# Capitolul 5. Testarea aplicației

Testarea reprezintă o componentă fundamentală în procesul de dezvoltare software, având rolul de a asigura calitatea, stabilitatea și funcționarea corectă a aplicației. Indiferent de cât de bine este proiectat sau dezvoltat un sistem, testarea este esențială pentru a detecta erorile, pentru a valida comportamentul aplicației în diverse scenarii și pentru a garanta o experiență coerentă și sigură pentru utilizatorii finali.

În general, testarea se împarte în două mari categorii:

* Testarea manuală, în care un tester uman verifică funcționalitățile aplicației, urmând scenarii prestabilite sau explorând liber comportamentul sistemului;
* Testarea automată, în care teste predefinite sunt executate automat prin intermediul unor framework-uri specializate (ex: JUnit, PyTest, Jasmine, PHPUnit), permițând validarea repetitivă, rapidă și fiabilă a componentelor aplicației.

Ambele tipuri de testare sunt complementare. Testarea manuală oferă flexibilitate și detectează mai ușor problemele de interfață sau de logică neprevăzută, în timp ce testarea automată asigură o verificare rapidă și constantă a funcționalităților critice, fiind ideală pentru procesele de integrare continuă și livrare rapidă.

În cadrul acestei lucrări, testarea a vizat atât partea de frontend (interfața utilizatorului), cât și backend-ul (API-ul și logica de generare a orarului), pentru a asigura buna funcționare a aplicației de la introducerea datelor până la exportul orarului final.

**5.1 Testarea manuală**

Testarea manuală a fost utilizată pentru a verifica funcționarea corectă a aplicației din perspectiva utilizatorului final. Aceasta a constat în parcurgerea interfeței grafice, introducerea datelor în diverse componente și observarea reacțiilor aplicației în diferite scenarii de utilizare. Au fost vizate componente esențiale precum autentificarea, gestionarea entităților (profesori, săli, grupe), generarea orarului și funcțiile de export și salvare.

Testele au fost realizate în mod sistematic, urmărind pași concreți și evaluând răspunsurile aplicației pentru fiecare caz.

Testul 1 – Autentificare

Scenariul 1 – Date corecte

* Pasul 1: Se deschide pagina de autentificare.
* Pasul 2: Se introduce admin / parola123.
* Pasul 3: Se apasă pe „Autentificare”.
* Rezultat: Se accesează pagina principală (dashboard).

Scenariul 2 – Parolă greșită

* Pasul 1: Se introduce admin / gresit.
* Rezultat: Se afișează mesajul: „Date de autentificare incorecte”.

Scenariul 3 – Câmpuri goale

* Pasul 1: Se lasă necompletat user/parolă.
* Rezultat: Se afișează mesajul: „Completați toate câmpurile”.

Scenariul 4 – Acces direct la pagină securizată

* Pasul 1: Se încearcă accesarea /dashboard fără autentificare.
* Rezultat: Redirecționare către login.

Testul 2 – Gestionarea profesorilor

Scenariul 1 – Adăugare validă

* Pasul 1: Se completează toate câmpurile: nume, discipline, tipuri, nivel.
* Rezultat: Profesorul apare în listă, cu mesaj de confirmare.

Scenariul 2 – Lipsă nume

* Pasul 1: Se lasă câmpul „Nume” gol.
* Rezultat: Mesaj de eroare vizibil: „Numele este obligatoriu”.

Scenariul 3 – Mai multe discipline

* Pasul 1: Se adaugă „Programare, Baze de date”.
* Rezultat: Disciplinele apar separate și corect afișate.

Testul 3 – Generarea orarului

Scenariul 1 – Generare completă

* Pasul 1: Se introduc profesori, săli, grupe, reguli.
* Pasul 2: Se accesează pagina de generare.
* Pasul 3: Se apasă „Generează orar cu AI”.
* Rezultat: Orarul este generat pentru toate grupele și zilele.

Scenariul 2 – Lipsă săli

* Pasul 1: Nu se introduc săli.
* Rezultat: Butonul de generare nu este disponibil.

Scenariul 3 – Lipsă regulă selectată

* Pasul 1: Se accesează pagina fără selectarea unei reguli.
* Rezultat: Apare Swal cu mesajul: „Regulă neselectată”.

Testul 4 – Exportul orarului

Scenariul 1 – Export PDF

* Pasul 1: Se generează un orar complet.
* Pasul 2: Se apasă pe „Exportă PDF”.
* Rezultat: Se descarcă fișierul orar.pdf.

Scenariul 2 – Export Excel

* Pasul 1: Se apasă „Exportă Excel”.
* Rezultat: Se descarcă orar.xlsx.

Scenariul 3 – Fără orar generat

* Pasul 1: Se accesează pagina fără a genera orar.
* Rezultat: Butonul de export nu apare deloc.

Scenariul 4 – Orar incomplet/invalid

* Pasul 1: Se generează orar cu date lipsă.
* Rezultat: Exportul nu este disponibil.

Testul 5 – Salvarea și încărcarea orarelor anterioare

Scenariul 1 – Salvare implicită

* Pasul 1: Se generează un orar.
* Rezultat: Orarul apare în lista „Orare salvate anterior”.

Scenariul 2 – Încărcare orar existent

* Pasul 1: Se apasă „Încarcă” pe un orar din listă.
* Rezultat: Orarul se afișează în aplicație.

Scenariul 3 – Editare orar

* Pasul 1: Se apasă „Editează” pe un orar salvat.
* Rezultat: Datele pot fi modificate și salvate din nou.

Scenariul 4 – Ștergere orar

* Pasul 1: Se apasă „Șterge” și se confirmă.
* Rezultat: Orarul este eliminat din listă.

Scenariul 5 – Căutare orar

* Pasul 1: Se caută după nume.
* Rezultat: Lista este filtrată instant.

## 5.2. Testarea automată

Testarea automată reprezintă un proces esențial în dezvoltarea aplicațiilor software moderne, asigurând validarea funcționalităților sistemului fără intervenție umană. În cadrul acestui proiect, testarea automată a fost utilizată pentru a verifica corectitudinea componentelor backend și frontend ale aplicației web pentru generarea orarelor universitare, contribuind astfel la creșterea fiabilității și mentenanței codului.

Scopul principal al testării automate este de a detecta rapid erorile și regresiile, facilitând dezvoltarea iterativă și introducerea de noi funcționalități fără riscul deteriorării celor existente. În plus, testele automate oferă un grad ridicat de încredere în comportamentul aplicației în scenarii diverse.

Pentru acest proiect, testele automate au fost structurate în două mari categorii:

* Testarea backendului: realizată cu ajutorul bibliotecii pytest în Python, a vizat testarea rutelor API, interacțiunii cu baza de date, integritatea datelor și răspunsurile corecte la solicitările HTTP.
* Testarea frontendului: realizată cu ajutorul bibliotecii Jest și a utilitarului React Testing Library, a urmărit verificarea componentelor vizuale, interacțiunilor utilizatorului, validarea formularelor și afișarea corectă a datelor în interfață.

Testarea automată a permis dezvoltarea controlată a aplicației și a contribuit la menținerea calității codului pe parcursul întregului ciclu de dezvoltare. În continuare, vor fi prezentate testele relevante implementate pentru componentele backend și frontend, împreună cu explicațiile aferente.

### 5.2.1 Testarea backend

Testarea backendului a fost realizată cu ajutorul frameworkului pytest, o bibliotecă Python eficientă și extensibilă, folosită pentru a valida corectitudinea logicii aplicației. Testele au fost organizate pe fișiere separate, în funcție de funcționalitățile testate, fiecare acoperind o zonă esențială a aplicației:

* rutare și API (endpoints)
* interacțiuni cu baza de date
* reguli de generare și validare a orarului

### 5.2.1 Testarea funcționalității de generare orar

Fișierul test\_api\_generator.py include testele unitare pentru rutele API responsabile de generarea orarului, din blueprint-ul generator\_bp. Testele au fost realizate cu pytest și Flask test\_client, permițând simularea cererilor HTTP fără rularea serverului.

Au fost verificate următoarele scenarii:

* GET /genereaza\_orar\_propriu – pagina de generare se afișează corect.
* POST valid – trimite date complete și primește un orar valid.
* POST invalid – trimite date lipsă și primește eroare 400.

Toate testele au fost executate cu succes, validând funcționarea corectă a API-ului în scenarii diverse.

#### A screenshot of a computer AI-generated content may be incorrect.

5.2.2 Testarea gestionării profesorilor – test\_api\_profesori.py

Fișierul test\_api\_profesori.py conține teste automate pentru operațiile CRUD legate de profesori (creare, listare, actualizare, ștergere). Testele au fost realizate cu pytest și Flask test\_client, simulând cereri HTTP fără rularea serverului.

Testele acoperă:

* Adăugare profesor cu date complete (inclusiv discipline și disponibilitate).
* Verificare în listă după adăugare (/toti\_profesorii).
* Actualizare profesor existent (nume, activități, disponibilitate).
* Ștergere profesor pentru curățare (cleanup).

Toate testele au trecut cu succes, validând funcționarea corectă a backendului pentru gestionarea profesorilor.

#### A screenshot of a computer AI-generated content may be incorrect.

5.2.3 Testarea interacțiunii directe cu baza de date – test\_database.py

Fișierul test\_database.py verifică operațiile SQL fundamentale folosind clasa personalizată Database, fără a implica API-ul Flask.

Testele acoperă:

* test\_show\_tables – verifică existența tabelei profesori.
* test\_select\_profesori – validează interogările SELECT și structura rezultatelor.
* test\_insert\_and\_delete\_profesor – simulează adăugarea și ștergerea unui profesor de test, confirmând integritatea operațiilor.

Toate testele au fost rulate cu succes, confirmând funcționarea corectă a conexiunii și manipulării datelor în MySQL.

#### A screenshot of a computer AI-generated content may be incorrect.

5.2.4 Testarea generatorului propriu de orar – test\_orar\_generator.py

Fișierul test\_orar\_generator.py conține teste pentru clasa OrarGenerator, care implementează logica de generare personalizată a orarului. Au fost utilizate tehnici de mocking (cu unittest.mock) pentru a izola logica de baza de date.

Testele acoperă:

* Inițializarea corectă a generatorului cu reguli implicite (pauză miercuri, max. 8h/zi).
* Extragerea nivelului și anului din denumirea grupei (ex: LM2a → Master, anul II).
* Actualizarea criteriilor (ore pe zi, pauze, reguli).
* Generarea efectivă a orarului conform regulilor (inclusiv pauza de miercuri).
* Limitarea numărului de cursuri diferite (ex: max. 9).

Toate testele au trecut, confirmând că logica de generare funcționează conform cerințelor și poate fi configurată flexibil.

#### A screenshot of a computer AI-generated content may be incorrect.

5.2.5 Testarea validării orarului generat – test\_validator\_orar.py

Fișierul test\_validator\_orar.py conține teste pentru clasa ValidatorOrar, care verifică dacă cursurile comune sunt sincronizate corect între grupele unui an.

Testele acoperă:

* Sincronizare validă – același curs apare în același interval, zi și sală pentru toate grupele (ex: LI1a și LI1b).
* Sincronizare invalidă – același curs apare în zile diferite la grupe diferite, iar validarea semnalează eroarea.

Ambele teste au fost rulate cu succes, demonstrând că sistemul detectează atât cazurile corecte, cât și erorile de aliniere.

#### A screenshot of a computer AI-generated content may be incorrect.

5.3 Testarea frontend

Testarea automată a componentelor frontend are un rol esențial în asigurarea funcționării corecte a interfeței utilizator. Într-o aplicație modernă construită cu React, testele frontend validează dacă elementele vizuale sunt afișate corect, dacă interacțiunile utilizatorului produc rezultatele așteptate și dacă starea aplicației se actualizează corespunzător în urma acestor interacțiuni.

Pentru acest proiect, testarea frontend a fost realizată folosind biblioteca Jest, împreună cu React Testing Library, două dintre cele mai populare unelte pentru testarea componentelor React. Acestea permit simularea comportamentului real al utilizatorului (de exemplu: apăsarea butoanelor, completarea câmpurilor, selectarea opțiunilor din formulare), precum și validarea conținutului vizual afișat în pagină.

Testele au fost organizate pe fișiere separate, câte unul pentru fiecare componentă importantă din aplicație, cum ar fi:

* pagina principală (Home.test.jsx),
* formularul de introducere a profesorilor (Profesori.test.jsx),
* formularul de setare reguli (SetareReguli.test.jsx),
* pagina de generare și afișare a orarului (GeneratedTimetable.test.jsx).

În cadrul fiecărui test, s-au verificat:

* prezența în DOM a elementelor cheie (titluri, butoane, formulare),
* comportamentul interactiv al utilizatorului,
* reacția componentelor la date de intrare și acțiuni,
* eventuale mesaje de eroare sau confirmare.

Prin această suită de teste, se asigură că aplicația răspunde corect la acțiunile utilizatorului, iar modificările aduse interfeței nu introduc erori de afișare sau funcționare.

5.3.1 Testarea componentei GeneratedTimetable.jsx

Componenta GeneratedTimetable este una dintre cele mai complexe, responsabilă pentru generarea, afișarea, validarea și gestionarea orarelor universitare. Testarea s-a realizat cu Jest și React Testing Library, utilizând mock-uri pentru toate hook-urile logice (useOrarGenerator, useValidareOrar, useOrarSalvat etc.).

Testele acoperă:

* Afișarea titlului și butoanelor principale: „Generează cu AI” și „Generează clasic”.
* Apelul corect al funcțiilor genereazaOrar și genereazaOrarClasic la click.
* Funcționarea butoanelor auxiliare: „Reîncarcă”, „Înapoi”.
* Afișarea listei de orare salvate (cu opțiuni de editare, ștergere, încărcare).
* Filtrarea orarelor după denumire.

Toate testele au trecut cu succes, validând comportamentul complet și robust al acestei componente critice.

#### A screenshot of a computer AI-generated content may be incorrect.

5.3.2 Testarea componentei Home.jsx

Componenta Home.jsx este pagina principală și afișează conținut diferit în funcție de starea de autentificare a utilizatorului. Testele au fost realizate cu Jest și React Testing Library, folosind mock pentru logica de autentificare (useHomeLogic).

Testele acoperă:

* Utilizator neautentificat: titlul aplicației, butonul „Autentifică-te”, mesaj informativ, buton de generare dezactivat.
* Utilizator autentificat: mesaj de bun venit, butoanele „Logout” și „Orarul meu”, butonul de generare activ și mesajul de redirecționare.

Toate testele au fost executate cu succes, confirmând afișarea corectă a interfeței în ambele scenarii.

#### A screenshot of a computer AI-generated content may be incorrect.

5.3.3 Testarea componentei Profesori.jsx

Testarea componentei Profesori – Profesori.test.jsx

Componenta Profesori.jsx permite adăugarea și configurarea profesorilor (nume, discipline, nivel, tip activitate, disponibilitate). Este una dintre cele mai complexe componente, motiv pentru care a fost testată automat cu Jest și React Testing Library, folosind mock pentru useProfesoriLogic.

Testele acoperă:

* Afișarea formularului și titlului principal.
* Comportamentul corect când lista este goală (*„Niciun profesor găsit”*).
* Funcționalitatea butonului „Salvează profesor” (apelează adaugaProfesor).
* Posibilitatea de a adăuga mai multe discipline.
* Afișarea tabelului de selecție a disponibilității săptămânale.

Toate testele au trecut cu succes, confirmând funcționalitatea completă a interfeței și reacția corectă la diverse stări ale aplicației.

#### A screenshot of a computer AI-generated content may be incorrect.

5.3.4 Testarea componentei SetareReguli.jsx

Componenta SetareReguli permite utilizatorului să definească, salveze și reutilizeze reguli personalizate în format JSON pentru generarea orarului. Testarea s-a realizat cu Jest și React Testing Library, folosind mock pentru useSetariReguli.

Testele acoperă:

* Afișarea titlului și descrierii componentei.
* Prezența câmpului pentru denumirea regulii (cu placeholder sugestiv).
* Funcționarea butonului „Salvează” (apelează funcția salveazaReguli).
* Afișarea mesajului *„Nu există reguli salvate”* când lista este goală.

Toate testele au fost executate cu succes, validând afișarea și interactivitatea componentei în toate scenariile relevante.

#### A screenshot of a computer AI-generated content may be incorrect.