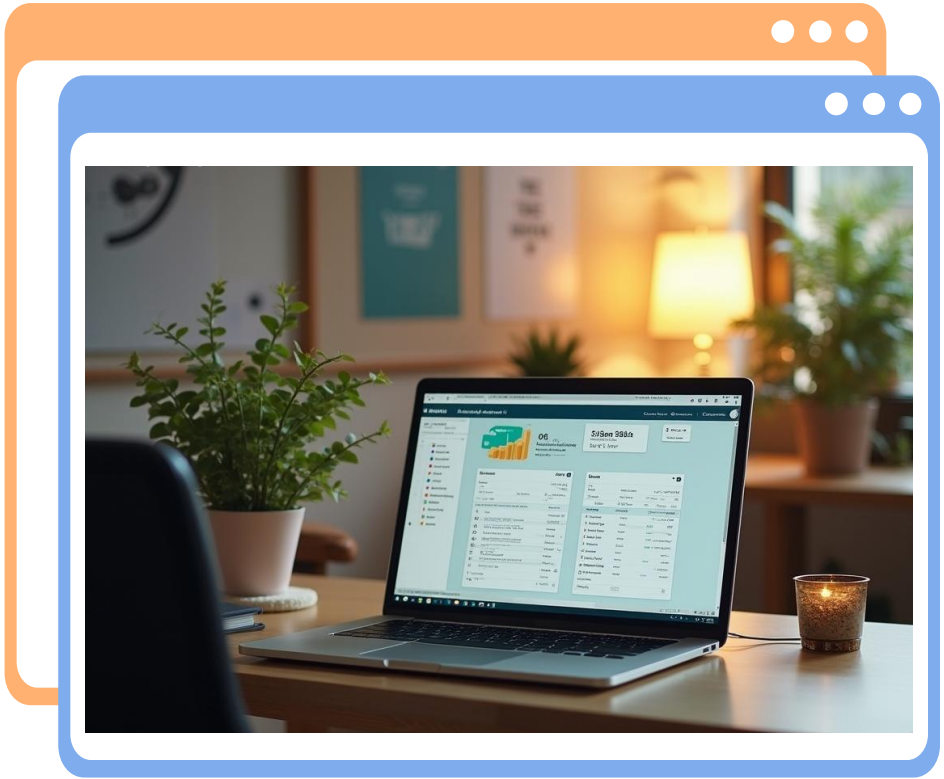
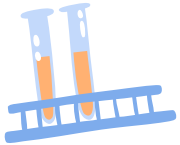
Finance Tracker

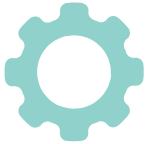
A Comprehensive Tool for Managing Personal Finances

Introduction



01 Overview





Purpose of Finance Tracker

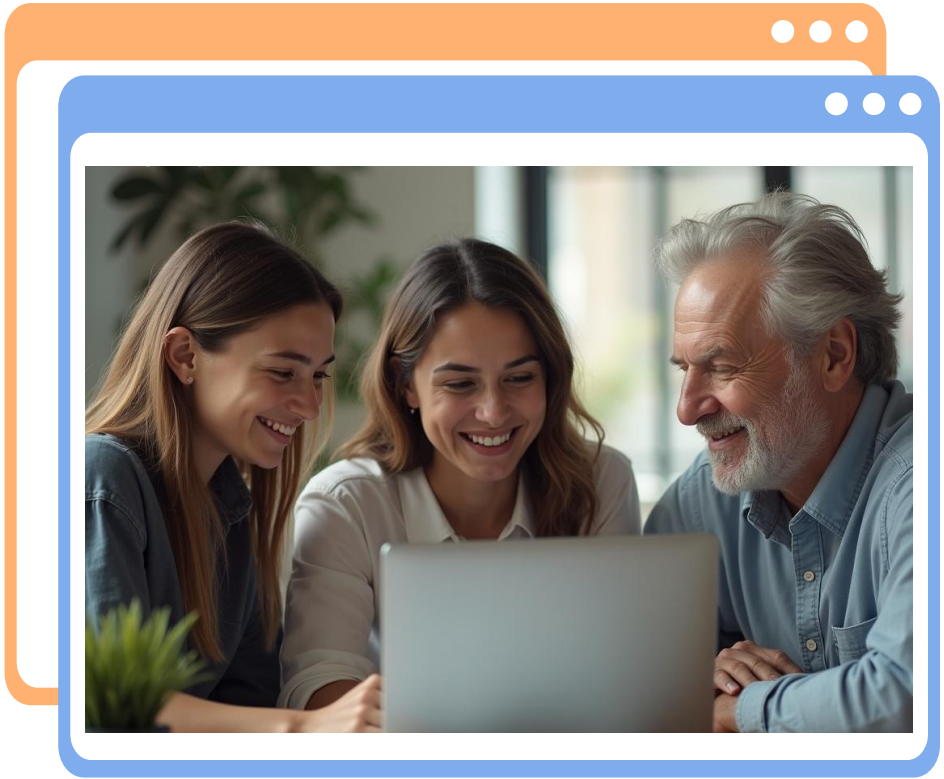
The Finance Tracker aims to empower users in managing their personal finances effectively. It provides tools to track income and expenses, helping users visualize their financial health and make informed decisions.

Core functionalities

Key functionalities include real-time tracking of expenses, income categorization, and financial health visualization through interactive dashboards. The application also supports secure user authentication and offers persistent data storage.

Target audience

The Finance Tracker is designed for individuals seeking effective financial management solutions. This includes students, professionals, and anyone looking to gain a clearer understanding of their financial situation.




02
Tech Stack

Frontend technologies

The frontend is developed using Streamlit and Altair, enabling the creation of responsive and interactive data visualizations. This combination allows users to engage with their financial data dynamically.

Backend architecture

The backend is powered by FastAPI and Pydantic. This modern setup ensures efficient API creation, data validation, and a modular architecture that promotes scalability and maintainability.

default

POST	/register	Register
GET	/metrics	Metrics
GET	/metrics/errors	Error Metrics
POST	/login	Login
POST	/records	Create Record
PUT	/records/{record_id}	Update Record
DELETE	/records/{record_id}	Delete Record
GET	/categories	Get Categories
POST	/categories	Create Category
GET	/dashboard	Get Dashboard
GET	/report	Get Report
GET	/profile	Get Profile
PUT	/profile	Update Profile

Database and authentication

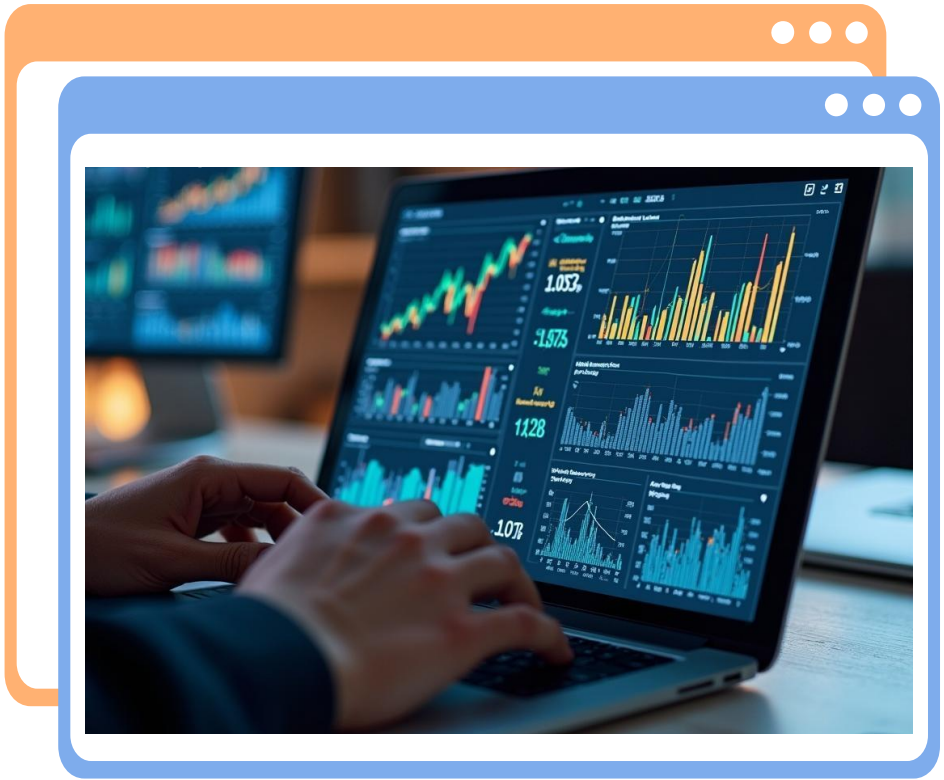
Database management is facilitated by SQLAlchemy with SQLite as the underlying database. Authentication is securely handled using Python-JOSE and Passlib with bcrypt, ensuring user data protection.

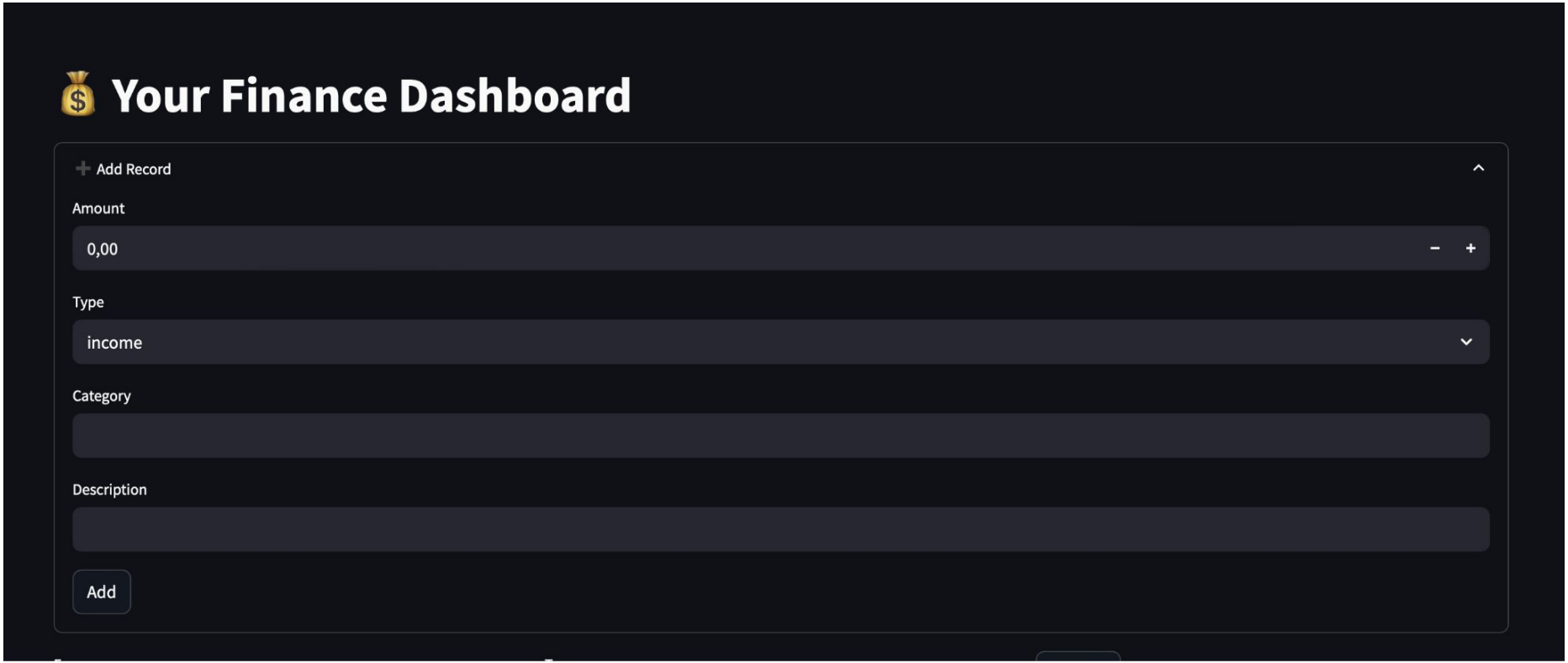
03

Features

Interactive dashboards

The Finance Tracker features interactive dashboards that allow users to visualize their financial data intuitively. Leveraging Streamlit and Altair, users can interact with charts and graphs to track their spending patterns, income sources, and financial trends, fostering a proactive approach to personal finance management.




Your Finance Dashboard
Add Record
Amount
0,00
Type
income
Category
Description
Add

Deploy

From

YYYY/MM/DD

To

YYYY/MM/DD

Refresh

Total Income

💚 100.0

Total Expense

❤️ 20.0

Balance

💰 80.0

Income vs Expense

Records

income • Salary • 100.0 (2025-05-08)

Work hard too

Amount

100,00

Type

income

Category

Salary

Description

Work hard too

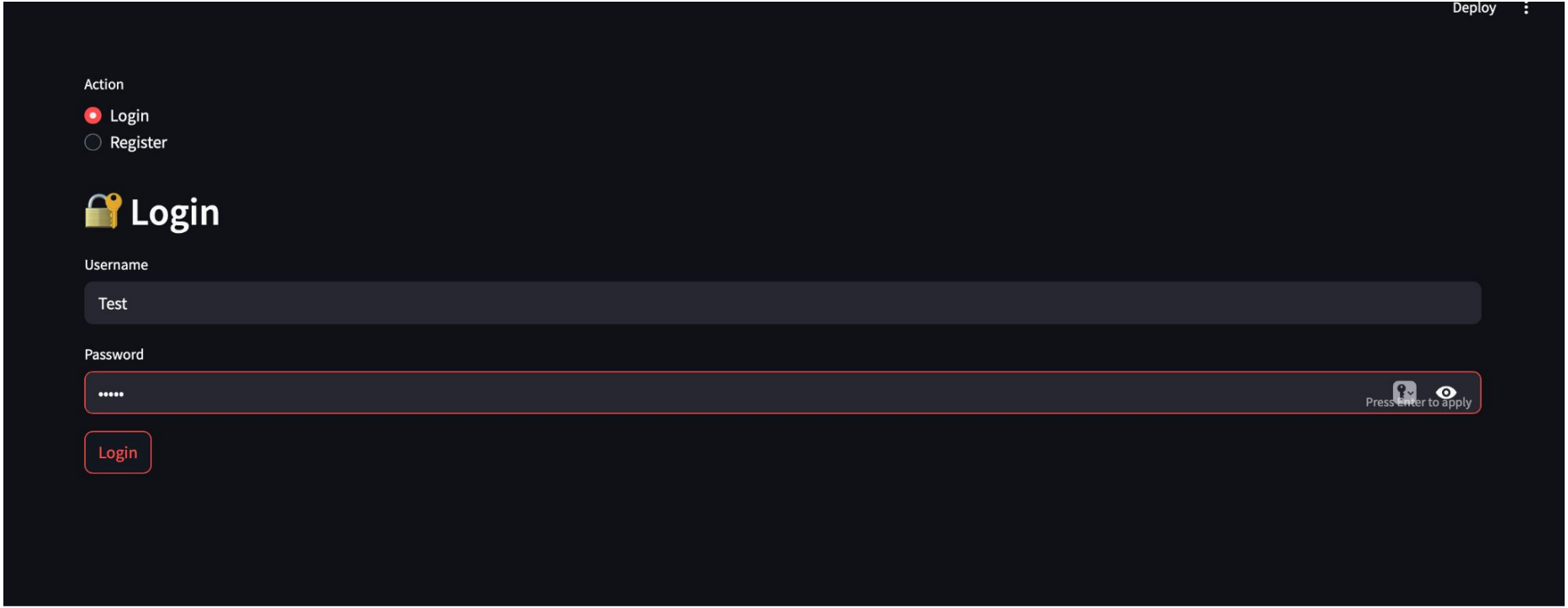
Save

Delete

expense • Flat • 20.0 (2025-05-08)

Data persistence and security

Data persistence is ensured through SQLAlchemy, with SQLite providing a reliable database solution. This setup allows for secure storage of user financial data. Additionally, security measures such as JWT authentication and encryption protocols are implemented to protect user information and ensure safe access to the application.


Deploy
Action
Login
Register
Login
Username
Test
Password
Press Enter to apply
Login


Action
Login
Register
Register
Username
Email
Password
Confirm
Register

Testing and monitoring capabilities

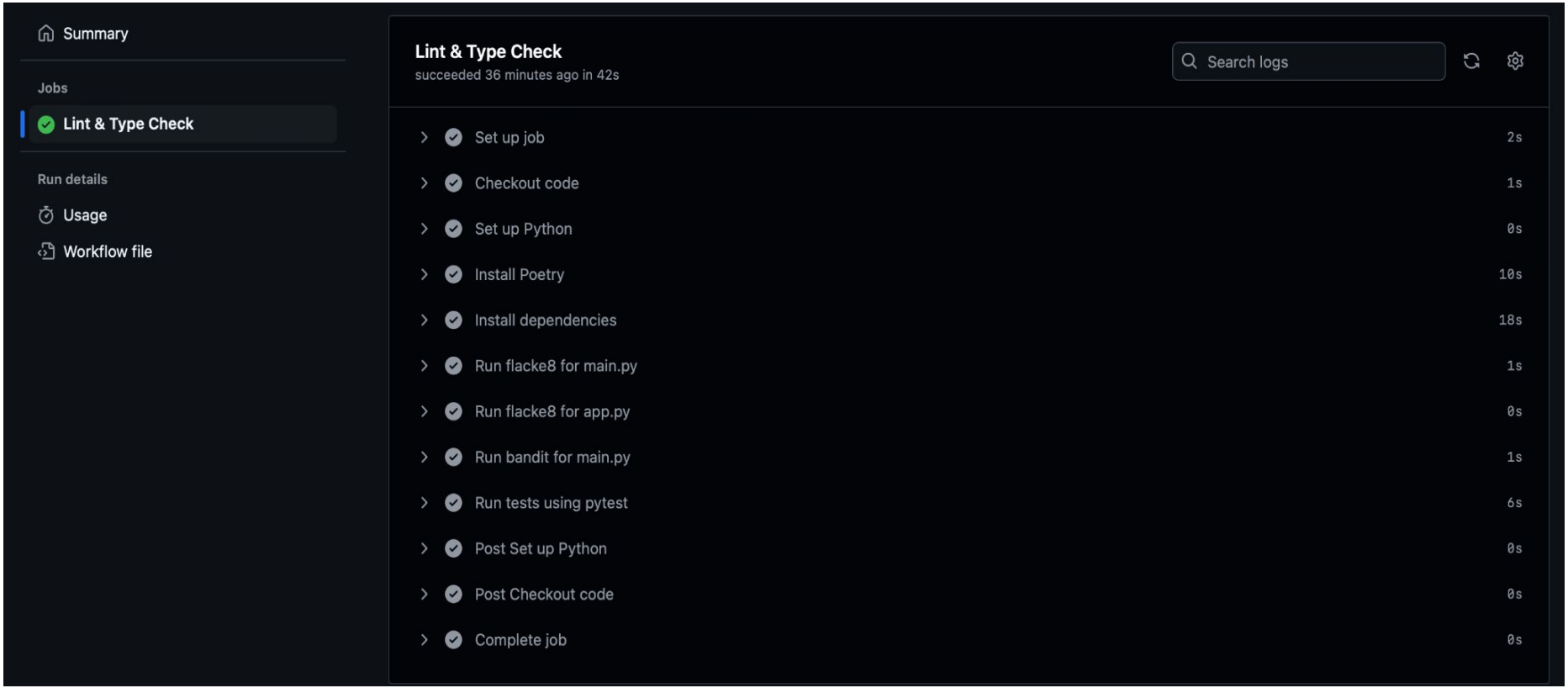
The Finance Tracker is rigorously tested using Pytest, ensuring that all features function as intended and meet quality standards. The inclusion of Prometheus monitoring allows for real-time insights into application performance, helping developers to quickly identify and address any issues that may arise.

```
http_response_size_bytes_sum{handler="/login"} 164.0
http_response_size_bytes_count{handler="/dashboard"} 1.0
http_response_size_bytes_sum{handler="/dashboard"} 307.0
# HELP http_response_size_bytes_created Content length of outgoing responses by handler. Only value of header is respected. Otherwise ignored. No percentile calculated.
# TYPE http_response_size_bytes_created gauge
http_response_size_bytes_created{handler="/login"} 1.746728246848684e+09
http_response_size_bytes_created{handler="/dashboard"} 1.746728248318635e+09
# HELP http_request_duration_highr_seconds Latency with many buckets but no API specific labels. Made for more accurate percentile calculations.
# TYPE http_request_duration_highr_seconds histogram
http_request_duration_highr_seconds_bucket{le="0.01"} 1.0
http_request_duration_highr_seconds_bucket{le="0.025"} 1.0
http_request_duration_highr_seconds_bucket{le="0.05"} 1.0
http_request_duration_highr_seconds_bucket{le="0.075"} 1.0
http_request_duration_highr_seconds_bucket{le="0.1"} 1.0
http_request_duration_highr_seconds_bucket{le="0.25"} 1.0
http_request_duration_highr_seconds_bucket{le="0.5"} 1.0
http_request_duration_highr_seconds_bucket{le="0.75"} 1.0
http_request_duration_highr_seconds_bucket{le="1.0"} 1.0
http_request_duration_highr_seconds_bucket{le="1.5"} 2.0
http_request_duration_highr_seconds_bucket{le="2.0"} 2.0
http_request_duration_highr_seconds_bucket{le="2.5"} 2.0
http_request_duration_highr_seconds_bucket{le="3.0"} 2.0
http_request_duration_highr_seconds_bucket{le="3.5"} 2.0
http_request_duration_highr_seconds_bucket{le="4.0"} 2.0
http_request_duration_highr_seconds_bucket{le="4.5"} 2.0
http_request_duration_highr_seconds_bucket{le="5.0"} 2.0
http_request_duration_highr_seconds_bucket{le="7.5"} 2.0
http_request_duration_highr_seconds_bucket{le="10.0"} 2.0
http_request_duration_highr_seconds_bucket{le="30.0"} 2.0
http_request_duration_highr_seconds_bucket{le="60.0"} 2.0
http_request_duration_highr_seconds_bucket{le="+Inf"} 2.0
http_request_duration_highr_seconds_count 2.0
http_request_duration_highr_seconds_sum 1.1577517910009192
# HELP http_request_duration_highr_seconds_created Latency with many buckets but no API specific labels. Made for more accurate percentile calculations.
# TYPE http_request_duration_highr_seconds_created gauge
http_request_duration_highr_seconds_created 1.746728129327355e+09
# HELP http_request_duration_seconds Latency with only few buckets by handler. Made to be only used if aggregation by handler is important.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{handler="/login",le="0.1",method="POST"} 0.0
http_request_duration_seconds_bucket{handler="/login",le="0.5",method="POST"} 0.0
http_request_duration_seconds_bucket{handler="/login",le="1.0",method="POST"} 0.0
http_request_duration_seconds_bucket{handler="/login",le="+Inf",method="POST"} 1.0
http_request_duration_seconds_count{handler="/login",method="POST"} 1.0
http_request_duration_seconds_sum{handler="/login",method="POST"} 1.1480368330012425
http_request_duration_seconds_bucket{handler="/dashboard",le="0.1",method="GET"} 1.0
http_request_duration_seconds_bucket{handler="/dashboard",le="0.5",method="GET"} 1.0
http_request_duration_seconds_bucket{handler="/dashboard",le="1.0",method="GET"} 1.0
http_request_duration_seconds_bucket{handler="/dashboard",le="+Inf",method="GET"} 1.0
http_request_duration_seconds_count{handler="/dashboard",method="GET"} 1.0
http_request_duration_seconds_sum{handler="/dashboard",method="GET"} 0.009714957999676699
# HELP http_request_duration_seconds_created Latency with only few buckets by handler. Made to be only used if aggregation by handler is important.
# TYPE http_request_duration_seconds_created gauge
http_request_duration_seconds_created{handler="/login",method="POST"} 1.746728246848701e+09
http_request_duration_seconds_created{handler="/dashboard",method="GET"} 1.74672824831866e+09
```


04
Continuous
Integration

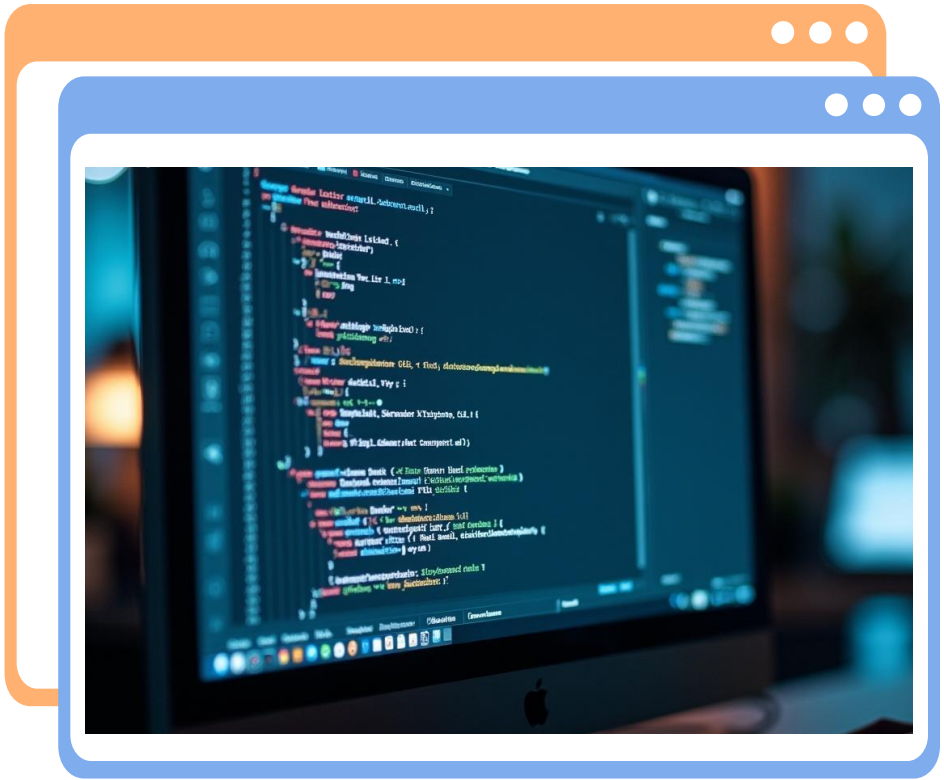
GitHub Actions workflow

The Finance Tracker employs a CI/CD pipeline via GitHub Actions, enabling automated processes for testing and deployment. Each push or pull request triggers a series of checks, ensuring that code changes do not introduce errors or degrade functionality.


Summary
Jobs
Lint & Type Check
Run details
Usage
Workflow file
Lint & Type Check
succeeded 36 minutes ago in 42s
Search logs
Set up job 2s
Checkout code 1s
Set up Python 0s
Install Poetry 10s
Install dependencies 18s
Run flacke8 for main.py 1s
Run flacke8 for app.py 0s
Run bandit for main.py 1s
Run tests using pytest 6s
Post Set up Python 0s
Post Checkout code 0s
Complete job 0s

Automated testing

Automated tests are integrated into the workflow, utilizing Pytest to verify application logic and performance continuously. This practice ensures that any new code adheres to established quality standards and reduces the likelihood of bugs in the production environment.



```
38
39  Name                Stmts   Miss  Cover   Missing
40  ------------------------------------------------
41  app.py                109    109     0%   1-236
42  main.py               279     38    86%   234-235, 256-259, 264, 270-272, 295, 327-330, 342, 381, 475, 509, 587, 589, 647, 667-668,
    705-731
43  tests/conftest.py       3      0   100%
44  tests/test_app.py      85      0   100%
45  ------------------------------------------------
46  TOTAL                 476    147    69%
47  ======================== 10 passed, 3 warnings in 4.74s ========================
```

Maintaining code quality

Code quality is maintained through regular linting and security checks using tools like Flake8 and Bandit. These tools help in identifying potential issues early in the development process, promoting a clean and secure codebase, which is essential for the application's long-term sustainability.

Run flacke8 for main.py

```
1  ▼ Run poetry run flake8 main.py
2    poetry run flake8 main.py
3    shell: /usr/bin/bash -e {0}
4    env:
5      pythonLocation: /opt/hostedtoolcache/Python/3.11.12/x64
6      PKG_CONFIG_PATH: /opt/hostedtoolcache/Python/3.11.12/x64/lib/pkgconfig
7      Python_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
8      Python2_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
9      Python3_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
10     LD_LIBRARY_PATH: /opt/hostedtoolcache/Python/3.11.12/x64/lib
```

Run flacke8 for app.py

```
1  ▼ Run poetry run flake8 app.py
2    poetry run flake8 app.py
3    shell: /usr/bin/bash -e {0}
4    env:
5      pythonLocation: /opt/hostedtoolcache/Python/3.11.12/x64
6      PKG_CONFIG_PATH: /opt/hostedtoolcache/Python/3.11.12/x64/lib/pkgconfig
7      Python_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
8      Python2_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
9      Python3_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.12/x64
10     LD_LIBRARY_PATH: /opt/hostedtoolcache/Python/3.11.12/x64/lib
```

Run bandit for main.py

```
▶ Run poetry run bandit -r main.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.12
Run started:2025-05-08 17:37:30.528070

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 569
        Total lines skipped (#nosec): 0
        Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
Files skipped (0):
```

Conclusions

The Finance Tracker stands as an innovative solution for managing personal finances. With a robust tech stack, comprehensive features, and rigorous testing and monitoring, it provides users with the tools necessary for effective financial management. Engaging with this application not only simplifies tracking but also enhances financial literacy and decision-making.

Thank you!

Do you have any questions?