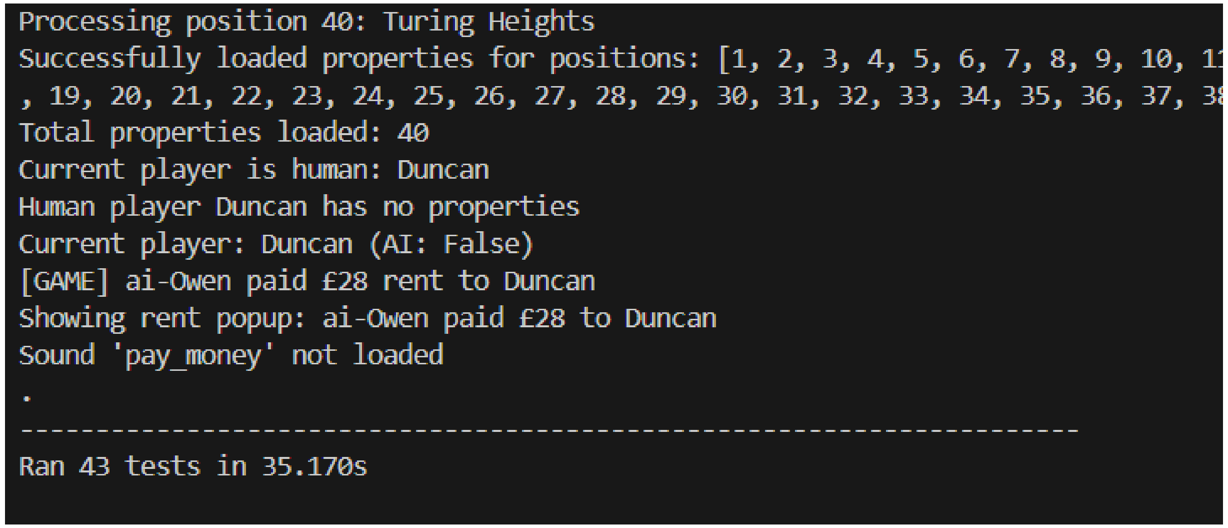
Group 5 – Software Engineering Project- Property Tycoon Testing Report

Date: 09/04/2025

We have tested our game and the code in multiple ways to ensure we know the most things in our game.

Here is the test we used to test and the result:

1. **Unit Test:**



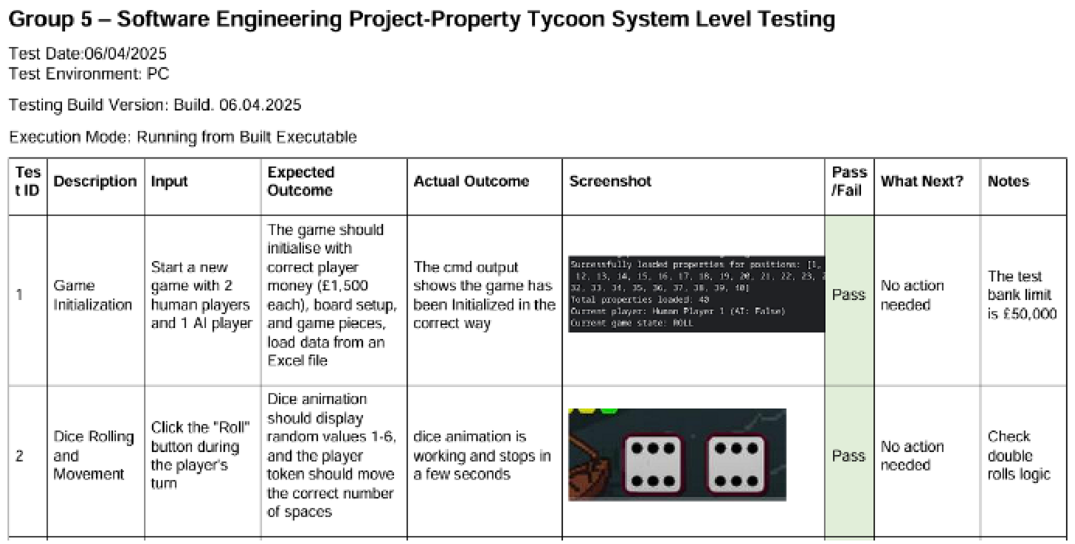
```
Processing position 40: Turing Heights
Successfully loaded properties for positions: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1
, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 3
Total properties loaded: 40
Current player is human: Duncan
Human player Duncan has no properties
Current player: Duncan (AI: False)
[GAME] ai-Owen paid £28 rent to Duncan
Showing rent popup: ai-Owen paid £28 to Duncan
Sound 'pay_money' not loaded
.
----------------------------------------------------------------------
Ran 43 tests in 35.170s
```

The unit testing phase successfully verified the core game logic. The main objectives were to isolate components, validate game mechanics, handle edge cases gracefully, and detect regression issues early.

We utilized Python's unittest framework for testing. The tests were organized within the TestGame class located in tests/Test_Game.py. Fixtures, including setUp and tearDown methods, managed the test environment. Mocking functions, such as random. randint and Pygame functions, ensured that the tests were deterministic. Helper methods were used to manage the synchronization of object-dictionary states.

In total, 43 unit tests comprehensively covered the Game, Significant attention was given to anticipating and mitigating potential edge cases during the initial coding phase. All 43 tests passed successfully when executed against build 09.04.2025.

2. Level testing:

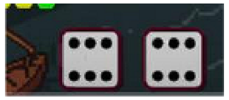


Group 5 – Software Engineering Project-Property Tycoon System Level Testing

Test Date:06/04/2025
Test Environment: PC

Testing Build Version: Build. 06.04.2025

Execution Mode: Running from Built Executable

Test ID	Description	Input	Expected Outcome	Actual Outcome	Screenshot	Pass /Fail	What Next?	Notes
1	Game Initialization	Start a new game with 2 human players and 1 AI player	The game should initialise with correct player money (£1,500 each), board setup, and game pieces, load data from an Excel file	The cmd output shows the game has been Initialized in the correct way	Successfully loaded properties for positions: [1, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 32, 33, 34, 35, 36, 37, 38, 39, 40] Total properties loaded: 40 Current player: Human Player 1 (AI: False) Current game state: ROLL	Pass	No action needed	The test bank limit is £50,000
2	Dice Rolling and Movement	Click the "Roll" button during the player's turn	Dice animation should display random values 1-6, and the player token should move the correct number of spaces	dice animation is working and stops in a few seconds		Pass	No action needed	Check double rolls logic

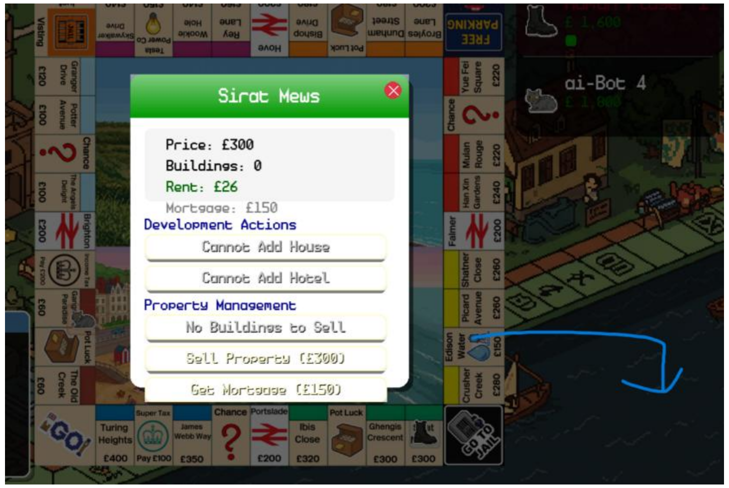
System-level testing focused on validating the overall gameplay experience and ensuring that different components interact correctly within the full application context. A series of test cases were executed, covering key gameplay scenarios. Detailed results for each test case are documented across two System Level Testing docs.

Most system-level test cases passed, and the core gameplay loop and main features function as expected when running the built application. However, we have identified several integration and timing-related bugs.

The specific issues found in version V1.0 09/04/2025 include:

- Card/Property Pop-up Delay: Information pop-ups related to landing on card spaces or properties sometimes appear one game round later than they should. While triggered, the display was delayed until the player's next turn. Notably, the tax payment pop-up functioned correctly.

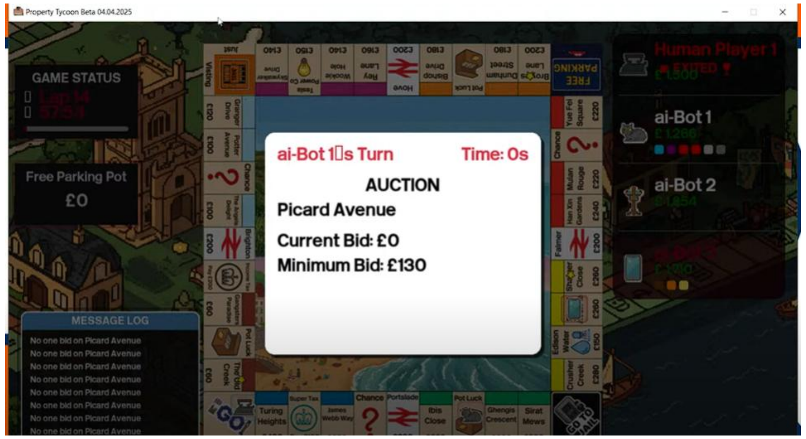
- Hunman Player Developer Mode Button Disappearance:



Price: £300
Buildings: 0
Rent: £26
Mortgage: £150

The end turn button could disappear when a human player is in Developer Mode, making the game unplayable since the human can't end their round in Developer Mode.

- Auction Screen Lag:

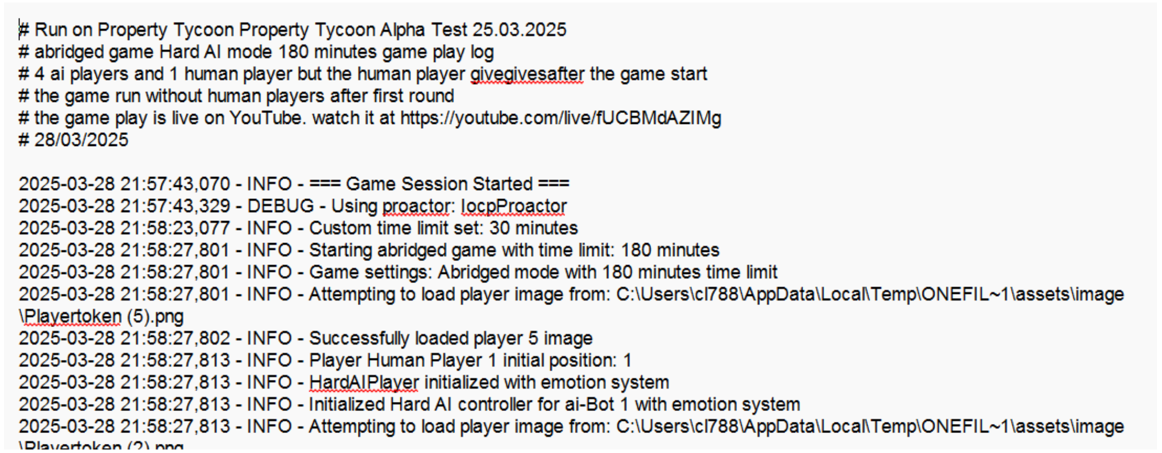


ai-Bot 1□s Turn Time: 0s
AUCTION
Picard Avenue
Current Bid: £0
Minimum Bid: £130

No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue
No one bid on Picard Avenue

The auction interface encountered potential freezing issues when AI players were involved in the bidding. However, it would start working again if a human player clicked on the screen with a mouse.

3. Game play Testing Debug log:



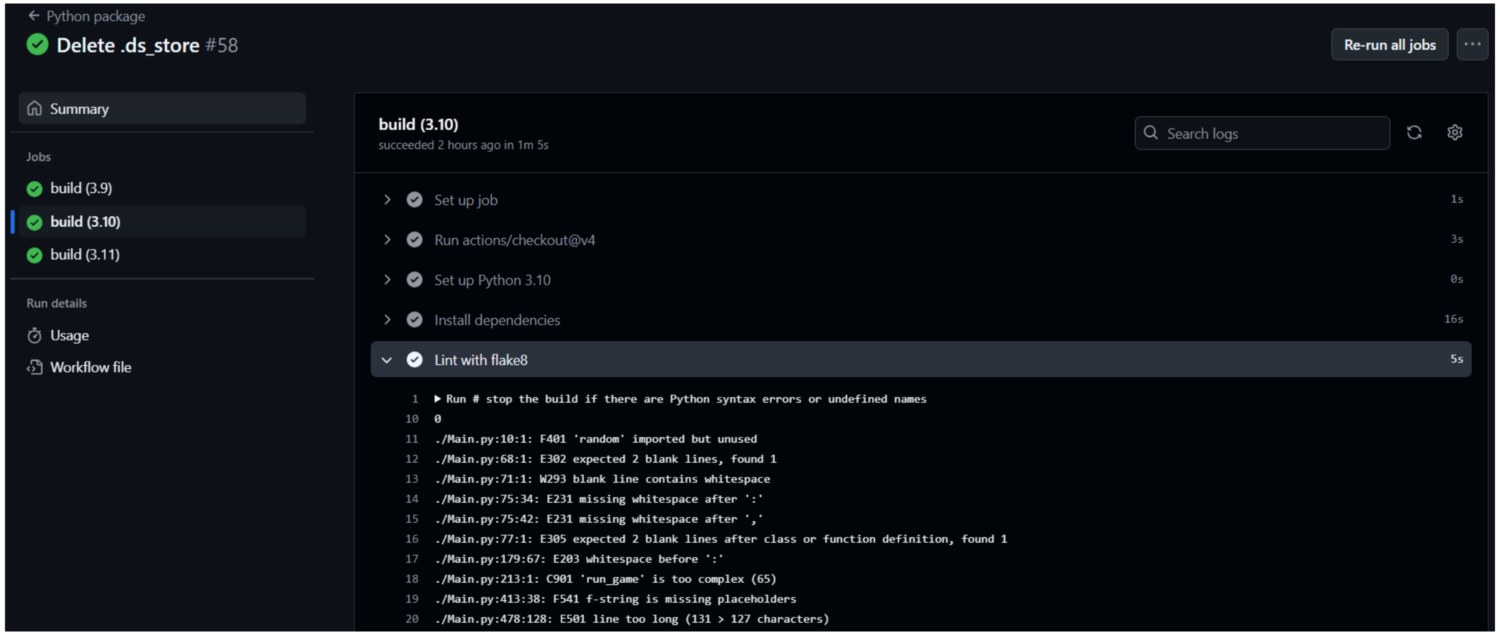
```
# Run on Property Tycoon Property Tycoon Alpha Test 25.03.2025
# abridged game Hard AI mode 180 minutes game play log
# 4 ai players and 1 human player but the human player givegivesafter the game start
# the game run without human players after first round
# the game play is live on YouTube. watch it at https://youtube.com/live/fUCBMdAZIMg
# 28/03/2025

2025-03-28 21:57:43,070 - INFO - === Game Session Started ===
2025-03-28 21:57:43,329 - DEBUG - Using proactor: IocpProactor
2025-03-28 21:58:23,077 - INFO - Custom time limit set: 30 minutes
2025-03-28 21:58:27,801 - INFO - Starting abridged game with time limit: 180 minutes
2025-03-28 21:58:27,801 - INFO - Game settings: Abridged mode with 180 minutes time limit
2025-03-28 21:58:27,801 - INFO - Attempting to load player image from: C:\Users\cl788\AppData\Local\Temp\ONEFIL~1\assets\image
\Playertoken (5).png
2025-03-28 21:58:27,802 - INFO - Successfully loaded player 5 image
2025-03-28 21:58:27,813 - INFO - Player Human Player 1 initial position: 1
2025-03-28 21:58:27,813 - INFO - HardAIPlayer initialized with emotion system
2025-03-28 21:58:27,813 - INFO - Initialized Hard AI controller for ai-Bot 1 with emotion system
2025-03-28 21:58:27,813 - INFO - Attempting to load player image from: C:\Users\cl788\AppData\Local\Temp\ONEFIL~1\assets\image
\Playertoken (2).png
```

During the game development process, we extensively used debugging print statements to gain insights into the game's behavior. In the early iterations of the code, we incorporated detailed and easily understandable debug output. As the project progressed, we transitioned to utilizing Python's logging module to manage these outputs more effectively.

We captured and saved the command-line output from each gameplay session to a dedicated log file. This approach allowed us to trace the flow of the game easily, analyze player actions, and identify potential issues or unexpected behaviors within the code. We consistently added additional debug print statements based on our observations of actual game progress, which helped us identify and understand bugs.

The gameplay testing debug logs can be found in the "6. Testing Evidence\Gameplay Testing Debug Log" directory. Additionally, gameplay test videos are available on the YouTube channel: https://www.youtube.com/channel/UCfA_oAs-qoOGMWJ_UueYHng.

4. Semantic and stylistic Test with Lint:



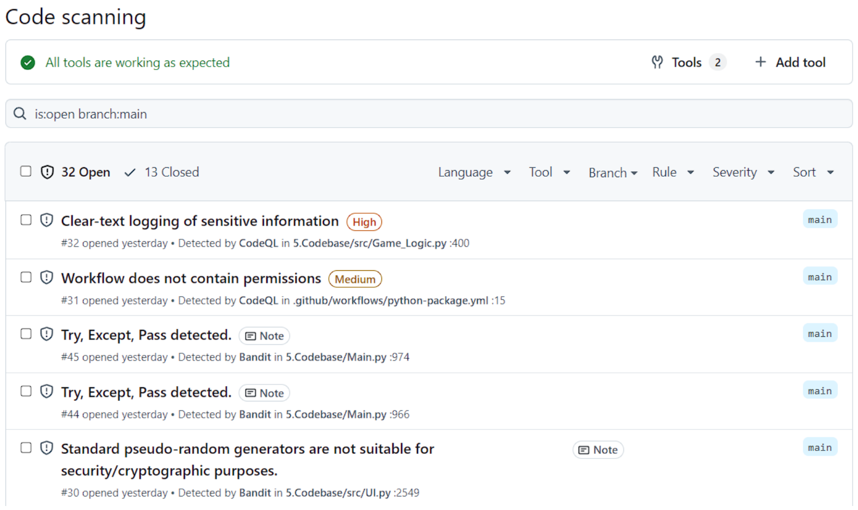
```
▶ Run # stop the build if there are Python syntax errors or undefined names
0
./Main.py:10:1: F401 'random' imported but unused
./Main.py:68:1: E302 expected 2 blank lines, found 1
./Main.py:71:1: W293 blank line contains whitespace
./Main.py:75:34: E231 missing whitespace after ':'
./Main.py:75:42: E231 missing whitespace after ','
./Main.py:77:1: E305 expected 2 blank lines after class or function definition, found 1
./Main.py:179:67: E203 whitespace before ':'
./Main.py:213:1: C901 'run_game' is too complex (65)
./Main.py:413:38: F541 f-string is missing placeholders
./Main.py:478:128: E501 line too long (131 > 127 characters)
```

To enhance code quality beyond functionality, we automated semantic and stylistic checks using linting. This was integrated via a GitHub Action, following standard Python testing practices (https://docs.github.com/en/actions/use-cases-and-examples/building-and-testing/building-and-testing-python).

Triggered automatically on every push, this action performs static analysis. It uses a linter to check the code against PEP 8 style guidelines and identify potential coding issues without executing the code.

This enforcement of consistent style and early detection significantly improves the codebase's readability, understandability, and maintainability for the entire team.

5. Test Code security results:



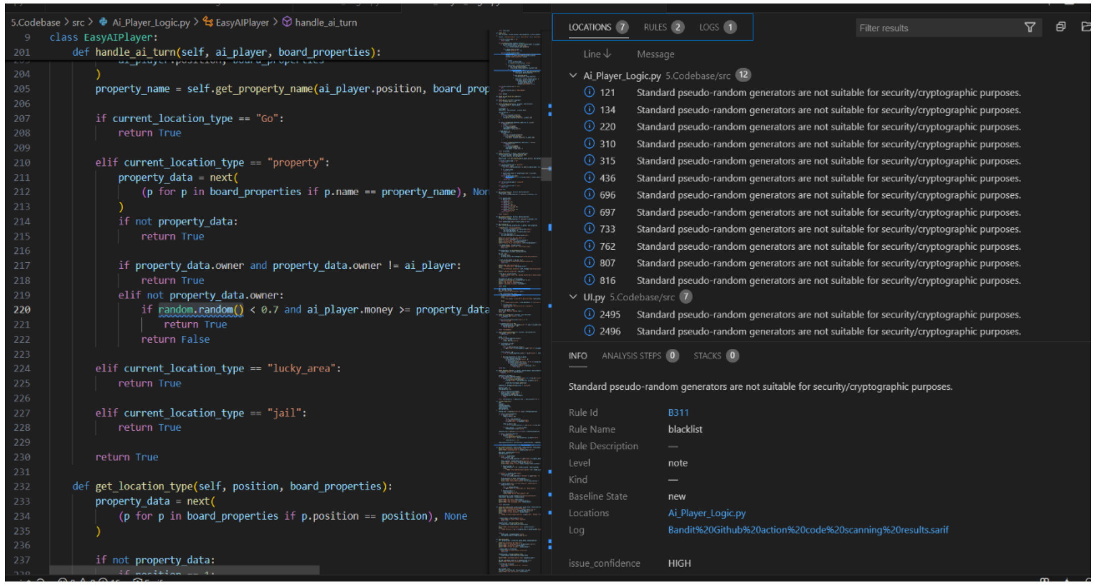
Code scanning

All tools are working as expected — Tools 2 — Add tool

is:open branch:main

32 Open ✓ 13 Closed — Language, Tool, Branch, Rule, Severity, Sort

- Clear-text logging of sensitive information (High) — main
  #32 opened yesterday • Detected by CodeQL in 5.Codebase/src/Game_Logic.py :400
- Workflow does not contain permissions (Medium) — main
  #31 opened yesterday • Detected by CodeQL in .github/workflows/python-package.yml :15
- Try, Except, Pass detected. (Note) — main
  #45 opened yesterday • Detected by Bandit in 5.Codebase/Main.py :974
- Try, Except, Pass detected. (Note) — main
  #44 opened yesterday • Detected by Bandit in 5.Codebase/Main.py :966
- Standard pseudo-random generators are not suitable for security/cryptographic purposes. (Note) — main
  #30 opened yesterday • Detected by Bandit in 5.Codebase/src/UI.py :2549



5.Codebase > src > Ai_Player_Logic.py > EasyAIPlayer > handle_ai_turn

```
9   class EasyAIPlayer:
201     def handle_ai_turn(self, ai_player, board_properties):
204         )
205         property_name = self.get_property_name(ai_player.position, board_prop
206
207         if current_location_type == "Go":
208             return True
209
210         elif current_location_type == "property":
211             property_data = next(
212                 (p for p in board_properties if p.name == property_name), Non
213             )
214             if not property_data:
215                 return True
216
217             if property_data.owner and property_data.owner != ai_player:
218                 return True
219             elif not property_data.owner:
220                 if random.random() < 0.7 and ai_player.money >= property_data
221                     return True
222                 return False
223
224         elif current_location_type == "lucky_area":
225             return True
226
227         elif current_location_type == "jail":
228             return True
229
230         return True
231
232     def get_location_type(self, position, board_properties):
233         property_data = next(
234             (p for p in board_properties if p.position == position), None
235         )
236
237         if not property_data:
```

LOCATIONS 7 — RULES 2 — LOGS 1 — Filter results

Line	Message
Ai_Player_Logic.py 5.Codebase/src 12	
121	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
134	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
220	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
310	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
315	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
436	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
696	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
697	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
733	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
762	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
807	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
816	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
UI.py 5.Codebase/src 7	
2495	Standard pseudo-random generators are not suitable for security/cryptographic purposes.
2496	Standard pseudo-random generators are not suitable for security/cryptographic purposes.

INFO — ANALYSIS STEPS 0 — STACKS 0

Standard pseudo-random generators are not suitable for security/cryptographic purposes.

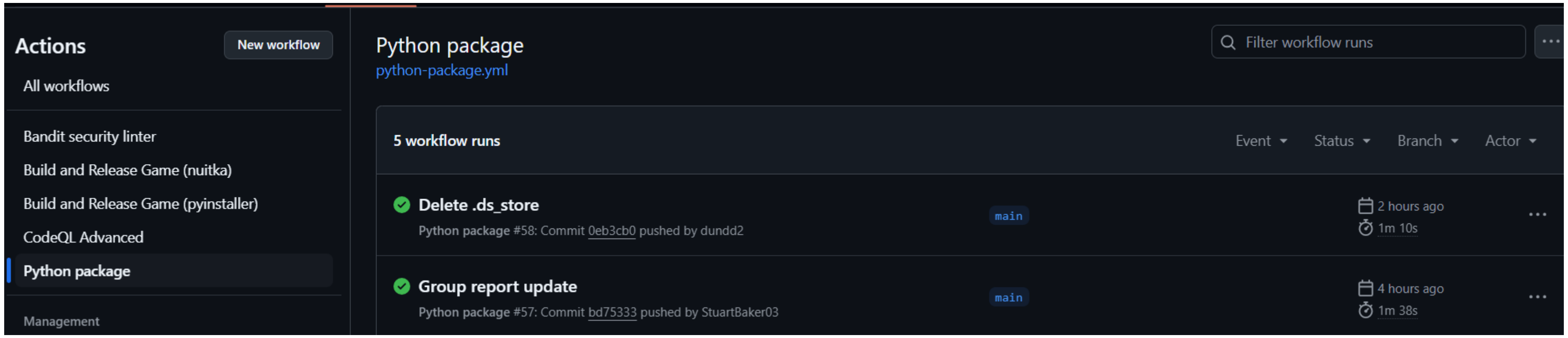
Rule Id	B311
Rule Name	blacklist
Rule Description	—
Level	note
Kind	—
Baseline State	new
Locations	Ai_Player_Logic.py
Log	Bandit%20Github%20action%20code%20scanning%20results.sarif
issue_confidence	HIGH

Given the potential public release of 'Property Tycoon' by Watson Games, codebase security is critical to prevent vulnerabilities that could lead to malware and harm user trust.

To address this, we automated security scanning using Bandit, a Python-specific tool (https://github.com/PyCQA/bandi). This scan was integrated via a dedicated GitHub Action (https://github.com/Minosaji/Software-Engineering-Project/actions/workflows/bandit%20security%20linter.yml) triggered on code pushes, ensuring continuous checks.

Scan results are generated in SARIF format. Using extensions like the Microsoft SARIF Viewer for VS Code (https://marketplace.visualstudio.com/items/?itemName=MS-SarifVSCode.sarif-viewer), these reports pinpoint potential vulnerabilities directly within the code editor. Detailed results are available in the "Test Code security results" folder.

6. Continuous Integration (CI)test using GitHub Actions



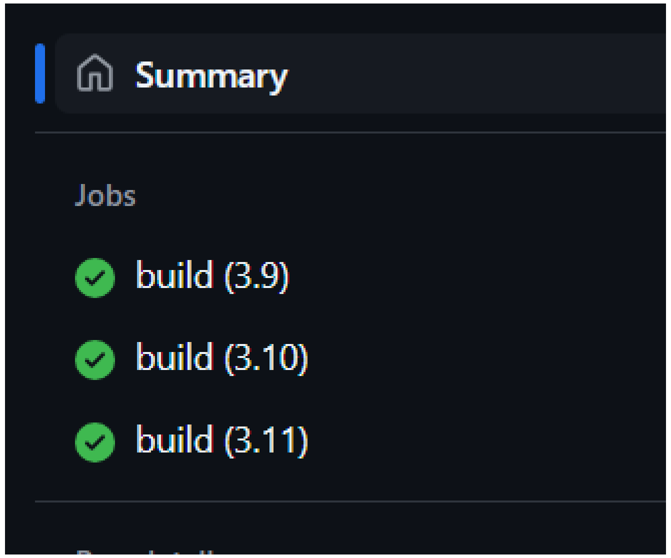
Actions
New workflow
All workflows
Bandit security linter
Build and Release Game (nuitka)
Build and Release Game (pyinstaller)
CodeQL Advanced
Python package
Management

Python package
python-package.yml
Filter workflow runs

5 workflow runs
Event Status Branch Actor

Delete .ds_store
Python package #58: Commit 0eb3cb0 pushed by dundd2
main
2 hours ago
1m 10s

Group report update
Python package #57: Commit bd75333 pushed by StuartBaker03
main
4 hours ago
1m 38s



Summary

Jobs

✓ build (3.9)

✓ build (3.10)

✓ build (3.11)

To improve code quality and stability, we implemented a CI pipeline using GitHub Actions. This automatically tests our code on every push and pull request to the main branch. Running on ubuntu-latest, it uses a matrix strategy to test against Python 3.9, 3.10, and 3.11 for compatibility. This CI setup ensures consistent testing environments (reducing "works on my machine" issues) and enables early detection of bugs and integration problems, making fixes easier and maintaining code stability. Workflow runs are visible here: https://github.com/Minosaji/Software-Engineering-Project/actions/workflows/python-package.yml.