Social Media App - Full Django Project Guide

App Description

This Django-based Social Media App replicates core features of Instagram, including user authentication (with optional Google Sign-In), profile management, posts with media, likes, comments, messaging, and post sharing with mutual friends.

Core Features

1. User authentication: login, logout, signup, password change and reset.
2. Optional Google Sign-In (OAuth2.0) via REST API.
3. Profile page with editable user info and profile picture.
4. Follow/unfollow functionality with counts.
5. Post creation with multiple images/videos.
6. Home feed showing posts from followers and public.
7. Like and comment system.
8. Messaging between mutual followers with 3-minute edit/delete window.
9. Share posts with mutual friends.
10. Complete user search and friend search functionality.

Django Models

The following are the core models used in this project:

```
from django.db import models
from django.contrib.auth.models import User
from datetime import timedelta
from django.utils import timezone

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(blank=True)
    profile_pic = models.ImageField(upload_to='profile_pics/', blank=True, null=True)
    date_of_birth = models.DateField(blank=True, null=True)
```

```
class Post(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    caption = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

class PostMedia(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='media')
    media_file = models.FileField(upload_to='post_media/')

class Follow(models.Model):
    follower = models.ForeignKey(User, related_name='following_set', on_delete=models.CASCADE)
    following = models.ForeignKey(User, related_name='follower_set', on_delete=models.CASCADE)
    class Meta:
        unique_together = ('follower', 'following')

class Like(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    class Meta:
        unique_together = ('user', 'post')

class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

class Message(models.Model):
    sender = models.ForeignKey(User, related_name='sent_messages', on_delete=models.CASCADE)
    receiver = models.ForeignKey(User, related_name='received_messages', on_delete=models.CASCADE)
    message_text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    is_edited = models.BooleanField(default=False)
    is_deleted = models.BooleanField(default=False)

    def can_edit_or_delete(self):
        return timezone.now() <= self.created_at + timedelta(minutes=3)

class SharedPost(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
```

```
    sender = models.ForeignKey(User, related_name='shared_posts_sent', on_delete=models.CASCADE)
                receiver    =    models.ForeignKey(User,    related_name='shared_posts_received',
on_delete=models.CASCADE)
    shared_at = models.DateTimeField(auto_now_add=True)
```

Google Sign-In (Optional)

1. Visit https://console.cloud.google.com/
2. Create a new project.
3. Enable "Google+ API" or "People API".
4. Go to OAuth consent screen and fill the details.
5. Go to Credentials > Create OAuth 2.0 Client ID (type: Web Application).
6. Add Authorized redirect URIs (for frontend/backend as needed).
7. Use the client ID to implement sign-in in frontend.
8. Send token to backend for verification via Google OAuth2 API.

Verify Google Token in Django Backend

```
from google.oauth2 import id_token
from google.auth.transport import requests

def verify_google_token(token):
    try:
        idinfo = id_token.verify_oauth2_token(token, requests.Request(), 'YOUR_GOOGLE_CLIENT_ID')
        if idinfo['iss'] not in ['accounts.google.com', 'https://accounts.google.com']:
            raise ValueError('Wrong issuer.')
        return idinfo  # Contains email, name, etc.
    except ValueError:
        return None
```