10 – Email and Password Reset

အခု Password ကို reset လုပ်ပါမယ်။ အဲဒီအတွက် flask package တစ်ခုဖြစ်တဲ့ itsdangerous နဲ့၊ comfirmation token ကို generate လုပ်ပြီး၊ account ကို confirm လုပ်၊ နောက် password ကို reset လုပ်ပါမယ်။ အရင်ဆုံး python shell မှာ token ကို generate လုပ်ကြည့်ရအောင်။ itsdangerous package က token generators အမျိုးမျိုးကို ထောက်ပံ့ထားပါတယ်။ အဲဒီ အမျိုးအစားတွေထဲကမှ time expire နဲ့ JSON Web Signatures (JWSs) ကို generate လုပ်ဖို့ TimeJSONWebSignatureSerializer ကို သုံးပါမယ်။

```
>>> from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
>>> s = Serializer('secret', 30)


>>> token = s.dumps({'user_id': 1}).decode('utf–8')
>>> s.loads(token)


# Signature expired
>>> s.loads(token)
```

class constructor (Serializer) မှာ encryption key (secret) နဲ့ expire time (30 s) ကို argument ထည့်ပေးထားပါမယ်။ dumps() method နဲ့ token string ကို generate လုပ်ပြီး၊ decode လုပ်ပါ ။ token အတွက် expire time က စက္ကန့် (30) ပေးထားပါတယ်။ decode လုပ်ပြီး token ကို ပြန်ယူဖို့ loads() method ကို သုံးရပါမယ်။ expire ဖြစ်ပြီး token အတွက်ဆိုရင်တော့ exception raise ဖြစ်ပါမယ်။

ဒါဆိုရင် flask မှာ token generation နဲ့ verification လုပ်ဖို့ model.py မှာ code သွားရေးပါမယ်။ လိုအပ်တဲ့ package ကို import လုပ်ပါ။

```
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from flaskblog import db, login_manager, app
```

နောက် User model မှာ token generation အတွက် get_reset_token() method နဲ့ verification အတွက် verify_reset_token() method နှစ်ခုကို ရေးပါမယ်။ verify_reset_token() က static method ပါ ။

```
    def get_reset_token(self, expires_sec=1800):
        s = Serializer(app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(app.config['SECRET_KEY'])
        try:
            user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)
```

Token ဖန်တီးပြီးရင် form.py မှာ request လုပ်ဖို့ form နဲ့ password reset လုပ်ဖို့ form တွေ ရေးပါမယ်။

```
class RequestResetForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that email. You must register first.')
```

```
class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Reset Password')
```

routes.py မှာ "/reset_password" route ကို ရေးပါမယ်။ url ကို request လုပ်လိုက်တာနဲ့ authenticate user ဟုတ်မဟုတ်စစ်ပါမယ်။ authenticate မဖြစ်ရင် ပထမ register လုပ်ရပါမယ်။ current_user ဆိုရင်တော့ RequestResetForm ကို ချပေးပါမယ်။ အဲဒီမှာ email ရိုက်ထည့်ပြီး submit လုပ်လိုက်ရင် အဲဒီ mail နဲ့ user ကို ဆွဲထုတ်ပြီး send_reset_email() method နဲ့ mail ပို့ပါမယ်။ ပြီးရင် login form ကို ပြပေးပါမယ်။

```
@app.route("/reset_password", methods=['GET', 'POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your password.', 'info')
        return redirect(url_for('login'))
    return render_template('reset_request.html', title='Reset Password', form=form)
```

RequestResetForm ကိုချပေးဖို့ reset_request.html template file ကို ရေးပါမယ်။ code ကို file မှာ ကြည့်ပါ။

ပြီးရင် mail မှာ ပါလာမယ့် token နဲ့ အလုပ်လုပ်ဖို့ "/reset_password/<token>" route ကို ရေးပါမယ်။

token ကို verify_reset_token() method နဲ့ စစ်ပြီး user create လုပ်ပါမယ်။ အကယ်၍ token expired ဖြစ်နေရင်တော့ user ရှိမှာ မဟုတ်တဲ့အတွက် warning message ပြပြီး request ပြန်လုပ်ခိုင်းပါမယ်။

token signature ရှိနေရင်တော့ ResetPasswordForm ကို ချပေးပါမယ်။ form မှာ password ကို ပြောင်းပေးပြီး submit လုပ်ပြီးလို့ success ဖြစ်ရင် password ကို reset လုပ်လို့ရသွားပါပြီ။ password အသစ်နဲ့ login လုပ်လို့ရပါပြီ။ ResetPasswordForm အတွက် render template ဖြစ်တဲ့ reset_token.html file ကို ရေးပါ။ code ကိုတော့ အောက်မှာပေးထားတဲ့ file list မှာ ကြည့်ပါ။

```
@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('reset_token.html', title='Reset Password', form=form)
```

အခု send_reset_email() method နဲ့ mail ပို့ဖို့ Flask-Mail ကို သုံးမှာ ဖြစ်တဲ့အတွက် flask-mail ကို install လုပ်ပါ ။

```
pip install flask-mail
```

flask-mail extension က SMTP server ကို ချိတ်ပြီး email တွေကို delivery လုပ်ပေးပါမယ်။ flask-mail ကို configuration မလုပ်ပေးရင်တော့ default အနေနဲ့ localhost server နဲ့ ချိတ်ဆက်ပြီး authenticate မလုပ်ပဲ email ကို ပို့ပေးပါလိမ့်မယ်။ development အချိန်မှာတော့ external SMTP server နဲ့ ချိတ်ဆက်အသုံးပြုတာ ပိုပြီး အဆင်ပြေပါလိမ့်မယ်။ ဒါ ကြောင့် flask-mail ကို configure လုပ်ရအောင် __init__.py file ကိုသွားပါ ။ Google Gmail account ကနေ email ပို့ဖို့ လိုအပ်တဲ့ configuration code ကို ရေးပါမယ်။

```
import os
from flask_mail import Mail
app.config['MAIL_SERVER'] = 'smtp.googlemail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = os.environ.get('EMAIL_USER')
app.config['MAIL_PASSWORD'] = os.environ.get('EMAIL_PASS')
mail = Mail(app)
```

ပြီးရင်တော့ send_reset_email() method ကို implement လုပ်ကြရအောင်။

```
from flaskblog import app, db, bcrypt, mail
from flask_mail import Message


def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request', sender='test@gmail.com',
                  recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
```

```
{url_for('reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes will be made.
'''
    mail.send(msg)
```

login.html ကိုသွားပါ ။ Forgot Password? ကို reset_request route နဲ့ link ချိတ်ပါ။

```
      <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
                <small class="text-muted ml-2">
                    <a href="{{ url_for('reset_request') }}">Forgot Password?</a>
                </small>
            </div>
```

ပြီးရင် run ကြည့်လို့ရပါပြီ။

__init__.py

```
import os
...
from flask_mail import Mail
...

login_manager.login_view = 'login'
login_manager.login_message_category = 'info'
app.config['MAIL_SERVER'] = 'smtp.googlemail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = os.environ.get('EMAIL_USER')
app.config['MAIL_PASSWORD'] = os.environ.get('EMAIL_PASS')
mail = Mail(app)

from flaskblog import routes
```

models.py

```
from datetime import datetime
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from flaskblog import db, login_manager, app
from flask_login import UserMixin

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    password = db.Column(db.String(60), nullable=False)
    posts = db.relationship('Post', backref='author', lazy=True)
    def get_reset_token(self, expires_sec=1800):
        s = Serializer(app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(app.config['SECRET_KEY'])
        try:
            user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)

    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"
     ...
```

forms.py

```
from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileAllowed
from flask_login import current_user
from wtforms import StringField, PasswordField, SubmitField, BooleanField, TextAreaField
from wtforms.validators import DataRequired, Length, Email, EqualTo, ValidationError
from flaskblog.models import User

...
```

```
class RequestResetForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that email. You must register first.')

class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Reset Password')
```

routes.py

```
import os
import secrets
from PIL import Image
from flask import render_template, url_for, flash, redirect, request, abort
from flaskblog import app, db, bcrypt, mail
from flaskblog.forms import (RegistrationForm, LoginForm, UpdateAccountForm,
                             PostForm, RequestResetForm, ResetPasswordForm)
from flaskblog.models import User, Post
from flask_login import login_user, current_user, logout_user, login_required
from flask_mail import Message

@app.route("/")
@app.route("/home")
...

@app.route("/about")
def about():
    return render_template('about.html', title='About')

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

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

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

def save_picture(form_picture):
    ...
    return picture_fn

@app.route("/account", methods=['GET', 'POST'])
@login_required
def account():
    ...

@app.route("/post/new", methods=['GET', 'POST'])
@login_required
def new_post():
    ...

@app.route("/post/<int:post_id>")
def post(post_id):
    post = Post.query.get_or_404(post_id)
    return render_template('post.html', title=post.title, post=post)

@app.route("/post/<int:post_id>/update", methods=['GET', 'POST'])
@login_required
def update_post(post_id):
    ...
@app.route("/post/<int:post_id>/delete", methods=['POST'])
@login_required
def delete_post(post_id):
    ...

@app.route("/user/<string:username>")
```

```
def user_posts(username):
    ...

def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
                  sender='test@gmail.com',
                  recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes will be
made.
'''
    mail.send(msg)

@app.route("/reset_password", methods=['GET', 'POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your password.', 'info')
        return redirect(url_for('login'))
    return render_template('reset_request.html', title='Reset Password', form=form)
```

```
@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('reset_token.html', title='Reset Password', form=form)
```

reset_request.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
            <legend class="border-bottom mb-4">Reset Password</legend>
            <div class="form-group">
                {{ form.email.label(class="form-control-label") }}
                {% if form.email.errors %}
                    {{ form.email(class="form-control form-control-lg is-invalid") }}
                    <div class="invalid-feedback">
                        {% for error in form.email.errors %}
                            <span>{{ error }}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{ form.email(class="form-control form-control-lg") }}
                {% endif %}
            </div>
        </fieldset>

        <div class="form-group">
            {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
</div>
{% endblock content %}
```

reset_token.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}

            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Reset Password</legend>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}
                    {% if form.password.errors %}
                        {{ form.password(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.password.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.password(class="form-control form-control-lg") }}
                    {% endif %}
                </div>

                <div class="form-group">
                    {{ form.confirm_password.label(class="form-control-label") }}
                    {% if form.confirm_password.errors %}
                        {{ form.confirm_password(class="form-control form-control-lg is-invalid")
}}
                        <div class="invalid-feedback">
                            {% for error in form.confirm_password.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.confirm_password(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>

            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
            </div>
        </form>
    </div>
{% endblock content %}
```

login.html

```
        <div class="form-group">
            {{ form.submit(class="btn btn-outline-info") }}
            <small class="text-muted ml-2">
                <a href="{{ url_for('reset_request') }}">Forgot Password?</a>
            </small>
```

</div>

Resources

```
>>>from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
s = Serializer('secret', 30)

token = s.dumps({'user_id': 1}).decode('utf-8')
s.loads(token)

# Signature expired
s.loads(token)

# install mail
pip install flask-mail
```