High-Level Test Plan: DayStride

Test Plan Identifier: DayStride_HLTP_v1.0
Date: 03.07.2025
Owner: Edon Fetaji, Software Quality and Testing Course

Introduction

Project: DayStride – A productivity platform for habit, goal, and todo tracking.
Stack: Django REST backend + React frontend (Vite, Mantine, Tailwind).

Purpose of Testing:

- Verify that the software meets functional and non-functional requirements.
- Detect and remove defects before delivery.
- Ensure the product is reliable, secure, and user-friendly.

References:

- Project README and API Docs
- Software Requirements Specification (SRS) – available in the GitHub repository
- Lecture: *Testing in the Lifecycle (TnS_02_Testing_in_the_lifecycle.pdf)*
- ISTQB / ISEB testing standards

Test Items

Backend:

- Django REST API endpoints (auth, CRUD for habits, goals, todos, dashboard)
- Business logic in models, serializers, utilities

Frontend:

- React UI components, pages, and flows
- Contexts, hooks, and API integration

CI/CD:

- GitHub Actions workflows for automated testing, linting, and reporting
- Load testing and security testing pipelines on the staging environment

Features to be Tested

- User registration and JWT-based authentication
- Security validation of exposed web endpoints using DAST
- Backend integration with Database and Frontend
- CRUD operations for habits, goals, and todos
- Dashboard data consistency and correctness
- API validation, error handling, and permission enforcement
  UI correctness across critical pages and states
- Load handling for `/api/dashboard/` and CRUD endpoints

Non-functional aspects:

- Usability of primary flows
- Performance under expected load
- Basic security for authentication and endpoint protection

Features Not to be Tested

- Detailed UI/UX evaluations beyond core workflows
- Extensive cross-browser testing (focus on Chromium-based browsers)
- Advanced penetration testing (focus remains on core authentication flows)

Approach

Testing Levels

Component Testing:

- Backend: Model, serializer, and utility testing using pytest and pytest-django.
- Frontend: Component and hook testing using Vitest and React Testing Library.

Integration Testing:

- **In the Small:**
  - Backend: Endpoint + database testing.
  - Frontend: Components with mocked API interactions.

- **In the Large:**
  - Backend and frontend workflows using containerized environments.

System Testing:

- End-to-end user flows (registration, login, CRUD, logout) using Playwright.

UI Testing:

- Main user task-flows (registration, login, CRUD, logout) using Playwright.

Non-functional Testing:

- Load testing with `k6` on selected API endpoints.
- Security testing using OWASP ZAP on the staging environment.

Test Design Techniques

- Black-box testing for functional correctness
- Equivalence partitioning and boundary value analysis for input validation
- Statement and branch coverage tracking on critical modules

Item Pass/Fail Criteria

- **Pass:** Test case meets expected outputs and behavior.
- **Fail:** Test case does not meet expected outputs or triggers errors, logged as defects.

Suspension Criteria and Resumption Requirements

- Testing will be suspended if critical defects block progress (e.g., persistent CI failures, server unavailability).
- Testing will resume upon resolution and verification of blocking defects.

Test Deliverables

- High-Level Test Plan (this document)
- Test case specifications
- Test execution results and logs
- Coverage Reports
- Load and security test reports
- Final test summary report with coverage and outcomes

Testing Tasks

- Designing, implementing, and executing tests
- Integrating tests with CI pipelines
- Analyzing and reporting test results

Environment

- Development Environment - Dockerized development environment using github workflows
- Local testing on Windows with PyCharm - for test creation and validation
- Staging environment - Amazon Aws EC2 instance - using docker compose

Responsibilities

- **Test Lead:** Edon Fetaji
- **Tester 1 (Edon Fetaji):** Backend (unit, integration in the small & large, api testing),load and security testing **CI/CD Automation and Maintenance**
- **Tester 2 (Artan Ebibi):** Frontend (unit, integration in the small & large, api testing), Ui and System / End to End testing

Staffing and Training Needs

- Familiarity with pytest, factory_boy, and pytest-django for backend testing
- Knowledge of React testing with Vitest and React Testing Library
- Experience using Playwright for end-to-end testing
- Understanding of `k6` for load testing and OWASP ZAP for security scans

Risks and Contingencies

- **Risk:** Incomplete test coverage due to time constraints
  Mitigation: Prioritize core user flows and endpoints.

- **Risk:** CI failures from environment inconsistencies
  Mitigation: Local validation before CI runs.

Approvals

Test Plan Approved By:

- Edon Fetaji – Test Lead
- Artan Ebibi – Co-tester