GITGOOD

Help gitGood, gitBetter

TESTING POLICY

DEMO 4

WEATHER TO WEAR

EPI-USE®

KYLE LIEBENBERG
DIYA BUDHIA
ALISHA PERUMAL

IBRAHIM SAID
TAYLOR SERGEL

INTRODUCTION

The Weather-to-Wear system is an integrated application that provides personalized clothing recommendations based on real-time weather conditions while offering social, wardrobe management, and event planning features. Given the diverse range of modules .

This document outlines the testing policy followed during the development of the Weather-to-Wear system. It describes the testing philosophy, the different testing levels applied, the strategies used in execution, and the outcomes of the testing effort. Each section elaborates on the rationale for the chosen approach, how it was implemented, and why it is relevant to system success.

The testing activities did not only focus on confirming that the system works, but also on ensuring that it performs consistently under realistic conditions, scales with user growth, and delivers a positive, secure, and reliable user experience.

TETSING STRATEGY OVERVIEW

The testing strategy followed a layered approach, aligned with industry best practices:

1. **Unit Testing** was applied to verify the correctness of individual functions across modules in isolation.
2. **Integration Testing** validated that modules interact correctly with one another, especially where data is exchanged across system boundaries.
3. **End-to-End (E2E) Testing** simulated real-world user scenarios, validating complete workflows.
4. **Non-Functional Testing** covered *performance*, *security*, *usability*, *scalability* and *latency* to ensure the system meets quality expectations beyond functionality.
5. **Usability Testing** helped identify areas for improvement and aspects that users found intuitive.

This strategy ensured that defects were detected early and at the appropriate level, significantly reducing the risk of cascading errors during integration or production deployment.

1. UNIT TESTING
OVERVIEW

Unit testing targeted the smallest functional components of the system, such as service functions and utility methods. By isolating these units, we were able to confirm that each function worked as intended under a variety of inputs and error conditions.

APPROACH

- Implemented using Jest for backend modules and supporting mocking libraries for external dependencies (e.g., file system, AWS S3, external APIs).
- Each function was tested under:
  - Normal conditions (expected input and output).
  - Error conditions (missing data, invalid user ID).
  - Edge cases (empty closets, invalid weather API responses).
- External services such as the weather API and S3 bucket uploads were mocked to ensure repeatability and consistency of results.

RELEVANCE

Unit testing was indispensable in building confidence at the foundation level. Since the system heavily relies on business logic, correctness here prevented downstream failures and improved maintainability.

COVERAGE ACROSS MODULES

AUTHORIZATION MODULE

- signup - handles valid and invalid credentials.
- login - handles valid and invalid credentials.

CLOSET MODULE

- uploadImage – Confirmed single image uploads are saved to the database and S3.
- uploadImagesBatch – Validated multiple file uploads, ensuring all images are processed and stored correctly.
- getByCategory – Ensured items are retrieved accurately by specified category (e.g., shoes, tops).
- deleteItem – Tested item deletion, including cleanup of local files and database references.
- updateItem – Confirmed item properties (category, tags, etc.) are updated correctly.
- favourite – Verified that items can be marked/unmarked as favorites.

EVENTS MODULE

- getEvents – Confirmed retrieval of all events belonging to a user.
- getEventById – Verified retrieval of a specific event by ID.
- createEvent – Tested successful creation of new events with required fields.
- updateEvent – Ensured event details are updated correctly.
- deleteEvent – Confirmed events are properly deleted, including linked packing lists.

OUTFIT MODULE

- create – Tested outfit creation using selected closet items.
- getAll – Verified retrieval of all outfits for a user.
- update – Confirmed outfits can be updated with new names, tags, or items.
- delete – Ensured outfits can be removed from the system.
- getItems – Verified retrieval of items belonging to a specific outfit.
- addItem – Confirmed adding new closet items to an outfit.
- removeItem – Tested removal of items from an outfit.
- recommend – Validated weather-based recommendations to suggest outfits.
- favourite – Verified outfits can be favorited/unfavorited.

PACKING MODULE

- createPackingList – Tested creation of packing lists for events.
- getPackingList – Verified retrieval of packing lists by event ID.
- updatePackingList – Ensured updates to packing list items (add/remove/update).
- deletePackingList – Confirmed packing lists are deleted when no longer needed.

SOCIAL MEDIA MODULE

- getPostById – Verified retrieval of a specific post.
- getPosts – Tested retrieval of multiple posts (feed view).
- createPost – Confirmed creation of posts with images/text.
- updatePost – Ensured posts can be edited.
- deletePost – Verified deletion of posts including cleanup of linked data.
- addComment – Tested adding comments to posts.
- updateComment – Confirmed comments can be updated.
- deleteComment – Verified comment deletion.
- getLikesForPost – Ensured accurate like counts and retrieval.
- likePost – Tested liking functionality.
- unlikePost – Confirmed unliking functionality.
- getFollowing – Verified retrieval of users that the current user follows.
- getFollowers – Verified retrieval of users that follow the current user.
- followUser – Tested follow functionality and database update.
- unfollowUser – Confirmed unfollow functionality.
- searchUsers – Verified that users can be searched by username/email.
- getNotifications – Ensured notifications are generated and retrieved correctly.
- acceptFollowRequest – Tested acceptance of pending follow requests.
- rejectFollowRequest – Verified rejection of follow requests.

USERS MODULE

- setUsername – Tested changing usernames.
- setEmailAddress – Confirmed updating email addresses.
- updateProfilePhoto – Verified profile photo uploads and updates.
- updatePrivacy – Tested updating account privacy settings (public/private).
- getMyPreferences – Verified retrieval of a user's color and style preferences.
- updatePreferences – Confirmed updates to preference settings persist.

WEATHER MODULE

- getWeather – Verified retrieval of current weather for a given location.
- getWeatherForDay – Confirmed accurate weather forecast retrieval for a day.
- getWeatherForWeek – Tested weekly forecast data.
- getCityMatches – Ensured city search returns valid location matches.

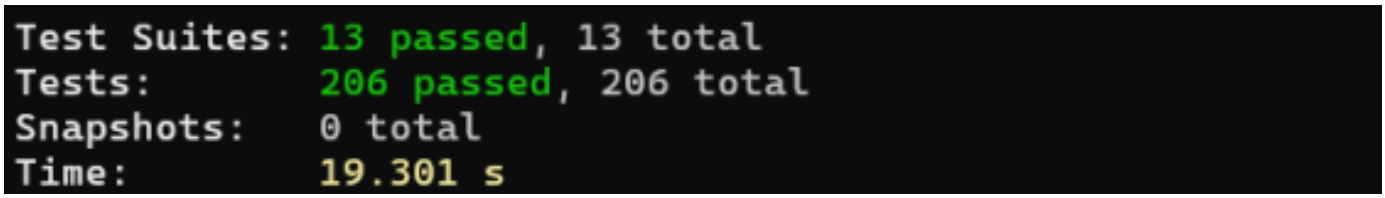
RESULTS

- Achieved >75% coverage for core service functions.
- Identified missing error handling in several delete/update operations, which were corrected early.
- Reduced later integration errors, especially in modules that depended on external APIs.

UNIT TEST RESULTS

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	77.47	54.63	83.4	80.22	
src	100	100	100	100	
prisma.ts	100	100	100	100	
src/middleware	50	23.52	0	50	
upload.middleware.ts	50	23.52	0	50	20-23,34-44
src/modules/auth	89.55	83.33	100	88.88	
auth.middleware.ts	77.41	75	100	77.41	31-32,39-40,46-51
auth.service.ts	100	100	100	100	
auth.utils.ts	100	100	100	100	
src/modules/closet	77.88	52.38	90	78.01	
closet.controller.ts	72.17	43.75	90.9	71.92	...05-106,127-128,163,170-171,205-206,226-227,235,237,239,241,243,245,262,271-285
closet.route.ts	100	100	100	100	
closet.service.ts	82.85	61.29	88.88	84.12	46,51,75,90-91,112-113,194-199
src/modules/events	91.33	86.48	100	94.21	
events.controller.ts	91.33	86.48	100	94.21	32,70,133,147-148,216,249
src/modules/outfit	79.66	60.42	80.86	82.85	
collabFiltering.ts	58.06	56.52	44.44	62.22	51-57,72-87
itemItemKnn.ts	97.91	50	100	100	17-34,40-80
outfit.controller.ts	82.75	60.97	80	82.85	21-30,72,84,97,119,135,181,240,247-260
outfit.service.ts	85.24	92.85	91.66	84.61	153,193,217,234-235,239-241
outfitRecommender.service.ts	78.91	56.72	80.3	83.6	28-33,137-138,148-149,245-247,254,256,319-338,356-379,396-398
src/modules/packing	100	89.74	100	100	
packing.controller.ts	100	89.74	100	100	57,85,129,153
src/modules/social	65.35	34.78	75	68.35	
social.controller.ts	65.35	34.78	75	68.35	...36,156-158,168-184,216-225,270,297-301,326,354,438-443,450-455,460-465,470-496
src/modules/userPreference	100	100	100	100	
userPref.controller.ts	100	100	100	100	
src/modules/weather	74.09	41.53	95	79.22	
weather.service.ts	74.09	41.53	95	79.22	...36-153,239,288,336,379,434,467,510-518,537,587,619-620,634,649-666,677,708-709
src/utils	43.18	14.28	25	47.5	
logger.ts	100	100	100	100	
s3.ts	39.02	14.28	25	43.24	20-31,35,62-96

```
Test Suites: 13 passed, 13 total
Tests:       206 passed, 206 total
Snapshots:   0 total
Time:        19.301 s
Ran all test suites.
```



2. INTEGRATION TESTING

OVERVIEW

Integration testing was performed to validate the interaction between individual modules within the Weather-to-Wear system. This level of testing was especially critical due to the system's reliance on multiple interconnected modules such as user authentication, closet management, social media interactions, weather services, and event scheduling.

APPROACH

The integration testing approach combined API testing, database verification, authentication validation, and service mocking. The methodology was as follows:

1. API-Level Testing
   - All endpoints were tested using Supertest, simulating HTTP requests and responses.
   - Both happy-path scenarios and negative/error scenarios were covered.
2. Authentication & Authorization Verification
   - JWT tokens were generated for test users to validate protected endpoints.
   - Tests ensured that requests without valid tokens were correctly blocked with 401 Unauthorized responses.
3. Database Interaction Testing
   - Prisma ORM operations were validated by checking that create, read, update, and delete operations produced expected results.
   - Each test ran on a clean database state to maintain independence and reproducibility.
4. External Service Mocking
   - External dependencies such as the weather service were mocked using Jest.
   - This allowed predictable responses (e.g., weather forecasts) while verifying the integration logic that consumes them.
5. Test Isolation and Cleanup
   - Before each test, relevant database tables (users, events, posts, likes, comments, etc.) were cleared.
   - After all tests, database connections were closed, and cleanup routines ensured no residual test data remained.

This approach ensured that integration testing validated inter-module communication, data consistency, and workflow reliability in a controlled and repeatable environment.

RELEVANCE

Integration testing was critical to ensure that complex user workflows spanning multiple modules functioned seamlessly. This includes:

- Event creation linked to weather forecasts and packing lists.
- Closet items correctly flowing into outfit recommendations.
- Social interactions triggering proper notifications.

By validating inter-module communication, the integration tests ensured that users experienced a reliable and coherent system, laying the foundation for higher-level end-to-end testing.

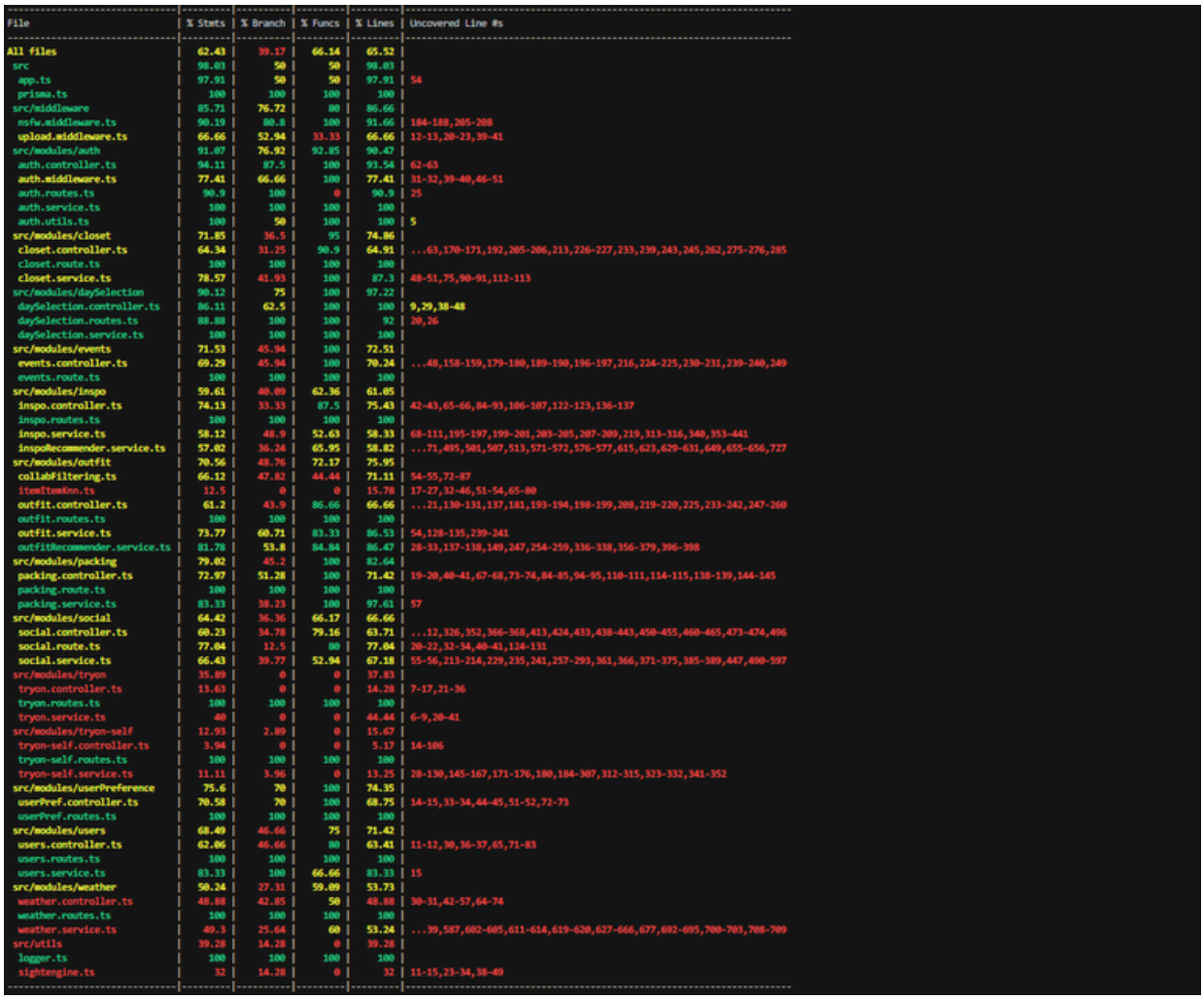
RESULTS

Integration testing yielded highly positive results:

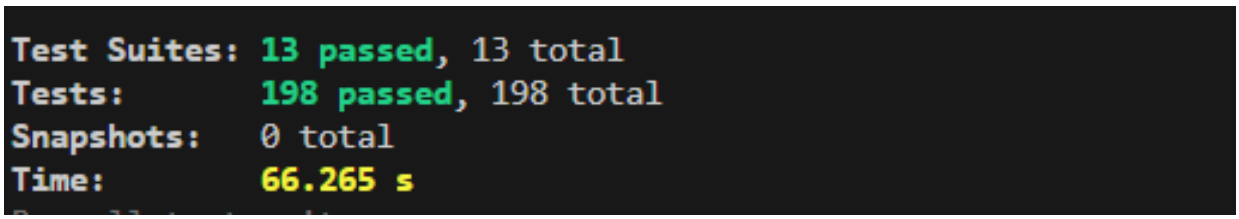
- High Pass Rate: The majority of integration scenarios executed successfully, demonstrating stable interactions between modules.
- Robust Error Handling: Endpoints consistently returned informative error messages for invalid requests, enhancing system resilience.
- Authentication Enforced: Tests confirmed that all protected routes enforced JWT verification correctly.
- Data Consistency: Database entries created, updated, and deleted via API requests matched expectations without leaving residual artifacts.

A small number of test iterations uncovered edge case issues which were corrected by refining request validation middleware.

INTEGRATION TEST RESULTS



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	62.43	39.17	66.14	65.52	
src	98.03	50	50	98.03	
app.ts	97.91	50	50	97.91	54
prisma.ts	100	100	100	100	
src/middleware	85.71	76.72	80	86.66	
nsfw.middleware.ts	90.19	80.8	100	91.66	184-188,205-208
upload.middleware.ts	66.66	52.94	33.33	66.66	12-13,20-23,39-41
src/modules/auth	91.07	76.92	92.85	90.47	
auth.controller.ts	94.11	87.5	100	93.54	62-63
auth.middleware.ts	77.41	66.66	100	77.41	31-32,39-40,46-51
auth.routes.ts	90.9	100	0	90.9	25
auth.service.ts	100	100	100	100	
auth.utils.ts	100	50	100	100	5
src/modules/closet	71.85	36.5	95	74.86	
closet.controller.ts	64.34	31.25	90.9	64.91	...63,170-171,192,205-206,213,226-227,233,239,243,245,262,275-276,285
closet.route.ts	100	100	100	100	
closet.service.ts	78.57	41.93	100	87.3	48-51,75,90-91,112-113
src/modules/daySelection	90.12	75	100	97.22	
daySelection.controller.ts	86.11	62.5	100	100	9,29,38-48
daySelection.routes.ts	88.88	100	100	92	20,26
daySelection.service.ts	100	100	100	100	
src/modules/events	71.53	45.94	100	72.51	
events.controller.ts	69.29	45.94	100	70.24	...48,158-159,179-180,189-190,196-197,216,224-225,230-231,239-240,249
events.route.ts	100	100	100	100	
src/modules/inspo	59.61	40.09	62.36	61.05	
inspo.controller.ts	74.13	33.33	87.5	75.43	42-43,65-66,84-93,106-107,122-123,136-137
inspo.routes.ts	100	100	100	100	
inspo.service.ts	58.12	48.9	52.63	58.33	68-111,195-197,199-201,203-205,207-209,219,313-316,340,353-441
inspoRecommender.service.ts	57.02	36.24	65.95	58.82	...71,495,501,507,513,571-572,576-577,615,623,629-631,649,655-656,727
src/modules/outfit	70.56	48.76	72.17	75.95	
collabFiltering.ts	66.12	47.82	44.44	71.11	54-55,72-87
itemItemKnn.ts	12.5	0	0	15.78	17-27,32-46,51-54,65-80
outfit.controller.ts	61.2	43.9	86.66	66.66	...21,130-131,137,181,193-194,198-199,208,219-220,225,233-242,247-260
outfit.routes.ts	100	100	100	100	
outfit.service.ts	73.77	60.71	83.33	86.53	54,128-135,239-241
outfitRecommender.service.ts	81.78	53.8	84.84	86.47	28-33,137-138,140,247,254-259,336-338,356-379,396-398
src/modules/packing	79.02	45.2	100	82.64	
packing.controller.ts	72.97	51.28	100	71.42	19-20,40-41,67-68,73-74,84-85,94-95,110-111,114-115,138-139,144-145
packing.route.ts	100	100	100	100	
packing.service.ts	83.33	38.23	100	97.61	57
src/modules/social	64.42	36.36	66.17	66.66	
social.controller.ts	60.23	34.78	79.16	63.71	...12,326,352,366-368,413,424,433,438-443,450-455,460-465,473-474,496
social.route.ts	77.04	12.5	80	77.04	20-22,32-34,40-41,124-131
social.service.ts	66.43	39.77	52.94	67.18	55-56,213-214,229,235,241,257-293,361,366,371-375,385-389,447,490-597
src/modules/tryon	35.89	0	0	37.83	
tryon.controller.ts	13.63	0	0	14.28	7-17,21-36
tryon.routes.ts	100	100	100	100	
tryon.service.ts	40	0	0	44.44	6-9,20-41
src/modules/tryon-self	12.93	2.89	0	15.67	
tryon-self.controller.ts	3.94	0	0	5.17	14-106
tryon-self.routes.ts	100	100	100	100	
tryon-self.service.ts	11.11	3.96	0	13.25	28-130,145-167,171-176,180,184-307,312-315,323-332,341-352
src/modules/userPreference	75.6	70	100	74.35	
userPref.controller.ts	70.58	70	100	68.75	14-15,33-34,44-45,51-52,72-73
userPref.routes.ts	100	100	100	100	
src/modules/users	68.49	46.66	75	71.42	
users.controller.ts	62.06	46.66	80	63.41	11-12,30,36-37,65,71-83
users.routes.ts	100	100	100	100	
users.service.ts	83.33	100	66.66	83.33	15
src/modules/weather	50.24	27.31	59.09	53.73	
weather.controller.ts	48.88	42.85	50	48.88	30-31,42-57,64-74
weather.routes.ts	100	100	100	100	
weather.service.ts	49.3	25.64	60	53.24	...39,587,602-605,611-614,619-620,627-666,677,692-695,700-703,708-709
src/utils	39.28	14.28	0	39.28	
logger.ts	100	100	100	100	
sightengine.ts	32	14.28	0	32	11-15,23-34,38-49



```
Test Suites: 13 passed, 13 total
Tests:       198 passed, 198 total
Snapshots:   0 total
Time:        66.265 s
```

3. END-TO-END TESTING

OVERVIEW

End-to-End testing simulated real user behavior across the entire application, validating workflows from login to wardrobe management, outfit recommendation, and social interactions.

APPROACH

- Tooling: Cypress for browser automation.
- Focused on end-user experience: workflows were defined to mimic typical user journeys.
- Covered happy-path, alternative, and failure paths.

RELEVANCE

E2E testing demonstrated that the system meets its core user promise — delivering weather-driven clothing recommendations while integrating wardrobe, events, and social features into a coherent workflow.

RESULTS

- Initial pass rate of ~85%; failures due to asynchronous notification updates and rendering bugs in try-on avatar.
- Post-fix regression testing showed 100% pass rate.
- Verified application stability and correctness in real-world scenarios.
- Check uploaded result video on GitHub.

Security

OVERVIEW

- Security testing was conducted to ensure that the Weather-to-Wear system protects user data, enforces secure access, and complies with modern web security standards. The tests focused on four main requirements:
  - Protected routes requiring valid JWT authentication
  - Proper hashing of passwords
  - Secure HTTPS traffic and deployment
  - Implementation of security headers and best practices

APPROACH

1. JWT Protection Tests

- Objective: Verify that all protected routes enforce JWT authentication and remain secure.
- Scope:
  - All API endpoints requiring authentication were tested.
  - Tokens tested included missing, invalid, tampered, and expired tokens.
  - Verified that public endpoints remain accessible without a token.
  - Edge cases were tested including case sensitivity, multiple spaces, and incorrect schemas.
- Outcome: Confirmed that only valid JWTs allow access to protected resources, and invalid tokens are rejected.

2. Password Security Tests

- Objective: Ensure passwords are securely stored and handled.
- Scope:
  - Passwords hashed using bcrypt with proper salting.
  - Validated password rules and defenses against timing attacks.
  - Tested edge cases: Unicode characters, special characters, and very long passwords.
  - Ensured plaintext passwords are never stored in the database.
  - Verified error messages do not allow user enumeration.
- Outcome: Password handling was secure, resistant to attacks, and met industry best practices.

APPROACH

3. Security Header Tests

- Objective: Ensure headers protect against information disclosure and security misconfigurations.
- Scope:
  - Verified HTTPS headers for proper configuration.
  - Ensured no server technologies or sensitive information were exposed.
  - Checked CORS, MIME types, and header content for user input reflection.
  - Ensured no technology leaks in error messages.
- Outcome: Security headers were correctly implemented, mitigating risks of data exposure or injection attacks.

4. HTTPS Security Tests

- Objective: Validate secure communication and HTTPS deployment.
- Scope:
  - Ensured HTTPS deployment checklist compliance.
  - Tested TLS configurations and certificate security (CA, expiration, renewal).
  - Verified no mixed content loading (all resources over HTTPS).
  - Tools used included SSL Labs and Mozilla Observatory.
- Outcome: HTTPS configuration met security standards, ensuring encrypted communication and preventing mixed content vulnerabilities.

RELEVANCE

Relevance:

Security testing ensured that:

- Sensitive data (passwords, personal info) is protected at rest and in transit.
- Only authorized users can access protected resources.
- System communications are encrypted and resilient against attacks.
- Headers and HTTPS configurations prevent leakage of server information and reduce attack surface.

By performing these tests, the system maintains user trust, regulatory compliance, and resilience against common security threats, which is critical for a platform handling personal wardrobe, social, and event data.

SLO

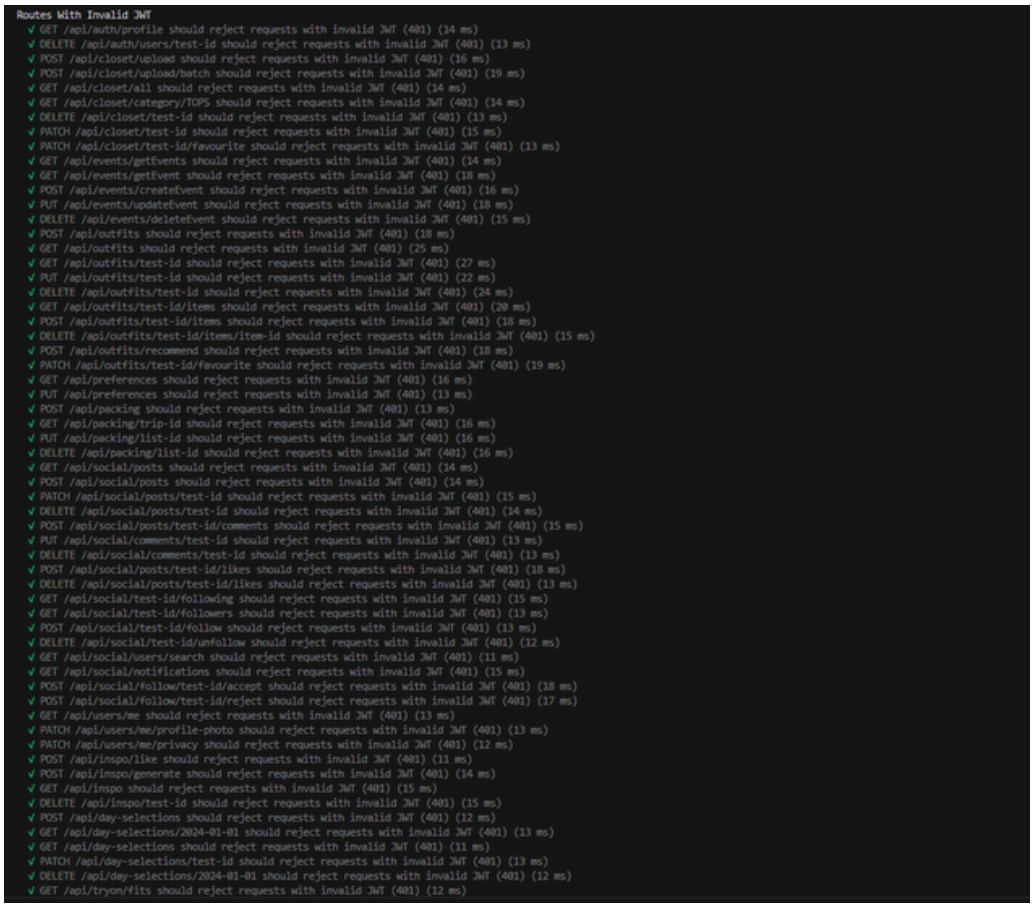
- Protected route access without JWT authentication: 0% build success
- Plaintext passwords stored in the database: 0 occurrences
- API Calls served via HTTPS should have 100% coverage.
- All security headers should be valid and present.
- Access using expired tokens should have a 0% success rate.
- No critical technical issues should be found after post-scan.

SECURITY TEST RESULTS

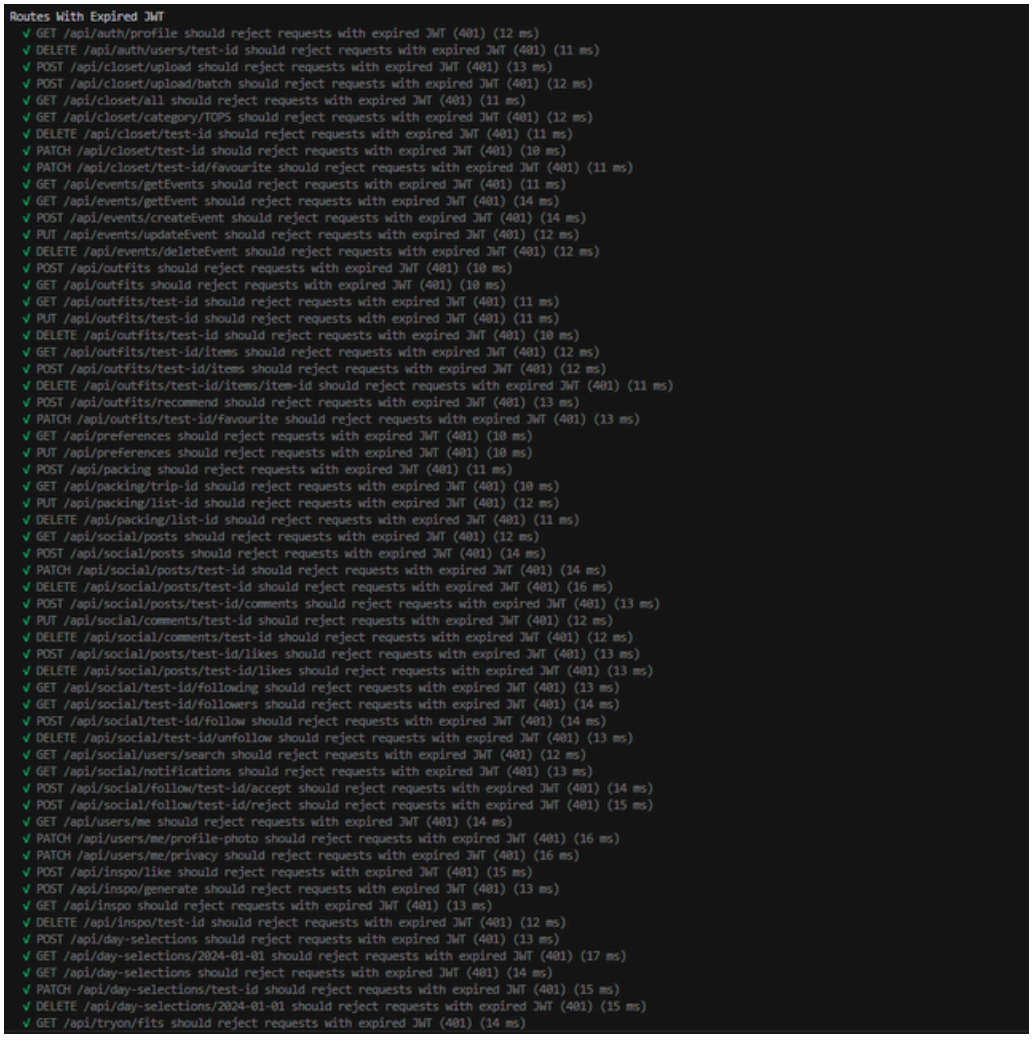
```
Security: JWT Protection Tests
  Routes Without Authorization Header
    √ GET /api/auth/profile should reject requests without Authorization header (401) (200 ms)
    √ DELETE /api/auth/users/test-id should reject requests without Authorization header (401) (38 ms)
    √ POST /api/closet/upload should reject requests without Authorization header (401) (23 ms)
    √ POST /api/closet/upload/batch should reject requests without Authorization header (401) (19 ms)
    √ GET /api/closet/all should reject requests without Authorization header (401) (26 ms)
    √ GET /api/closet/category/TOPS should reject requests without Authorization header (401) (19 ms)
    √ DELETE /api/closet/test-id should reject requests without Authorization header (401) (17 ms)
    √ PATCH /api/closet/test-id should reject requests without Authorization header (401) (16 ms)
    √ PATCH /api/closet/test-id/favourite should reject requests without Authorization header (401) (23 ms)
    √ GET /api/events/getEvents should reject requests without Authorization header (401) (19 ms)
    √ GET /api/events/getEvent should reject requests without Authorization header (401) (17 ms)
    √ POST /api/events/createEvent should reject requests without Authorization header (401) (18 ms)
    √ PUT /api/events/updateEvent should reject requests without Authorization header (401) (22 ms)
    √ DELETE /api/events/deleteEvent should reject requests without Authorization header (401) (22 ms)
    √ POST /api/outfits should reject requests without Authorization header (401) (16 ms)
    √ GET /api/outfits should reject requests without Authorization header (401) (16 ms)
    √ GET /api/outfits/test-id should reject requests without Authorization header (401) (19 ms)
    √ PUT /api/outfits/test-id should reject requests without Authorization header (401) (25 ms)
    √ DELETE /api/outfits/test-id should reject requests without Authorization header (401) (23 ms)
    √ GET /api/outfits/test-id/items should reject requests without Authorization header (401) (21 ms)
    √ POST /api/outfits/test-id/items should reject requests without Authorization header (401) (26 ms)
    √ DELETE /api/outfits/test-id/items/item-id should reject requests without Authorization header (401) (23 ms)
    √ POST /api/outfits/recommend should reject requests without Authorization header (401) (35 ms)
    √ PATCH /api/outfits/test-id/favourite should reject requests without Authorization header (401) (26 ms)
    √ GET /api/preferences should reject requests without Authorization header (401) (19 ms)
    √ PUT /api/preferences should reject requests without Authorization header (401) (20 ms)
    √ POST /api/packing should reject requests without Authorization header (401) (25 ms)
    √ GET /api/packing/trip-id should reject requests without Authorization header (401) (29 ms)
    √ PUT /api/packing/list-id should reject requests without Authorization header (401) (26 ms)
    √ DELETE /api/packing/list-id should reject requests without Authorization header (401) (26 ms)
    √ GET /api/social/posts should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/posts should reject requests without Authorization header (401) (28 ms)
    √ PATCH /api/social/posts/test-id should reject requests without Authorization header (401) (30 ms)
    √ DELETE /api/social/posts/test-id should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/posts/test-id/comments should reject requests without Authorization header (401) (20 ms)
    √ PUT /api/social/comments/test-id should reject requests without Authorization header (401) (24 ms)
    √ DELETE /api/social/comments/test-id should reject requests without Authorization header (401) (21 ms)
    √ POST /api/social/posts/test-id/likes should reject requests without Authorization header (401) (24 ms)
    √ DELETE /api/social/posts/test-id/likes should reject requests without Authorization header (401) (22 ms)
    √ GET /api/social/test-id/following should reject requests without Authorization header (401) (26 ms)
    √ GET /api/social/test-id/followers should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/test-id/follow should reject requests without Authorization header (401) (28 ms)
    √ DELETE /api/social/test-id/unfollow should reject requests without Authorization header (401) (29 ms)
    √ GET /api/social/users/search should reject requests without Authorization header (401) (23 ms)
    √ GET /api/social/notifications should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/follow/test-id/accept should reject requests without Authorization header (401) (24 ms)
    √ POST /api/social/follow/test-id/reject should reject requests without Authorization header (401) (36 ms)
    √ GET /api/users/me should reject requests without Authorization header (401) (32 ms)
    √ PATCH /api/users/me/profile-photo should reject requests without Authorization header (401) (21 ms)
    √ PATCH /api/users/me/privacy should reject requests without Authorization header (401) (20 ms)
    √ POST /api/inspo/like should reject requests without Authorization header (401) (18 ms)
    √ POST /api/inspo/generate should reject requests without Authorization header (401) (23 ms)
    √ GET /api/inspo should reject requests without Authorization header (401) (21 ms)
    √ DELETE /api/inspo/test-id should reject requests without Authorization header (401) (17 ms)
    √ POST /api/day-selections should reject requests without Authorization header (401) (19 ms)
    √ GET /api/day-selections/2024-01-01 should reject requests without Authorization header (401) (15 ms)
    √ GET /api/day-selections should reject requests without Authorization header (401) (21 ms)
    √ PATCH /api/day-selections/test-id should reject requests without Authorization header (401) (31 ms)
    √ DELETE /api/day-selections/2024-01-01 should reject requests without Authorization header (401) (21 ms)
```


```
Security: JWT Protection Tests
  Routes Without Authorization Header
    √ GET /api/auth/profile should reject requests without Authorization header (401) (200 ms)
    √ DELETE /api/auth/users/test-id should reject requests without Authorization header (401) (38 ms)
    √ POST /api/closet/upload should reject requests without Authorization header (401) (23 ms)
    √ POST /api/closet/upload/batch should reject requests without Authorization header (401) (19 ms)
    √ GET /api/closet/all should reject requests without Authorization header (401) (26 ms)
    √ GET /api/closet/category/TOPS should reject requests without Authorization header (401) (19 ms)
    √ DELETE /api/closet/test-id should reject requests without Authorization header (401) (17 ms)
    √ PATCH /api/closet/test-id should reject requests without Authorization header (401) (16 ms)
    √ PATCH /api/closet/test-id/favourite should reject requests without Authorization header (401) (23 ms)
    √ GET /api/events/getEvents should reject requests without Authorization header (401) (19 ms)
    √ GET /api/events/getEvent should reject requests without Authorization header (401) (17 ms)
    √ POST /api/events/createEvent should reject requests without Authorization header (401) (18 ms)
    √ PUT /api/events/updateEvent should reject requests without Authorization header (401) (22 ms)
    √ DELETE /api/events/deleteEvent should reject requests without Authorization header (401) (22 ms)
    √ POST /api/outfits should reject requests without Authorization header (401) (16 ms)
    √ GET /api/outfits should reject requests without Authorization header (401) (16 ms)
    √ GET /api/outfits/test-id should reject requests without Authorization header (401) (19 ms)
    √ PUT /api/outfits/test-id should reject requests without Authorization header (401) (25 ms)
    √ DELETE /api/outfits/test-id should reject requests without Authorization header (401) (23 ms)
    √ GET /api/outfits/test-id/items should reject requests without Authorization header (401) (21 ms)
    √ POST /api/outfits/test-id/items should reject requests without Authorization header (401) (26 ms)
    √ DELETE /api/outfits/test-id/items/item-id should reject requests without Authorization header (401) (23 ms)
    √ POST /api/outfits/recommend should reject requests without Authorization header (401) (35 ms)
    √ PATCH /api/outfits/test-id/favourite should reject requests without Authorization header (401) (26 ms)
    √ GET /api/preferences should reject requests without Authorization header (401) (19 ms)
    √ PUT /api/preferences should reject requests without Authorization header (401) (20 ms)
    √ POST /api/packing should reject requests without Authorization header (401) (25 ms)
    √ GET /api/packing/trip-id should reject requests without Authorization header (401) (29 ms)
    √ PUT /api/packing/list-id should reject requests without Authorization header (401) (26 ms)
    √ DELETE /api/packing/list-id should reject requests without Authorization header (401) (26 ms)
    √ GET /api/social/posts should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/posts should reject requests without Authorization header (401) (28 ms)
    √ PATCH /api/social/posts/test-id should reject requests without Authorization header (401) (30 ms)
    √ DELETE /api/social/posts/test-id should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/posts/test-id/comments should reject requests without Authorization header (401) (20 ms)
    √ PUT /api/social/comments/test-id should reject requests without Authorization header (401) (24 ms)
    √ DELETE /api/social/comments/test-id should reject requests without Authorization header (401) (21 ms)
    √ POST /api/social/posts/test-id/likes should reject requests without Authorization header (401) (24 ms)
    √ DELETE /api/social/posts/test-id/likes should reject requests without Authorization header (401) (22 ms)
    √ GET /api/social/test-id/following should reject requests without Authorization header (401) (26 ms)
    √ GET /api/social/test-id/followers should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/test-id/follow should reject requests without Authorization header (401) (28 ms)
    √ DELETE /api/social/test-id/unfollow should reject requests without Authorization header (401) (29 ms)
    √ GET /api/social/users/search should reject requests without Authorization header (401) (23 ms)
    √ GET /api/social/notifications should reject requests without Authorization header (401) (23 ms)
    √ POST /api/social/follow/test-id/accept should reject requests without Authorization header (401) (24 ms)
    √ POST /api/social/follow/test-id/reject should reject requests without Authorization header (401) (36 ms)
    √ GET /api/users/me should reject requests without Authorization header (401) (32 ms)
    √ PATCH /api/users/me/profile-photo should reject requests without Authorization header (401) (21 ms)
    √ PATCH /api/users/me/privacy should reject requests without Authorization header (401) (20 ms)
    √ POST /api/inspo/like should reject requests without Authorization header (401) (18 ms)
    √ POST /api/inspo/generate should reject requests without Authorization header (401) (23 ms)
    √ GET /api/inspo should reject requests without Authorization header (401) (21 ms)
    √ DELETE /api/inspo/test-id should reject requests without Authorization header (401) (17 ms)
    √ POST /api/day-selections should reject requests without Authorization header (401) (19 ms)
    √ GET /api/day-selections/2024-01-01 should reject requests without Authorization header (401) (15 ms)
    √ GET /api/day-selections should reject requests without Authorization header (401) (21 ms)
    √ PATCH /api/day-selections/test-id should reject requests without Authorization header (401) (31 ms)
    √ DELETE /api/day-selections/2024-01-01 should reject requests without Authorization header (401) (21 ms)
```

SECURITY TEST RESULTS



```
Routes With Invalid JWT
  √ GET /api/auth/profile should reject requests with invalid JWT (401) (14 ms)
  √ DELETE /api/auth/users/test-id should reject requests with invalid JWT (401) (13 ms)
  √ POST /api/closet/upload should reject requests with invalid JWT (401) (16 ms)
  √ POST /api/closet/upload/batch should reject requests with invalid JWT (401) (19 ms)
  √ GET /api/closet/all should reject requests with invalid JWT (401) (14 ms)
  √ GET /api/closet/category/TOPS should reject requests with invalid JWT (401) (14 ms)
  √ DELETE /api/closet/test-id should reject requests with invalid JWT (401) (13 ms)
  √ PATCH /api/closet/test-id should reject requests with invalid JWT (401) (15 ms)
  √ PATCH /api/closet/test-id/favourite should reject requests with invalid JWT (401) (13 ms)
  √ GET /api/events/getEvents should reject requests with invalid JWT (401) (14 ms)
  √ GET /api/events/getEvent should reject requests with invalid JWT (401) (18 ms)
  √ POST /api/events/createEvent should reject requests with invalid JWT (401) (16 ms)
  √ PUT /api/events/updateEvent should reject requests with invalid JWT (401) (18 ms)
  √ DELETE /api/events/deleteEvent should reject requests with invalid JWT (401) (15 ms)
  √ POST /api/outfits should reject requests with invalid JWT (401) (18 ms)
  √ GET /api/outfits should reject requests with invalid JWT (401) (25 ms)
  √ GET /api/outfits/test-id should reject requests with invalid JWT (401) (27 ms)
  √ PUT /api/outfits/test-id should reject requests with invalid JWT (401) (22 ms)
  √ DELETE /api/outfits/test-id should reject requests with invalid JWT (401) (24 ms)
  √ GET /api/outfits/test-id/items should reject requests with invalid JWT (401) (20 ms)
  √ POST /api/outfits/test-id/items should reject requests with invalid JWT (401) (18 ms)
  √ DELETE /api/outfits/test-id/items/item-id should reject requests with invalid JWT (401) (15 ms)
  √ POST /api/outfits/recommend should reject requests with invalid JWT (401) (18 ms)
  √ PATCH /api/outfits/test-id/favourite should reject requests with invalid JWT (401) (19 ms)
  √ GET /api/preferences should reject requests with invalid JWT (401) (16 ms)
  √ PUT /api/preferences should reject requests with invalid JWT (401) (13 ms)
  √ POST /api/packing should reject requests with invalid JWT (401) (13 ms)
  √ GET /api/packing/trip-id should reject requests with invalid JWT (401) (16 ms)
  √ PUT /api/packing/list-id should reject requests with invalid JWT (401) (16 ms)
  √ DELETE /api/packing/list-id should reject requests with invalid JWT (401) (16 ms)
  √ GET /api/social/posts should reject requests with invalid JWT (401) (14 ms)
  √ POST /api/social/posts should reject requests with invalid JWT (401) (14 ms)
  √ PATCH /api/social/posts/test-id should reject requests with invalid JWT (401) (15 ms)
  √ DELETE /api/social/posts/test-id should reject requests with invalid JWT (401) (14 ms)
  √ POST /api/social/posts/test-id/comments should reject requests with invalid JWT (401) (15 ms)
  √ PUT /api/social/comments/test-id should reject requests with invalid JWT (401) (13 ms)
  √ DELETE /api/social/comments/test-id should reject requests with invalid JWT (401) (13 ms)
  √ POST /api/social/posts/test-id/likes should reject requests with invalid JWT (401) (18 ms)
  √ DELETE /api/social/posts/test-id/likes should reject requests with invalid JWT (401) (13 ms)
  √ GET /api/social/test-id/following should reject requests with invalid JWT (401) (15 ms)
  √ GET /api/social/test-id/followers should reject requests with invalid JWT (401) (13 ms)
  √ POST /api/social/test-id/follow should reject requests with invalid JWT (401) (13 ms)
  √ DELETE /api/social/test-id/unfollow should reject requests with invalid JWT (401) (12 ms)
  √ GET /api/social/users/search should reject requests with invalid JWT (401) (11 ms)
  √ GET /api/social/notifications should reject requests with invalid JWT (401) (15 ms)
  √ POST /api/social/follow/test-id/accept should reject requests with invalid JWT (401) (18 ms)
  √ POST /api/social/follow/test-id/reject should reject requests with invalid JWT (401) (17 ms)
  √ GET /api/users/me should reject requests with invalid JWT (401) (13 ms)
  √ PATCH /api/users/me/profile-photo should reject requests with invalid JWT (401) (13 ms)
  √ PATCH /api/users/me/privacy should reject requests with invalid JWT (401) (12 ms)
  √ POST /api/inspo/like should reject requests with invalid JWT (401) (11 ms)
  √ POST /api/inspo/generate should reject requests with invalid JWT (401) (14 ms)
  √ GET /api/inspo should reject requests with invalid JWT (401) (15 ms)
  √ DELETE /api/inspo/test-id should reject requests with invalid JWT (401) (15 ms)
  √ POST /api/day-selections should reject requests with invalid JWT (401) (12 ms)
  √ GET /api/day-selections/2024-01-01 should reject requests with invalid JWT (401) (13 ms)
  √ GET /api/day-selections should reject requests with invalid JWT (401) (11 ms)
  √ PATCH /api/day-selections/test-id should reject requests with invalid JWT (401) (13 ms)
  √ DELETE /api/day-selections/2024-01-01 should reject requests with invalid JWT (401) (12 ms)
  √ GET /api/tryon/fits should reject requests with invalid JWT (401) (12 ms)
```



```
Routes With Expired JWT
  √ GET /api/auth/profile should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/auth/users/test-id should reject requests with expired JWT (401) (11 ms)
  √ POST /api/closet/upload should reject requests with expired JWT (401) (13 ms)
  √ POST /api/closet/upload/batch should reject requests with expired JWT (401) (12 ms)
  √ GET /api/closet/all should reject requests with expired JWT (401) (11 ms)
  √ GET /api/closet/category/TOPS should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/closet/test-id should reject requests with expired JWT (401) (11 ms)
  √ PATCH /api/closet/test-id should reject requests with expired JWT (401) (10 ms)
  √ PATCH /api/closet/test-id/favourite should reject requests with expired JWT (401) (11 ms)
  √ GET /api/events/getEvents should reject requests with expired JWT (401) (11 ms)
  √ GET /api/events/getEvent should reject requests with expired JWT (401) (14 ms)
  √ POST /api/events/createEvent should reject requests with expired JWT (401) (14 ms)
  √ PUT /api/events/updateEvent should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/events/deleteEvent should reject requests with expired JWT (401) (12 ms)
  √ POST /api/outfits should reject requests with expired JWT (401) (10 ms)
  √ GET /api/outfits should reject requests with expired JWT (401) (10 ms)
  √ GET /api/outfits/test-id should reject requests with expired JWT (401) (11 ms)
  √ PUT /api/outfits/test-id should reject requests with expired JWT (401) (11 ms)
  √ DELETE /api/outfits/test-id should reject requests with expired JWT (401) (10 ms)
  √ GET /api/outfits/test-id/items should reject requests with expired JWT (401) (12 ms)
  √ POST /api/outfits/test-id/items should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/outfits/test-id/items/item-id should reject requests with expired JWT (401) (11 ms)
  √ POST /api/outfits/recommend should reject requests with expired JWT (401) (13 ms)
  √ PATCH /api/outfits/test-id/favourite should reject requests with expired JWT (401) (13 ms)
  √ GET /api/preferences should reject requests with expired JWT (401) (10 ms)
  √ PUT /api/preferences should reject requests with expired JWT (401) (10 ms)
  √ POST /api/packing should reject requests with expired JWT (401) (11 ms)
  √ GET /api/packing/trip-id should reject requests with expired JWT (401) (10 ms)
  √ PUT /api/packing/list-id should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/packing/list-id should reject requests with expired JWT (401) (11 ms)
  √ GET /api/social/posts should reject requests with expired JWT (401) (12 ms)
  √ POST /api/social/posts should reject requests with expired JWT (401) (14 ms)
  √ PATCH /api/social/posts/test-id should reject requests with expired JWT (401) (14 ms)
  √ DELETE /api/social/posts/test-id should reject requests with expired JWT (401) (16 ms)
  √ POST /api/social/posts/test-id/comments should reject requests with expired JWT (401) (13 ms)
  √ PUT /api/social/comments/test-id should reject requests with expired JWT (401) (12 ms)
  √ DELETE /api/social/comments/test-id should reject requests with expired JWT (401) (12 ms)
  √ POST /api/social/posts/test-id/likes should reject requests with expired JWT (401) (13 ms)
  √ DELETE /api/social/posts/test-id/likes should reject requests with expired JWT (401) (13 ms)
  √ GET /api/social/test-id/following should reject requests with expired JWT (401) (13 ms)
  √ GET /api/social/test-id/followers should reject requests with expired JWT (401) (14 ms)
  √ POST /api/social/test-id/follow should reject requests with expired JWT (401) (14 ms)
  √ DELETE /api/social/test-id/unfollow should reject requests with expired JWT (401) (13 ms)
  √ GET /api/social/users/search should reject requests with expired JWT (401) (12 ms)
  √ GET /api/social/notifications should reject requests with expired JWT (401) (13 ms)
  √ POST /api/social/follow/test-id/accept should reject requests with expired JWT (401) (14 ms)
  √ POST /api/social/follow/test-id/reject should reject requests with expired JWT (401) (15 ms)
  √ GET /api/users/me should reject requests with expired JWT (401) (14 ms)
  √ PATCH /api/users/me/profile-photo should reject requests with expired JWT (401) (16 ms)
  √ PATCH /api/users/me/privacy should reject requests with expired JWT (401) (16 ms)
  √ POST /api/inspo/like should reject requests with expired JWT (401) (15 ms)
  √ POST /api/inspo/generate should reject requests with expired JWT (401) (13 ms)
  √ GET /api/inspo should reject requests with expired JWT (401) (13 ms)
  √ DELETE /api/inspo/test-id should reject requests with expired JWT (401) (12 ms)
  √ POST /api/day-selections should reject requests with expired JWT (401) (13 ms)
  √ GET /api/day-selections/2024-01-01 should reject requests with expired JWT (401) (17 ms)
  √ GET /api/day-selections should reject requests with expired JWT (401) (14 ms)
  √ PATCH /api/day-selections/test-id should reject requests with expired JWT (401) (15 ms)
  √ DELETE /api/day-selections/2024-01-01 should reject requests with expired JWT (401) (15 ms)
  √ GET /api/tryon/fits should reject requests with expired JWT (401) (14 ms)
```

SECURITY TEST RESULTS

```
Routes With Valid JWT
  √ GET /api/auth/profile should accept requests with valid JWT (not 401) (11 ms)
  √ DELETE /api/auth/users/test-id should accept requests with valid JWT (not 401) (11 ms)
  √ POST /api/closet/upload should accept requests with valid JWT (not 401) (12 ms)
  √ POST /api/closet/upload/batch should accept requests with valid JWT (not 401) (113 ms)
  √ GET /api/closet/all should accept requests with valid JWT (not 401) (47 ms)
  √ GET /api/closet/category/TOPS should accept requests with valid JWT (not 401) (38 ms)
  √ DELETE /api/closet/test-id should accept requests with valid JWT (not 401) (34 ms)
  √ PATCH /api/closet/test-id should accept requests with valid JWT (not 401) (47 ms)
  √ PATCH /api/closet/test-id/favourite should accept requests with valid JWT (not 401) (32 ms)
  √ GET /api/events/getEvents should accept requests with valid JWT (not 401) (64 ms)
  √ GET /api/events/getEvent should accept requests with valid JWT (not 401) (29 ms)
  √ POST /api/events/createEvent should accept requests with valid JWT (not 401) (33 ms)
  √ PUT /api/events/updateEvent should accept requests with valid JWT (not 401) (40 ms)
  √ DELETE /api/events/deleteEvent should accept requests with valid JWT (not 401) (26 ms)
  √ POST /api/outfits should accept requests with valid JWT (not 401) (75 ms)
  √ GET /api/outfits should accept requests with valid JWT (not 401) (51 ms)
  √ GET /api/outfits/test-id should accept requests with valid JWT (not 401) (29 ms)
  √ PUT /api/outfits/test-id should accept requests with valid JWT (not 401) (24 ms)
  √ DELETE /api/outfits/test-id should accept requests with valid JWT (not 401) (27 ms)
  √ GET /api/outfits/test-id/items should accept requests with valid JWT (not 401) (22 ms)
  √ POST /api/outfits/test-id/items should accept requests with valid JWT (not 401) (22 ms)
  √ DELETE /api/outfits/test-id/items/item-id should accept requests with valid JWT (not 401) (23 ms)
  √ POST /api/outfits/recommend should accept requests with valid JWT (not 401) (86 ms)
  √ PATCH /api/outfits/test-id/favourite should accept requests with valid JWT (not 401) (36 ms)
  √ GET /api/preferences should accept requests with valid JWT (not 401) (52 ms)
  √ PUT /api/preferences should accept requests with valid JWT (not 401) (62 ms)
  √ POST /api/packing should accept requests with valid JWT (not 401) (56 ms)
  √ GET /api/packing/trip-id should accept requests with valid JWT (not 401) (36 ms)
  √ PUT /api/packing/list-id should accept requests with valid JWT (not 401) (39 ms)
  √ DELETE /api/packing/list-id should accept requests with valid JWT (not 401) (25 ms)
  √ GET /api/social/posts should accept requests with valid JWT (not 401) (82 ms)
  √ POST /api/social/posts should accept requests with valid JWT (not 401) (65 ms)
  √ PATCH /api/social/posts/test-id should accept requests with valid JWT (not 401) (33 ms)
  √ DELETE /api/social/posts/test-id should accept requests with valid JWT (not 401) (124 ms)
  √ POST /api/social/posts/test-id/comments should accept requests with valid JWT (not 401) (59 ms)
  √ PUT /api/social/comments/test-id should accept requests with valid JWT (not 401) (41 ms)
  √ DELETE /api/social/comments/test-id should accept requests with valid JWT (not 401) (26 ms)
  √ POST /api/social/posts/test-id/likes should accept requests with valid JWT (not 401) (33 ms)
  √ DELETE /api/social/posts/test-id/likes should accept requests with valid JWT (not 401) (41 ms)
  √ GET /api/social/test-id/following should accept requests with valid JWT (not 401) (41 ms)
  √ GET /api/social/test-id/followers should accept requests with valid JWT (not 401) (38 ms)
  √ POST /api/social/test-id/follow should accept requests with valid JWT (not 401) (27 ms)
  √ DELETE /api/social/test-id/unfollow should accept requests with valid JWT (not 401) (39 ms)
  √ GET /api/social/users/search should accept requests with valid JWT (not 401) (23 ms)
  √ GET /api/social/notifications should accept requests with valid JWT (not 401) (33 ms)
  √ POST /api/social/follow/test-id/accept should accept requests with valid JWT (not 401) (30 ms)
  √ POST /api/social/follow/test-id/reject should accept requests with valid JWT (not 401) (37 ms)
  √ GET /api/users/me should accept requests with valid JWT (not 401) (17 ms)
  √ PATCH /api/users/me/profile-photo should accept requests with valid JWT (not 401) (16 ms)
  √ PATCH /api/users/me/privacy should accept requests with valid JWT (not 401) (37 ms)
  √ POST /api/inspo/like should accept requests with valid JWT (not 401) (20 ms)
  √ POST /api/inspo/generate should accept requests with valid JWT (not 401) (63 ms)
  √ GET /api/inspo should accept requests with valid JWT (not 401) (30 ms)
  √ DELETE /api/inspo/test-id should accept requests with valid JWT (not 401) (19 ms)
  √ POST /api/day-selections should accept requests with valid JWT (not 401) (38 ms)
  √ GET /api/day-selections/2024-01-01 should accept requests with valid JWT (not 401) (37 ms)
  √ GET /api/day-selections should accept requests with valid JWT (not 401) (22 ms)
  √ PATCH /api/day-selections/test-id should accept requests with valid JWT (not 401) (39 ms)
  √ DELETE /api/day-selections/2024-01-01 should accept requests with valid JWT (not 401) (30 ms)
  √ GET /api/tryon/fits should accept requests with valid JWT (not 401) (21 ms)
```


```
Routes With Valid JWT
  √ GET /api/auth/profile should accept requests with valid JWT (not 401) (11 ms)
  √ DELETE /api/auth/users/test-id should accept requests with valid JWT (not 401) (11 ms)
  √ POST /api/closet/upload should accept requests with valid JWT (not 401) (12 ms)
  √ POST /api/closet/upload/batch should accept requests with valid JWT (not 401) (113 ms)
  √ GET /api/closet/all should accept requests with valid JWT (not 401) (47 ms)
  √ GET /api/closet/category/TOPS should accept requests with valid JWT (not 401) (38 ms)
  √ DELETE /api/closet/test-id should accept requests with valid JWT (not 401) (34 ms)
  √ PATCH /api/closet/test-id should accept requests with valid JWT (not 401) (47 ms)
  √ PATCH /api/closet/test-id/favourite should accept requests with valid JWT (not 401) (32 ms)
  √ GET /api/events/getEvents should accept requests with valid JWT (not 401) (64 ms)
  √ GET /api/events/getEvent should accept requests with valid JWT (not 401) (29 ms)
  √ POST /api/events/createEvent should accept requests with valid JWT (not 401) (33 ms)
  √ PUT /api/events/updateEvent should accept requests with valid JWT (not 401) (40 ms)
  √ DELETE /api/events/deleteEvent should accept requests with valid JWT (not 401) (26 ms)
  √ POST /api/outfits should accept requests with valid JWT (not 401) (75 ms)
  √ GET /api/outfits should accept requests with valid JWT (not 401) (51 ms)
  √ GET /api/outfits/test-id should accept requests with valid JWT (not 401) (29 ms)
  √ PUT /api/outfits/test-id should accept requests with valid JWT (not 401) (24 ms)
  √ DELETE /api/outfits/test-id should accept requests with valid JWT (not 401) (27 ms)
  √ GET /api/outfits/test-id/items should accept requests with valid JWT (not 401) (22 ms)
  √ POST /api/outfits/test-id/items should accept requests with valid JWT (not 401) (22 ms)
  √ DELETE /api/outfits/test-id/items/item-id should accept requests with valid JWT (not 401) (23 ms)
  √ POST /api/outfits/recommend should accept requests with valid JWT (not 401) (86 ms)
  √ PATCH /api/outfits/test-id/favourite should accept requests with valid JWT (not 401) (36 ms)
  √ GET /api/preferences should accept requests with valid JWT (not 401) (52 ms)
  √ PUT /api/preferences should accept requests with valid JWT (not 401) (62 ms)
  √ POST /api/packing should accept requests with valid JWT (not 401) (56 ms)
  √ GET /api/packing/trip-id should accept requests with valid JWT (not 401) (36 ms)
  √ PUT /api/packing/list-id should accept requests with valid JWT (not 401) (39 ms)
  √ DELETE /api/packing/list-id should accept requests with valid JWT (not 401) (25 ms)
  √ GET /api/social/posts should accept requests with valid JWT (not 401) (82 ms)
  √ POST /api/social/posts should accept requests with valid JWT (not 401) (65 ms)
  √ PATCH /api/social/posts/test-id should accept requests with valid JWT (not 401) (33 ms)
  √ DELETE /api/social/posts/test-id should accept requests with valid JWT (not 401) (124 ms)
  √ POST /api/social/posts/test-id/comments should accept requests with valid JWT (not 401) (59 ms)
  √ PUT /api/social/comments/test-id should accept requests with valid JWT (not 401) (41 ms)
  √ DELETE /api/social/comments/test-id should accept requests with valid JWT (not 401) (26 ms)
  √ POST /api/social/posts/test-id/likes should accept requests with valid JWT (not 401) (33 ms)
  √ DELETE /api/social/posts/test-id/likes should accept requests with valid JWT (not 401) (41 ms)
  √ GET /api/social/test-id/following should accept requests with valid JWT (not 401) (41 ms)
  √ GET /api/social/test-id/followers should accept requests with valid JWT (not 401) (38 ms)
  √ POST /api/social/test-id/follow should accept requests with valid JWT (not 401) (27 ms)
  √ DELETE /api/social/test-id/unfollow should accept requests with valid JWT (not 401) (39 ms)
  √ GET /api/social/users/search should accept requests with valid JWT (not 401) (23 ms)
  √ GET /api/social/notifications should accept requests with valid JWT (not 401) (33 ms)
  √ POST /api/social/follow/test-id/accept should accept requests with valid JWT (not 401) (30 ms)
  √ POST /api/social/follow/test-id/reject should accept requests with valid JWT (not 401) (37 ms)
  √ GET /api/users/me should accept requests with valid JWT (not 401) (17 ms)
  √ PATCH /api/users/me/profile-photo should accept requests with valid JWT (not 401) (16 ms)
  √ PATCH /api/users/me/privacy should accept requests with valid JWT (not 401) (37 ms)
  √ POST /api/inspo/like should accept requests with valid JWT (not 401) (20 ms)
  √ POST /api/inspo/generate should accept requests with valid JWT (not 401) (63 ms)
  √ GET /api/inspo should accept requests with valid JWT (not 401) (30 ms)
  √ DELETE /api/inspo/test-id should accept requests with valid JWT (not 401) (19 ms)
  √ POST /api/day-selections should accept requests with valid JWT (not 401) (38 ms)
  √ GET /api/day-selections/2024-01-01 should accept requests with valid JWT (not 401) (37 ms)
  √ GET /api/day-selections should accept requests with valid JWT (not 401) (22 ms)
  √ PATCH /api/day-selections/test-id should accept requests with valid JWT (not 401) (39 ms)
  √ DELETE /api/day-selections/2024-01-01 should accept requests with valid JWT (not 401) (30 ms)
  √ GET /api/tryon/fits should accept requests with valid JWT (not 401) (21 ms)
```

SECURITY TEST RESULTS

```
PASS tests/security/password-security.security.test.ts (7.361 s)
  Security: Password Hashing Tests
    Password Hashing Implementation
      √ should hash passwords using bcrypt (179 ms)
      √ should use appropriate salt rounds (minimum 10) (170 ms)
      √ should generate different hashes for the same password (341 ms)
      √ should verify hashed passwords correctly (487 ms)
      √ should handle empty password gracefully (659 ms)
      √ should handle special characters in passwords (343 ms)
      √ should handle unicode characters in passwords (339 ms)
      √ should be case-sensitive (454 ms)
    Password Validation Rules
      √ should enforce minimum length requirement (23 ms)
      √ should require uppercase letter (9 ms)
      √ should require lowercase letter (8 ms)
      √ should require special character (9 ms)
      √ should accept valid strong password (269 ms)
    Password Storage Security
      √ should never store plaintext passwords (170 ms)
      √ should use salt for password hashing (342 ms)
    Timing Attack Prevention
      √ should take similar time for password comparison regardless of correctness (451 ms)
    Authentication Flow Security
      √ should not reveal whether email exists during login attempts (4 ms)
      √ should handle password comparison errors gracefully (10 ms)
    Hash Format Validation
      √ should handle invalid hash formats during comparison (2 ms)
      √ should handle corrupted hash gracefully (172 ms)
      √ should validate bcrypt hash format structure (170 ms)
    Edge Cases
      √ should handle very long passwords (336 ms)
      √ should handle password with only minimum requirements (347 ms)
      √ should handle concurrent hashing operations (555 ms)
```


```
PASS tests/security/password-security.security.test.ts (7.361 s)
  Security: Password Hashing Tests
    Password Hashing Implementation
      √ should hash passwords using bcrypt (179 ms)
      √ should use appropriate salt rounds (minimum 10) (170 ms)
      √ should generate different hashes for the same password (341 ms)
      √ should verify hashed passwords correctly (487 ms)
      √ should handle empty password gracefully (659 ms)
      √ should handle special characters in passwords (343 ms)
      √ should handle unicode characters in passwords (339 ms)
      √ should be case-sensitive (454 ms)
    Password Validation Rules
      √ should enforce minimum length requirement (23 ms)
      √ should require uppercase letter (9 ms)
      √ should require lowercase letter (8 ms)
      √ should require special character (9 ms)
      √ should accept valid strong password (269 ms)
    Password Storage Security
      √ should never store plaintext passwords (170 ms)
      √ should use salt for password hashing (342 ms)
    Timing Attack Prevention
      √ should take similar time for password comparison regardless of correctness (451 ms)
    Authentication Flow Security
      √ should not reveal whether email exists during login attempts (4 ms)
      √ should handle password comparison errors gracefully (10 ms)
    Hash Format Validation
      √ should handle invalid hash formats during comparison (2 ms)
      √ should handle corrupted hash gracefully (172 ms)
      √ should validate bcrypt hash format structure (170 ms)
    Edge Cases
      √ should handle very long passwords (336 ms)
      √ should handle password with only minimum requirements (347 ms)
      √ should handle concurrent hashing operations (555 ms)
```

SECURITY TEST RESULTS

```
PASS tests/security/https-security.security.test.ts
  Security: HTTPS and Transport Security Tests
    HTTPS Enforcement Documentation
      √ should document HTTPS requirements for production (5 ms)
      √ should verify current security configuration status (18 ms)
    Transport Layer Security
      √ should document TLS configuration requirements (6 ms)
    Secure Cookie Configuration
      √ should document cookie security requirements (5 ms)
      √ should verify JWT tokens are not stored in cookies without proper security (4 ms)
    Mixed Content Prevention
      √ should document mixed content prevention requirements (4 ms)
    Security Headers for HTTPS
      √ should implement Strict-Transport-Security header (19 ms)
      √ should handle protocol-relative URLs securely (6 ms)
    Certificate and Domain Security
      √ should document certificate requirements (5 ms)
      √ should document domain security practices (5 ms)
    Production HTTPS Checklist
      √ should provide comprehensive HTTPS deployment checklist (8 ms)
    Security Testing Integration
      √ should document security testing tools (16 ms)
      √ should validate current deployment is ready for production security (28 ms)
```

```
Test Suites: 4 passed, 4 total
Tests:       416 passed, 416 total
Snapshots:   0 total
Time:        32.41 s
Ran all test suites.
```

Usability

APPROACH

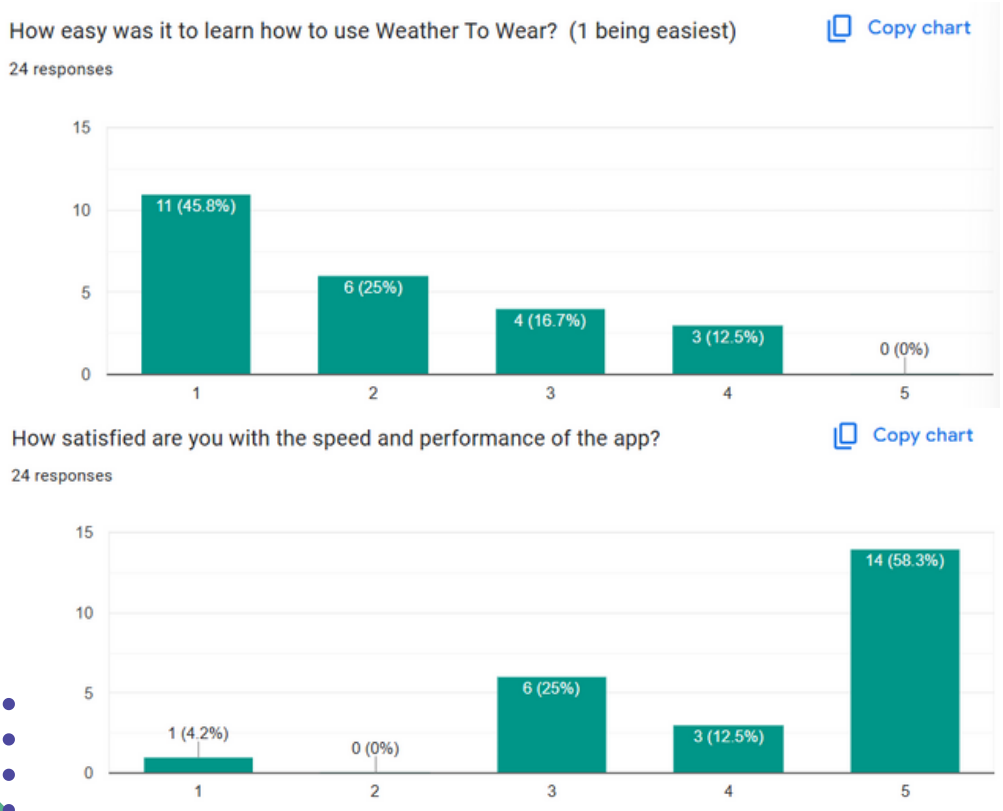
- Conducted heuristic evaluations and informal user testing with target users to assess the intuitiveness and accessibility of the interface.
- Focused on key workflows: account creation, closet management, outfit generation, event creation, and social interactions.
- Evaluated the onboarding experience, labeling, navigation, and overall satisfaction.

RELEVANCE

Usability testing ensured that users can efficiently and effectively interact with the system, increasing adoption and engagement.

RESULTS

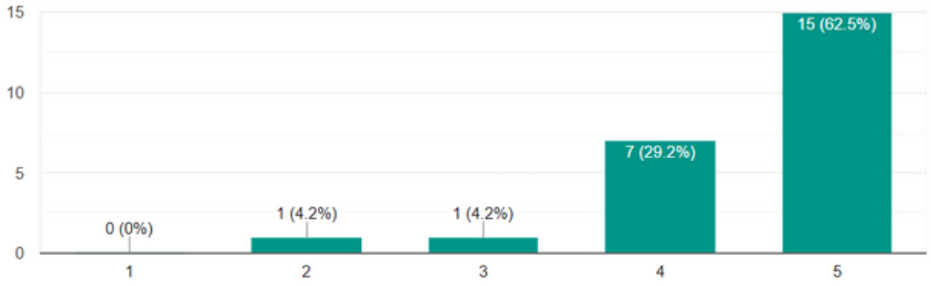
- Users found the interface intuitive for managing wardrobe items and creating outfits.
- Minor improvements were made to the closet upload interface, event creation wizard, and notification visibility.
- Onboarding tutorial enhanced understanding of virtual try-on features.



Overall, how satisfied are you with the app?

Copy chart

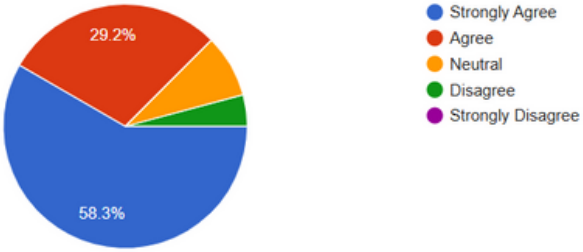
24 responses



The layout of the app is clear and intuitive.

Copy chart

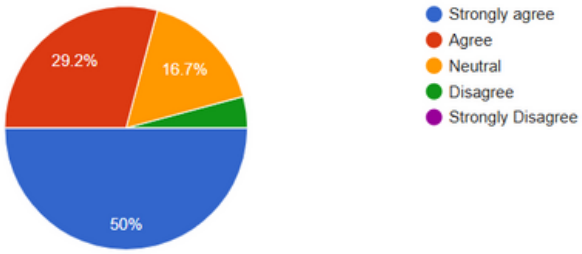
24 responses



I was able to find what I was looking for without difficulty.

Copy chart

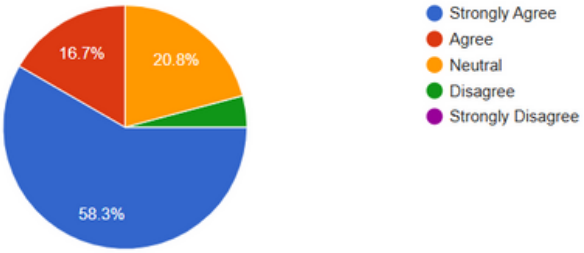
24 responses



The outfit recommendations were useful and relevant.

Copy chart

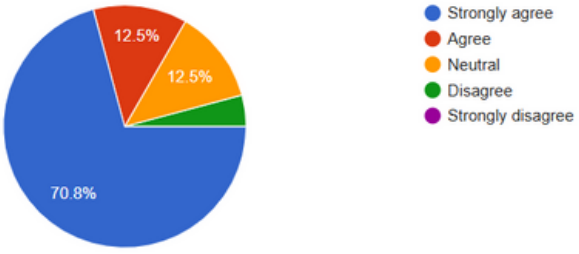
24 responses



I would continue to use this app in the future.

Copy chart

24 responses



Performance

APPROACH

- Measured API response times under varying load conditions for critical endpoints such as auth, weather, outfit recommendation, closet
- Measure end-to-end upload experience and async processing, and seed items for the smoke test.

RELEVANCE

Performance testing confirmed that the system remains responsive and efficient under heavy usage, ensuring a smooth user experience and reliable operation during peak demand.

RESULTS

API Reads:

- Closet (`GET /api/closet/all`)
  - SLO: p95 < 800 ms, p99 < 1500
  - Result: p95 = 222.34 ms, p99 = 240.74 ms
- Weather (`GET /api/weather?city=<city>`)
  - SLO: p95 < 3 500 ms, p99 < 4 000 ms
  - Result: p95 = 3 200 ms, p99 =3 220 ms
- Outfit (`POST /api/outfits/recommend`):
  - SLO: p95 < 1500 ms
  - Result: p95 = 441.04 ms

Upload Pipeline:

- e2e_total_ms (POST start → enrichment visible):
  - SLO: p50 ≤ 9 s, p95 ≤ 12 s
  - Result: p50 = 8.37 s, p95 = 8.74 s
- pipeline_total_ms (post-response → enrichment visible):
  - SLO: p50 ≤ 7 s, p95 ≤ 10 s
  - Result: p50 = 0.752 s, p95 = 0.780 s
- Error rate during burst
  - SLO: < 0.05%
  - Result: 0.00%

```
     execution: local
        script: /scripts/smoke-read.k6.js
        output: -

     scenarios: (100.00%) 1 scenario, 5 max VUs, 1m30s max duration (incl. graceful stop):
              * default: 5 looping VUs for 1m0s (gracefulStop: 30s)


  THRESHOLDS

    http_req_duration{endpoint:closet_all}
    ✓ 'p(95)<800' p(95)=222.34ms
    ✓ 'p(99)<1500' p(99)=240.74ms

    http_req_duration{endpoint:outfits_recommend}
    ✓ 'p(95)<1500' p(95)=441.04ms

    http_req_duration{endpoint:weather}
    ✓ 'p(95)<3500' p(95)=3.2s
    ✓ 'p(99)<4000' p(99)=3.22s

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


  TOTAL RESULTS

    checks_total.......: 546     8.540183/s
    checks_succeeded...: 100.00% 546 out of 546
    checks_failed......: 0.00%   0 out of 546

    ✓ weather 200
    ✓ weather repeat 200
    ✓ closet 200
    ✓ recommend 200
    ✓ recommend returns array
    ✓ recommend <=1.5s

    CUSTOM
    weather_first_ms.....................: avg=232.065934  min=183      med=190      max=842      p(90)=229      p(95)=742
    weather_repeat_ms....................: avg=2160.791209 min=199      med=3188     max=3759     p(90)=3209     p(95)=3221.5

    HTTP
    http_req_duration....................: avg=710.78ms    min=182.93ms med=212.4ms  max=3.35s    p(90)=3.19s    p(95)=3.19s
      { endpoint:closet_all }............: avg=197.73ms    min=187.12ms med=194.79ms max=242.51ms p(90)=212.26ms p(95)=222.34ms
      { endpoint:outfits_recommend }.....: avg=308.94ms    min=271.35ms med=279.36ms max=765.55ms p(90)=366.38ms p(95)=441.04ms
      { endpoint:weather }...............: avg=1.17s       min=182.93ms med=204.27ms max=3.35s    p(90)=3.19s    p(95)=3.2s
      { expected_response:true }.........: avg=710.78ms    min=182.93ms med=212.4ms  max=3.35s    p(90)=3.19s    p(95)=3.19s
    http_req_failed......................: 0.00%  0 out of 365
    http_reqs............................: 365    5.709097/s

    EXECUTION
    iteration_duration...................: avg=3.43s       min=1.11s    med=4.19s    max=5.05s    p(90)=4.9s     p(95)=4.95s
    iterations...........................: 91     1.423364/s
    vus..................................: 3      min=3        max=5
    vus_max..............................: 5      min=5        max=5

    NETWORK
    data_received........................: 2.8 MB 43 kB/s
    data_sent............................: 194 kB 3.0 kB/s



running (1m03.9s), 0/5 VUs, 91 complete and 0 interrupted iterations
default ✓ [======================================] 5 VUs  1m0s
PS C:\Users\kylea\Documents\w2w\Weather-to-Wear\app-backend>
```

```
     execution: local
        script: upload-pipeline.k6.js
        output: -

     scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
              * default: 3 iterations shared among 1 VUs (maxDuration: 10m0s, gracefulStop: 30s)


  THRESHOLDS

    e2e_total_ms
    ✓ 'p(50)<9000' p(50)=8372
    ✓ 'p(95)<12000' p(95)=8739.2

    http_req_failed
    ✓ 'rate<0.05' rate=0.00%

    pipeline_total_ms
    ✓ 'p(50)<7000' p(50)=752
    ✓ 'p(95)<10000' p(95)=780.8


  TOTAL RESULTS

    checks_total.......: 12      0.43589/s
    checks_succeeded...: 100.00% 12 out of 12
    checks_failed......: 0.00%   0 out of 12

    ✓ upload 201
    ✓ upload returns id
    ✓ upload returned id
    ✓ pipeline finished <=10s

    CUSTOM
    e2e_total_ms.................: avg=8450        min=8198     med=8372     max=8780  p(90)=8698.4 p(95)=8739.2
    pipeline_total_ms............: avg=759         min=741      med=752      max=784   p(90)=777.6  p(95)=780.8
    upload_latency_ms............: avg=7690.666667 min=7446     med=7587     max=8039  p(90)=7948.6 p(95)=7993.8

    HTTP
    http_req_duration............: avg=3.34s       min=192.05ms med=278.24ms max=7.58s p(90)=7.52s  p(95)=7.55s
      { expected_response:true }.: avg=3.34s       min=192.05ms med=278.24ms max=7.58s p(90)=7.52s  p(95)=7.55s
    http_req_failed..............: 0.00% 0 out of 7
    http_reqs....................: 7     0.254269/s

    EXECUTION
    iteration_duration...........: avg=8.87s       min=8.59s    med=8.77s    max=9.26s p(90)=9.16s  p(95)=9.21s
    iterations...................: 3     0.108972/s
    vus..........................: 1     min=1      max=1
    vus_max......................: 1     min=1      max=1

    NETWORK
    data_received................: 57 kB 2.1 kB/s
    data_sent....................: 90 kB 3.3 kB/s



running (00m27.5s), 0/1 VUs, 3 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs  00m26.6s/10m0s  3/3 shared iters
PS C:\Users\kylea\Documents\w2w\Weather-to-Wear\app-backend>
```

Scalability

APPROACH

- Conducted load testing to evaluate system performance under increasing user and data volumes.
- Simulated concurrent users making API reads and uploads to image pipeline.
- *Target traffic: ~20 concurrent users at ~5 req/s total; bursts of 6 uploads/min for 5 minutes.*

RELEVANCE

Validate that Weather to Wear keeps latency within targets as concurrency and request rates increase, and that the image upload pipeline handles short bursts without building an unhealthy backlog.

RESULTS

API Reads

- Closet (`GET /api/closet/all`):
  - SLO: p95 ≤ 400 ms, p99 ≤ 900 ms, errors < 1%
  - Result: p95 = 224.41 ms, p99 = 263.9 ms, errors = 0%
- Recommend (`POST /api/outfits/recommend`):
  - SLO: p95 ≤ 1 000 ms, errors < 1%
  - Result: p95 = 341.91 ms
- Weather (`GET /api/weather?city=...`):
  - SLO: p95 ≤ 3 500 ms
  - Result: p95 = 3 220 ms

Upload Pipeline

- e2e_total_ms (POST start → enrichment visible):
  - SLO: p50 ≤ 9 s, p95 ≤ 12 s
  - Result: p50 = 8.11 s, p95 = 8.55 s
- pipeline_total_ms (post-response → enrichment visible):
  - SLO: p50 ≤ 7 s, p95 ≤ 10 s
  - Result: p50 = 1.02 s, p95 = 1.14 s
- Error rate during burst
  - SLO: < 2%
  - Result: 0.00%

```
     execution: local
        script: burst-uploads.k6.js
        output: -

     scenarios: (100.00%) 1 scenario, 12 max VUs, 5m30s max duration (incl. graceful stop):
              * uploads: 0.10 iterations/s for 5m0s (maxVUs: 6-12, exec: uploadOne, gracefulStop: 30s)


  THRESHOLDS

    e2e_total_ms
    ✓ 'p(50)<9000' p(50)=8110.5
    ✓ 'p(95)<12000' p(95)=8559.6

    http_req_failed
    ✓ 'rate<0.02' rate=0.00%

    pipeline_total_ms
    ✓ 'p(50)<7000' p(50)=1027
    ✓ 'p(95)<10000' p(95)=1144.65


  TOTAL RESULTS

    checks_total.......: 60      0.192906/s
    checks_succeeded...: 100.00% 60 out of 60
    checks_failed......: 0.00%   0 out of 60

    ✓ upload 2xx
    ✓ upload returned id

    CUSTOM
    e2e_total_ms.....................: avg=8150.966667 min=7764    med=8110.5   max=9194  p(90)=8419.6 p(95)=8559.6
    pipeline_total_ms................: avg=959.466667  min=206     med=1027     max=1202  p(90)=1131.1 p(95)=1144.65
    upload_latency_ms................: avg=7191.233333 min=6939    med=7060.5   max=8370  p(90)=7690.7 p(95)=7718.3

    HTTP
    http_req_duration...............: avg=2.51s       min=190.7ms med=410.72ms max=7.68s p(90)=7.06s  p(95)=7.08s
      { expected_response:true }...: avg=2.51s       min=190.7ms med=410.72ms max=7.68s p(90)=7.06s  p(95)=7.08s
    http_req_failed.................: 0.00%  0 out of 92
    http_reqs.......................: 92     0.295789/s

    EXECUTION
    iteration_duration..............: avg=8.57s       min=8.04s   med=8.54s    max=9.65s p(90)=8.79s  p(95)=9s
    iterations......................: 30     0.096453/s
    vus.............................: 0      min=0       max=1
    vus_max.........................: 6      min=6       max=6

    NETWORK
    data_received...................: 758 kB 2.4 kB/s
    data_sent.......................: 215 kB 691 B/s



running (5m11.0s), 00/06 VUs, 30 complete and 0 interrupted iterations
uploads ✓ [======================================] 00/06 VUs  5m0s  0.10 iters/s
PS C:\Users\kylea\Documents\w2w\Weather-to-Wear\app-backend>
```


```
     execution: local
        script: burst-uploads.k6.js
        output: -

     scenarios: (100.00%) 1 scenario, 12 max VUs, 5m30s max duration (incl. graceful stop):
              * uploads: 0.10 iterations/s for 5m0s (maxVUs: 6-12, exec: uploadOne, gracefulStop: 30s)


  THRESHOLDS

    e2e_total_ms
    ✓ 'p(50)<9000' p(50)=8110.5
    ✓ 'p(95)<12000' p(95)=8559.6

    http_req_failed
    ✓ 'rate<0.02' rate=0.00%

    pipeline_total_ms
    ✓ 'p(50)<7000' p(50)=1027
    ✓ 'p(95)<10000' p(95)=1144.65


  TOTAL RESULTS

    checks_total.......: 60      0.192906/s
    checks_succeeded...: 100.00% 60 out of 60
    checks_failed......: 0.00%   0 out of 60

    ✓ upload 2xx
    ✓ upload returned id

    CUSTOM
    e2e_total_ms.....................: avg=8150.966667 min=7764    med=8110.5   max=9194  p(90)=8419.6 p(95)=8559.6
    pipeline_total_ms................: avg=959.466667  min=206     med=1027     max=1202  p(90)=1131.1 p(95)=1144.65
    upload_latency_ms................: avg=7191.233333 min=6939    med=7060.5   max=8370  p(90)=7690.7 p(95)=7718.3

    HTTP
    http_req_duration...............: avg=2.51s       min=190.7ms med=410.72ms max=7.68s p(90)=7.06s  p(95)=7.08s
      { expected_response:true }...: avg=2.51s       min=190.7ms med=410.72ms max=7.68s p(90)=7.06s  p(95)=7.08s
    http_req_failed.................: 0.00%  0 out of 92
    http_reqs.......................: 92     0.295789/s

    EXECUTION
    iteration_duration..............: avg=8.57s       min=8.04s   med=8.54s    max=9.65s p(90)=8.79s  p(95)=9s
    iterations......................: 30     0.096453/s
    vus.............................: 0      min=0       max=1
    vus_max.........................: 6      min=6       max=6

    NETWORK
    data_received...................: 758 kB 2.4 kB/s
    data_sent.......................: 215 kB 691 B/s



running (5m11.0s), 00/06 VUs, 30 complete and 0 interrupted iterations
uploads ✓ [======================================] 00/06 VUs  5m0s  0.10 iters/s
PS C:\Users\kylea\Documents\w2w\Weather-to-Wear\app-backend>
```