get(name);
    }
};

// Make utils available globally
window.DardUtils = utils;

// Console log for debugging (remove in production)
console.log('Project Dard JavaScript loaded successfully');
```

Project Documentation

File: **replit.md**

```
# Project Dard - Replit Configuration

## Overview

Project Dard is a bilingual community discussion platform built with Flask that facilitates academic dis

## System Architecture

### Backend Architecture
- **Framework**: Flask (Python web framework)
- **Database ORM**: SQLAlchemy with Flask-SQLAlchemy
- **Authentication**: Flask-Login for session management
- **Form Handling**: Flask-WTF with WTForms for form validation
- **Security**: Werkzeug for password hashing and proxy handling

### Frontend Architecture
- **Template Engine**: Jinja2 (Flask's default)
- **CSS Framework**: Bootstrap 5 for responsive design
- **Icons**: Font Awesome for UI icons
- **Typography**: Google Fonts (Vazirmatn for Persian text)
- **JavaScript**: Vanilla JavaScript for interactive features

### Database Design
- **Primary Database**: SQLite (development) with PostgreSQL support configured
- **Connection Pooling**: Configured with pool recycling and pre-ping
- **Models**: Admin, WeeklyTopic, WeeklyStory, Comment, CommentVote, Appeal, Visitor

## Key Components

### Authentication System
- Admin-only login system using Flask-Login
- Session-based authentication with remember me functionality
- Password hashing using Werkzeug security utilities

### Content Management
- **Weekly Topics**: Bilingual discussion topics posted by admins
- **Weekly Stories**: Featured stories with author attribution
- **Comments**: User-generated content with voting system
- **Moderation**: Community-driven with like/dislike voting and appeals

### Internationalization
- Built-in translation system supporting English and Persian
- RTL text direction support for Persian content
- Language switching functionality with session persistence

### Admin Dashboard
- Content management for topics and stories
- Comment moderation and appeal review system
- Visitor analytics and statistics tracking
- PDF export functionality for discussions

## Data Flow

1. **User Visit**: Visitor tracking captures IP, user agent, and page data
2. **Content Display**: Language preference determines content display (EN/FA)
3. **User Interaction**: Comments submitted through validated forms
4. **Community Moderation**: Like/dislike system automatically hides heavily disliked comments
5. **Appeal Process**: Users can appeal hidden comments through formal process
6. **Admin Review**: Admins review appeals and manage content through dashboard

## External Dependencies

### Python Packages
- Flask ecosystem (Flask, Flask-SQLAlchemy, Flask-Login, Flask-WTF)
- Database: SQLAlchemy, psycopg2 (for PostgreSQL)
- PDF Generation: ReportLab
- Security: Werkzeug
- Forms: WTForms with validation

### Frontend Libraries
- Bootstrap 5.3.0 (via CDN)
- Font Awesome 6.0.0 (via CDN)
- Google Fonts API (Vazirmatn for Persian)

### Environment Variables
- `DATABASE_URL`: Database connection string
- `SESSION_SECRET`: Flask session secret key
## Deployment Strategy
```

Development Setup
- SQLite database for local development
- Debug mode enabled with hot reloading
- Logging configured for debugging

Production Considerations
- PostgreSQL database support configured
- ProxyFix middleware for reverse proxy deployment
- Session secret from environment variables
- Connection pooling for database efficiency

Database Migration
- SQLAlchemy model-based table creation
- Automatic table creation on app startup
- Default admin user creation process

Changelog
- July 01, 2025. Initial setup

User Preferences

Preferred communication style: Simple, everyday language.

Database Schema

The application uses the following database models: 1. Admin - Administrative users - id: Primary key - username: Unique username - email: Unique email address - password_hash: Hashed password - created_at: Account creation timestamp 2. WeeklyTopic - Discussion topics - id: Primary key - title_en/title_fa: Bilingual titles - content_en/content_fa: Bilingual content - created_at: Creation timestamp - is_active: Active status - admin_id: Foreign key to Admin 3. WeeklyStory - Featured stories - id: Primary key - title_en/title_fa: Bilingual titles - content_en/content_fa: Bilingual content - author_en/author_fa: Bilingual author names - created_at: Creation timestamp - is_active: Active status - admin_id: Foreign key to Admin 4. Comment - User comments - id: Primary key - content: Comment text - language: Comment language ('en' or 'fa') - author_name: Commenter name - author_email: Commenter email (optional) - created_at: Creation timestamp - is_hidden: Hidden status - likes/dislikes: Vote counts - topic_id: Foreign key to WeeklyTopic 5. CommentVote - Vote tracking - id: Primary key - comment_id: Foreign key to Comment - ip_address: Voter IP (to prevent duplicate votes) - vote_type: 'like' or 'dislike' - created_at: Vote timestamp 6. Appeal - Comment appeals - id: Primary key - comment_id: Foreign key to Comment - appellant_name: Appellant name - appellant_email: Appellant email - reason: Appeal reason - status: 'pending', 'approved', or 'rejected' - created_at: Appeal timestamp - reviewed_at: Review timestamp - admin_id: Foreign key to reviewing Admin 7. Visitor - Analytics - id: Primary key - ip_address: Visitor IP - user_agent: Browser information - page_visited: Visited page - visited_at: Visit timestamp

Installation and Setup

1. Prerequisites: - Python 3.11 or higher - pip package manager 2. Install Dependencies: pip install flask flask-sqlalchemy flask-login flask-wtf wtforms reportlab psycopg2-binary gunicorn 3. Environment Variables: - SESSION_SECRET: Flask session secret key - DATABASE_URL: Database connection string (optional, defaults to SQLite) 4. Run Application: python main.py Or using Gunicorn: gunicorn --bind 0.0.0.0:5000 --reuse-port --reload main:app 5. Default Admin Credentials: - Username: admin - Password: admin123 (Change these in production!) 6. Access: - Main site: http://localhost:5000 - Admin dashboard: http://localhost:5000/admin

Features and Usage

Home Page Features: - View current weekly topic and story - Post comments in English or Persian - Vote on comments (like/dislike) - Language switching (EN/FA) - Automatic comment hiding (5+ dislikes) Archive Page: - Browse all past discussions - Download discussions as PDF - Pagination for large datasets Info Page: - Platform information - Founder and team bios - Featured author spotlight - Platform features overview Appeal Page: - Submit appeals for hidden comments - Contact information - Social media links Admin Dashboard: - Post weekly topics and stories - Manage comments (hide/unhide/delete) - Review and process appeals - View visitor statistics - Content moderation tools Internationalization: - Full English and Persian support - RTL text direction for Persian - Bilingual content management - Language-specific date formatting