Test Strategy
Spotify Testing

Author: Annisa Tri Hapsari
Alya Ananda Putri
Jena Feronika
Date: 20250601
Version: 1.1

Document Control

Document location

Location
Bandar Lampung, Lampung, Indonesia

Author

Position	Name	Contact no
Team Lead	Annisa Tri Hapsari	6288286749573
Developer	Alya Ananda Putri	6289528618632
QA Engineer	Jena Feronika	6283179898649

Stakeholders and other contributors

Position	Name

Revision history

Version	Issue date	Author/editor	Description/Summary of changes

Reviewed by

Version	Issue date	Name	Position	Review date

Approvals

Version	Issue date	Name	Position	Approval date

Related documents

Document	Location

Test Strategy Template
Spotify Testing

Table of Contents

Test Strategy Template
Spotify Testing

1. Introduction

Dokumen ini menjelaskan strategi pengujian yang akan digunakan untuk memastikan kualitas dan fungsionalitas website Spotify. Tujuan utama dari strategi ini adalah untuk mendefinisikan pendekatan, sumber daya, jadwal, dan cakupan pengujian untuk pengujian otomatis dan manual, terutama menggunakan Selenium WebDriver.

1.1. Purpose

Tujuan dari dokumen ini adalah untuk menjelaskan strategi pengujian untuk website Spotify, dengan fokus pada pengujian otomatis menggunakan Selenium WebDriver. Strategi ini bertujuan untuk memastikan bahwa semua fitur utama yang tersedia pada versi desktop website Spotify dapat diuji secara efisien, akurat, dan dapat diandalkan. Dokumen ini juga berfungsi sebagai pedoman bagi tim QA dalam merencanakan, mengembangkan, menjalankan, dan mengevaluasi pengujian.

1.2. Objective

Memastikan bahwa fitur utama Spotify berfungsi sesuai dengan spesifikasi dan dapat digunakan oleh pengguna tanpa error.

1.3. Scope

Pengujian akan difokuskan pada antarmuka pengguna (UI) dan fungsionalitas inti dari website Spotify versi desktop, seperti login, pencarian lagu, pemutaran musik, dan manajemen playlist.

1.4. Underlying Testing Principles

Strategi pengujian ini didasarkan pada prinsip-prinsip dasar berikut:

- **Early Testing**: Pengujian akan dimulai sejak tahap awal pengembangan untuk mendeteksi dan memperbaiki bug lebih awal.
- **Risk-Based Testing**: Prioritas diberikan kepada area aplikasi yang memiliki risiko paling tinggi terhadap gangguan fungsionalitas utama (contoh: login, pencarian, dan pemutaran musik).
- **Automation First**: Untuk meningkatkan efisiensi regresi testing, otomatisasi akan diprioritaskan terutama untuk alur-alur pengguna yang sering digunakan.
- **Repeatability and Reusability**: Test case dan test script akan dibuat sedemikian rupa agar dapat digunakan kembali untuk pengujian di masa depan.
- **Measurable Results**: Semua hasil pengujian akan terdokumentasi dan dianalisis secara kuantitatif untuk mengukur tingkat keberhasilan pengujian dan kualitas sistem.

2. Test Strategy

2.1. Testing Methodology

Metodologi pengujian yang digunakan adalah Agile Testing. Pengujian dilakukan secara iteratif dan berkelanjutan seiring dengan siklus pengembangan aplikasi. Setiap sprint akan mencakup fase penulisan test case, pengembangan script otomatisasi dengan Selenium, dan eksekusi pengujian.
Pendekatan yang digunakan:

- **Manual Testing** untuk eksplorasi dan skenario kompleks yang belum stabil.
- **Automation Testing** untuk regresi dan skenario berulang, menggunakan Selenium WebDriver.
- **Black-box Testing** akan digunakan untuk memverifikasi fungsionalitas tanpa melihat kode sumber.
- **Data-driven Testing** untuk validasi input/output pada form (misalnya login, pencarian).

2.2. Develop Test Plans

2.2.1. Master Test Plan

Master Test Plan menjelaskan pendekatan umum, strategi, dan sumber daya untuk semua pengujian selama proyek berlangsung. Komponen utamanya:

- **Tujuan**: Memastikan website Spotify berjalan dengan baik, tanpa bug mayor.
- **Fokus Pengujian**: Fitur login, pencarian lagu, navigasi, pemutaran musik, playlist.
- **Lingkup Pengujian**: Website versi desktop.
- **Jadwal**: Lihat bagian “Schedule”.
- **Sumber Daya**: Tim QA, Selenium, Browser Driver, dan lingkungan pengujian.
- **Risiko**: Perubahan antarmuka, waktu eksekusi lambat, ketergantungan jaringan.

2.2.2. Detailed Test Plan

Detailed Test Plan merinci langkah-langkah teknis dan operasional:

- **Test Cases**: Disusun berdasarkan fitur Spotify (contoh: "Login with valid credentials").
- **Test Data**: Disiapkan untuk input valid dan invalid.
- **Test Scripts**: Ditulis menggunakan Selenium (Python/Java) dan framework Pytest/TestNG.
- **Prioritas**: Login dan pemutaran musik sebagai prioritas utama.
- **Eksekusi**: Manual dan otomatis, dengan laporan setiap iterasi/sprint.

2.3. Test Design & Preparation

Langkah-langkah dalam test design dan persiapan:

- **Identifikasi Skenario Uji**: Misalnya login sukses, login gagal, pencarian lagu, membuat playlist.
- **Penulisan Test Case**: Dalam format Given-When-Then atau tabel.
- **Penyiapan Test Data**: Berbagai kombinasi input valid dan invalid.
- **Penulisan Script Otomatis**: Menggunakan Page Object Model (POM) untuk meningkatkan reusabilitas.
- **Validasi Awal Script**: Script diuji pada environment lokal terlebih dahulu.

Test Case	Langkah	Input	Expected Result
Login valid	1. Buka Spotify → 2. Klik login → 3. Isi email & password → 4. Klik login	Email valid & password valid	Berhasil masuk ke dashboard pengguna

2.4. Test Summary Report

Setelah seluruh pengujian selesai, Test Summary Report akan dibuat dengan isi sebagai berikut:

- **Cakupan Pengujian**: Daftar fitur yang diuji.
- **Jumlah Test Case**: Misalnya, 30 test case (20 automated, 10 manual).
- **Status Test Case**: 25 Passed, 3 Failed, 2 Blocked.
- **Bug Terdeteksi**: Total 5 bug, 4 diperbaiki, 1 open (minor).
- **Tools yang Digunakan**: Selenium WebDriver, ChromeDriver, Pytest.
- **Rangkuman Temuan**: Tidak ada bug kritis. Sistem dianggap stabil untuk dirilis.
- **Rekomendasi**: Melanjutkan regression test rutin tiap kali ada update fitur baru.

3. Testing Types

3.1. Unit / Component Testing

- **Deskripsi:** Pengujian pada level terkecil dari aplikasi – fungsi atau komponen individual – seperti validator input, tombol, atau elemen UI.
- **Dilakukan oleh:** Tim pengembang (developer).
- **Tools:** Biasanya menggunakan framework seperti JUnit, Mocha, Jest.
- **Relevansi terhadap QA:** QA tidak melakukan langsung pengujian ini, namun mengasumsikan bahwa semua unit telah teruji sebelum dilakukan pengujian sistem.
- **Contoh di Spotify:** Komponen login form validator yang mengecek apakah input email sesuai format.

3.2. System Testing

- **Deskripsi:** Pengujian terhadap sistem secara keseluruhan berdasarkan spesifikasi fungsional.
- **Fokus QA:** Memastikan bahwa seluruh fitur utama Spotify berfungsi dengan baik sebagai satu kesatuan aplikasi web.
- **Tools & Pendekatan:** Selenium digunakan untuk mengotomatisasi test case end-to-end terhadap sistem, misalnya: login, search, play, manage playlist.
- **Contoh Test Case:** "User berhasil mencari lagu dan memutarnya setelah login."

3.3. Regression Testing

- **Deskripsi:** Pengujian untuk memastikan bahwa perubahan baru (fitur atau perbaikan bug) tidak menyebabkan kerusakan pada fungsi yang sudah ada.
- **Tools:** Selenium WebDriver.
- **Strategi:** QA akan membuat automation test suite (misalnya: login, search, play music) dan menjalankannya kembali setiap terjadi perubahan besar (code update).
- **Contoh di Spotify:** Setelah UI search bar diperbarui, test otomatis memastikan bahwa fungsi pencarian lagu masih berfungsi normal.

3.4. Integration Testing

- **Deskripsi:** Memastikan bahwa modul-modul individual yang telah diuji secara unit dapat berfungsi dengan baik secara bersama-sama.
- **Dilakukan oleh:** Tim QA dengan dukungan dari tim dev.
- **Pendekatan QA (dengan Selenium):**
  - Menjalankan test case yang memverifikasi alur fungsi-fungsi saling terhubung, seperti login → load data user → tampilkan playlist.
- **Contoh di Spotify:** Setelah login, informasi akun dan daftar lagu milik pengguna harus langsung muncul dari backend melalui integrasi API.

3.5. End to End Testing of current business processes

- **Deskripsi:** Pengujian menyeluruh terhadap seluruh proses yang dilakukan oleh pengguna biasa (real user scenario).
- **Tools:** Selenium untuk pengujian otomatis dari awal hingga akhir.
- **Contoh alur bisnis Spotify:**
  1. Login ke akun
  2. Cari lagu
  3. Putar lagu
  4. Tambahkan ke playlist
  5. Logout
- **Tujuan:** Menjamin pengalaman pengguna berjalan mulus dari awal hingga akhir, tanpa bug.

3.6. Performance Testing

- **Deskripsi:** Mengukur seberapa cepat dan stabil sistem bekerja di bawah beban tertentu.
- **Catatan: Selenium tidak digunakan untuk performance testing** karena tidak dirancang untuk itu.
- **Tools yang direkomendasikan:** Apache JMeter, Google Lighthouse (untuk page load), atau WebPageTest.
- **Contoh di Spotify:** Waktu respon saat membuka halaman artist, waktu load playlist, latency saat memutar lagu.

3.7. User Acceptance Testing

- **Deskripsi:** Dilakukan oleh stakeholder atau user akhir untuk memverifikasi apakah sistem memenuhi kebutuhan bisnis dan siap dirilis.
- **Pendekatan:** QA menyediakan skenario UAT berdasarkan proses bisnis nyata, lalu user mengujinya langsung di environment staging.
- **Contoh Skenario Spotify:** Pengguna melakukan login, mencoba fitur-fitur utama (search, play, create playlist), dan mengonfirmasi bahwa aplikasi berjalan sesuai ekspektasi.

4. Testing Approach

Strategi pendekatan pengujian terhadap website Spotify ini berfokus pada kombinasi pengujian otomatis dan manual, dengan pendekatan berbasis risiko dan regresi. Pengujian dilakukan untuk memastikan semua fitur utama berjalan sesuai fungsinya dan stabil setelah adanya perubahan kode.

4.1. Testing Objectives

Tujuan utama dari kegiatan pengujian ini adalah:

- Memastikan bahwa semua fungsi inti Spotify seperti login, pencarian lagu, pemutaran musik, pengelolaan playlist, dan logout dapat berjalan tanpa kesalahan.
- Menemukan dan mendokumentasikan bug atau anomali fungsionalitas seawal mungkin sebelum sistem dirilis ke publik.
- Menjamin stabilitas dan keandalan sistem terhadap perubahan kode dengan menjalankan pengujian regresi secara berkala.
- Meningkatkan efisiensi pengujian dengan menerapkan otomatisasi untuk skenario penting dan berulang menggunakan Selenium WebDriver.
- Memastikan bahwa pengalaman pengguna berjalan end-to-end dengan lancar, termasuk integrasi dengan backend/API.

4.2. Testing Framework

Pengujian otomatis dilakukan menggunakan Selenium dalam framework yang mendukung struktur pengujian modular dan terkelola dengan baik. Komponen framework yang digunakan meliputi:

- **Tool Otomasi:** Selenium WebDriver
- **Bahasa Pemrograman:** Python (atau Java, tergantung implementasi)
- **Framework Tambahan:**
  - Pytest atau TestNG untuk manajemen test suite
  - Allure atau ExtentReports untuk reporting
  - Page Object Model (POM) sebagai pola desain untuk memisahkan logika test dan elemen UI
- **CI/CD Integration:** Jenkins (opsional) untuk menjalankan testing otomatis saat ada push kode baru.
- **Struktur Folder:**
  - /pages: berisi representasi halaman-halaman Spotify
  - /tests: berisi file test case
  - /data: test data (valid/invalid credentials, nama lagu, dsb.)

- /reports: hasil test dalam format HTML/log

4.3. Test Execution

Eksekusi pengujian dilakukan secara bertahap sebagai berikut:

- **Persiapan Data dan Environment:**
  - QA menyiapkan data uji (akun pengguna, nama lagu, playlist).
  - Memastikan environment pengujian stabil (base URL, browser, driver).
- **Eksekusi Manual:**
  - Dilakukan untuk exploratory testing dan skenario edge case yang belum otomatis.
- **Eksekusi Otomatis (Selenium):**
  - Test case prioritas tinggi dijalankan secara otomatis (login, search, play music).
  - Dijalankan melalui command line atau CI pipeline.
  - Hasil test dicatat otomatis dalam report.
- **Evaluasi Hasil Testing:**
  - QA mengevaluasi status pass/fail dari setiap test.
  - Bug yang ditemukan dilaporkan ke tim developer menggunakan bug tracker (misal Jira/Trello).
- **Re-Test & Regression:**
  - Setelah bug diperbaiki, dilakukan pengujian ulang dan regresi untuk memastikan tidak ada dampak ke fitur lain.

4.4. Testing Challenges

Dalam proses pengujian website Spotify, beberapa tantangan yang dihadapi antara lain:

- **Perubahan UI yang cepat:** Spotify kerap melakukan perubahan tampilan atau struktur DOM, sehingga script Selenium menjadi cepat usang (fragile).
- **Dynamic Content:** Banyak elemen seperti daftar lagu, iklan, dan rekomendasi bersifat dinamis, sehingga sulit diidentifikasi secara stabil oleh selector.
- **Timeout dan Latency:** Kadang respon lambat dari server memerlukan implementasi eksplisit wait dan retry mechanism.
- **Keterbatasan pada Testing Performance:** Selenium tidak cocok untuk mengukur performa secara menyeluruh, dibutuhkan tool tambahan seperti JMeter.
- **Akses terhadap akun premium:** Beberapa fitur Spotify hanya tersedia untuk akun premium, sehingga pengujian fitur tertentu memerlukan konfigurasi khusus.
- **Dependency pada koneksi internet:** Karena Spotify adalah aplikasi berbasis streaming, kestabilan jaringan sangat berpengaruh terhadap hasil pengujian.

5. Automated Testing Tools

Penggunaan alat bantu otomasi pengujian sangat penting untuk memastikan efisiensi, skalabilitas, dan akurasi dalam proses pengujian Spotify Web. Berbagai jenis tools digunakan untuk mendukung aktivitas pengujian dari manajemen, otomasi, hingga pelaporan.

5.1. Test Management

5.1.1. Test Management Tool

Tool yang digunakan: TestRail

Fungsi:

- Mendokumentasikan test case dan skenario pengujian.
- Melacak progres eksekusi test case secara terstruktur.
- Mengelompokkan test suite berdasarkan modul Spotify (login, search, playlist).

Alternatif ringan: Google Sheets atau Excel digunakan jika tim bekerja tanpa tool enterprise.

5.1.2. Defect Management Tool

- **Tool yang digunakan:** GitHub Issues
- **Fungsi:**
  - Mencatat bug yang ditemukan saat pengujian.
  - Menyediakan sistem pelacakan (tracking) dan update status dari setiap issue/defect.
  - Menghubungkan defect dengan test case terkait.
- **Workflow Umum:**
  - QA mendeteksi bug → membuat issue → tim dev memperbaiki → QA melakukan re-test dan update status.

5.2. Test Automation Tools

5.2.1. Test Automation Tool

- **Tool utama:** Selenium WebDriver
- **Bahasa pemrograman:** Python
- **Framework pendukung:**
  - Pytest atau TestNG untuk mengatur test suite dan assert logic.
  - Allure atau ExtentReports untuk laporan uji otomatis dalam format HTML.
- **Design Pattern:** Page Object Model (POM) digunakan untuk memisahkan antara logika pengujian dan struktur UI.
- **Alur eksekusi:**
  - Script disimpan di repo (GitHub/GitLab) → dieksekusi lokal atau via Jenkins → laporan test dikumpulkan → bug dicatat jika ditemukan.

5.2.2. Performance Testing Tool

Tool yang direkomendasikan:

- **Apache JMeter** – untuk mengukur waktu respon, throughput, dan concurrent user pada endpoint Spotify Web.
- **Google Lighthouse** – untuk menguji performa page load, interaktivitas, dan aksesibilitas di browser.
- **WebPageTest** – untuk benchmarking visual performance.

Catatan: Karena Selenium tidak dirancang untuk load testing atau benchmarking performa, maka digunakan tool khusus untuk skenario ini.

5.3. Licensing and Installation

Selenium: Open-source, bebas digunakan tanpa lisensi.

Test Management & Defect Tool:

- **Jira, TestRail, Zephyr:** Berbayar, biasanya tersedia dalam versi trial/edukasi untuk tim kecil.
- **Trello, Google Sheets:** Gratis, cocok untuk tim atau proyek skala kecil.

Installation & Setup:

- **Selenium WebDriver:** Diinstal melalui pip (pip install selenium) atau Maven (untuk Java).
- **Driver:** ChromeDriver atau GeckoDriver diunduh sesuai browser yang digunakan.
- **Python/Java & IDE:** VSCode, IntelliJ, atau PyCharm digunakan untuk menulis dan mengeksekusi script.
- **CI/CD Tools:** Jenkins bisa dikonfigurasi untuk menjalankan automation script secara periodik.

Dependencies: Disimpan dalam file requirements.txt (Python) atau pom.xml (Java) agar bisa direplikasi di environment lain dengan mudah.

6. Test Environment Strategy

Strategi pengelolaan environment sangat penting untuk memastikan bahwa seluruh aktivitas pengujian berlangsung dalam kondisi yang konsisten dan representatif terhadap lingkungan produksi. Dalam konteks pengujian website Spotify, pengaturan test environment dilakukan secara terstruktur untuk menunjang otomatisasi dan manual testing secara efektif.

6.1. Testing Environments Provisioning Request Process

- **Deskripsi:** Proses ini menjelaskan bagaimana QA atau tim pengujian dapat mengajukan permintaan pembuatan, konfigurasi, atau akses ke test environment yang dibutuhkan.
- **Langkah-langkah umum:**
  - QA mengajukan request melalui tiket (Jira, Trello, atau email) kepada tim DevOps atau tim IT.
  - Request mencakup detail:
    - URL environment yang dibutuhkan (dev/staging)
    - Tipe akun Spotify (free/premium/testing)
    - Browser/platform spesifik (Chrome, Firefox, Edge, mobile)
    - Test data (akun dummy, playlist, lagu favorit, dsb.)
  - DevOps menyediakan dan mengonfigurasi environment dalam waktu SLA tertentu.
  - QA melakukan verifikasi awal bahwa environment siap untuk diuji.
- **Tujuan:** Menjamin bahwa QA tidak melakukan pengujian pada environment yang tidak stabil, belum siap, atau sedang digunakan oleh tim lain.

6.1.1. Test Lab – Testing

- **Deskripsi:** Test Lab adalah kumpulan sistem atau konfigurasi yang digunakan untuk menjalankan testing, baik otomatis maupun manual.
- **Struktur Test Lab:**
  - **Local Machines:** QA dapat menjalankan script Selenium di laptop masing-masing dengan konfigurasi driver yang sesuai (ChromeDriver, GeckoDriver).
  - **Virtual Test Lab / Cloud:** Untuk pengujian lintas browser dan perangkat, digunakan layanan seperti BrowserStack, SauceLabs, atau VM internal.
  - **CI Integration (optional):** Test Lab bisa dihubungkan dengan Jenkins/GitHub Actions agar automation berjalan otomatis setelah push kode.
- **Platform yang Diuji:**
  - **Desktop Browser:** Chrome, Firefox, Edge
  - **Mobile Browser (Responsiveness):** Simulasi via Developer Tools atau emulator
- **Tujuan:** Memastikan bahwa website Spotify berfungsi konsisten di berbagai platform dan versi browser yang berbeda.

6.1.2. Test Environment Management

- **Deskripsi:** Pengelolaan environment dilakukan secara terpusat untuk menghindari konflik, gangguan, atau inkonsistensi data saat pengujian.
- **Tanggung Jawab:**
  - **Tim QA:**
    - Menjaga test data tetap bersih dan tidak bertabrakan antar penguji.
    - Menghapus akun atau playlist dummy setelah pengujian.
  - **Tim DevOps/IT:**
    - Memastikan environment staging selalu sinkron dengan produksi (tanpa data sensitif).
    - Menjaga ketersediaan dan stabilitas server/web service/API saat pengujian.
    - Memberikan log akses/error jika dibutuhkan oleh QA.
- **Praktik Baik (Best Practices):**
  - Menyediakan environment khusus regression yang tidak terganggu oleh deployment harian.
  - Menggunakan akun user dummy dengan berbagai role (free, premium).

    - Melakukan snapshot/rollback environment jika diperlukan.
- **Monitoring & Logging:** QA berkoordinasi dengan tim backend atau DevOps untuk mengakses log kesalahan selama testing (seperti log API error, response time lambat, dsb.)

7. Test Data Strategy

Test data adalah elemen penting dalam pengujian fungsional maupun otomatisasi. Strategi pengelolaan test data bertujuan memastikan bahwa seluruh skenario pengujian dapat dijalankan secara akurat, repeatable, dan bebas risiko terhadap data produksi.

7.1. Test Data Strategy Objectives

Tujuan utama dari strategi test data adalah:

- Menyediakan **data uji yang relevan dan akurat** untuk seluruh skenario pengujian (otomatis dan manual).
- Menghindari ketergantungan terhadap **data produksi** yang bersifat sensitif.
- Mendukung **repeatability** dalam eksekusi otomatis dengan data yang konsisten.
- Mengelola data untuk berbagai environment (dev, staging) secara efisien.
- Menjamin bahwa test data tersedia sesuai kebutuhan dalam waktu yang cepat.

7.2. Types of Test Data

Jenis data yang digunakan dalam pengujian Spotify antara lain:

- **Data Akun Pengguna:**
    - Akun valid (free & premium)
    - Akun invalid (salah email, password salah)
- **Data Playlist:**
    - Playlist kosong
    - Playlist dengan beberapa lagu
    - Playlist publik vs privat
- **Data Lagu:**
    - Lagu yang tersedia di region tertentu
    - Lagu yang tidak tersedia (untuk validasi error)
- **Data Skenario Negatif:**
    - Nama lagu yang tidak ada
    - Playlist tanpa nama
- **Data Dummy (Simulasi):**
    - Data akun pengguna uji otomatis (contoh: test_user_001)

7.3. Test Data Sources

Sumber test data dapat berasal dari:

- **Manual Entry:** QA membuat data dummy secara manual melalui UI Spotify.
- **CSV/Excel File:** Digunakan untuk data input skenario otomatisasi (seperti login credentials).
- **Mock Data/API:** Data dihasilkan dari mock server atau response API tiruan untuk edge case.
- **Database Query:** (Jika tersedia) mengambil langsung dari database non-produktif (read-only).
- **Data Repository:** File JSON/CSV yang disimpan dalam folder /data/ pada proyek otomatisasi.

7.4. Environment Test Data Mapping

Mapping antara environment dan test data penting untuk menghindari inkonsistensi:

Environment	Data Digunakan	Catatan
Development	Data dummy bebas, akun test internal	Tidak stabil, hanya untuk pengujian awal
Staging	Akun test valid, data dikontrol QA	Environment utama untuk validasi fungsional
Production	Tidak digunakan untuk testing langsung	Data real, hanya monitoring, tidak untuk uji

7.5. Test Data Management

- **Pengelolaan Harian:**
  - Test data disimpan dalam format terstruktur (CSV, JSON, atau database uji).
  - QA bertanggung jawab atas pembaruan dan pembersihan data uji secara berkala.
- **Kebijakan Data Sensitif:**
  - Tidak diperbolehkan menggunakan data dari production (real user).
  - Semua akun test dibuat dengan nama/email dummy dan tidak terkait user sungguhan.
- **Penerapan pada Selenium:**
  - Test script membaca data login atau playlist dari file eksternal (data-driven testing).
  - Data dipisahkan dari script untuk memudahkan penggantian dan pengujian berulang.

7.6. Test Data Provisioning Request Process

7.6.1. Applicability

- Berlaku jika QA membutuhkan data tertentu yang tidak bisa di-generate langsung via UI.
- Misalnya: akun premium dengan playlist khusus, atau data lagu region-lock.

7.6.2. Provisioning Process

- QA mengajukan request ke tim DevOps/Data melalui form/ticket/email.
- Menjelaskan:
  - Jenis data yang dibutuhkan
  - Untuk environment apa (staging/dev)
  - Format atau struktur data (CSV, DB, API)
- DevOps atau Developer menyusun data dan memberikan akses/link ke QA.
- QA memverifikasi apakah data valid untuk digunakan dalam pengujian.

7.6.3. Backup and Restore Test Data

- **Backup:**
  - Semua data uji disimpan ke dalam repository /data/backup/ setiap minggu.
  - Otomatisasi backup dilakukan melalui script (jika menggunakan database).
- **Restore:**
  - Data dapat dipulihkan dari backup sebelumnya jika terjadi corrupt/kesalahan.
  - QA dapat menjalankan script restore sebelum pengujian regresi dimulai.

8. Testing Controls & Procedures

Bab ini mendeskripsikan kontrol dan prosedur pengujian yang diterapkan untuk memastikan bahwa proses pengujian dilakukan secara terkendali, terdokumentasi, dan sesuai dengan tujuan kualitas sistem yang diuji, dalam hal ini adalah Spotify Web Application.

8.1. Testing Success Criteria

Pengujian dinyatakan berhasil apabila:

- Semua test case kritis dan mayor berhasil dijalankan tanpa defect terbuka.
- Tidak ada defect dengan severity “Critical” atau “High” yang terbuka pada saat peluncuran.
- Automated test suite minimal memiliki tingkat keberhasilan 95%.
- Semua requirement fungsional memiliki minimal satu test case yang berhasil.

8.2. Defect Management

- Defect dicatat dalam tools seperti **Jira** atau **GitHub Issues**.
- QA bertanggung jawab mencatat dengan informasi lengkap: langkah reproduksi, environment, screenshot/log.
- Setiap defect diberikan **severity** dan **priority**, lalu didistribusikan ke tim developer.
- QA akan melakukan **retesting dan regression** setelah perbaikan dilakukan.

8.2.1. Defect Management Severity Definitions

Severity	Definisi
Critical	Bug menghentikan seluruh sistem atau fungsi inti (e.g. tidak bisa login).
High	Fungsi penting gagal, namun sistem masih bisa berjalan.
Medium	Fungsi sekunder error, tanpa memengaruhi jalannya aplikasi utama.
Low	Cosmetic issues atau error minor (e.g. salah label, ikon tidak muncul).

8.2.2. Priority Code Definitions

Priority	Deskripsi
P1 (Highest)	Harus diperbaiki segera untuk mencegah delay besar.
P2	Penting, namun dapat menunggu hingga rilis minor berikutnya.
P3	Perlu diperbaiki, tapi tidak mendesak.
P4 (Lowest)	Enhancement atau low-impact defect, tidak wajib diperbaiki langsung.

8.2.3. Defect Management of Testing Completion

- Defect Log akan dianalisis di akhir siklus testing untuk menilai kestabilan sistem.
- Semua defect dengan severity Critical dan High harus ditutup untuk menyatakan testing selesai.
- QA menyusun Test Summary Report yang mencantumkan status bug sebelum rilis.

8.3. Issues Management

- Isu selain defect (contoh: environment down, tool error) dicatat terpisah sebagai issues log.
- Setiap isu ditindak oleh pemiliknya (IT/DevOps) dan harus diselesaikan sebelum milestone testing kritis.
- QA mencatat delay atau dampak dari isu terhadap eksekusi testing.

8.4. Risks Management

- Risiko dikelola sejak awal pengujian melalui risk log.
- Contoh risiko: API Spotify tidak stabil, Selenium timeout, akun diblokir karena terlalu banyak login otomatis.
- Setiap risiko dinilai dari segi probabilitas dan dampak, serta ditentukan mitigasinya.

8.5. Issue and Risk Escalation and Governance

- QA dapat melakukan eskalasi kepada Test Lead jika:
  - Isu tertunda lebih dari 24–48 jam tanpa solusi.
  - Risiko berdampak terhadap timeline proyek.
- Eskalasi dilakukan melalui email formal atau ticket escalation.

8.6. Progress Reporting

Daily Status Update via standup atau chat internal.
Weekly Progress Report mencakup:

- Jumlah test case dijalankan vs direncanakan
- Bug summary berdasarkan severity
- Progres pengujian otomatis
- Hambatan/testing blocker

Digunakan oleh Project Manager untuk mengambil keputusan rilis.

8.7. Entry Criteria Risk Assessment

Pengujian hanya boleh dimulai jika:

- Environment dan test data siap.
- Tools (Selenium, browser driver, test case) dapat diakses.
- Tidak ada risiko blocker aktif yang memengaruhi eksekusi awal.

8.8. Exit Criteria Risks Assessment

Pengujian dianggap selesai jika:

- Semua test case telah dieksekusi.
- Tidak ada defect Critical/High terbuka.
- Semua risiko telah ditutup atau ditangani dengan mitigasi yang disetujui.
- Test Summary Report sudah direview oleh stakeholder.

8.9. Testing Requirements Traceability

- Setiap test case memiliki referensi langsung ke requirement fungsional (menggunakan ID dari BRD atau user story).
- Dilacak dalam Test Management Tool (contoh: TestRail/Jira/Xray).
- Traceability Matrix disusun untuk memastikan seluruh fitur diuji.

8.10. Test Coverage Analysis

- Coverage fungsional dievaluasi dari jumlah requirement yang sudah diuji vs total.
- Coverage juga dihitung dari sisi test case automatisasi (berapa persen skenario telah diautomasi).
- Target coverage minimal:

- Functional coverage: ≥ 95%
- Automated test coverage (smoke/regression): ≥ 70%

8.11. Exception Justification

- Jika terdapat test case yang tidak dijalankan (due to tool issue, waktu tidak cukup), harus diberikan justifikasi tertulis:
  - Alasan pengabaian
  - Dampak terhadap kualitas aplikasi
  - Rencana mitigasi atau penundaan

9. Key Roles, Accountabilities and Responsibilities

Pengujian yang efektif memerlukan struktur tim yang jelas, tanggung jawab yang terdefinisi, serta koordinasi lintas peran. Tim QA harus memiliki keahlian teknis dan komunikasi yang cukup untuk mendukung keberhasilan seluruh siklus pengujian.

9.1. Proposed Test Team Structure

Peran	Tanggung Jawab Utama
Test Lead	• Menyusun rencana pengujian dan strategi pengujian. • Menetapkan prioritas pengujian dan melakukan review defect. • Mengelola timeline dan pelaporan progres pengujian kepada stakeholder.
Developer	• Mendukung pembuatan test environment dan test data. • Memperbaiki defect yang ditemukan selama proses testing. • Membantu debugging terhadap kasus yang kompleks.
QA Engineer	• Menulis test case manual dan otomatis (Selenium). • Melaksanakan semua jenis pengujian (fungsional, regresi, integrasi). • Mencatat defect dan memverifikasi perbaikannya.

10. Staffing and Training Needs

Staffing

- **1 Test Lead:** Bertanggung jawab menyusun strategi dan koordinasi tim kecil.
- **1 QA Engineer:** Menjalankan pengujian manual dan membuat skrip Selenium.
- **1 Developer:** Fokus pada perbaikan bug dan dukungan teknis.

Training Needs

- QA Engineer dan Test Lead perlu memahami:
  - Selenium WebDriver (untuk pengujian otomatis)
  - Git (untuk version control)
  - Jira atau tools pelacakan defect lainnya
  - Dasar penggunaan Spotify Web App: login, pencarian lagu, playlist, dll.
- Developer perlu siap memberikan dukungan saat ditemukan defect kompleks.

11. Milestones and Schedule

Bab ini menjelaskan tahapan utama dalam proses pengujian dan estimasi waktu pelaksanaannya. Dengan tim yang ramping, pengujian difokuskan pada area fungsional utama, stabilitas, dan regresi dasar, serta eksekusi skrip otomatisasi menggunakan Selenium.

11.1. High Level Schedule for Testing

Aktivitas	Estimasi Waktu	Keterangan
Requirement & Test Planning	2 hari	Pengumpulan spesifikasi fitur & skenario uji
Test Case Design (Manual)	3 hari	Penyusunan test case fungsional
Test Script Development (Selenium)	4 hari	Pengembangan dan pengujian skrip otomatisasi
Test Environment Setup	1 hari	Konfigurasi staging dan test data
Test Execution (Manual + Auto)	5 hari	Pelaksanaan pengujian
Bug Fixing & Regression	3 hari	Pengujian ulang setelah perbaikan bug
Test Summary & Sign-Off	1 hari	Dokumentasi hasil uji, persetujuan rilis

12. Risks and Contingencies

Risiko	Dampak	Rencana Kontinjensi
API Spotify tidak stabil atau down	Script Selenium gagal eksekusi	Tambahkan retry logic dan fallback handling
Perubahan UI Spotify mendadak	Test script otomatis gagal	Gunakan locator yang stabil, lakukan review rutin
Keterlambatan akses environment	Pengujian tertunda	Koordinasi awal dengan tim DevOps
QA kekurangan resource atau cuti	Timeline terhambat	Backup QA atau alokasi ulang tugas
Data login diblokir karena overuse	Akun uji tidak dapat digunakan	Sediakan akun dummy cadangan & rotasi akun

13. Definitions and Terms

Istilah	Definisi
Selenium	Framework open-source untuk otomatisasi pengujian UI berbasis browser.
Test Case	Deskripsi langkah pengujian terhadap fungsi tertentu.
Regression Testing	Pengujian ulang setelah adanya perubahan untuk memastikan tidak ada error baru.
Test Coverage	Persentase requirement atau kode yang telah diuji.
Bug/Defect	Kesalahan fungsi, tampilan, atau performa sistem yang menyimpang dari ekspektasi.
Severity	Ukuran dampak bug terhadap sistem.
Priority	Ukuran urgensi bug untuk diperbaiki.
Test Summary Report	Laporan akhir berisi hasil pengujian dan kesimpulan kualitas aplikasi.