

AI-Assisted Task Management Platform

System Testing Documentation

1. Testing Overview

This document outlines the testing procedures conducted for the AI-Assisted Task Management Platform. The purpose of testing was to validate system functionality, verify database integrity, ensure session security, and confirm correct implementation of workload prioritization logic.

Testing was performed in a local development environment using XAMPP (Apache and MySQL) with PHP as the server-side language. All features were tested through manual functional testing and database verification using phpMyAdmin.

Feature	Test case	Expected Result	Actual Result	Status
Registration	Register new user	Account created successfully	Account created successfully	Passed
Login	Login with correct credentials	User redirected to dashboard	User redirected to dashboard	Passed
Login Validation	Incorrect password	Error message shown	Error message shown	Passed
Add Task	Add new task	Task saved in database	Task saved in database	Passed
Edit Task	Update task	Task updated correctly	Task updated correctly	Passed
Delete Task	Delete task	Task removed from database	Task removed	Passed
Recommendation	Ranking logic	Tasks sorted by workload score	Tasks sorted by workload score	Passed
Sorting	Sort by Priority	Tasks reordered	Tasks reordered correctly	Passed
Filtering	Filter by status	Only matching tasks displayed	Filter works	Passed

2. Authentication and Session Management Testing

The authentication module was tested to verify secure account creation and controlled system access.

The following validations were performed:

- Successful user registration inserts a new record into the `users` table.
- Passwords are stored using secure hashing mechanisms.
- Login with valid credentials establishes a session and redirects the user to the dashboard.
- Login with invalid credentials prevents access and displays appropriate feedback.
- Session variables persist during navigation across protected pages.
- Direct access to restricted pages without an active session results in redirection to the login page.
- Logout destroys the session and prevents further access without re-authentication.

All authentication flows were validated through database inspection and browser testing. No session bypass vulnerabilities were observed within the project scope.

3. CRUD Operation Testing

Full Create, Read, Update, and Delete functionality was tested to ensure correct interaction between the application layer and the database.

Create (Add Task)

- New tasks are inserted into the `tasks` table with the correct foreign key reference (`user_id`).
- All task attributes (priority, difficulty, due date, estimated hours, status) are stored accurately.
- Invalid or incomplete input is handled appropriately.

Read (View Tasks)

- Only tasks associated with the logged-in user are displayed.
- Database queries correctly filter by `user_id`.
- Tasks are retrieved in the expected order.

Update (Edit Task)

- Editing a task modifies the correct record in the database.
- Updated values are reflected immediately upon refresh.
- Users cannot edit tasks belonging to other accounts.

Delete (Remove Task)

- The selected task is permanently removed from the database.
- No orphaned records remain after deletion.
- Deletion confirmation prevents accidental removal.

All CRUD operations were verified by cross-checking the application output with the database contents.

4. Workload Balancing and Prioritization Testing

The workload balancing module was tested to validate the rule-based prioritization algorithm.

The system computes a workload score based on:

- Task priority level
- Task difficulty
- Proximity of due date
- Estimated completion hours

Testing scenarios included:

- High-priority tasks receiving greater scores than low-priority tasks.
- Tasks with closer deadlines increasing in urgency score.
- Difficulty level contributing proportionally to overall ranking.
- Completed tasks being excluded from recommendation results.
- The top-ranked task matching manual workload score calculations.

Multiple test cases were executed with varying combinations of priority, difficulty, and deadlines. The ranking output consistently reflected the expected scoring logic.

5. Sorting and Filtering Testing

The task organization module was tested to ensure dynamic query handling and correct data ordering.

Sorting was verified for:

- Due date (ascending order)
- Priority (High to Low hierarchy)
- Status (Pending, In Progress, Completed)

Filtering was verified for:

- Priority-specific filtering
- Status-specific filtering
- Combined filter scenarios
- Reset functionality restoring default task view

SQL queries were inspected to confirm proper WHERE and ORDER BY clause execution. Results displayed matched expected database subsets.

6. Data Integrity and Access Control

Additional validation was performed to ensure:

- All tasks maintain correct foreign key association with `user_id`.
- Cross-account data access is not possible.
- Database constraints prevent invalid data types.
- Session protection prevents unauthorized page access.

The system enforces basic access control and maintains consistent relational integrity between `users` and `tasks` tables.

7. User Interface and Functional Stability Testing

The user interface was evaluated for consistency and usability.

The following were confirmed:

- Navigation between pages functions correctly.
- Buttons and action links perform intended operations.
- No broken links or missing routes were detected.
- Layout consistency is maintained across major views.
- Error handling prevents system crashes during invalid operations.

The application demonstrated stable behavior under normal usage scenarios.

8. Conclusion

Testing confirmed that the AI-Assisted Task Management Platform meets the defined functional requirements. Authentication, CRUD operations, workload prioritization logic, sorting and filtering features, and session control mechanisms operate as intended.

The system demonstrates consistent database interaction, correct access control enforcement, and reliable task prioritization behavior. Within the scope of this project, the implementation is considered functionally complete and stable for deployment in a local academic environment.