សាកលវិទ្យាល័យន័រតុន

NORTON UNIVERSITY

Flask Project Structure for Expert Systems

By: SEK SOCHEAT

Advisor to DCS and Lecturer

Mobile: 017 879 967

Email: socheatsek@norton-u.com

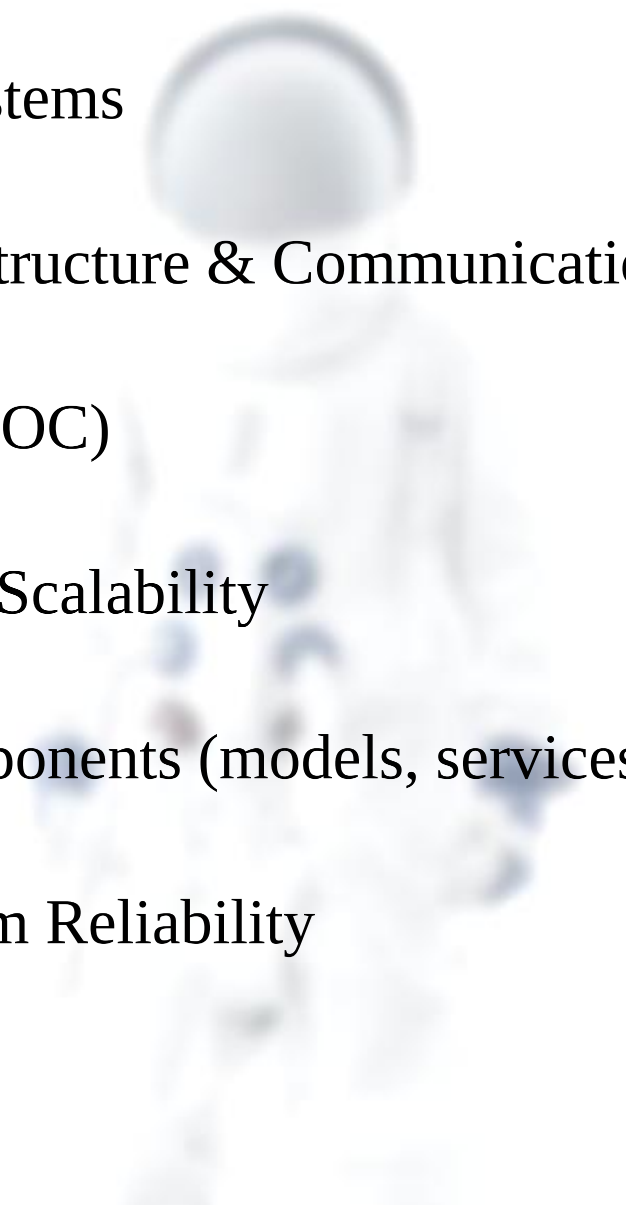
socheat.sek@gmail.com

Expert System

2025 – 2026

Y3 – DCS – NU

Table of Contents:



សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

Learning Outcome:

By the end of this lesson, students will be able to:

1. Explain the purpose of expert systems and how their workflow maps to a structured Flask application.

2. Identify and describe each major folder in the project structure and explain the role it plays in the expert system architecture.

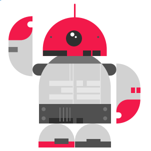
3. Apply the principle of Separation of Concerns (SOC) to keep knowledge, rules, logic, and interface code properly separated.

4. Demonstrate how structured design improves scalability, allowing new rules, facts, and features to be added without breaking the system.

5. Build and organize knowledge models, reasoning logic, and consultation routes into appropriate folders following expert-system best practices.

6. Evaluate system reliability by using tests, migrations, and configuration files to ensure the expert system remains stable as it grows.

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

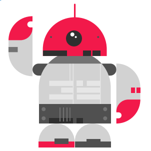


1 — Introduction to Expert Systems

What Is an Expert System?

- AI that mimics human decision-making
- Uses facts + rules
- Needs a clear structure
- ***Example:*** Permissions System

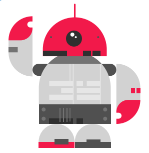
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY



Expert System Workflow

- User Input
- Knowledge Base (facts)
- Inference Engine (rules)
- Output
- Flask project maps to these steps

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

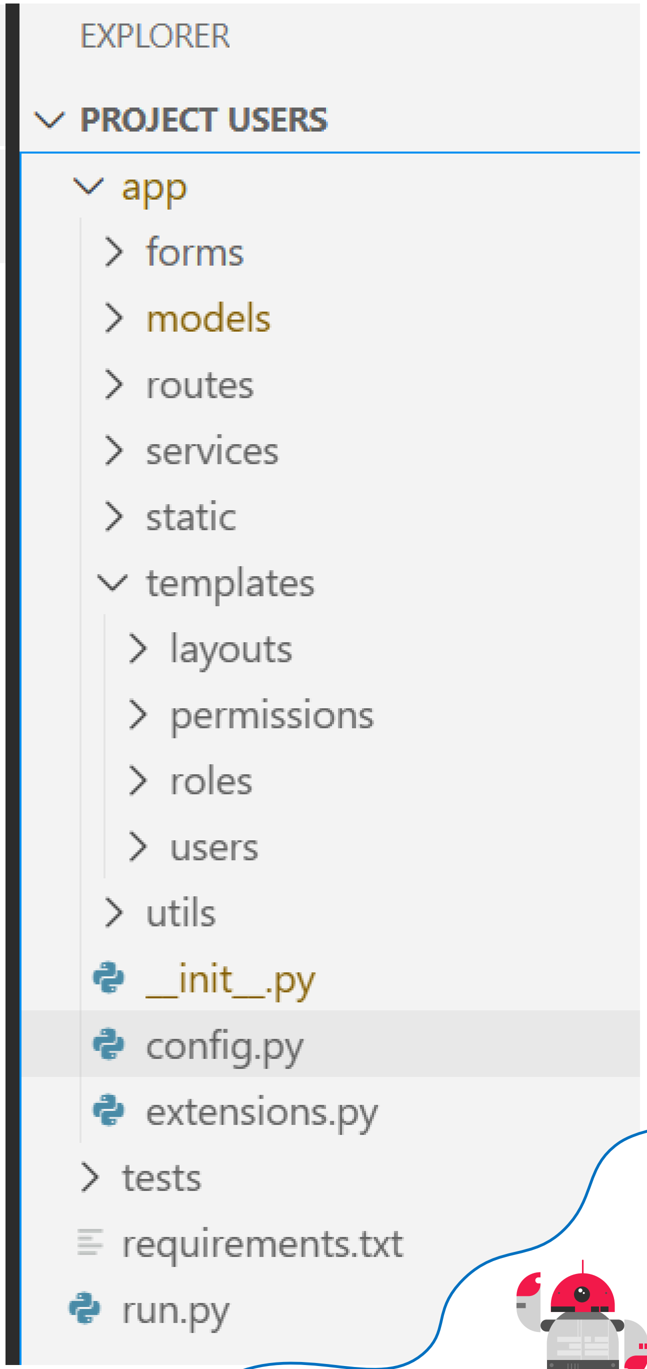


2 — Overview of the Project Structure & Communication

2 — Overview of the Project Structure & Communication

Full Project Structure Overview

- Display folder tree
- "Everything has a place"
- Supports real-world workflow



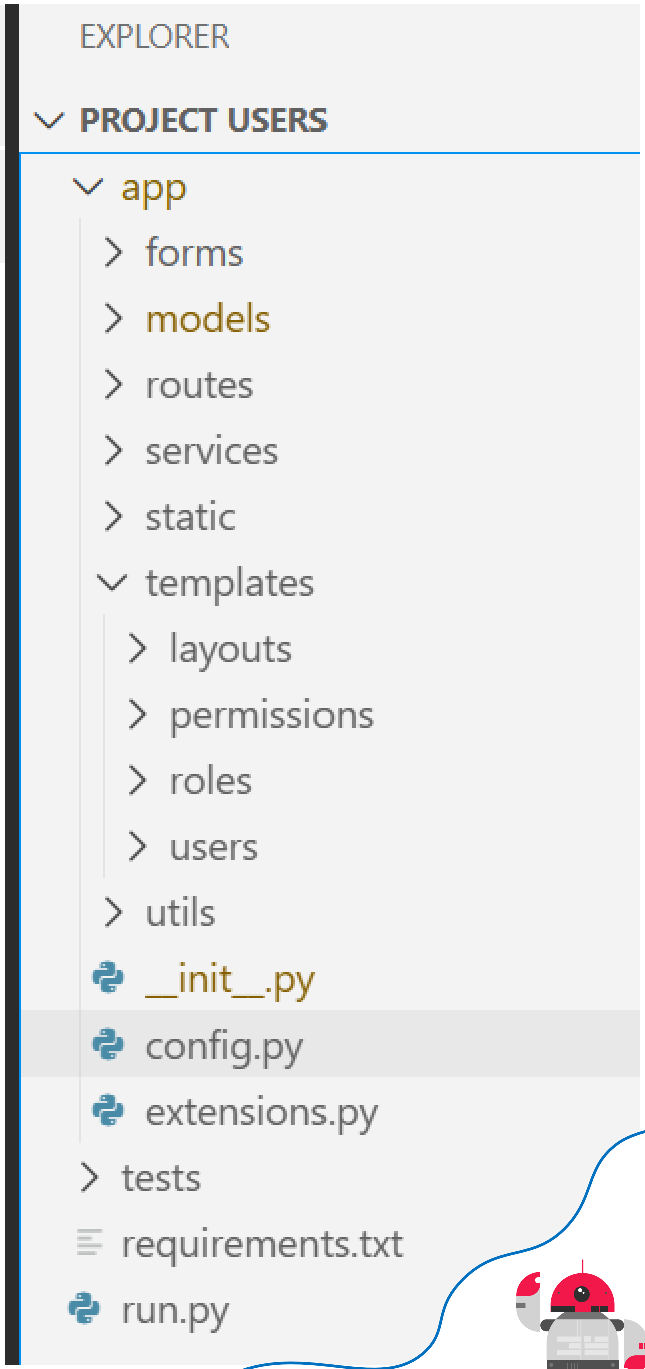
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

2 — Overview of the Project Structure & Communication

Structure = Communication

- Structure teaches developers
- Reduces confusion
- Makes teamwork easier
- Faster onboarding



```
EXPLORER
PROJECT USERS
  app
    forms
    models
    routes
    services
    static
    templates
      layouts
      permissions
      roles
      users
    utils
    __init__.py
    config.py
    extensions.py
  tests
  requirements.txt
  run.py
```

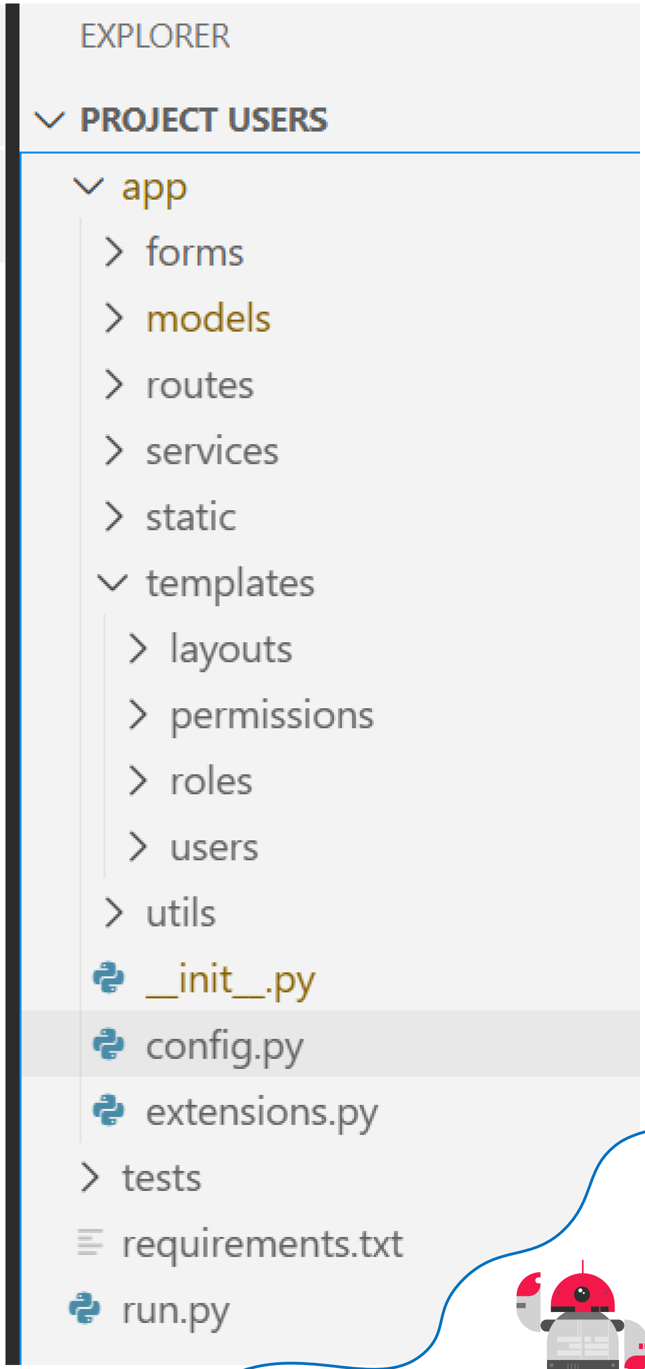
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

3 — Separation of Concerns (SOC)

3 — Separation of Concerns (SOC)

What is SOC?

SOC is vital in expert systems because mixing rules and interface can lead to incorrect reasoning or unpredictable behavior.

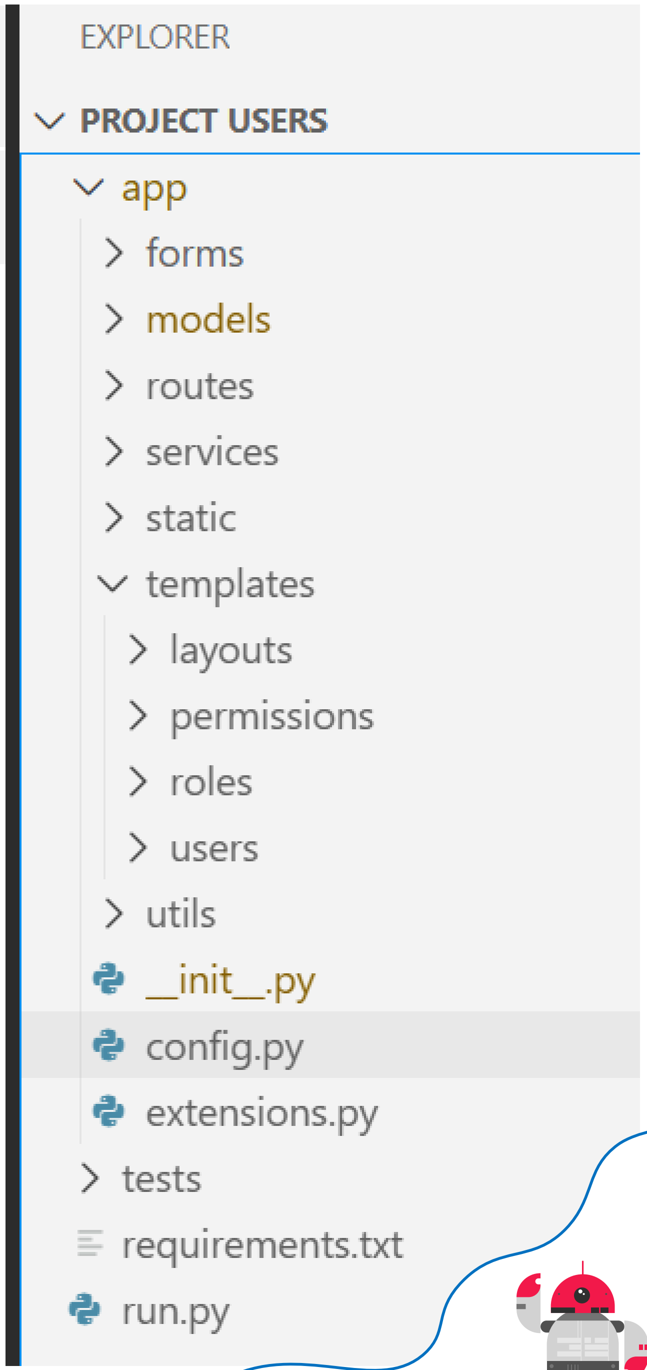

```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

3 — Separation of Concerns (SOC)

Why SOC Matters

- Each folder has one job
- Prevents mixing logic/UI/data
- Cleaner, safer code
- Easy to maintain



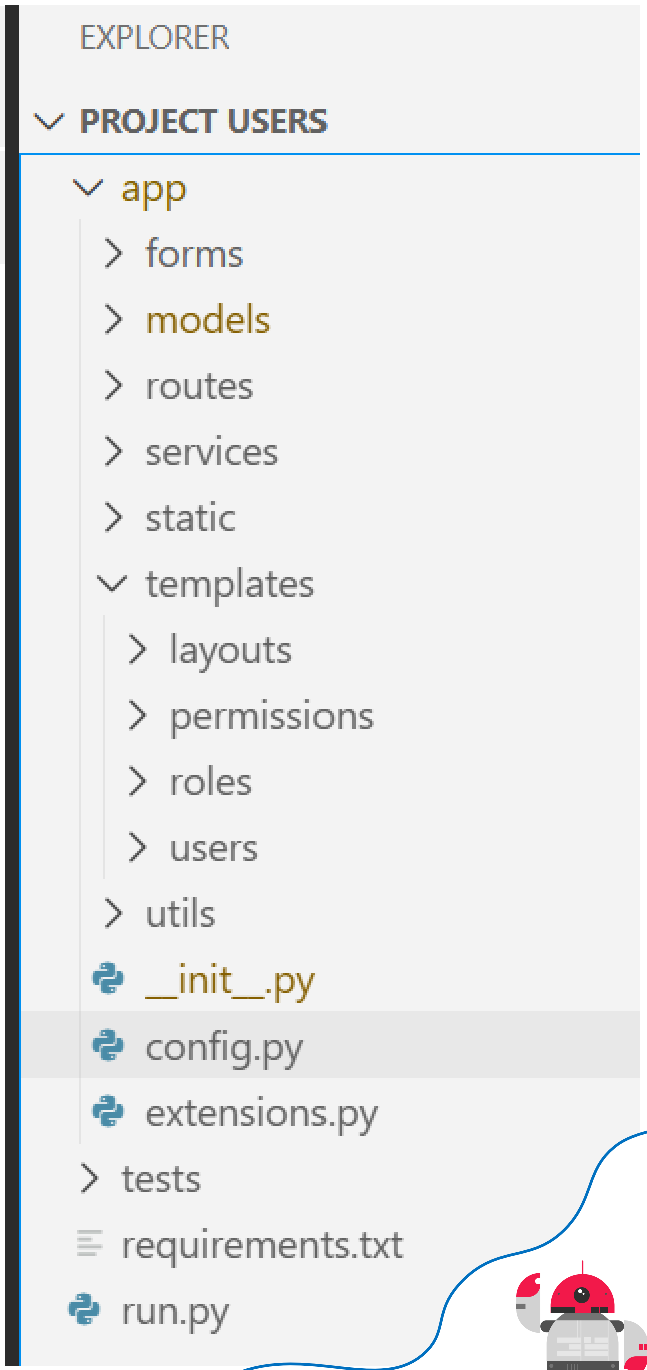
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

3 — Separation of Concerns (SOC)

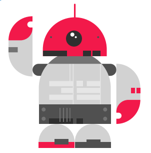
Example: Bad Structure

- Everything in app.py
- Hard to read
- Hard to test
- Easily breaks



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

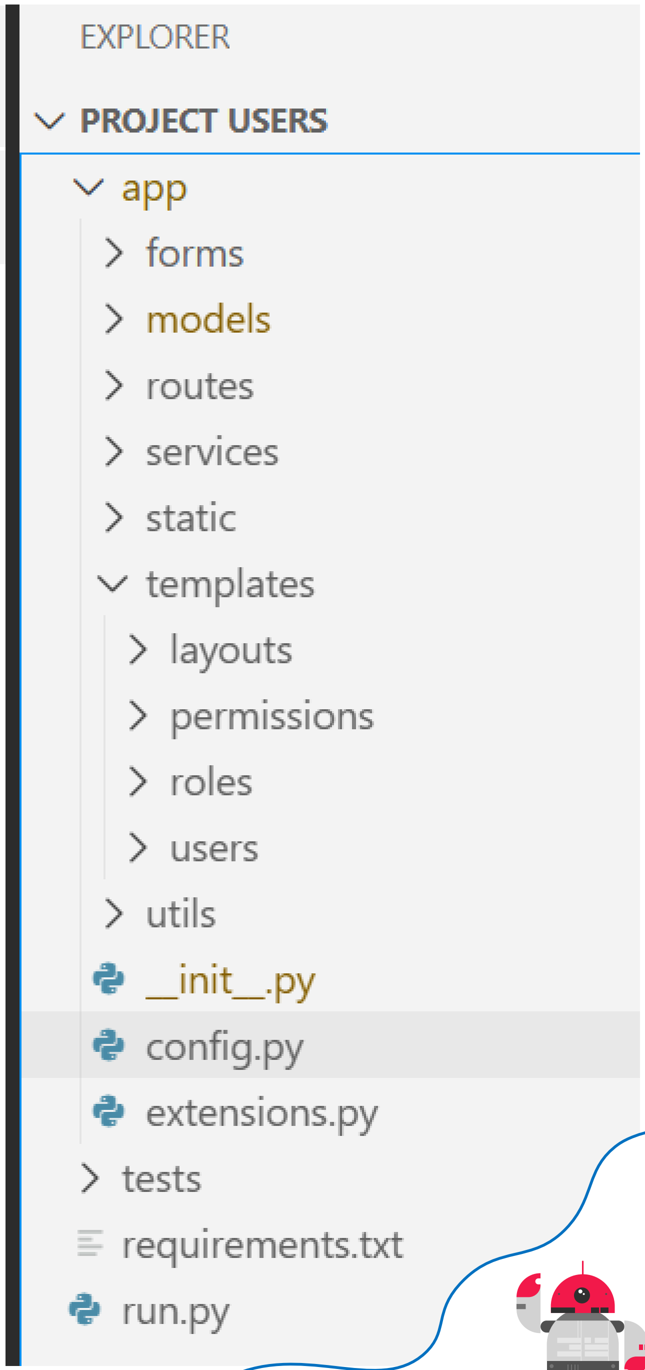
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY



3 — Separation of Concerns (SOC)

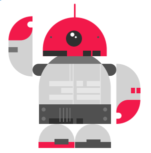
Example: Good Structure

- Clear folders
- Easy updates
- Safe changes
- Professional quality



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY



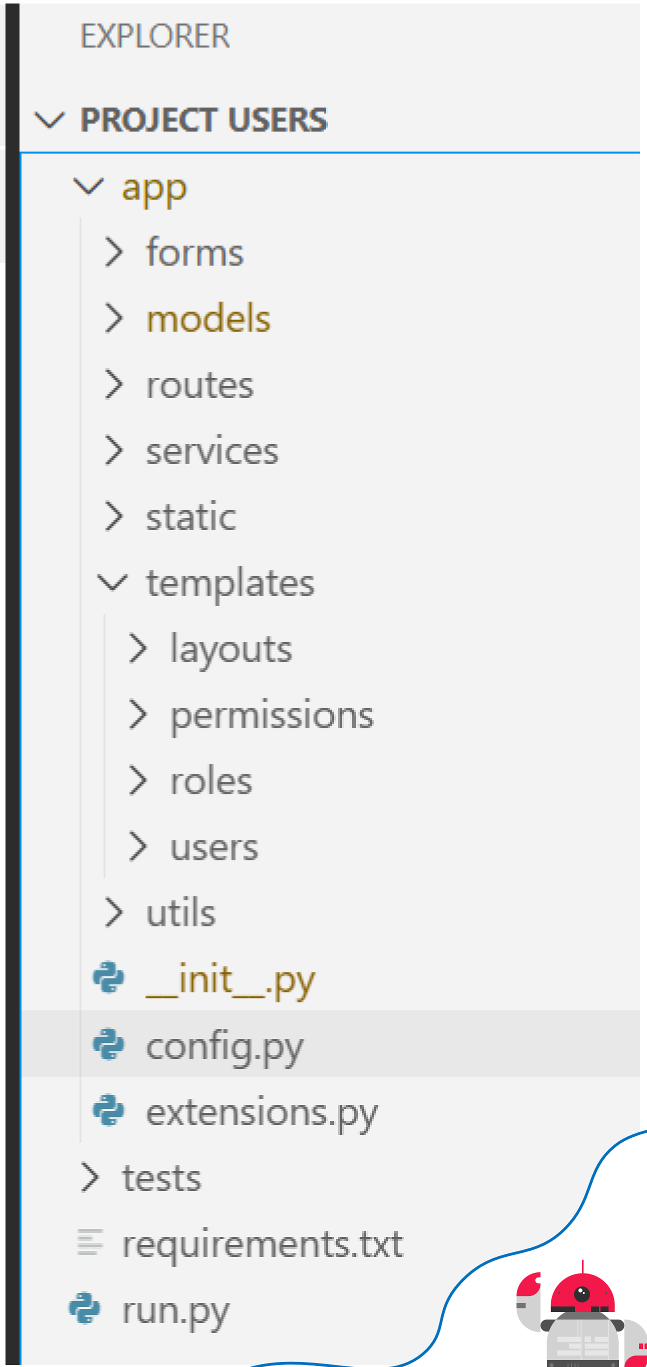
3 — Separation of Concerns (SOC)

Example: Good Structure

Separation of Concerns means each folder has one job.

This prevents mixing:

- Knowledge of UI
- Reasoning with templates
- Database code with business rules



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
      __init__.py
      config.py
      extensions.py
  > tests
    requirements.txt
    run.py
```

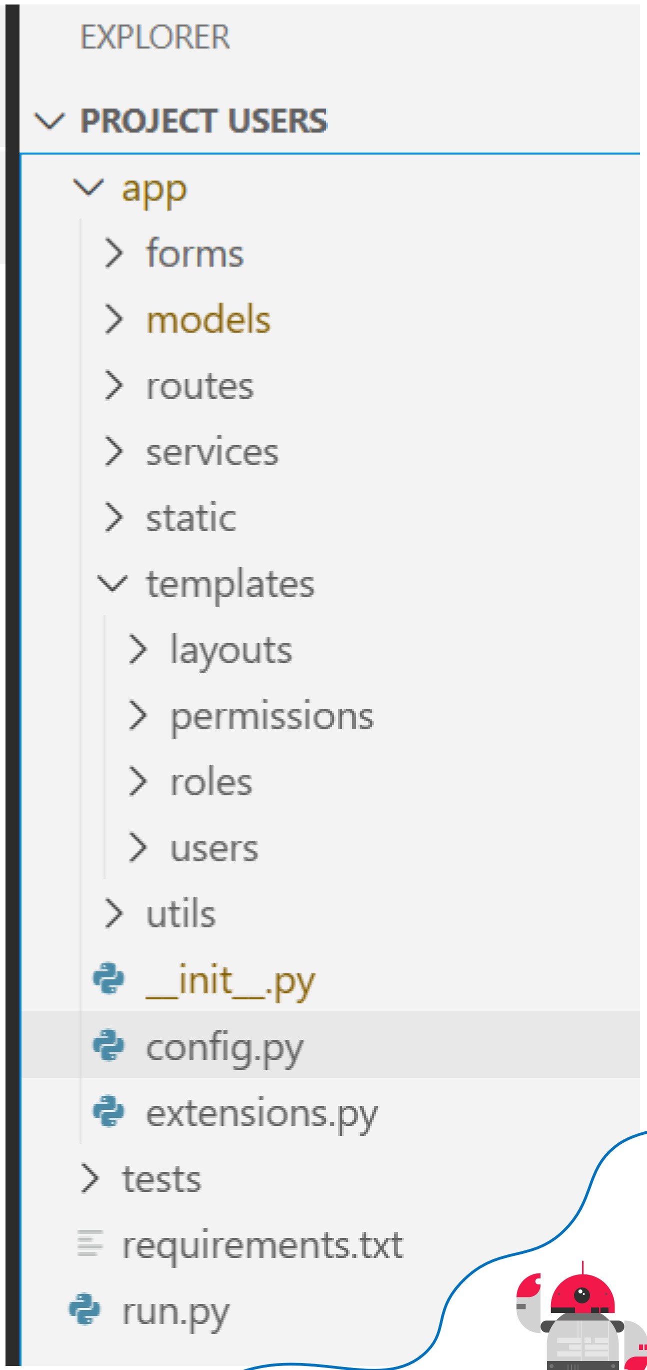
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

3 — Separation of Concerns (SOC)

Example: Good Structure

Result:

- Safer changes
- Cleaner logic
- Fewer bugs
- Clear flow from user → knowledge → reasoning → output



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

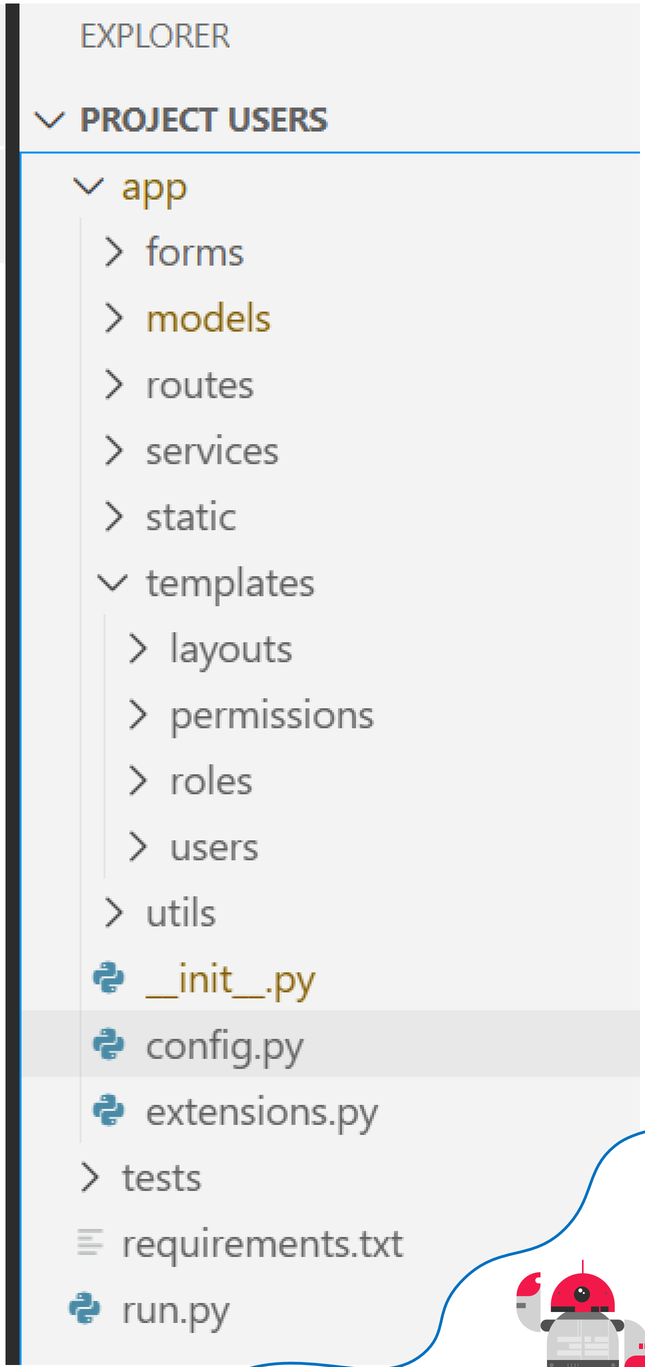
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

4 — Expert System Growth & Scalability

4 — Expert System Growth & Scalability

Scalability

- Projects grow
- More features
- More rules
- Structure supports expansion



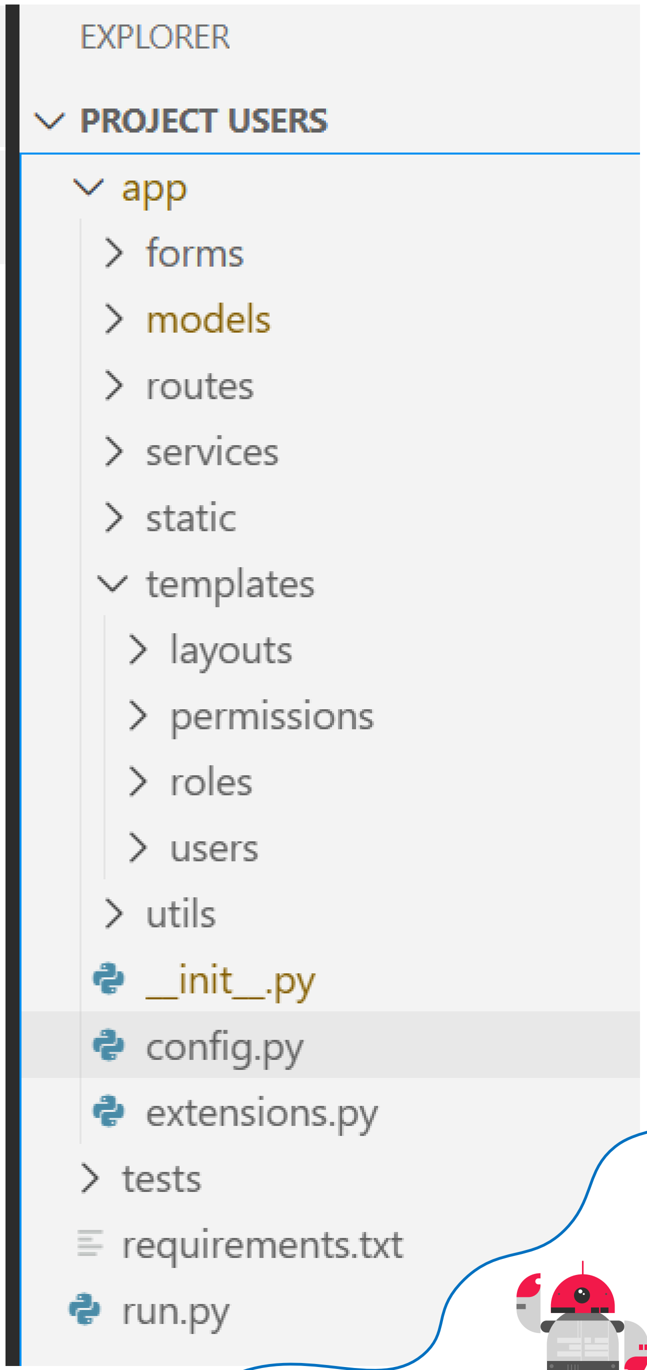
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

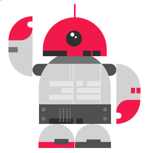
4 — Expert System Growth & Scalability

Expert systems grow quickly:

- More rules
- More facts
- More user roles
- More decisions
- More modules



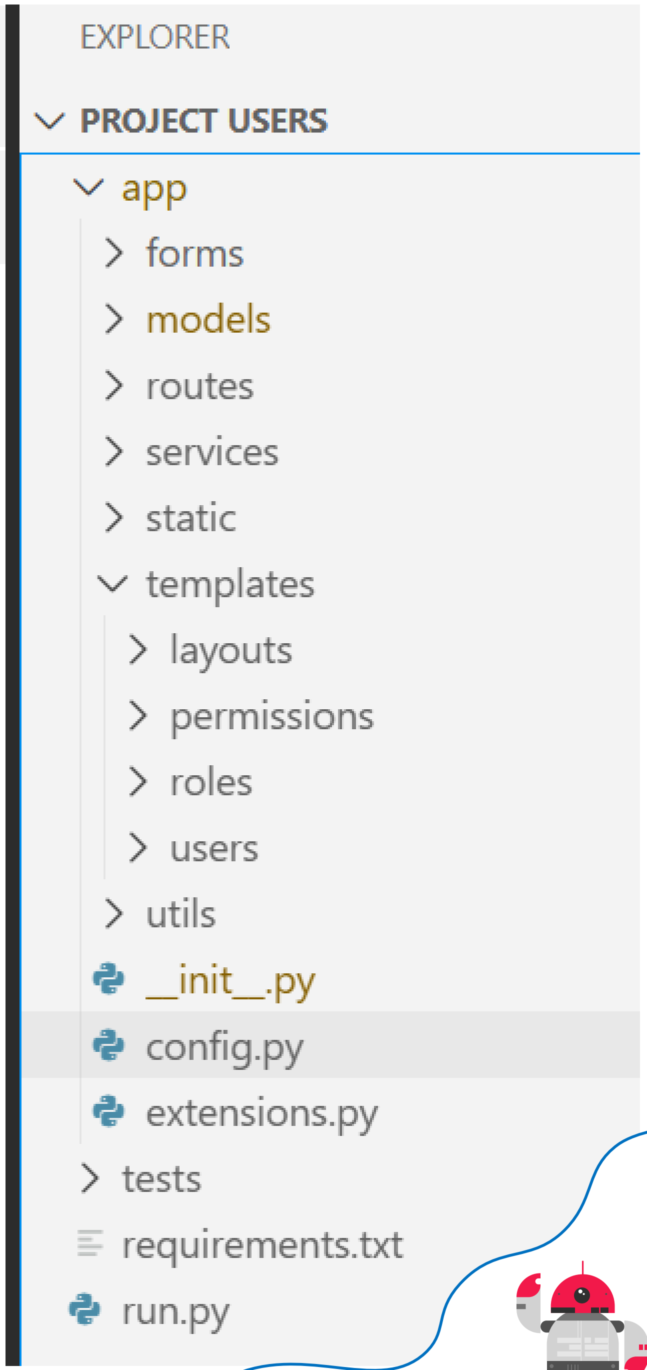
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```



4 — Expert System Growth & Scalability

Without structure:

- Files become huge
- Errors increase
- Updates break things
- Circular imports appear
- Students/developers get lost



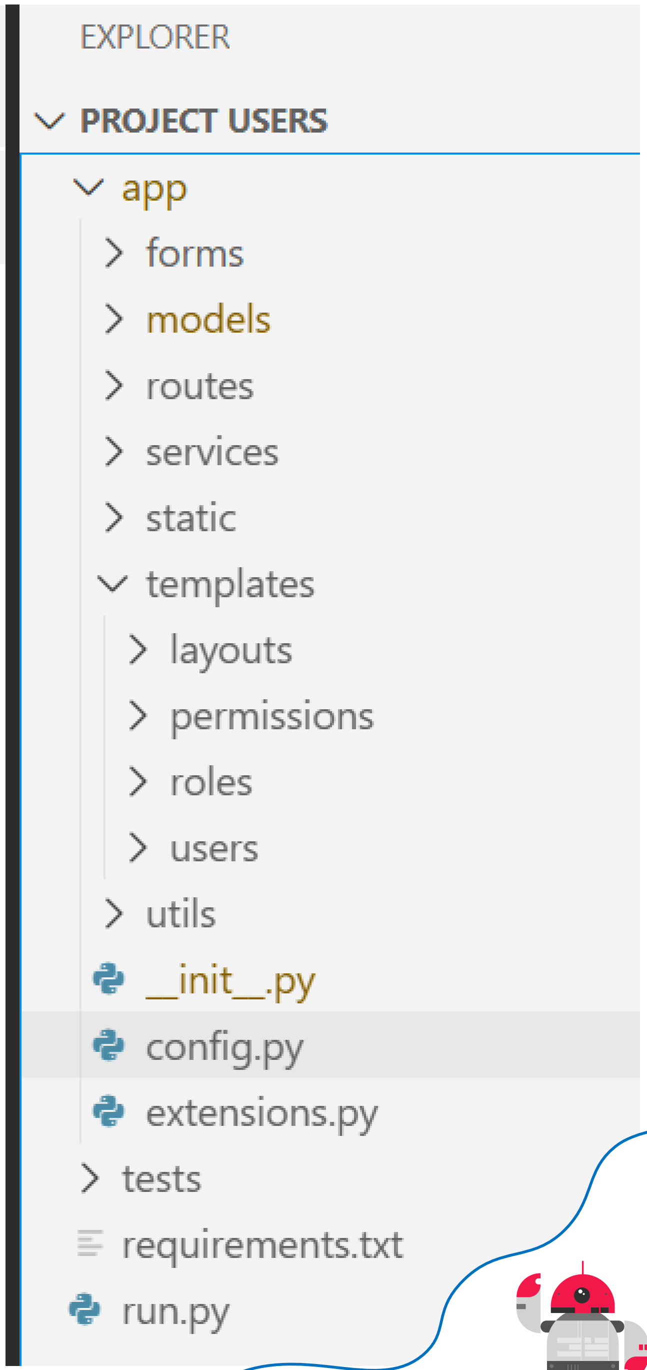
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន NORTON UNIVERSITY

4 — Expert System Growth & Scalability

With structure:

- Adding new rules is easy
- Adding new entities is safe
- UI, logic, and data stay separate
- System can scale without breaking



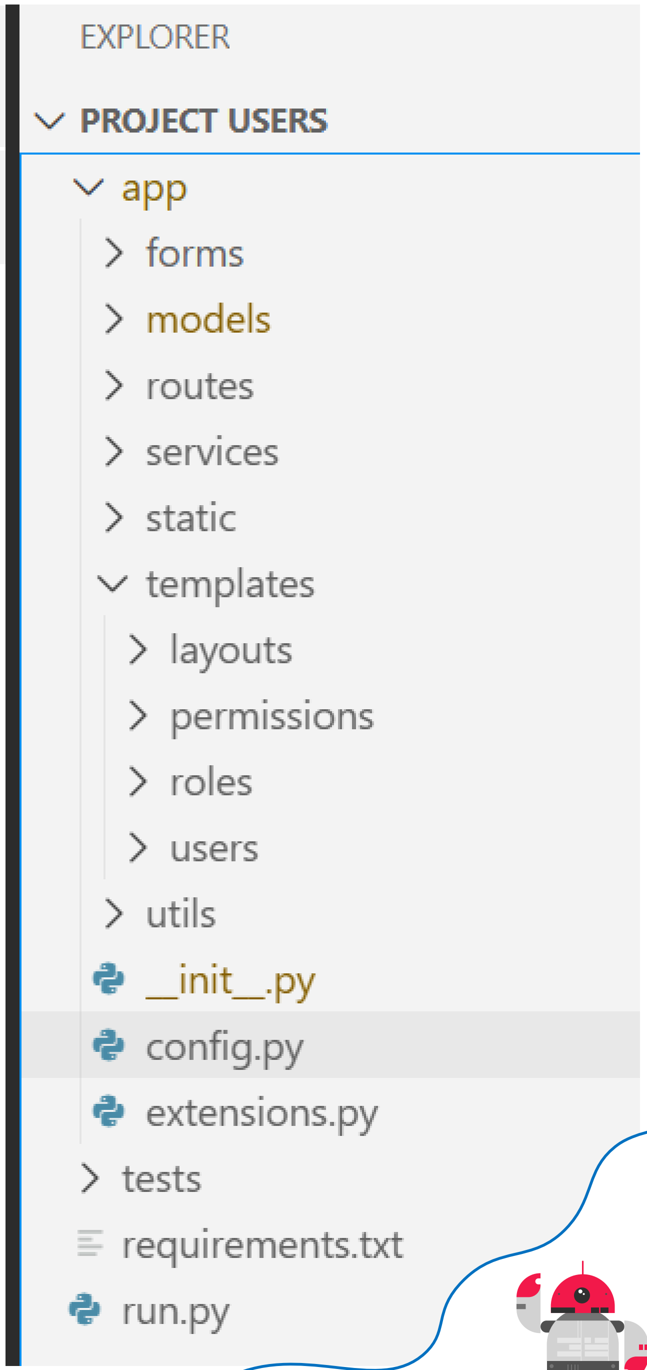
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

5 — Core Expert System Components (models, services, routes, templates, static)

5 — Core Expert System Components

services/ — Inference Engine

- Business rules
- Decision logic
- Assigning roles
- Permission checks



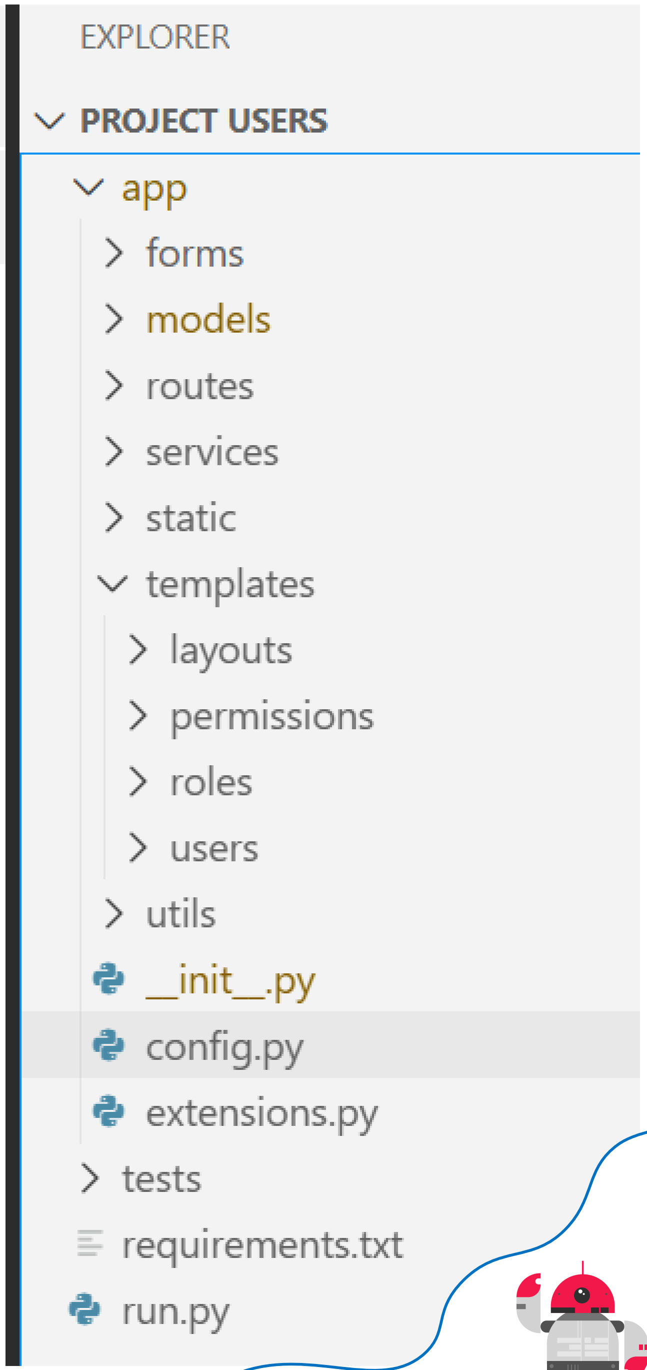
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

5 — Core Expert System Components

Why Logic Goes in services/

- Templates should not have logic
- Routes should not apply rules
- Clean reasoning layer
- Supports testing



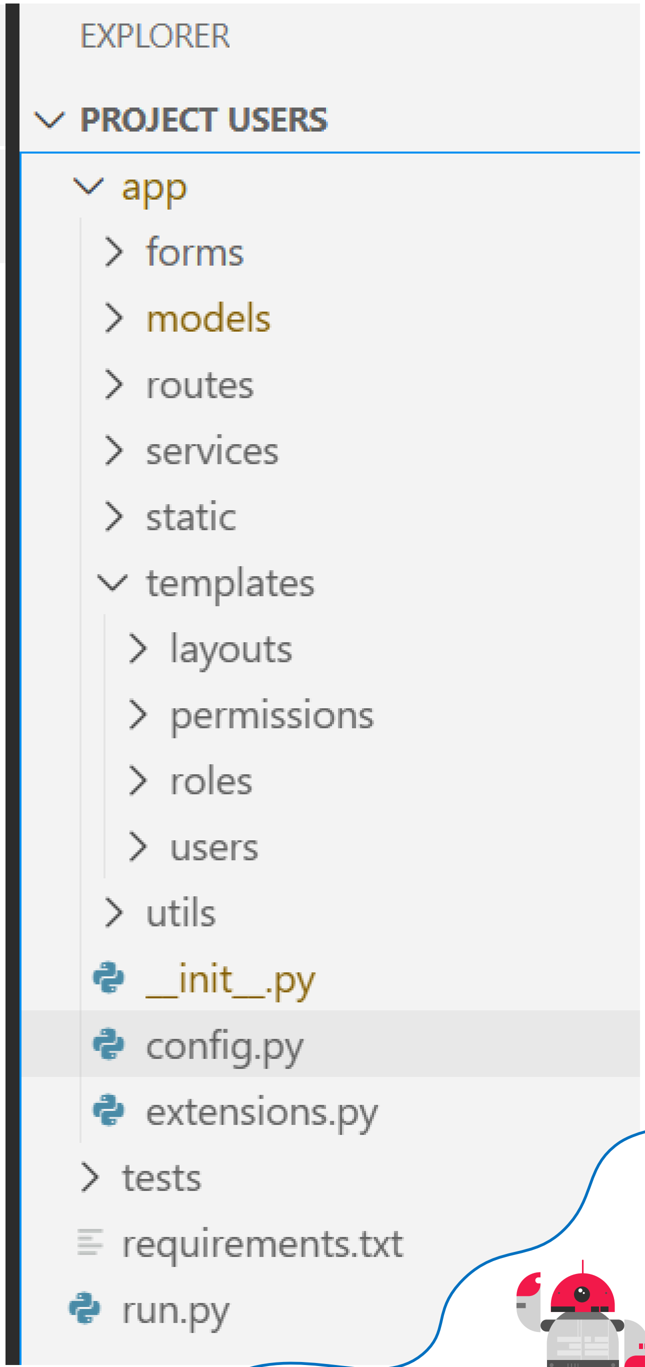
```
EXPLORER
PROJECT USERS
  app
    forms
    models
    routes
    services
    static
    templates
      layouts
      permissions
      roles
      users
    utils
    __init__.py
    config.py
    extensions.py
  tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

5 — Core Expert System Components

routes/ — Consultation Interface

- User asks questions
- System answers
- Bridges UI + logic
- Routes stay thin



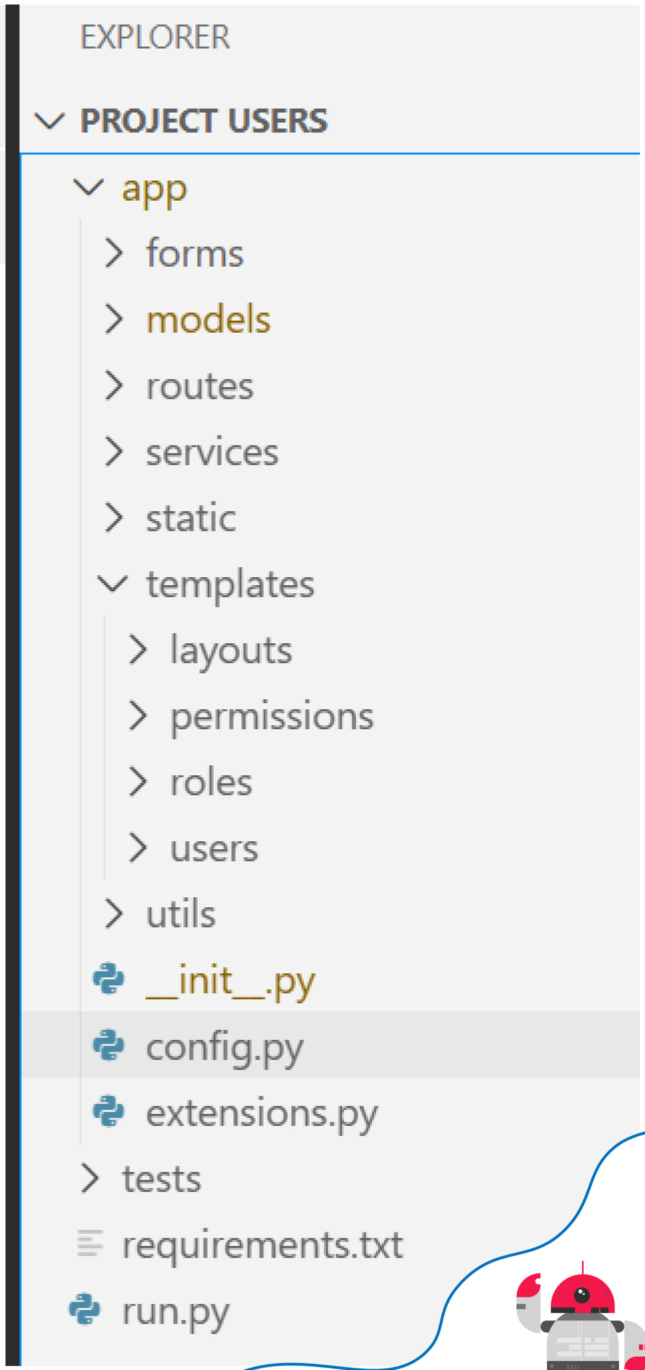
```
EXPLORER
PROJECT USERS
  app
    forms
    models
    routes
    services
    static
    templates
      layouts
      permissions
      roles
      users
    utils
    __init__.py
    config.py
    extensions.py
  tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

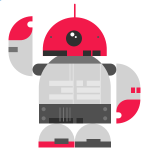
5 — Core Expert System Components

templates_custom/ — Presentation Layer

- HTML views
- Organized by domain
- Custom folder for modularity
- Clear mapping to routes/services



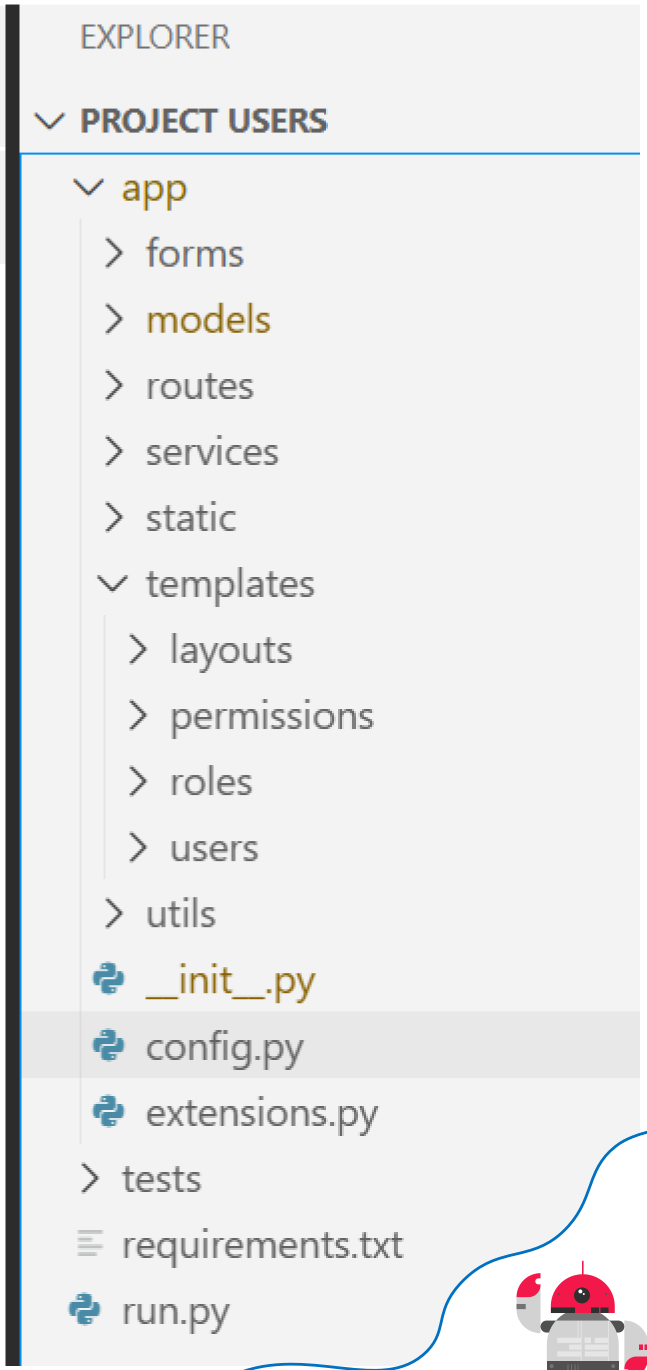
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```



5 — Core Expert System Components

static/ & forms/

- CSS/JS support user interaction
- Forms validate input
- Stops bad facts from entering the system



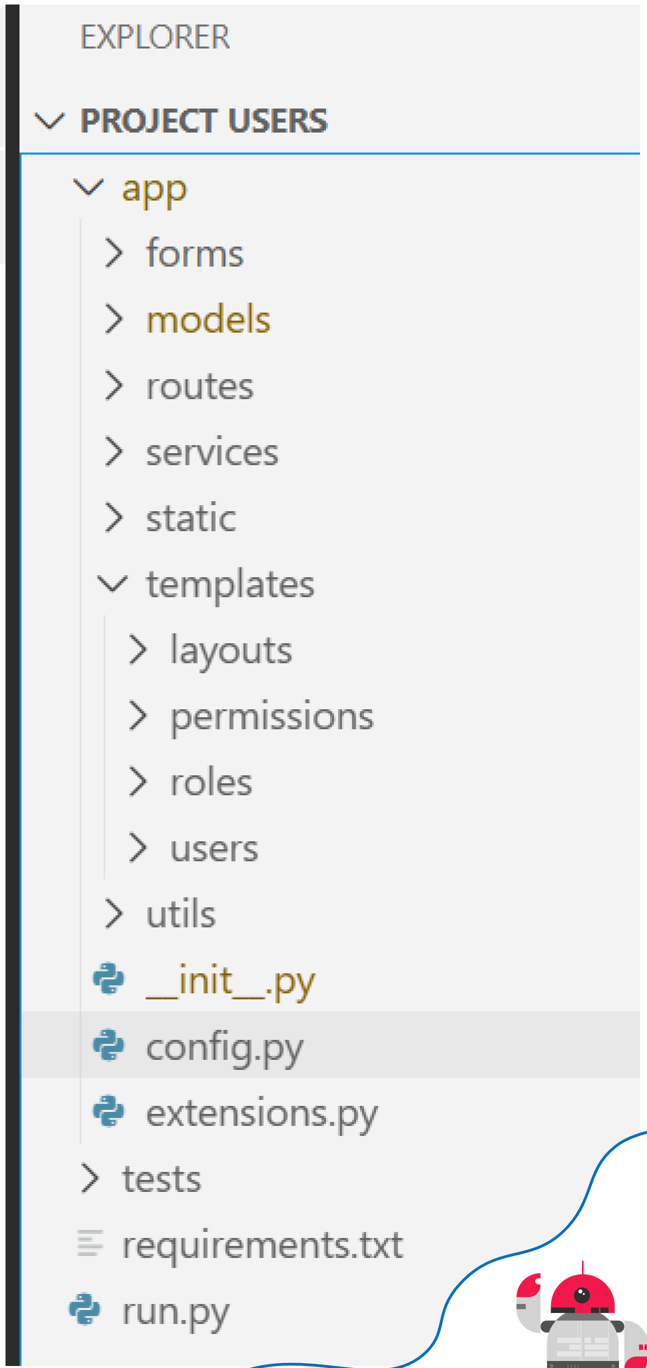
```
EXPLORER
PROJECT USERS
  app
    forms
    models
    routes
    services
    static
    templates
      layouts
      permissions
      roles
      users
    utils
    __init__.py
    config.py
    extensions.py
  tests
  requirements.txt
  run.py
```

 សាកលវិទ្យាល័យន័រតុន NORTON UNIVERSITY

5 — Core Expert System Components

utils/

- Reusable helper functions
- Decorators
- Small reasoning helpers



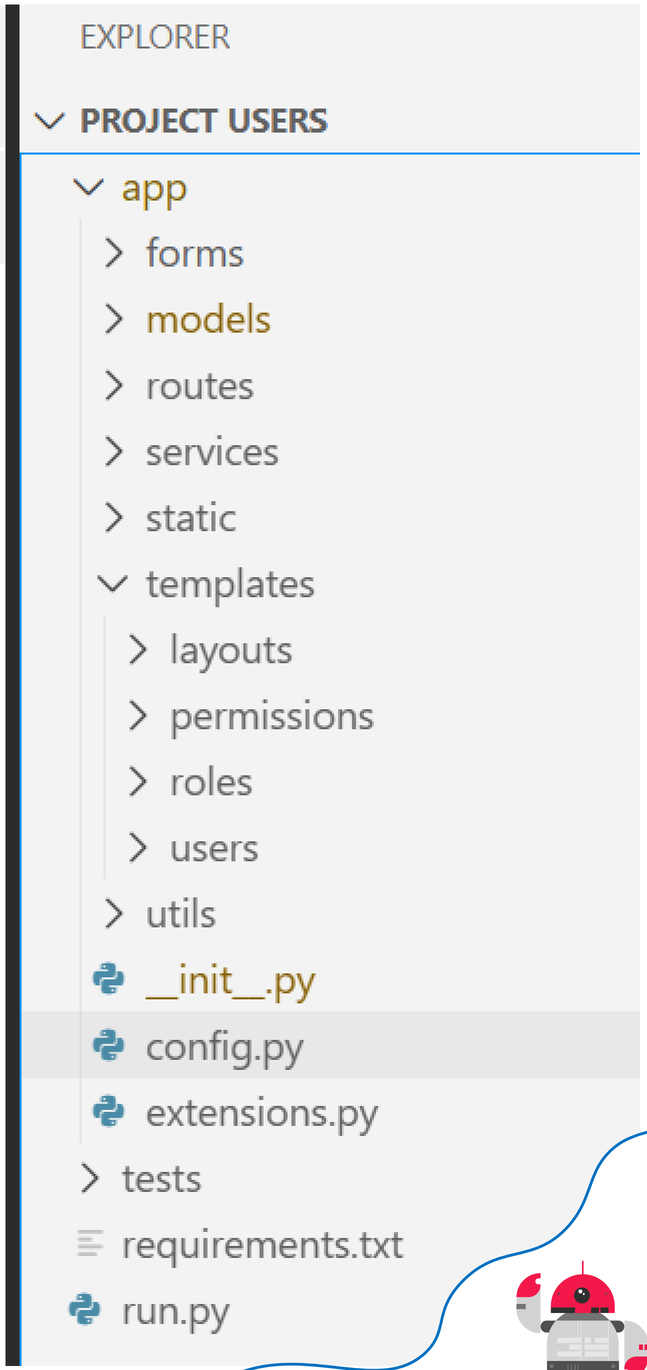
EXPLORER
PROJECT USERS

```
app
  forms
  models
  routes
  services
  static
  templates
    layouts
    permissions
    roles
    users
  utils
  __init__.py
  config.py
  extensions.py
tests
requirements.txt
run.py
```

5 — Core Expert System Components

instance/

- Holds database
- Environment specific
- Not in Git
- Protects data

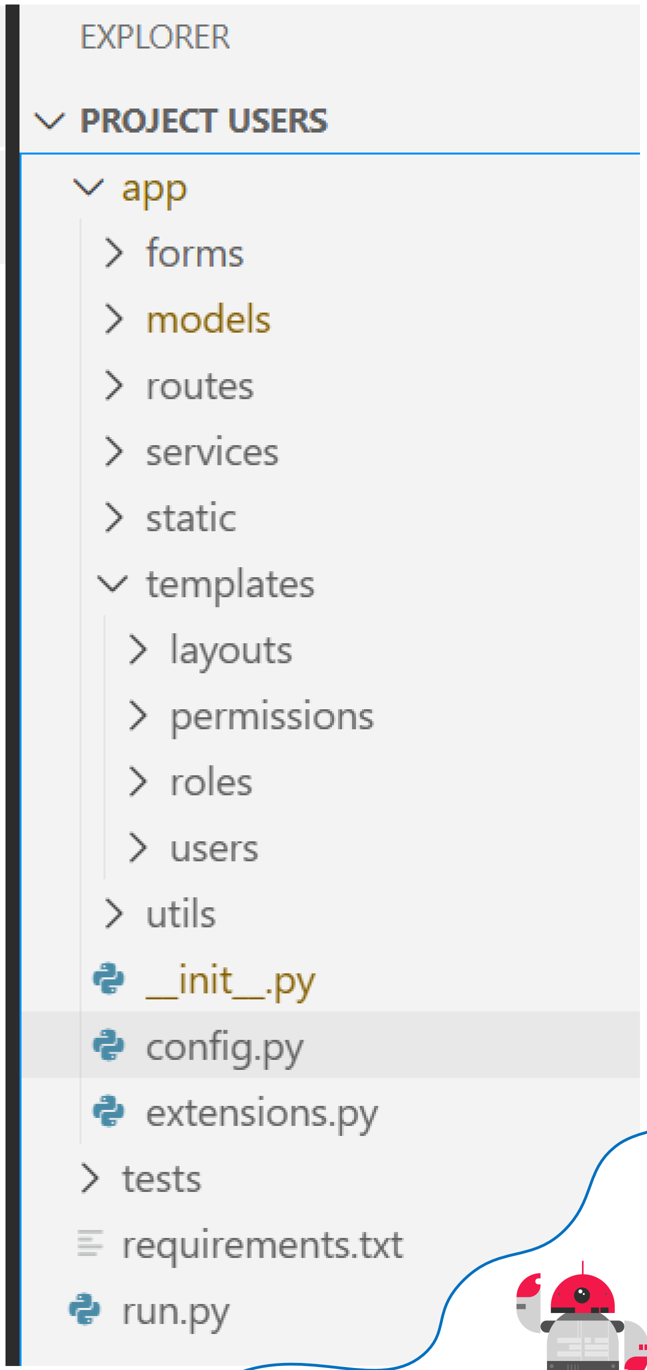


```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

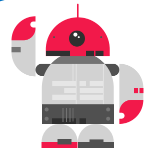
5 — Core Expert System Components

migrations/

- Tracks DB changes
- Supports schema evolution
- Essential for real systems



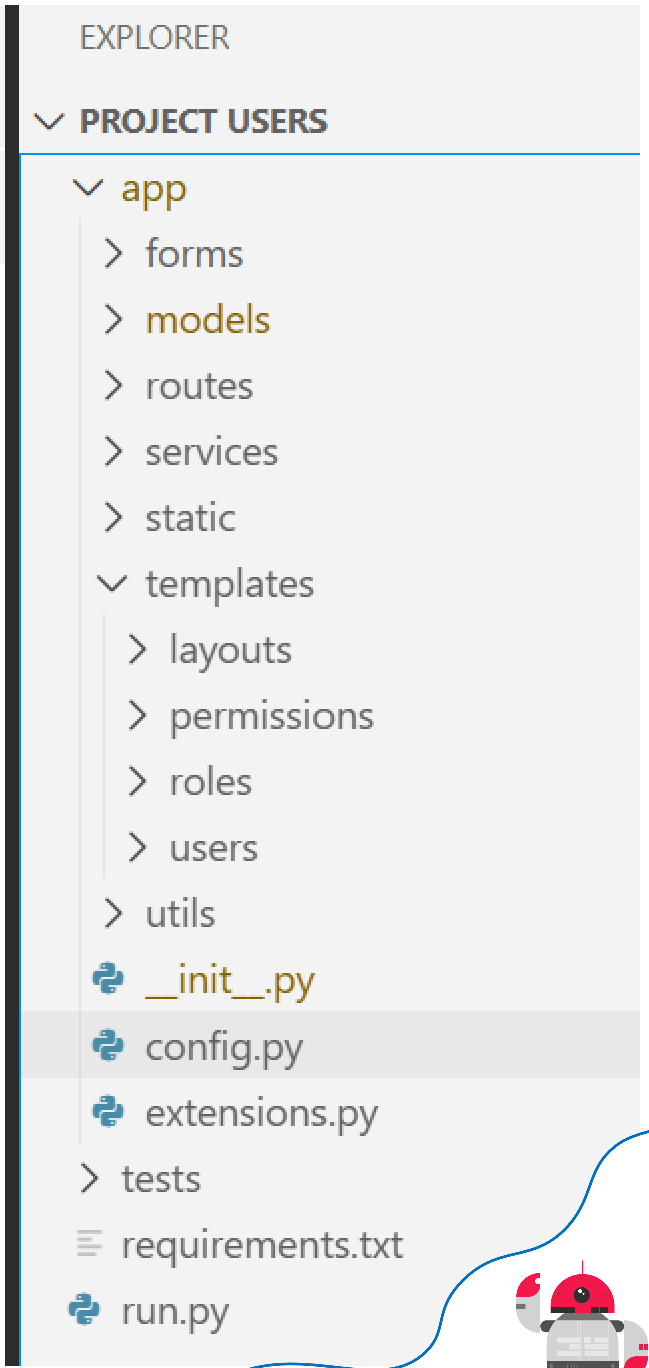
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```



5 — Core Expert System Components

tests/

- Expert systems must be correct
- Test rule logic
- Test knowledge accuracy
- Prevent regressions



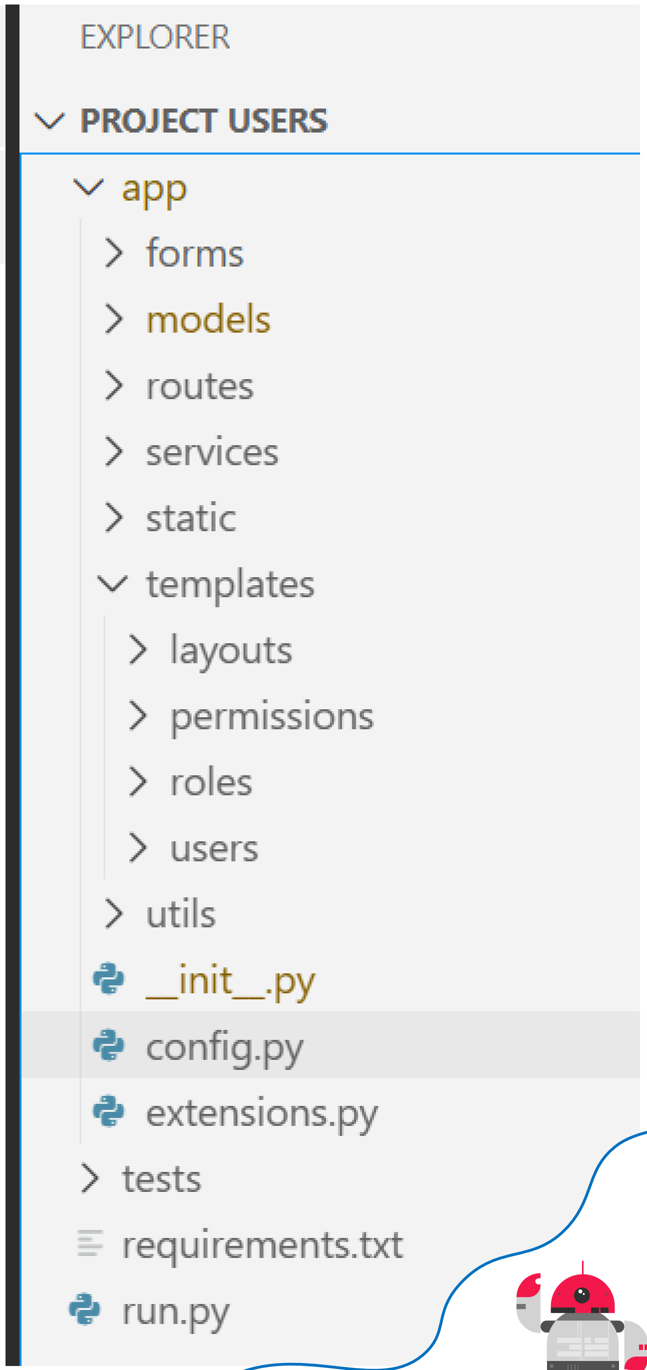
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

NORTON UNIVERSITY

5 — Core Expert System Components

config.py

- Central system settings
- DB URI
- Secret key
- Debug settings



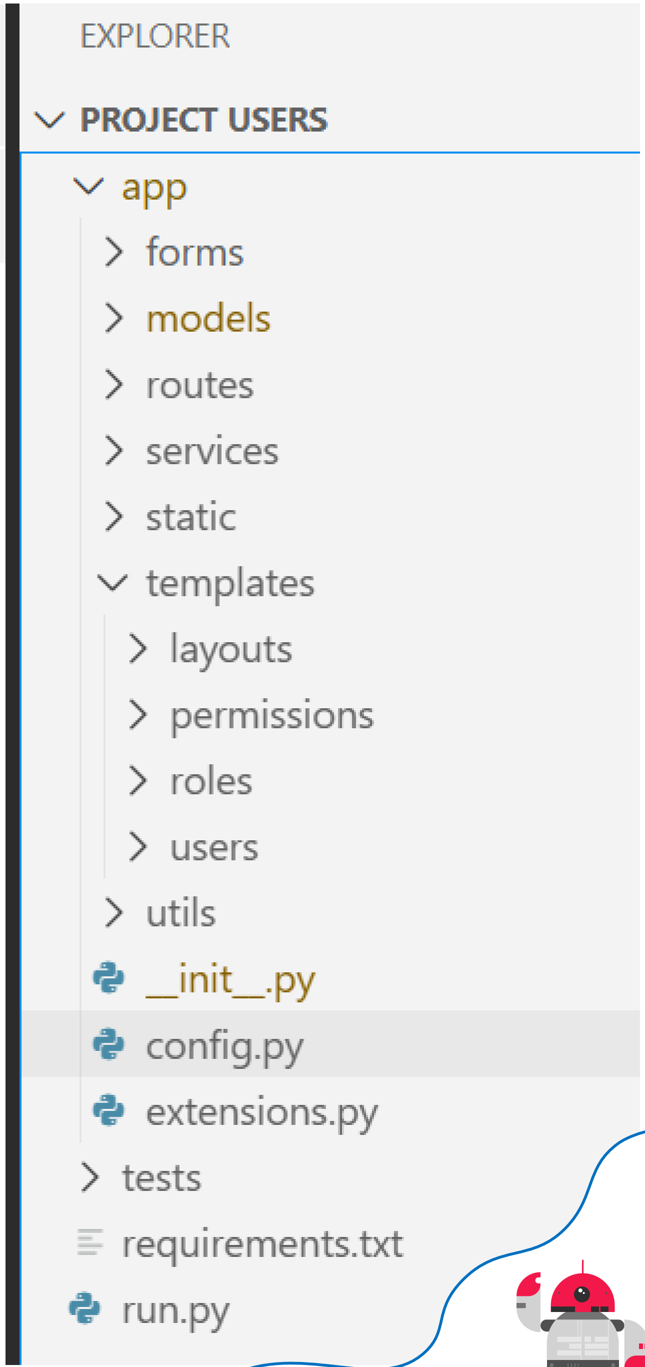
```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

5 — Core Expert System Components

extensions.py

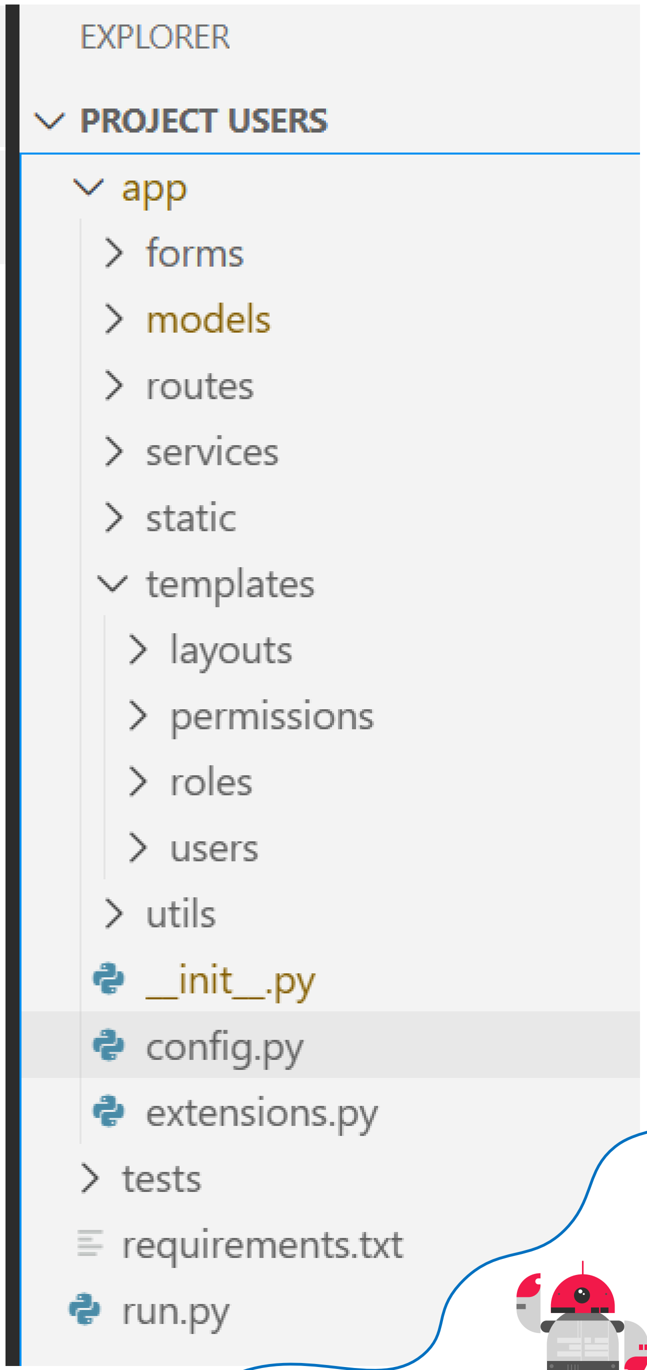
- Initializes SQLAlchemy
- Initializes Migrate
- Avoids circular imports



5 — Core Expert System Components

run.py & Requirements

- run.py starts the system
- requirements.txt = needed libraries
- README = how to use



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

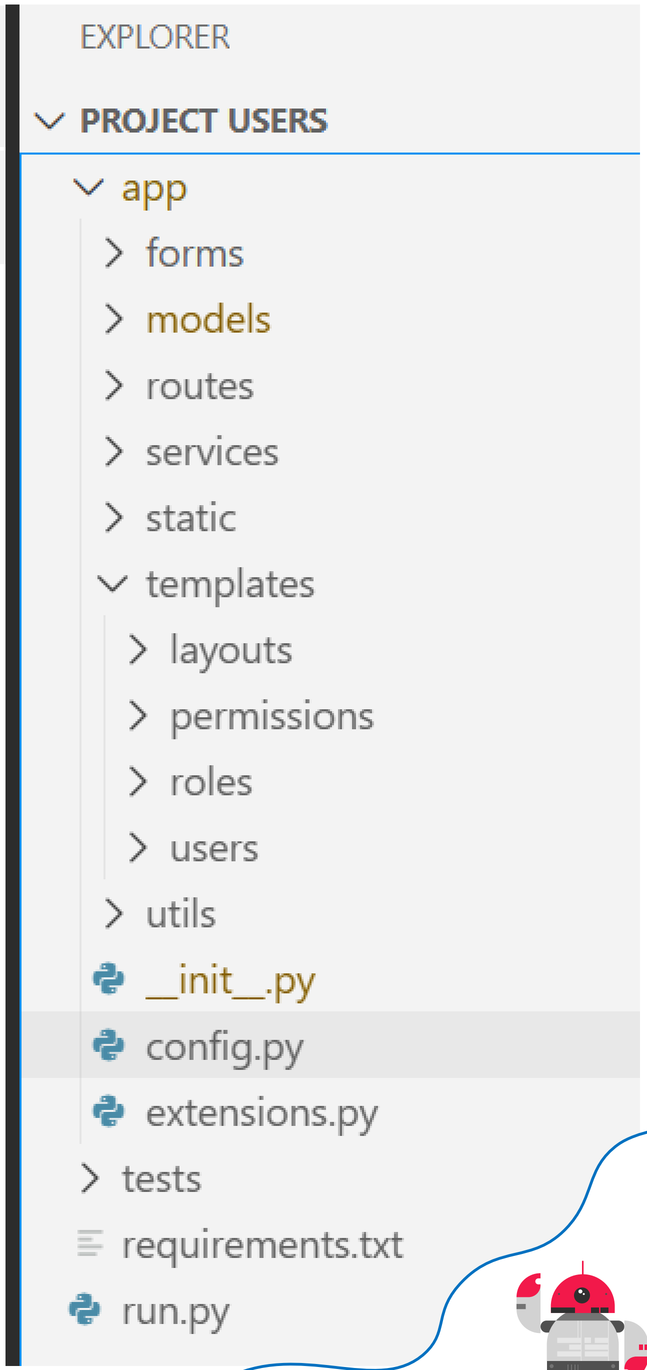
សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

6 — Support Systems & System Reliability

6 — Support Systems & System Reliability

Putting It All Together

- Expert system workflow
- Folder workflow
- Everything maps cleanly
- Easy to upgrade

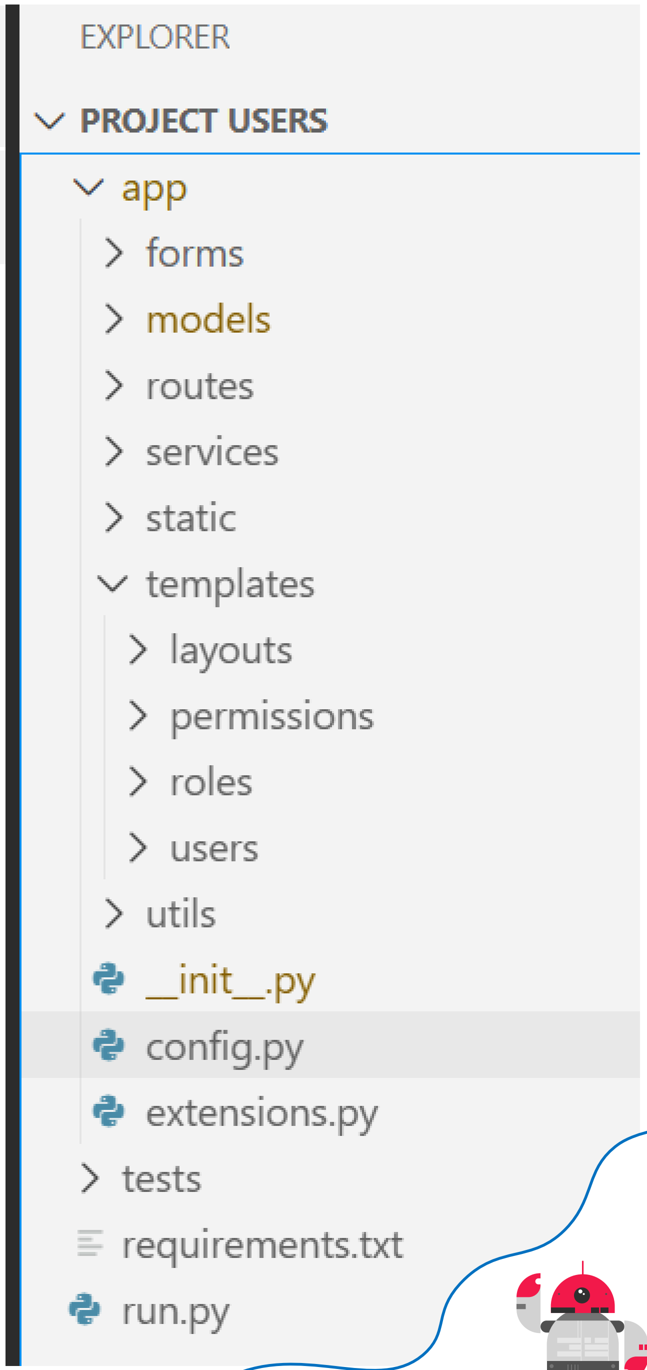


```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

Professional Benefits

- Team-friendly
- Future-proof
- Cleaner development
- Industry standard architecture

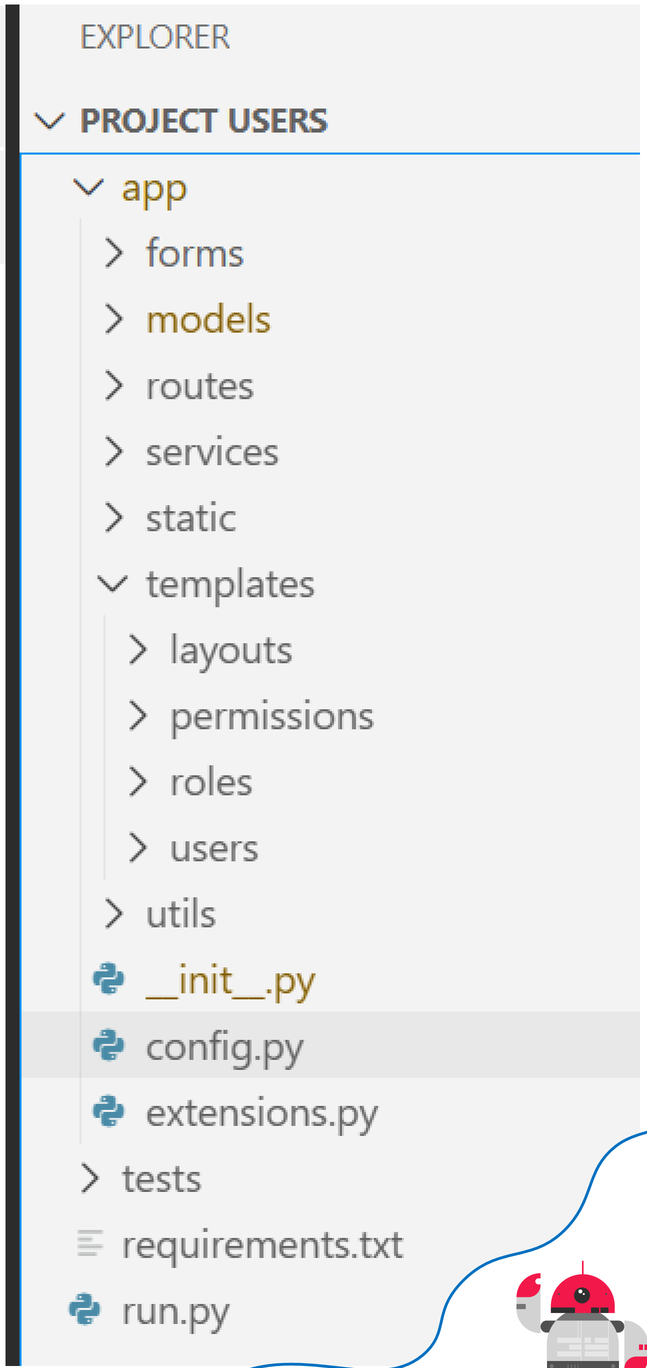


```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```

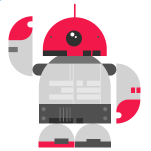
6 — Support Systems & System Reliability

Summary

- Structure is essential, not optional
- Makes expert systems reliable
- Makes development easier
- Builds real-world skills
- Questions & next steps



```
EXPLORER
∨ PROJECT USERS
  ∨ app
    > forms
    > models
    > routes
    > services
    > static
    ∨ templates
      > layouts
      > permissions
      > roles
      > users
    > utils
    __init__.py
    config.py
    extensions.py
  > tests
  requirements.txt
  run.py
```



Thank You

SEK SocheaT

socheatsek@norton-u.com