សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

User Registration Using Flask and SQLite3

User Registration System

2025 – 2026
Y3 – DCS – NU

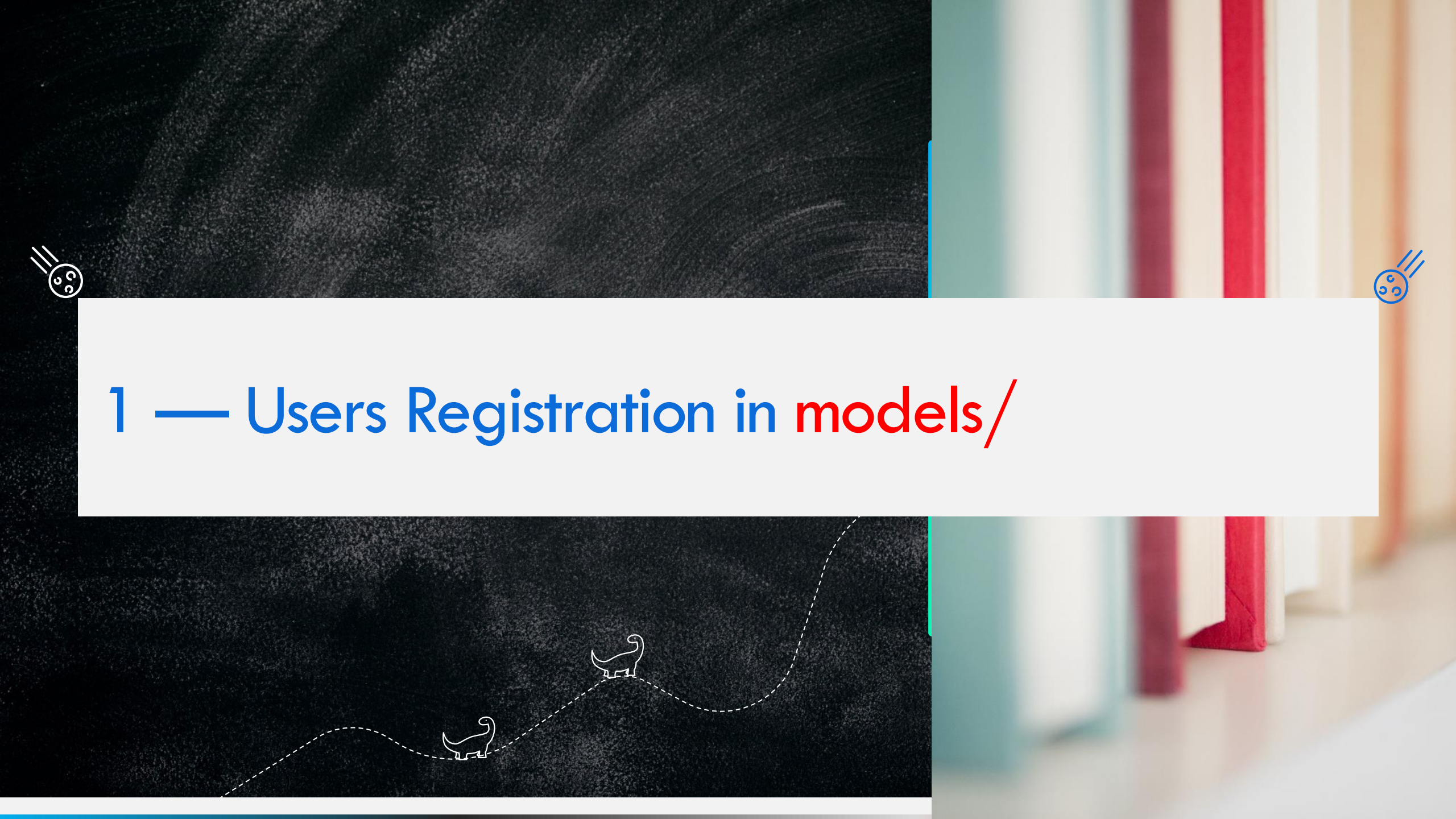
By: SEK SOCHEAT

Advisor to DCS and Lecturer

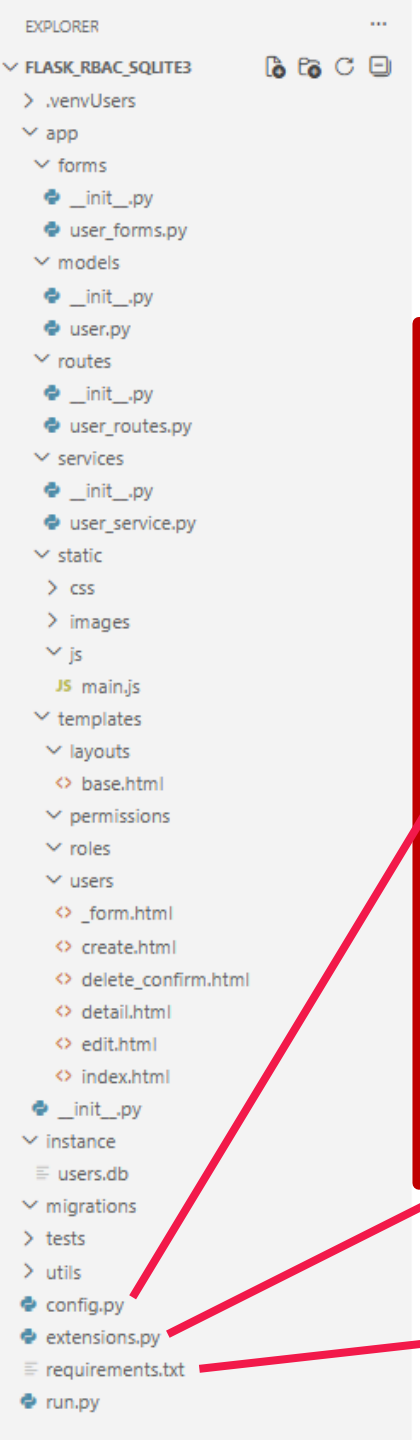
Mobile: 017 879 967

Email: socheatsek@norton-u.com

socheat.sek@gmail.com



1 — Users Registration in models/



config.py

```python
import os

BASE_DIR = os.path.abspath(os.path.dirname(__file__))


class Config:
    SECRET_KEY = os.environ.get("SECRET_KEY", "dev-secret-key-change-me")
    SQLALCHEMY_DATABASE_URI = (
        os.environ.get("DATABASE_URL")
        or "sqlite:///" + os.path.join(BASE_DIR, "instance\\users.db")
    )
    SQLALCHEMY_TRACK_MODIFICATIONS = False

```

extensions.py

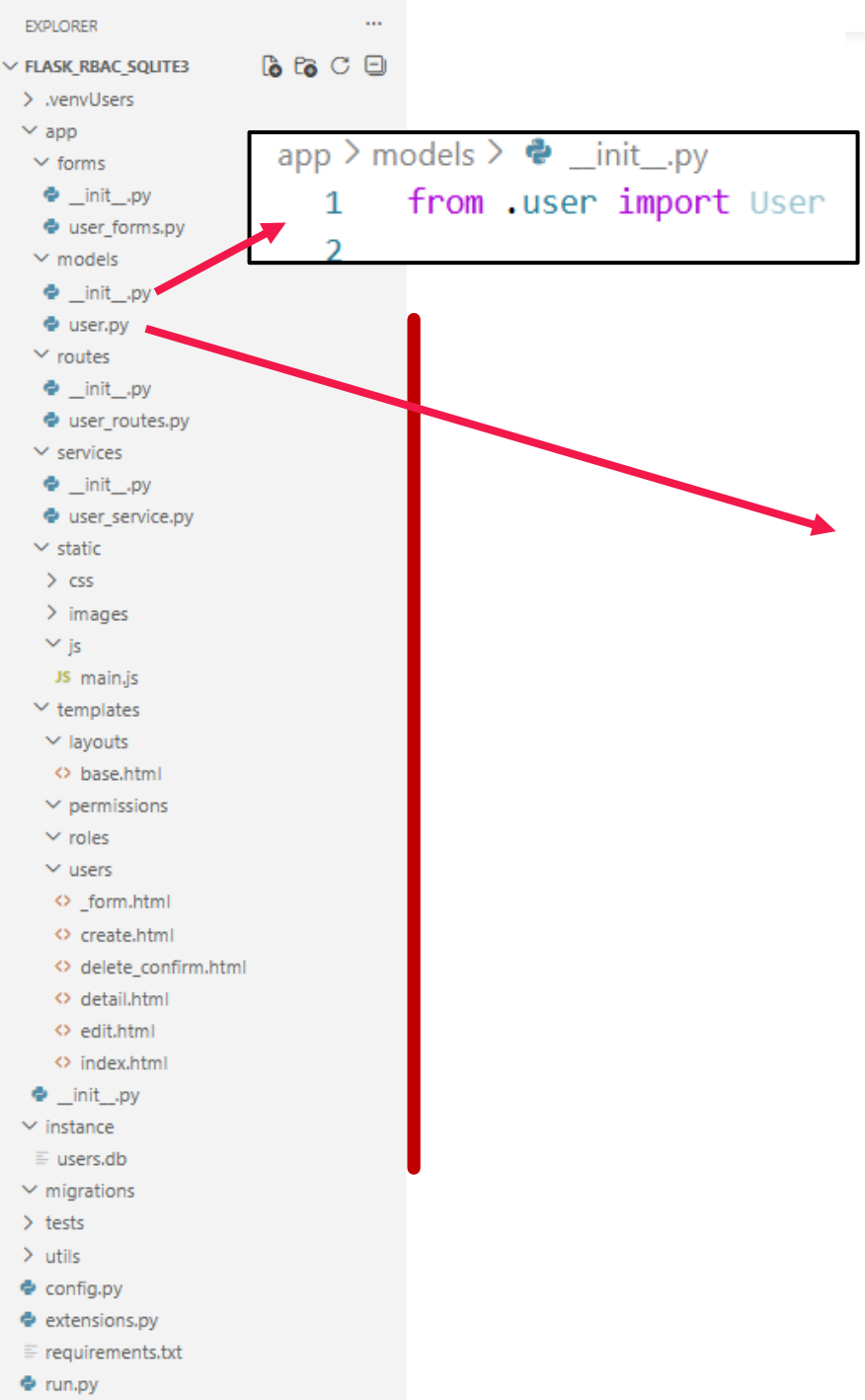
```python
from flask_sqlalchemy import SQLAlchemy
from flask_wtf import CSRFProtect

db = SQLAlchemy()
csrf = CSRFProtect()

```

requirements.txt

```
Flask==2.3.3
Flask-WTF==1.1.1
WTForms==3.1.2
email-validator==2.1.0.post1
Werkzeug==2.3.7
Flask-SQLAlchemy==3.1.1

```

Setting Up a Virtual Environment in Windows via Git-Bash

1. **Create new venv:** python –m venv .venvUser
2. **Activate venv via Git-Bash:** source .venvUser/scripts/activate
3. **Upgrade pip:** python –m pip install ––upgrade pip
4. **Installing requirements:** pip install –r requirements.txt

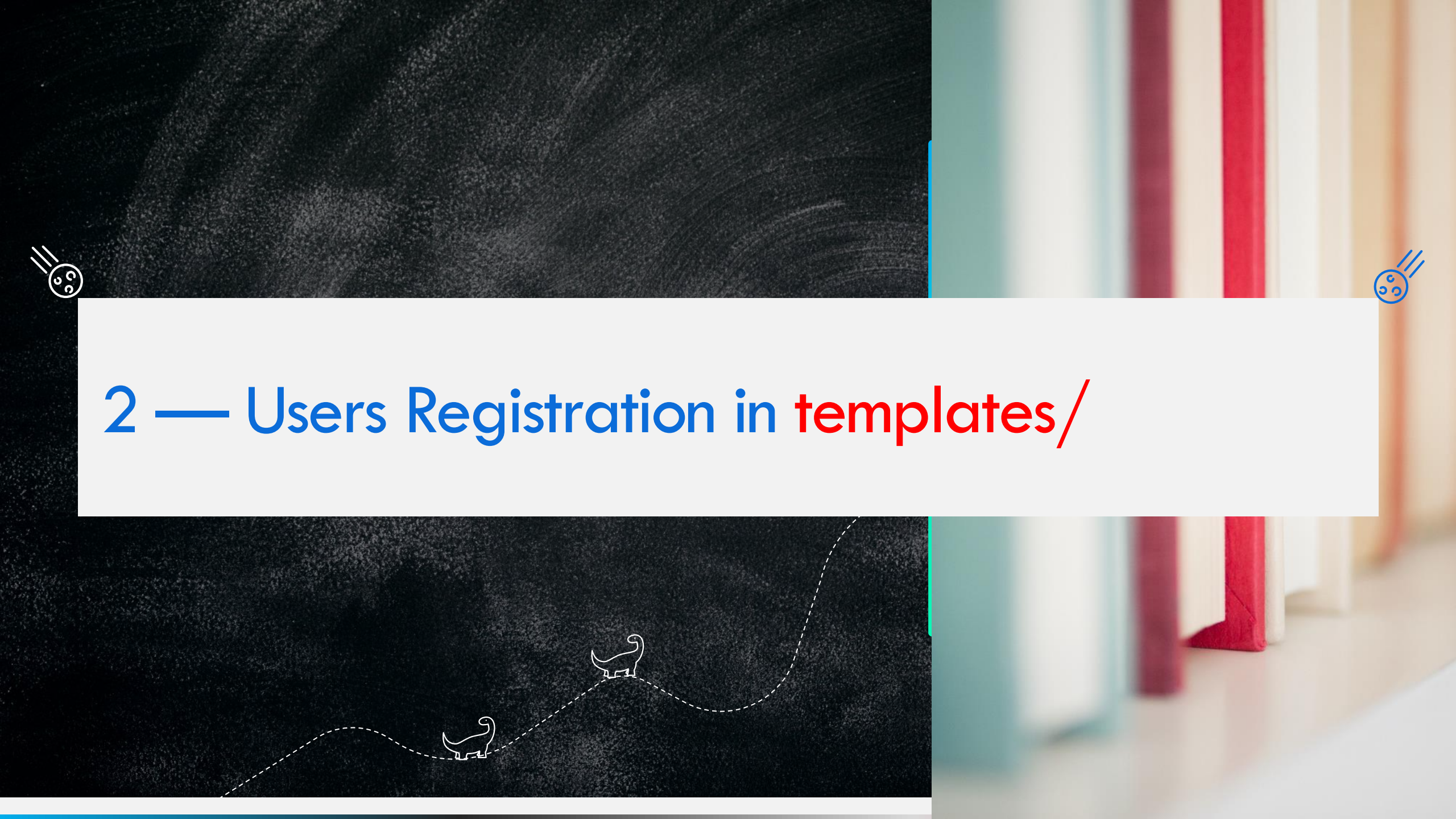


app > models > __init__.py

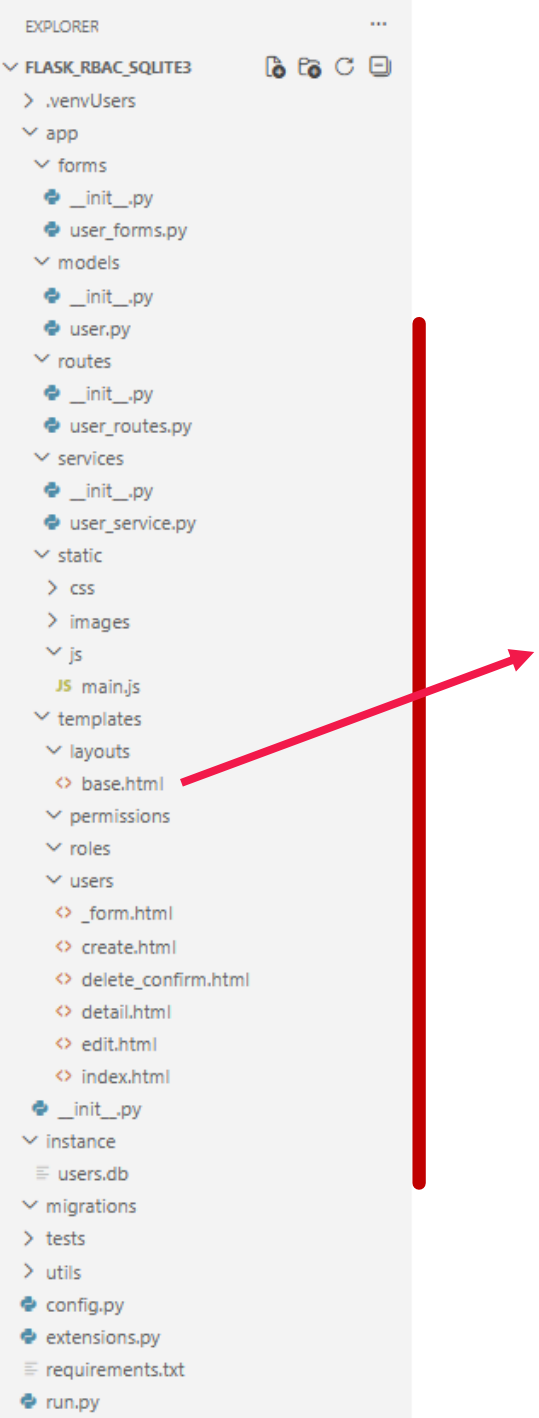
```python
from .user import User
```

app > models > user.py > ...

```python
from datetime import datetime
from werkzeug.security import generate_password_hash, check_password_hash

from extensions import db


class User(db.Model):
    __tablename__ = "users"

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    full_name = db.Column(db.String(120), nullable=False)
    is_active = db.Column(db.Boolean, default=True, nullable=False)

    # NEW: store only the hash, never plain text
    password_hash = db.Column(db.String(255), nullable=False)

    created_at = db.Column(db.DateTime, default=datetime.utcnow, nullable=False)
    updated_at = db.Column(
        db.DateTime, default=datetime.utcnow, onupdate=datetime.utcnow, nullable=False
    )

    # convenience helpers
    def set_password(self, password: str) -> None:
        self.password_hash = generate_password_hash(password)

    def check_password(self, password: str) -> bool:
        return check_password_hash(self.password_hash, password)

    def __repr__(self) -> str:
        return f"<User {self.username}>"
```

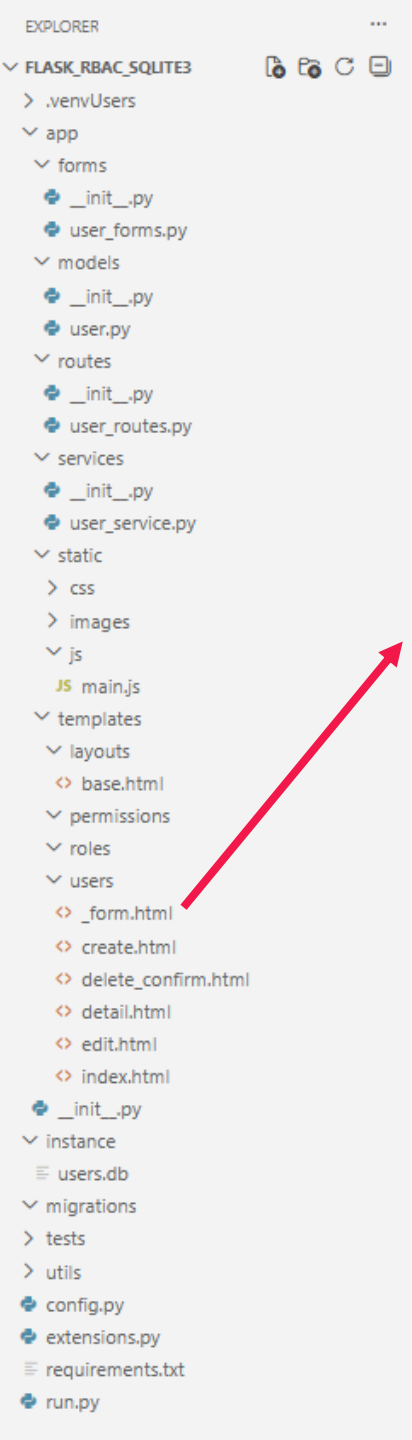


2 — Users Registration in templates/



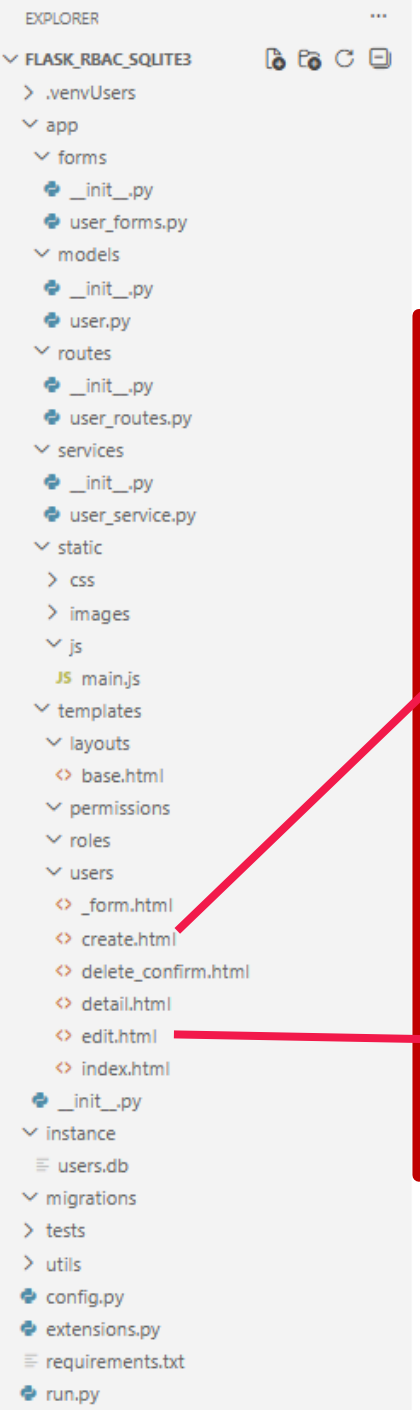
app > templates > layouts > base.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>{% block title %}User Management{% endblock %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <!-- Bootstrap 5 CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />

    <!-- Custom CSS -->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}" />
    {% block extra_css %}{% endblock %}
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary mb-4">
        <div class="container-fluid">
            <a class="navbar-brand" href="{{ url_for('users.index') }}">
                User Management
            </a>
        </div>
    </nav>

    <main class="container">
        {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
        {% for category, message in messages %}
        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
            {{ message }}
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>
        {% endfor %}
        {% endif %}
        {% endwith %}

        {% block content %}{% endblock %}
    </main>

    <!-- Bootstrap Bundle JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
    <script src="{{ url_for('static', filename='js/main.js') }}"></script>
    {% block extra_js %}{% endblock %}
</body>
</html>
```



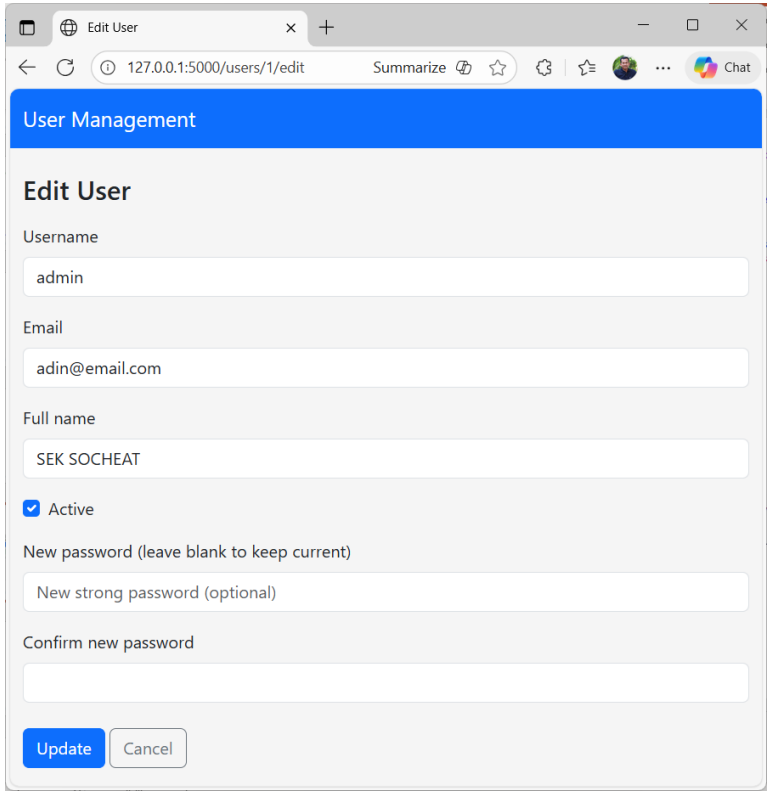
app > templates > users > _form.html

```
{{ form.hidden_tag() }}

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

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

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


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

{# Password fields (may be required or optional depending on form class) #}
<div class="mb-3">
  {{ form.password.label(class="form-label") }}
  {{ form.password(class="form-control") }}
  {% if form.password.errors %}
    <div class="text-danger small">
      {% for error in form.password.errors %}
        <div>{{ error }}</div>
      {% endfor %}
    </div>
  {% endif %}
</div>

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

<div class="mt-4">
    {{ form.submit(class="btn btn-primary") }}
    <a href="{{ url_for('users.index') }}" class="btn btn-outline-secondary">
        Cancel
    </a>
</div>
```



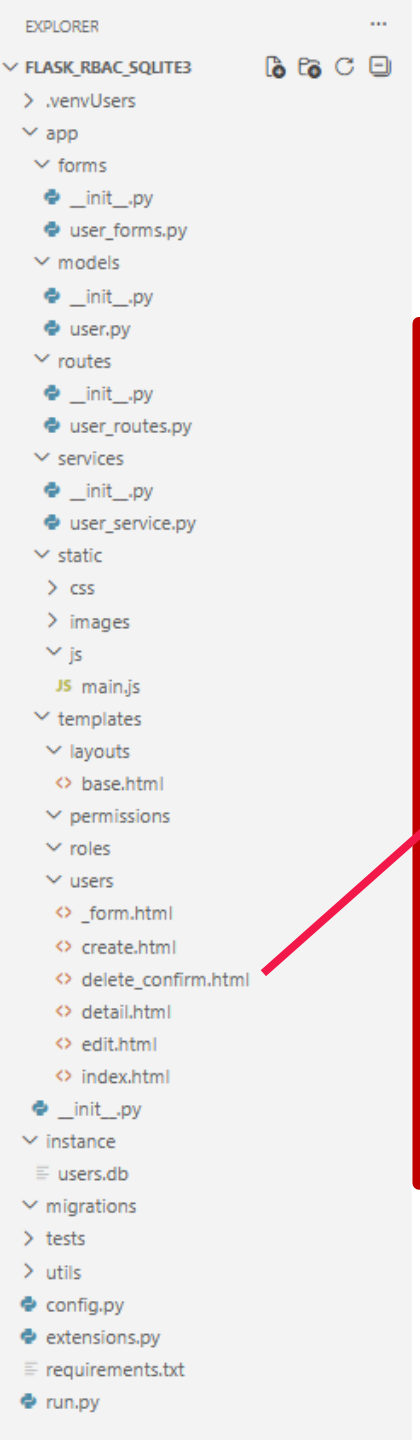
app > templates > users > create.html

```
1  {% extends "layouts/base.html" %}
2  {% block title %}Create User{% endblock %}
3
4  {% block content %}
5  <h1 class="h3 mb-3">Create User</h1>
6  <form method="post" novalidate>
7      {% include "users/_form.html" %}
8  </form>
9  {% endblock %}
```



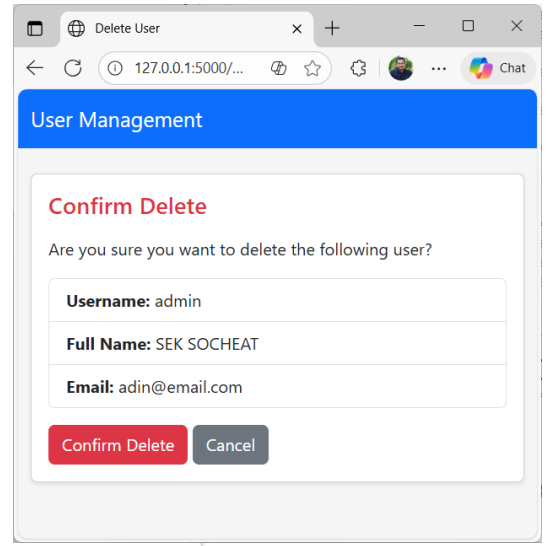
app > templates > users > edit.html

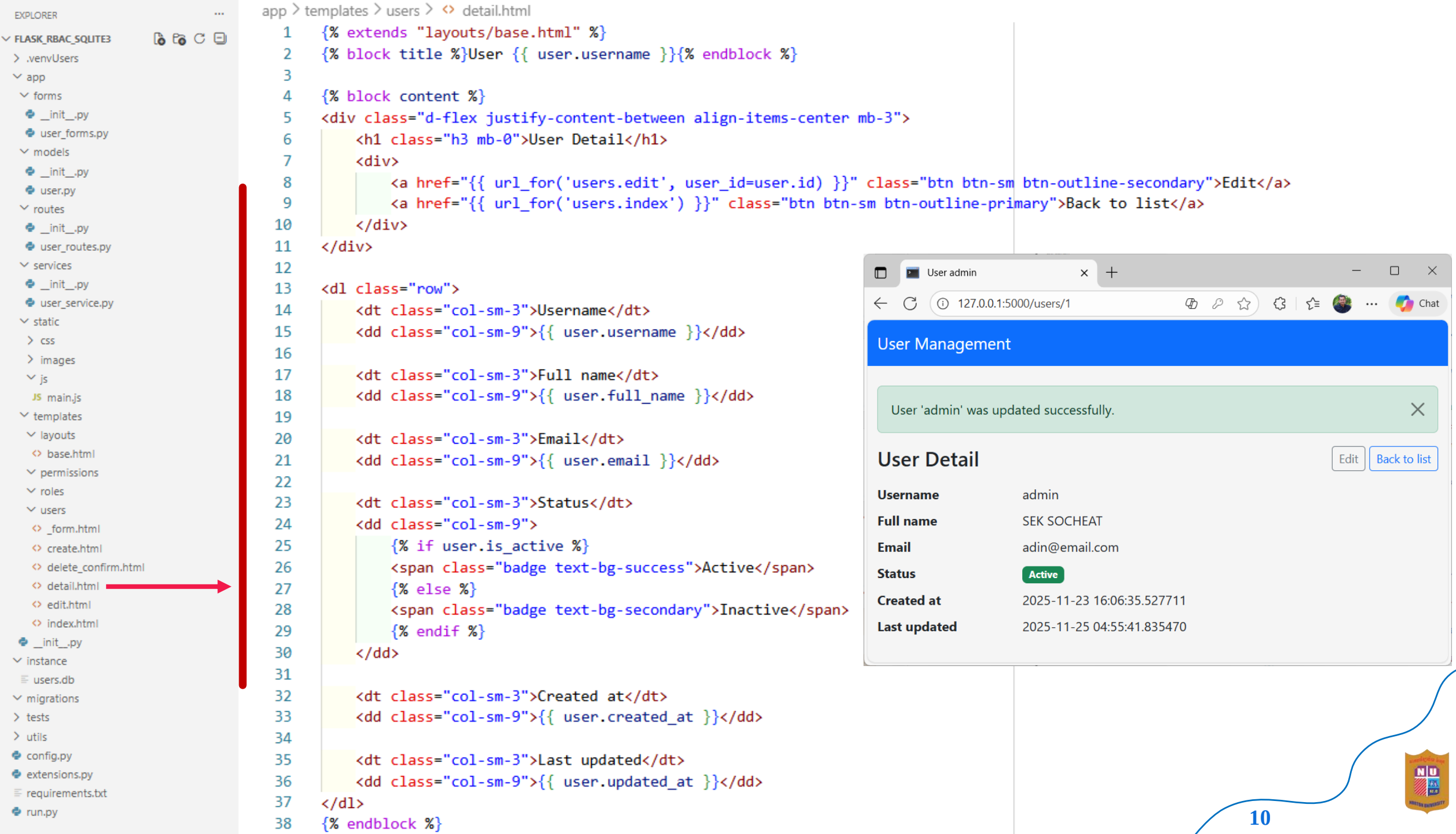
```
1  {% extends "layouts/base.html" %}
2  {% block title %}Edit User{% endblock %}
3
4  {% block content %}
5  <h1 class="h3 mb-3">Edit User</h1>
6  <form method="post" novalidate>
7      {% include "users/_form.html" %}
8  </form>
9  {% endblock %}
```

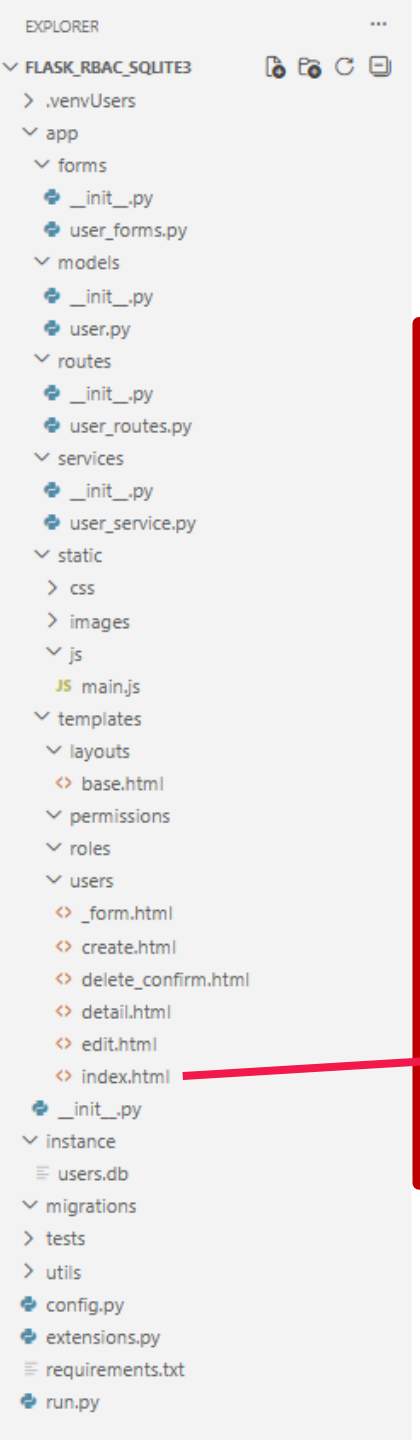


app > templates > users > delete_confirm.html

```
1  {% extends "layouts/base.html" %}
2  {% block title %}Delete User{% endblock %}
3
4  {% block content %}
5
6  <div class="card shadow-sm">
7    <div class="card-body">
8      <h3 class="h4 text-danger">Confirm Delete</h3>
9
10     <p class="mt-3">Are you sure you want to delete the following user?</p>
11
12     <ul class="list-group mb-3">
13       <li class="list-group-item"><strong>Username:</strong> {{ user.username }}</li>
14       <li class="list-group-item"><strong>Full Name:</strong> {{ user.full_name }}</li>
15       <li class="list-group-item"><strong>Email:</strong> {{ user.email }}</li>
16     </ul>
17
18     <form method="POST" action="{{ url_for('users.delete', user_id=user.id) }}">
19       {{ form.hidden_tag() }}
20       {{ form.submit(class="btn btn-danger") }}
21       <a href="{{ url_for('users.index') }}" class="btn btn-secondary">Cancel</a>
22     </form>
23   </div>
24 </div>
25
26 {% endblock %}
```

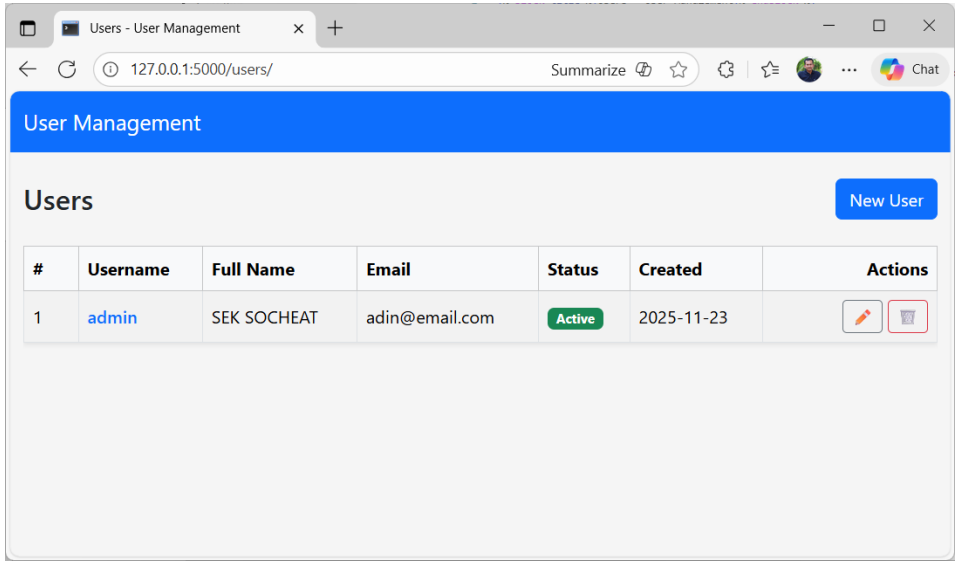


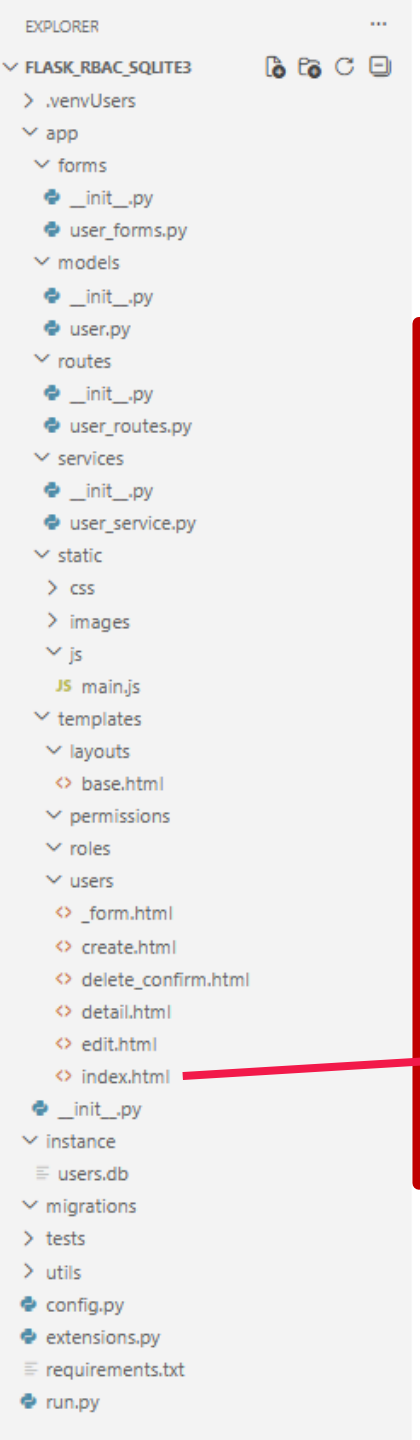

EXPLORER
FLASK_RBAC_SQLITE3
.venvUsers
app
forms
__init__.py
user_forms.py
models
__init__.py
user.py
routes
__init__.py
user_routes.py
services
__init__.py
user_service.py
static
css
images
js
main.js
templates
layouts
base.html
permissions
roles
users
_form.html
create.html
delete_confirm.html
detail.html
edit.html
index.html
__init__.py
instance
users.db
migrations
tests
utils
config.py
extensions.py
requirements.txt
run.py
app > templates > users > detail.html
1 {% extends "layouts/base.html" %}
2 {% block title %}User {{ user.username }}{% endblock %}
3
4 {% block content %}
5
6 <h1 class="h3 mb-0">User Detail</h1>
7
8 Edit
9 Back to list
10
11
12
13 <dl class="row">
14 <dt class="col-sm-3">Username</dt>
15 <dd class="col-sm-9">{{ user.username }}</dd>
16
17 <dt class="col-sm-3">Full name</dt>
18 <dd class="col-sm-9">{{ user.full_name }}</dd>
19
20 <dt class="col-sm-3">Email</dt>
21 <dd class="col-sm-9">{{ user.email }}</dd>
22
23 <dt class="col-sm-3">Status</dt>
24 <dd class="col-sm-9">
25 {% if user.is_active %}
26 Active
27 {% else %}
28 Inactive
29 {% endif %}
30 </dd>
31
32 <dt class="col-sm-3">Created at</dt>
33 <dd class="col-sm-9">{{ user.created_at }}</dd>
34
35 <dt class="col-sm-3">Last updated</dt>
36 <dd class="col-sm-9">{{ user.updated_at }}</dd>
37 </dl>
38 {% endblock %}
User admin
127.0.0.1:5000/users/1
Chat
User Management
User 'admin' was updated successfully.
User Detail
Edit
Back to list
Username admin
Full name SEK SOCHEAT
Email adin@email.com
Status Active
Created at 2025-11-23 16:06:35.527711
Last updated 2025-11-25 04:55:41.835470



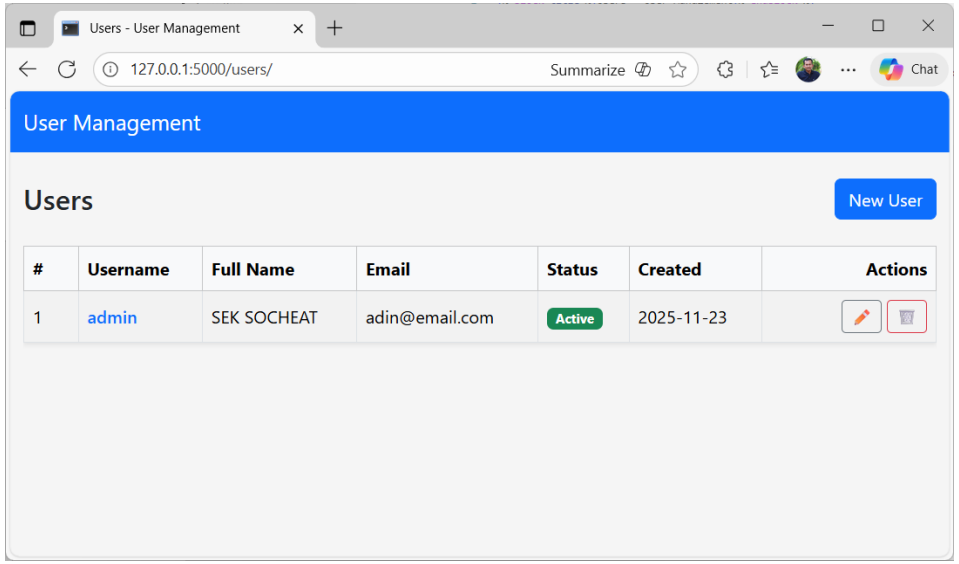
app > templates > users > index.html

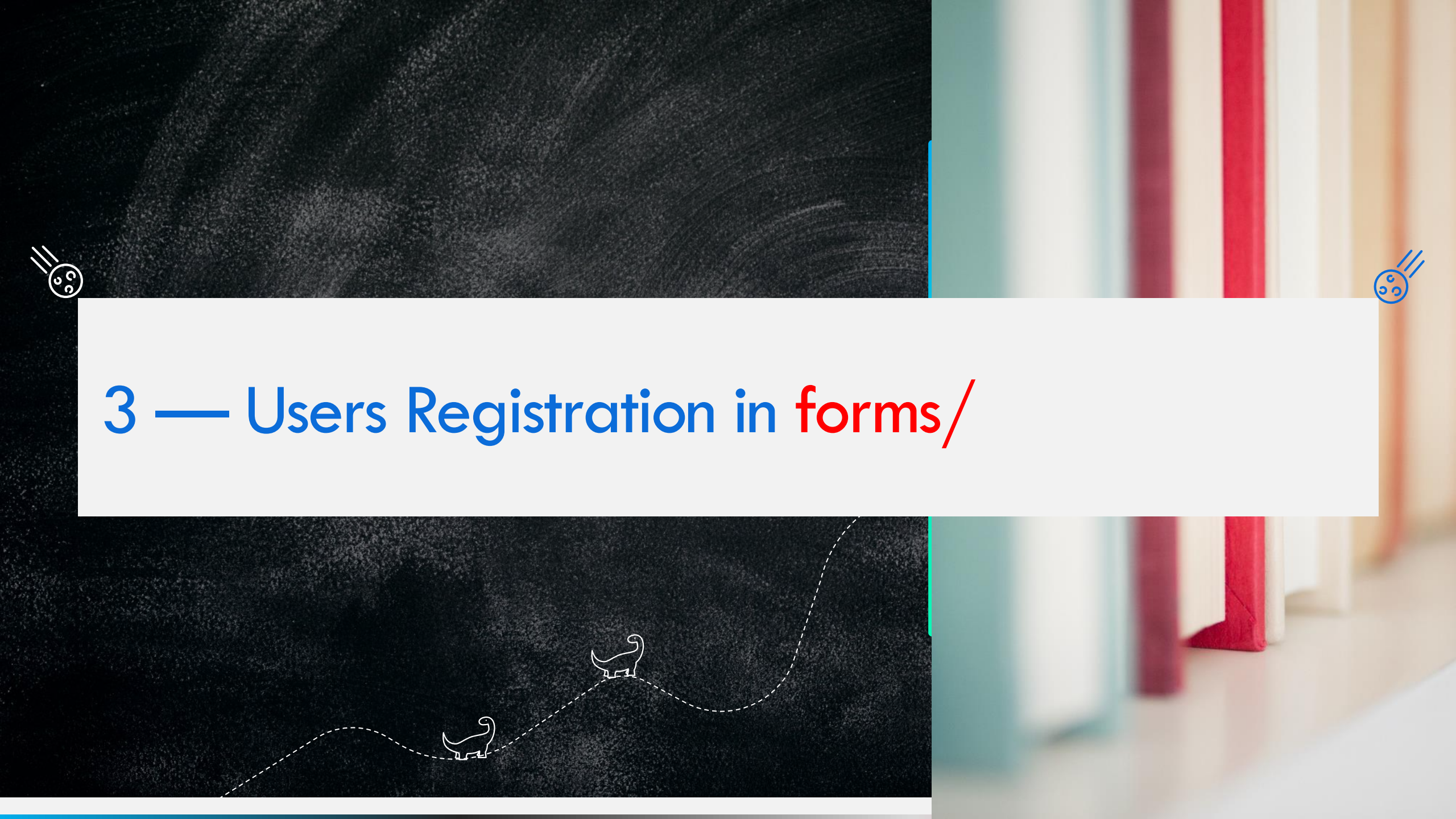
```
1  {% extends "layouts/base.html" %}
2  {% block title %}Users - User Management{% endblock %}
3
4  {% block content %}
5  <div class="d-flex justify-content-between align-items-center mb-4">
6    <h1 class="h3 mb-0">Users</h1>
7    <a href="{{ url_for('users.create') }}" class="btn btn-primary">New User</a>
8  </div>
9
10 {% if users %}
11   <div class="table-responsive">
12     <table class="table table-striped table-bordered align-middle shadow-sm">
13       <thead class="table-light">
14         <tr>
15           <th style="width: 50px;">#</th>
16           <th>Username</th>
17           <th>Full Name</th>
18           <th>Email</th>
19           <th>Status</th>
20           <th>Created</th>
21           <th class="text-end" style="width: 160px;">Actions</th>
22         </tr>
23       </thead>
24       <tbody>
25         {% for user in users %}
26           <tr>
27             <td>{{ loop.index }}</td>
28
29             <td>
30               <a
31                 href="{{ url_for('users.detail', user_id=user.id) }}"
32                 class="text-decoration-none fw-semibold"
33               >
34                 {{ user.username }}
35               </a>
36             </td>
37
38             <td>{{ user.full_name }}</td>
39             <td>{{ user.email }}</td>
40
```



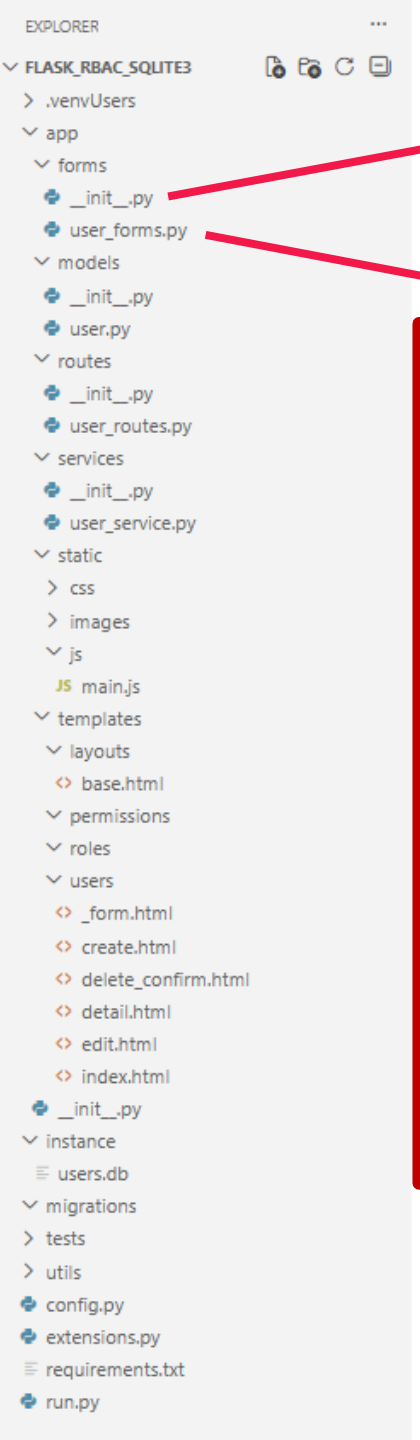



```
40
41              <td>
42                {% if user.is_active %}
43                  <span class="badge bg-success">Active</span>
44                {% else %}
45                  <span class="badge bg-secondary">Inactive</span>
46                {% endif %}
47              </td>
48
49              <td>{{ user.created_at.strftime('%Y-%m-%d') }}</td>
50
51              <td class="text-end">
52                <!-- Edit -->
53                <a
54                  href="{{ url_for('users.edit', user_id=user.id) }}"
55                  class="btn btn-sm btn-outline-secondary"
56                  title="Edit"
57                >
58                  ✏️
59                </a>
60
61                <!-- Delete (go to custom delete confirmation page) -->
62                <a
63                  href="{{ url_for('users.delete_confirm', user_id=user.id) }}"
64                  class="btn btn-sm btn-outline-danger"
65                  title="Delete"
66                >
67                  🗑️
68                </a>
69              </td>
70
71            </tr>
72          {% endfor %}
73        </tbody>
74      </table>
75    </div>
76  {% else %}
77    <div class="alert alert-info">
78      No users found.
79      <a href="{{ url_for('users.create') }}">Create the first user</a>.
80    </div>
81  {% endif %}
82  {% endblock %}
```





3 — Users Registration in forms/

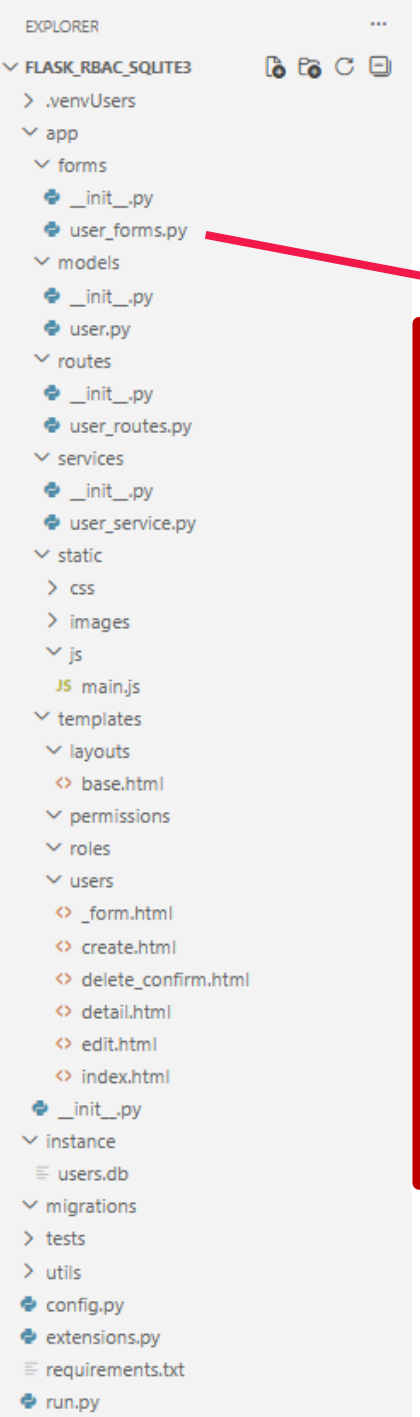


app > forms > __init__.py

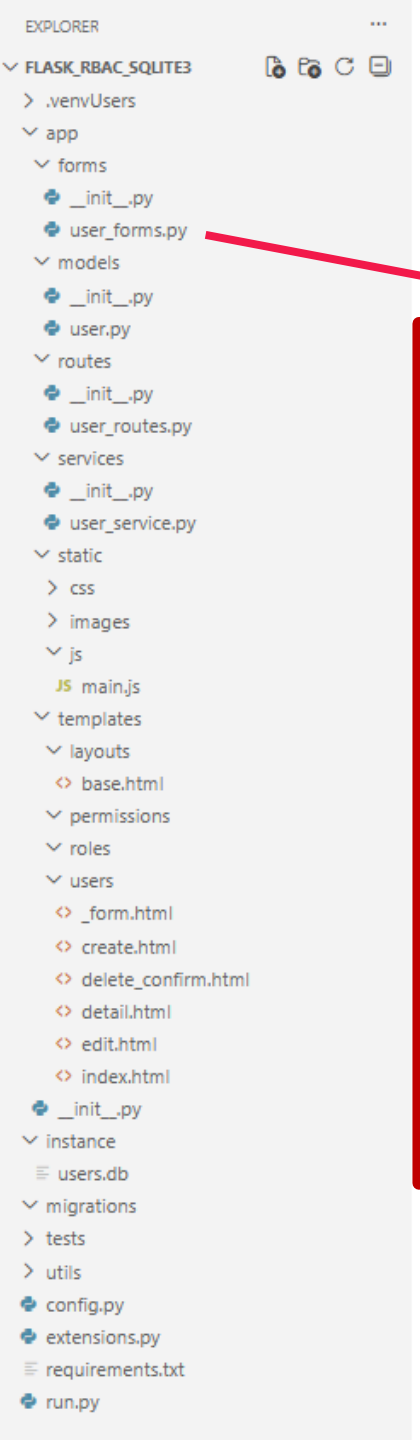
```python
from .user_forms import UserCreateForm, UserEditForm
from .user_forms import ConfirmDeleteForm
```

app > forms > user_forms.py > ...

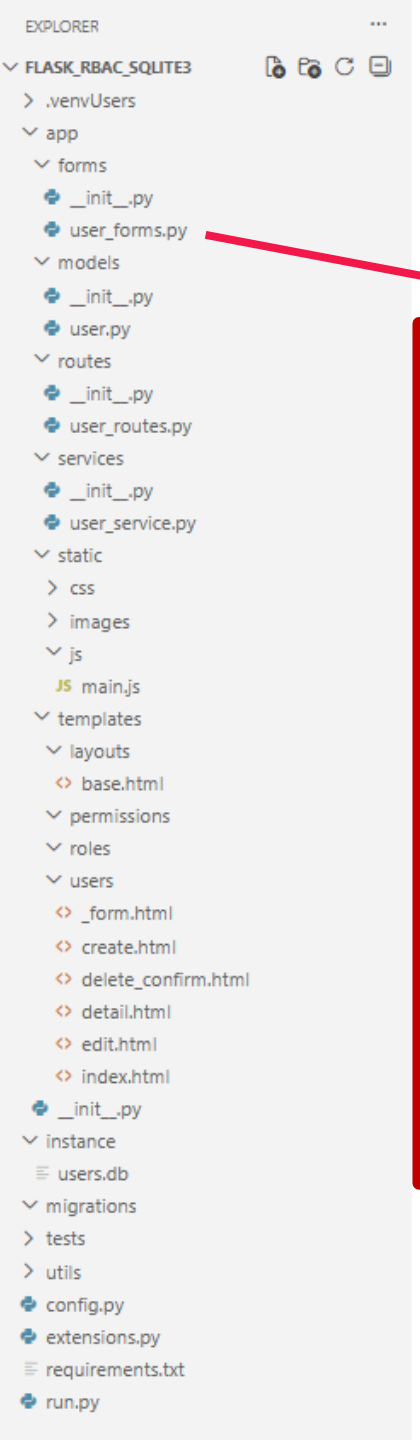
```python
import re
from flask_wtf import FlaskForm
from wtforms import BooleanField, StringField, SubmitField, PasswordField
from wtforms.validators import DataRequired, Email, Length, EqualTo, ValidationError

from app.models import User
from extensions import db

# ----- helpers -----

def strong_password(form, field):
    """Require: min 8 chars, upper, lower, digit, special."""
    password = field.data or ""
    if len(password) < 8:
        raise ValidationError("Password must be at least 8 characters long.")

    if not re.search(r"[A-Z]", password):
        raise ValidationError("Password must contain at least one uppercase letter.")

    if not re.search(r"[a-z]", password):
        raise ValidationError("Password must contain at least one lowercase letter.")

    if not re.search(r"[0-9]", password):
        raise ValidationError("Password must contain at least one digit.")

    if not re.search(r"[!@#$%^&*(),.?\":{}|<>_\-+=]", password):
        raise ValidationError("Password must contain at least one special character.")
```



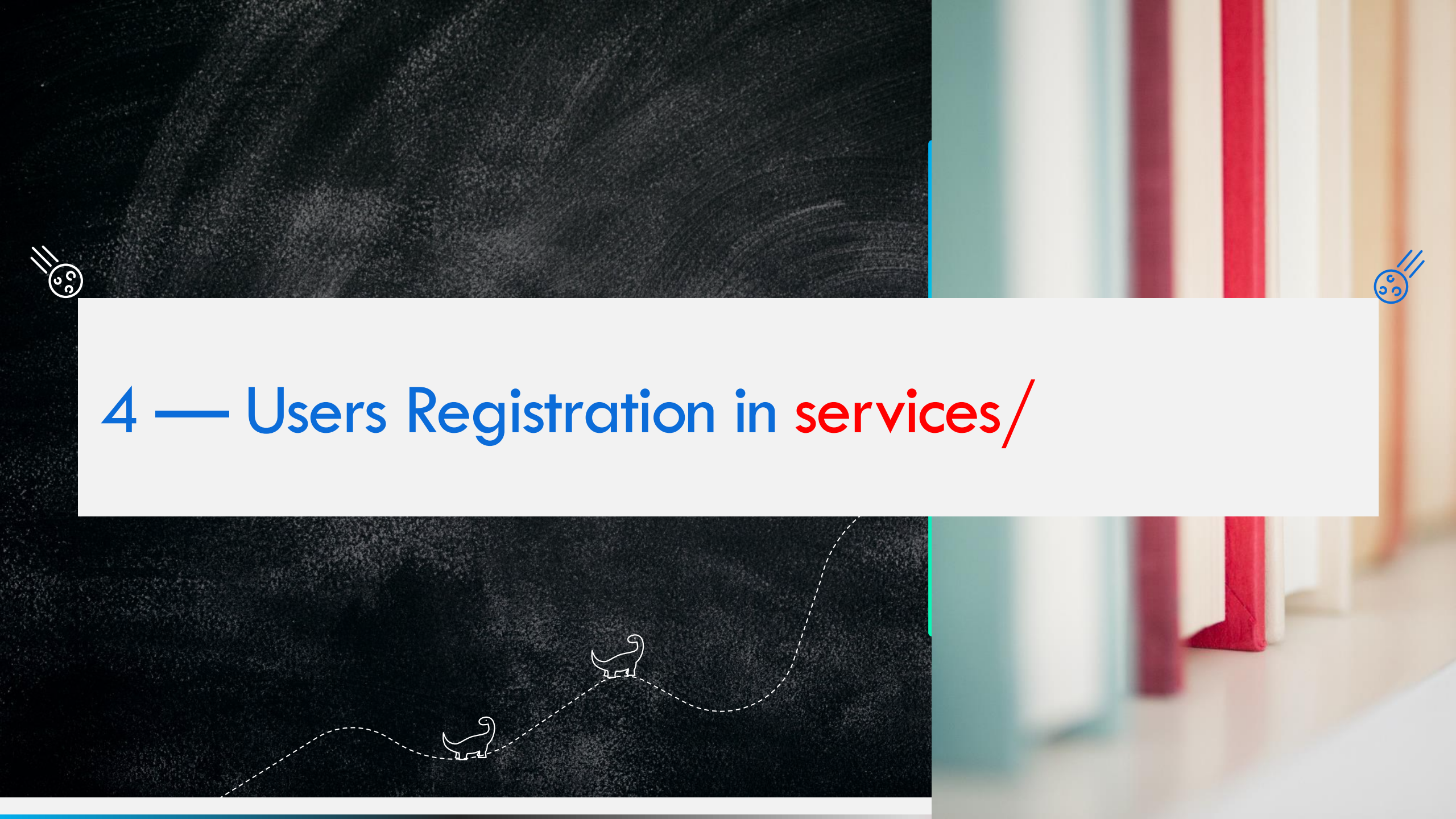

```python
# ----- create form (password required) -----

class UserCreateForm(FlaskForm):
    username = StringField(
        "Username",
        validators=[DataRequired(), Length(min=3, max=80)],
        render_kw={"placeholder": "Enter username"},
    )
    email = StringField(
        "Email",
        validators=[DataRequired(), Email(), Length(max=120)],
        render_kw={"placeholder": "Enter email"},
    )
    full_name = StringField(
        "Full name",
        validators=[DataRequired(), Length(min=3, max=120)],
        render_kw={"placeholder": "Enter full name"},
    )
    is_active = BooleanField("Active", default=True)

    password = PasswordField(
        "Password",
        validators=[
            DataRequired(),
            strong_password,
        ],
        render_kw={"placeholder": "Strong password"},
    )
    confirm_password = PasswordField(
        "Confirm password",
        validators=[
            DataRequired(),
            EqualTo("password", message="Passwords must match."),
        ],
        render_kw={"placeholder": "Confirm password"},
    )

    submit = SubmitField("Save")

```



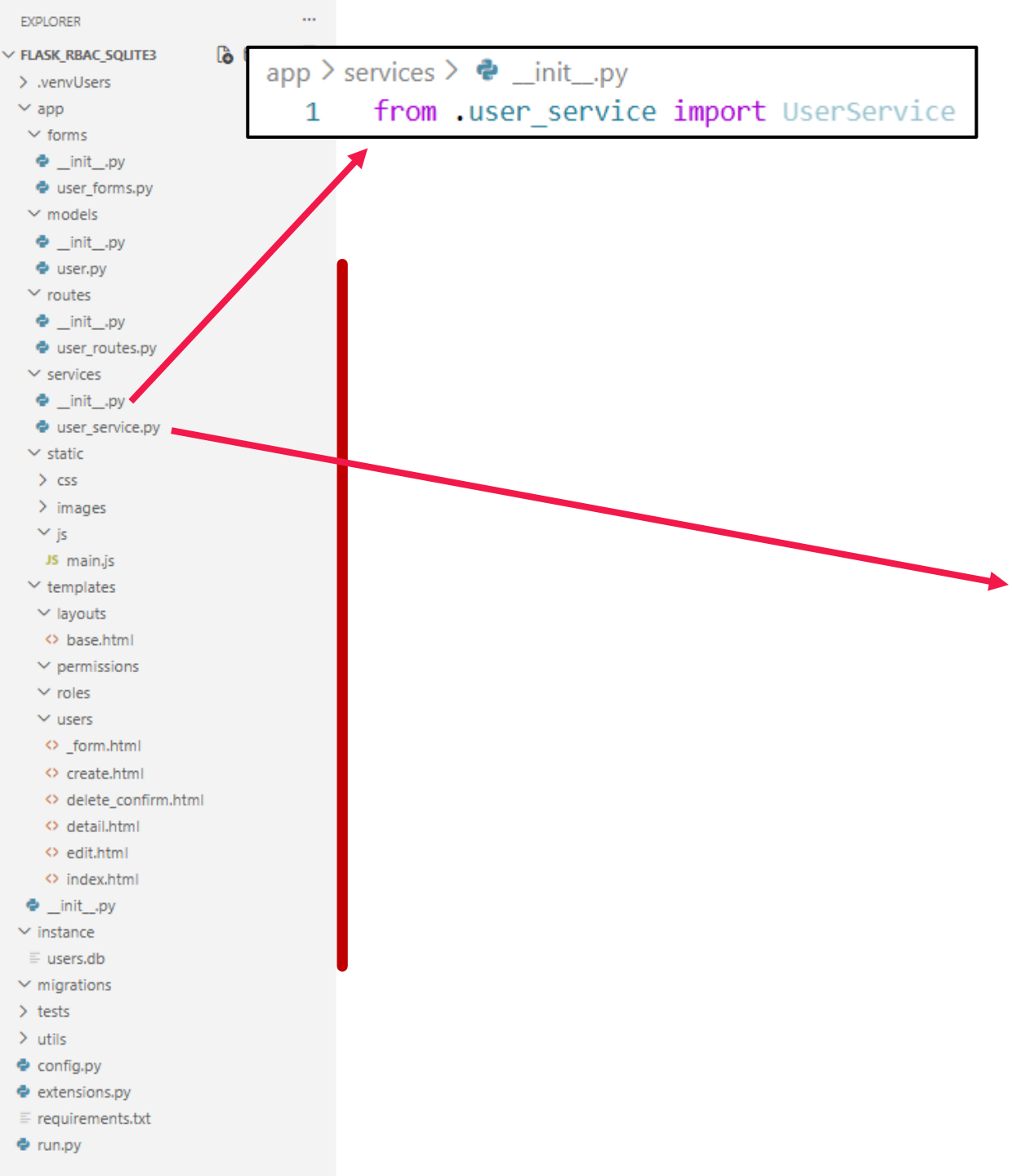

```python
    # ---- server-side uniqueness checks ----

    def validate_username(self, field):
        exists = db.session.scalar(
            db.select(User).filter(User.username == field.data)
        )
        if exists:
            raise ValidationError("This username is already taken.")

    def validate_email(self, field):
        exists = db.session.scalar(
            db.select(User).filter(User.email == field.data)
        )
        if exists:
            raise ValidationError("This email is already registered.")
```




```python
# ----- edit form (password optional) -----

class UserEditForm(FlaskForm):
    username = StringField(
        "Username",
        validators=[DataRequired(), Length(min=3, max=80)],
    )
    email = StringField(
        "Email",
        validators=[DataRequired(), Email(), Length(max=120)],
    )
    full_name = StringField(
        "Full name",
        validators=[DataRequired(), Length(min=3, max=120)],
    )
    is_active = BooleanField("Active")

    # optional password - only change if filled
    password = PasswordField(
        "New password (leave blank to keep current)",
        validators=[strong_password],
        render_kw={"placeholder": "New strong password (optional)"},
    )
    confirm_password = PasswordField(
        "Confirm new password",
        validators=[EqualTo("password", message="Passwords must match.")],
    )

    submit = SubmitField("Update")

    def __init__(self, original_user: User, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.original_user = original_user

    def validate_username(self, field):
        q = db.select(User).filter(User.username == field.data, User.id != self.original_user.id)
        exists = db.session.scalar(q)
        if exists:
            raise ValidationError("This username is already taken.")

    def validate_email(self, field):
        q = db.select(User).filter(User.email == field.data, User.id != self.original_user.id)
        exists = db.session.scalar(q)
        if exists:
            raise ValidationError("This email is already registered.")

class ConfirmDeleteForm(FlaskForm):
    submit = SubmitField("Confirm Delete")
```

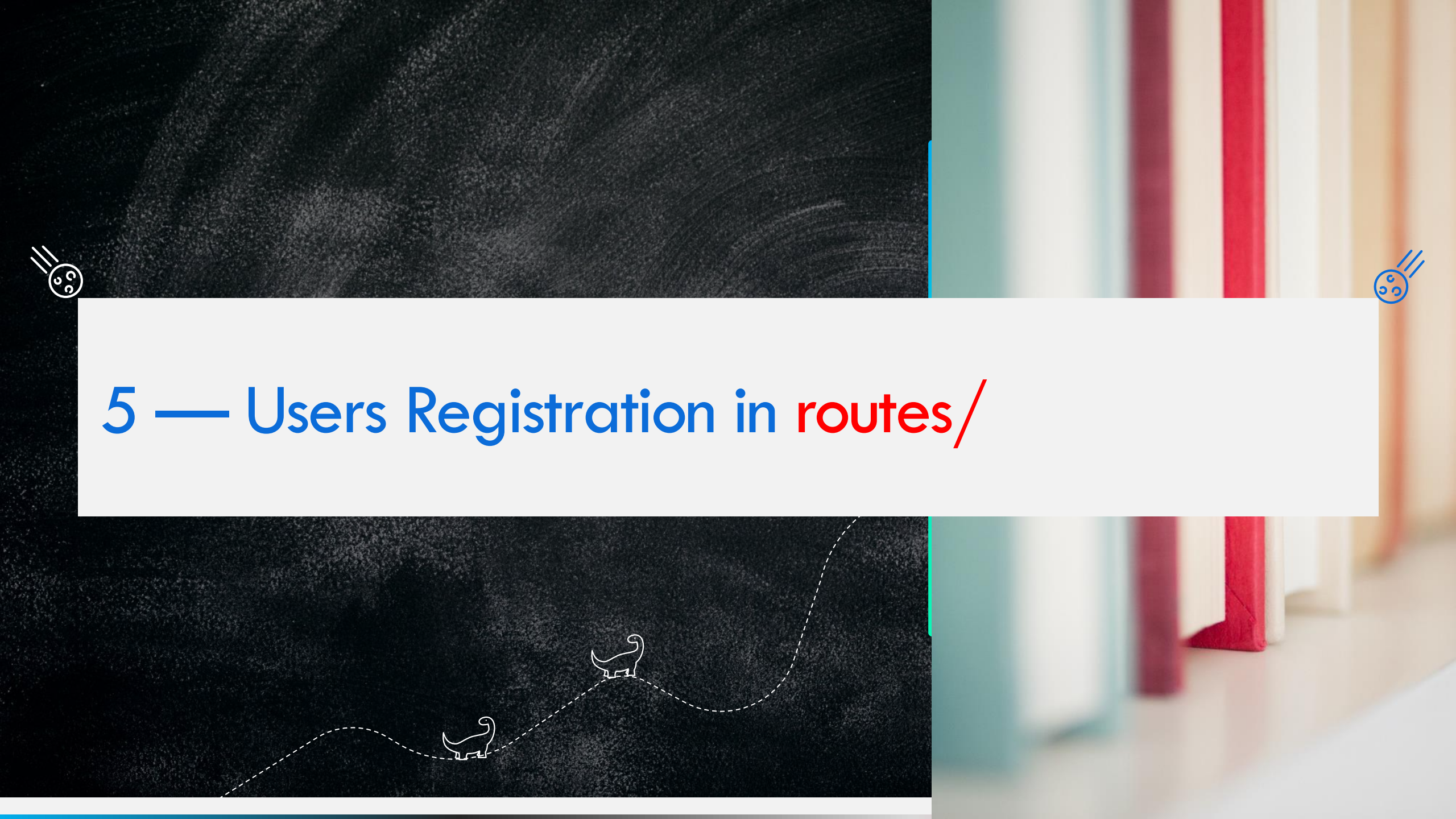


4 — Users Registration in services/

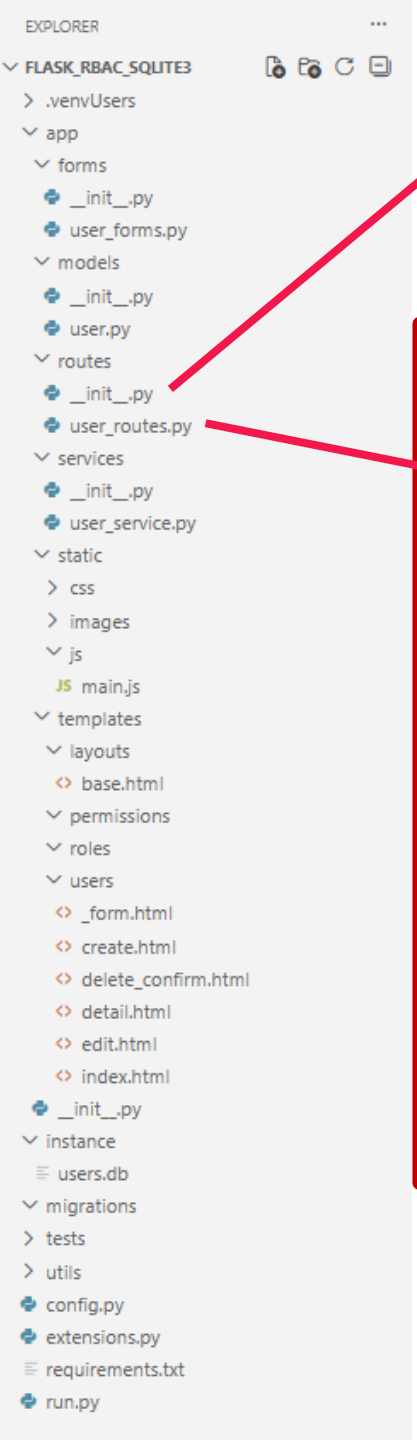


app > services > user_service.py > ...

```python
from typing import List, Optional
from app.models.user import User
from extensions import db

class UserService:
    @staticmethod
    def get_all() -> List[User]:
        return User.query.order_by(User.id.desc()).all()

    @staticmethod
    def get_by_id(user_id: int) -> Optional[User]:
        return User.query.get(user_id)

    @staticmethod
    def create(data: dict, password: str) -> User:
        user = User(
            username=data["username"],
            email=data["email"],
            full_name=data["full_name"],
            is_active=data.get("is_active", True),
        )
        user.set_password(password)
        db.session.add(user)
        db.session.commit()
        return user

    @staticmethod
    def update(user: User, data: dict, password: Optional[str] = None) -> User:
        user.username = data["username"]
        user.email = data["email"]
        user.full_name = data["full_name"]
        user.is_active = data.get("is_active", True)

        if password:
            user.set_password(password)

        db.session.commit()
        return user

    @staticmethod
    def delete(user: User) -> None:
        db.session.delete(user)
        db.session.commit()
```



5 — Users Registration in routes/

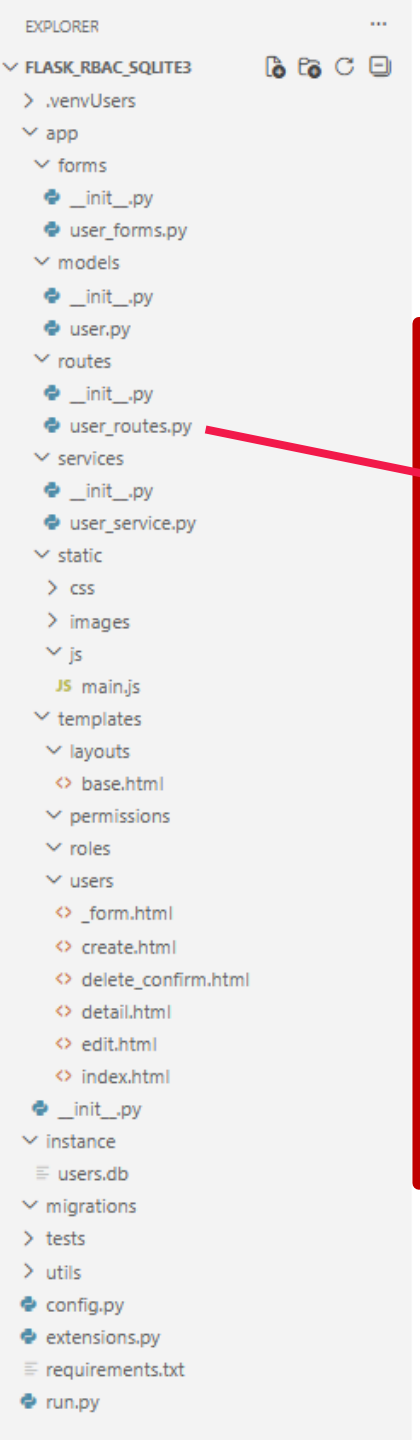


app > routes > __init__.py

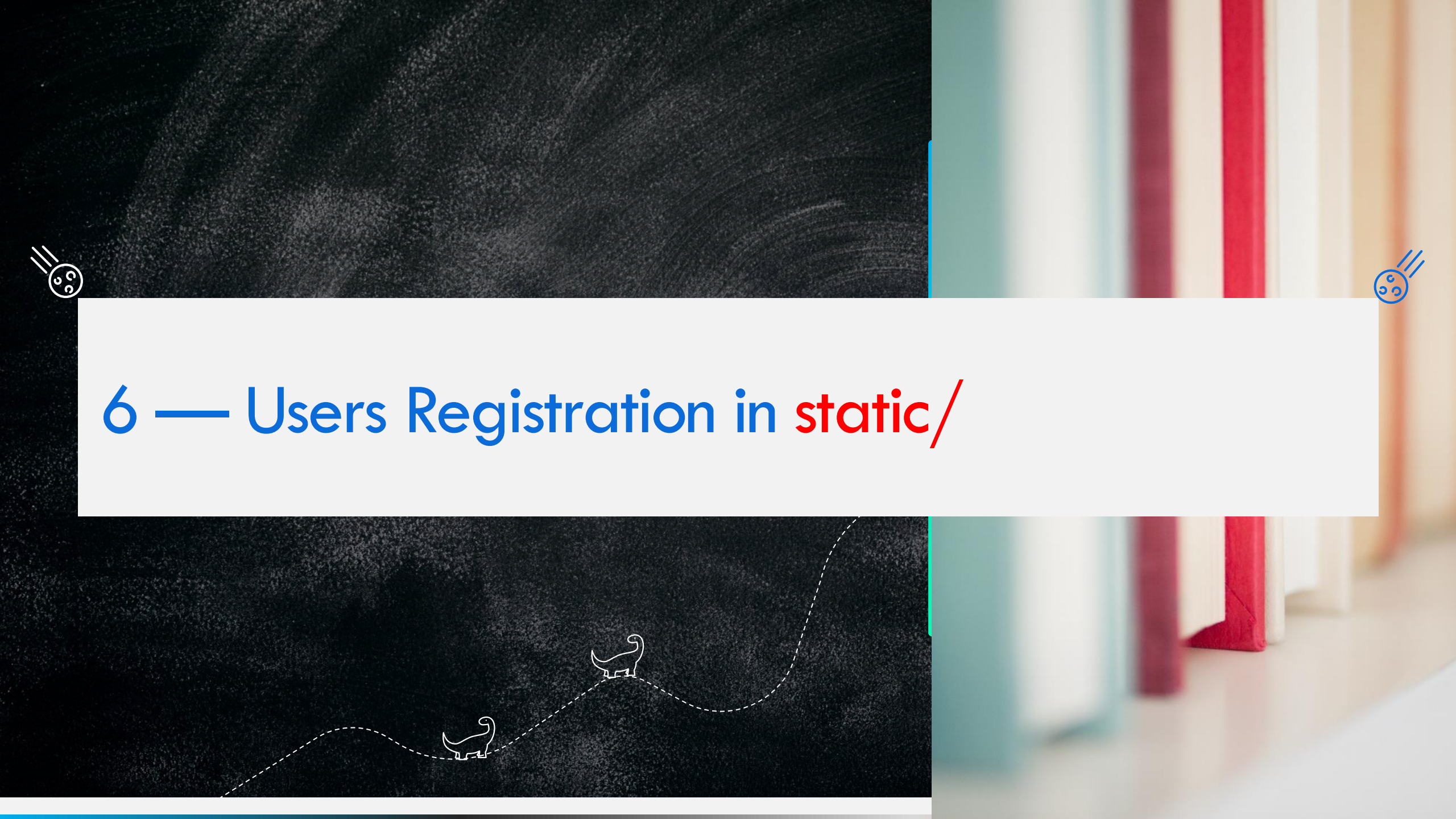
```python
from .user_routes import user_bp
```

app > routes > user_routes.py > edit

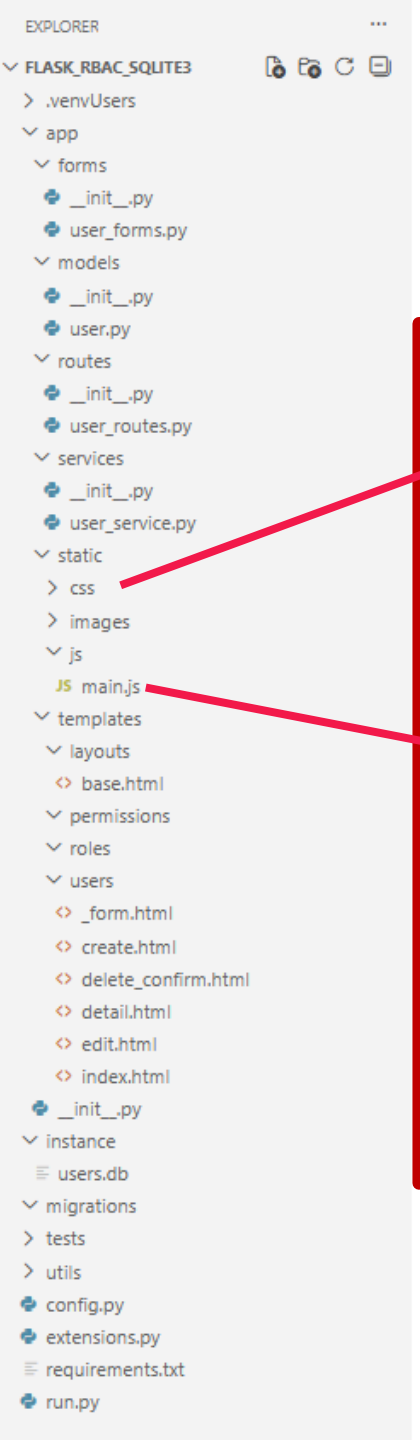
```python
from flask import (
    Blueprint,
    render_template,
    redirect,
    url_for,
    flash,
    abort,
)

from app.forms.user_forms import UserCreateForm, UserEditForm, ConfirmDeleteForm
from app.services.user_service import UserService

user_bp = Blueprint("users", __name__, url_prefix="/users")

@user_bp.route("/")
def index():
    users = UserService.get_all()
    return render_template("users/index.html", users=users)

@user_bp.route("/<int:user_id>")
def detail(user_id: int):
    user = UserService.get_by_id(user_id)
    if user is None:
        abort(404)
    return render_template("users/detail.html", user=user)

@user_bp.route("/create", methods=["GET", "POST"])
def create():
    form = UserCreateForm()
    if form.validate_on_submit():
        data = {
            "username": form.username.data,
            "email": form.email.data,
            "full_name": form.full_name.data,
            "is_active": form.is_active.data,
        }
        password = form.password.data
        user = UserService.create(data, password)
        flash(f"User '{user.username}' was created successfully.", "success")
        return redirect(url_for("users.index"))

    return render_template("users/create.html", form=form)
```




```python
@user_bp.route("/<int:user_id>/edit", methods=["GET", "POST"])
def edit(user_id: int):
    user = UserService.get_by_id(user_id)
    if user is None:
        abort(404)

    form = UserEditForm(original_user=user, obj=user)

    if form.validate_on_submit():
        data = {
            "username": form.username.data,
            "email": form.email.data,
            "full_name": form.full_name.data,
            "is_active": form.is_active.data,
        }
        password = form.password.data or None
        UserService.update(user, data, password)
        flash(f"User '{user.username}' was updated successfully.", "success")
        return redirect(url_for("users.detail", user_id=user.id))

    return render_template("users/edit.html", form=form, user=user)

@user_bp.route("/<int:user_id>/delete", methods=["GET"])
def delete_confirm(user_id: int):
    user = UserService.get_by_id(user_id)
    if user is None:
        abort(404)
    form = ConfirmDeleteForm()
    return render_template("users/delete_confirm.html", user=user, form=form)


@user_bp.route("/<int:user_id>/delete", methods=["POST"])
def delete(user_id: int):
    user = UserService.get_by_id(user_id)
    if user is None:
        abort(404)

    UserService.delete(user)
    flash("User was deleted successfully.", "success")
    return redirect(url_for("users.index"))
```



6 — Users Registration in static/



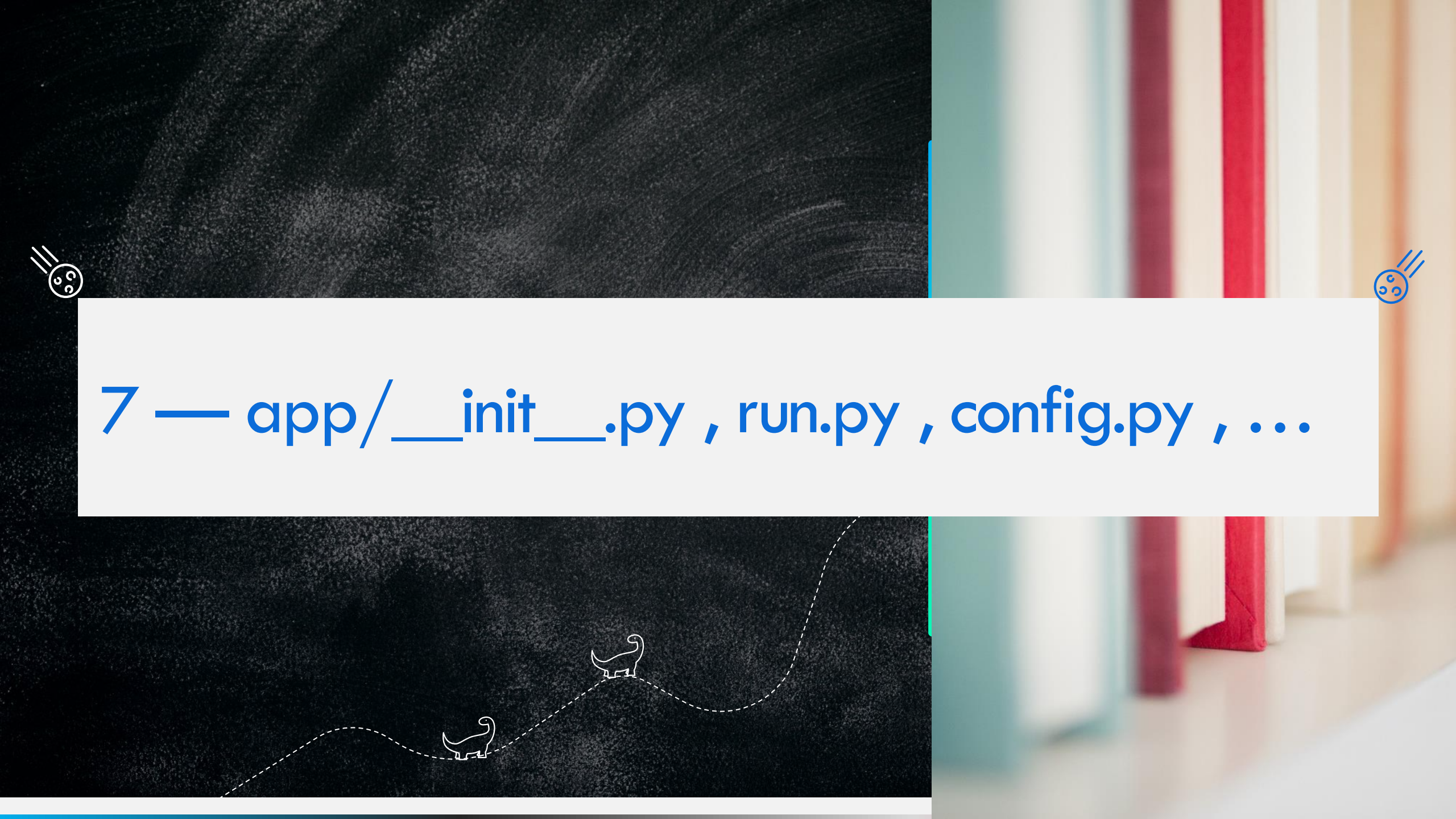
style.css

app > static > css > # style.css > ...

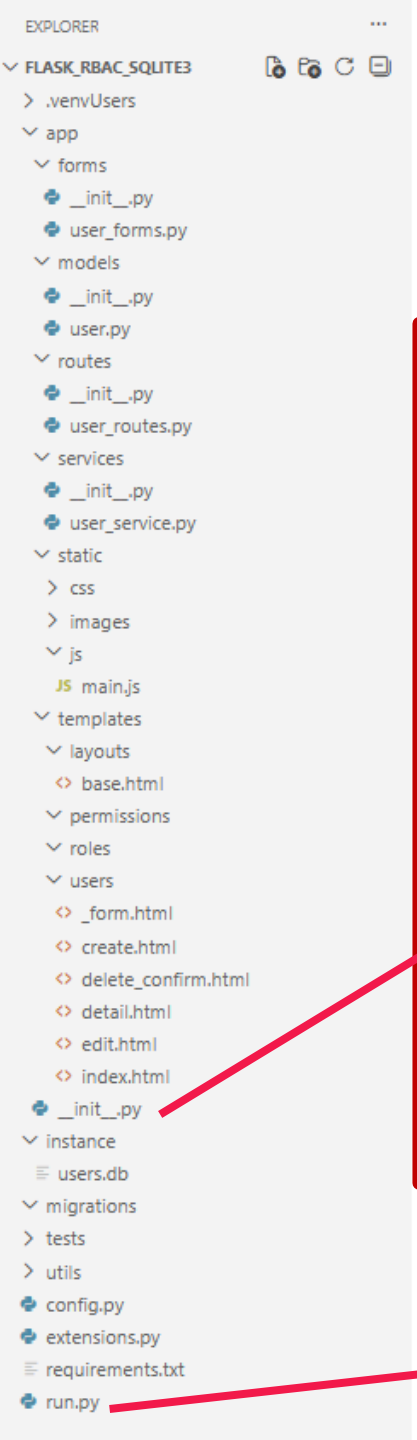
```css
body {
    background-color: #f5f5f5;
}

main.container {
    max-width: 960px;
}

.table td,
.table th {
    vertical-align: middle;
}
```

app > static > js > JS main.js > ...

```js
document.addEventListener("DOMContentLoaded", () => {
    const pwd = document.querySelector('input[name="password"]');
    if (!pwd) return;

    pwd.addEventListener("input", () => {
        const msg = document.querySelector("#passwordHelp");
        if (!msg) return;

        const v = pwd.value;
        const strong =
            v.length >= 8 &&
            /[A-Z]/.test(v) &&
            /[a-z]/.test(v) &&
            /[0-9]/.test(v) &&
            /[!@#$%^&*(),.?":{}|<>_\-+=]/.test(v);

        msg.textContent = strong
            ? "Strong password ✅"
            : "Use at least 8 chars, with upper, lower, number, and special symbol.";
        msg.className = strong ? "form-text text-success" : "form-text text-danger";
    });
});
```



7 — app/__init__.py , run.py , config.py , ...



app > __init__.py > ...

```python
from flask import Flask, redirect, url_for
from config import Config
from extensions import db, csrf

def create_app(config_class: type[Config] = Config):
    app = Flask(__name__)
    app.config.from_object(config_class)

    db.init_app(app)
    csrf.init_app(app)

    # Register blueprints
    from app.routes.user_routes import user_bp
    app.register_blueprint(user_bp)

    # 👇 Add this block so "/" goes to the Users list
    @app.route("/")
    def home():
        return redirect(url_for("users.index"))

    # Create tables
    with app.app_context():
        from app.models import User  # noqa: F401
        db.create_all()

    return app
```

run.py > ...

```python
from app import create_app

app = create_app()

if __name__ == "__main__":
    app.run(debug=True)

```

Thank You

SEK SocheaT

socheatsek@norton-u.com