Requirements Document — Coursera Testing Project

---

1. Introduction

This document defines the functional and non-functional requirements for the Coursera web automation testing project. The goal is to ensure that critical site features — such as search, filters, navigation, and course browsing — work reliably and deliver a smooth user experience.

---

2. Functional Requirements (Features to Test)

> ***Core UI & user-facing features*** *we will test (safe scope)*

1. **Homepage & Navigation**
    - Homepage loads and displays banners/promotions.
    - Top navigation menus work (Categories, Catalog, For Business, etc.).
    - Footer links (About, Help, Terms) are present and clickable.
    - Featured course cards and links navigate to course pages.
2. **Search Functionality**
    - Keyword search box accepts queries.
    - Auto-suggestions / typeahead appear and are selectable.
    - Search results page loads relevant courses for the query.
    - Pagination / infinite scroll (if present) functions correctly.
3. **Filters & Sorting**

- Filter options: Level (Beginner/Intermediate/Advanced), Language, Duration, Topic/Subject, Price (free/paid), Rating.
- Applying one or multiple filters refines results.
- Clear/Reset filters works.
- Sorting options (Relevance, Highest Rated, Newest, Popular) reorder results correctly.

4. **Course Details Page**
    - Course title, subtitle, instructor(s), institution, rating, number of reviews, duration, syllabus outline, learning objectives appear.
    - Media elements (thumbnail, preview video) load and open (no playback validation of full video).
    - Action buttons such as **Save**, **Share**, **Enroll** are present (we will not complete paid enrollment).
    - Reviews and FAQ sections load.
5. **Authentication (Login / Logout Only)**
    - Login with dedicated test account works (successful login → dashboard).
    - Invalid login shows correct error messages.
    - Logout ends session and returns user to public view.
    - Session expiration handling tested (if accessible).
6. **User Profile (Safe Fields)**
    - View profile page (name, bio, location, picture placeholder).
    - Edit non-sensitive fields (name, bio) and save changes (then revert to avoid polluting real data).
    - Validate input constraints and error messages.
7. **My Learning / Dashboard (Read-only verification)**
    - Verify that enrolled courses (if any for the test account) appear.
    - Validate navigation from dashboard to course content pages (no downloading or submitting content).

8. **API Testing (Postman / RESTful Endpoints)**
    - Course search API: send queries and verify JSON response contains expected course metadata.
    - Course details API: verify response fields (title, instructors, syllabus) match UI.
    - Filter endpoints: validate filters produce consistent results server-side.
    - Response status codes, schema validation, and response time thresholds.
9. **Static / Informational Pages**
    - FAQ, Help, About, Terms pages load and links inside those pages are valid.
10. **Accessibility & Keyboard Navigation (basic checks)**
    - Major controls reachable by keyboard (tab order).
    - Presence of ARIA attributes on dynamic widgets (where applicable).
    - Images have alt text where required.

---

3. Non-Functional Requirements

Performance

- Search results page should render within **3 seconds** under normal network conditions.
- Applying filters should update results within **2 seconds** on average.
- API GET responses for search/details should be under **1.5 seconds** ideally.

Reliability

- Core features (search, filters, login, profile view) must pass repeatedly across multiple runs with stable results.
- Tests should be deterministic where possible (use stable test accounts and seeded data).

Usability

- All primary workflows should be intuitive and maintain consistent UI behavior.
- Error messages should be clear and informative.

Compatibility

- Support and test across latest stable versions of:
  - Chrome
  - Firefox
  - Microsoft Edge

Security (test scope)

- Verify login occurs over HTTPS.
- Ensure sensitive fields are not exposed in logs (do not log passwords or tokens).
- Validate session handling (logout ends session cookie).

Accessibility

- Basic keyboard navigation checks and presence of semantic HTML/ARIA for dynamic controls.

---

4. User Stories

- **US-01**: As a *visitor*, I want to search for courses so I can find topics that interest me.
- **US-02**: As a *visitor*, I want to filter search results (by level, language) so I can reduce irrelevant results.
- **US-03**: As a *returning user*, I want to log in so I can access my dashboard.
- **US-04**: As a *user*, I want to view course details so I can evaluate course suitability.
- **US-05**: As a *tester*, I want to validate that API search responses match the UI so we ensure backend-frontend consistency.
- **US-06**: As a *tester*, I want to verify the profile edit flow (safe fields) so that data validation and UI persistency works.

---

5. Use Cases (step-by-step)

Use Case A — Login (Valid credentials)

- **Actor**: Registered user (test account)
- **Preconditions**: Test account exists and is not admin/production-critical.
- **Steps**:
    1. Navigate to Coursera login page.
    2. Enter email and password (from test credentials store).
    3. Click **Login**.
- **Postconditions / Expected**:
    - User is authenticated and redirected to the dashboard/home.
    - Session cookie is set; logout is possible.

Use Case B — Login (Invalid credentials)

- **Actor**: Visitor
- **Steps**:
    1. Navigate to login page.
    2. Enter incorrect credentials.
    3. Submit and observe response.
- **Expected**:
    - Appropriate error message shown (e.g., "Incorrect email or password").
    - No session is created.

Use Case C — Search & Filter

- **Actor**: Visitor / Student
- **Steps**:
    1. Enter keyword in search box (e.g., "Python").

2. Wait for auto-suggestions; select one or press Enter.
3. Apply filter(s) (e.g., Beginner, English).

- **Expected**:
  - Search results include relevant courses.
  - Filters narrow results; UI updates without errors.

Use Case D — View Course Details

- **Actor**: Visitor / Student
- **Steps**:
  1. From results, click a course card.
  2. Course details page loads.
- **Expected**:
  - Title, instructors, syllabus, ratings shown.
  - Buttons (Save, Share, Enroll) visible. Do not complete paid enrollment.

Use Case E — Profile Edit (Safe)

- **Actor**: Authenticated user
- **Steps**:
  1. Login with test account.
  2. Navigate to profile/edit page.
  3. Modify non-sensitive fields (e.g., display name, bio).
  4. Save changes and verify persistence.
  5. Revert changes to original values to avoid data pollution.
- **Expected**:
  - Inputs validate correctly; changes persist and are visible.

Use Case F — API Search Validation

- **Actor**: Test automation (Postman / script)
- **Steps**:
  1. Send GET request to the course search API with keyword and filter params.
  2. Parse JSON response and validate schema (title, id, instructors, rating, language).
  3. Cross-check top N results against UI results for consistency (spot-check).
- **Expected**:
  - API returns 200 OK; schema fields present.
  - Response time within threshold; data aligns with UI for tested samples.

---

6. Expected Behavior / Acceptance Criteria

- Search and filter results are accurate and relevant within acceptable response times.
- Login/logout flows operate correctly for test accounts and do not expose sensitive information.
- Course detail pages present complete and consistent data.
- API endpoints for search/details return valid JSON, match UI data, and meet response time requirements.
- Profile edits on safe fields persist and are reversible.
- Out-of-scope sensitive operations (payments, signups) are never executed during automated runs.

---

8. Deliverables

- Detailed list of functional & non-functional requirements (this document).
- Manual test cases mapped to user stories & use cases.
- Automated Selenium test suite for: search, filters, course details, login/logout, profile edits (safe).

- Postman collection for API testing (search, course details) with assertions and reports.
- Test execution reports and KPI summary.