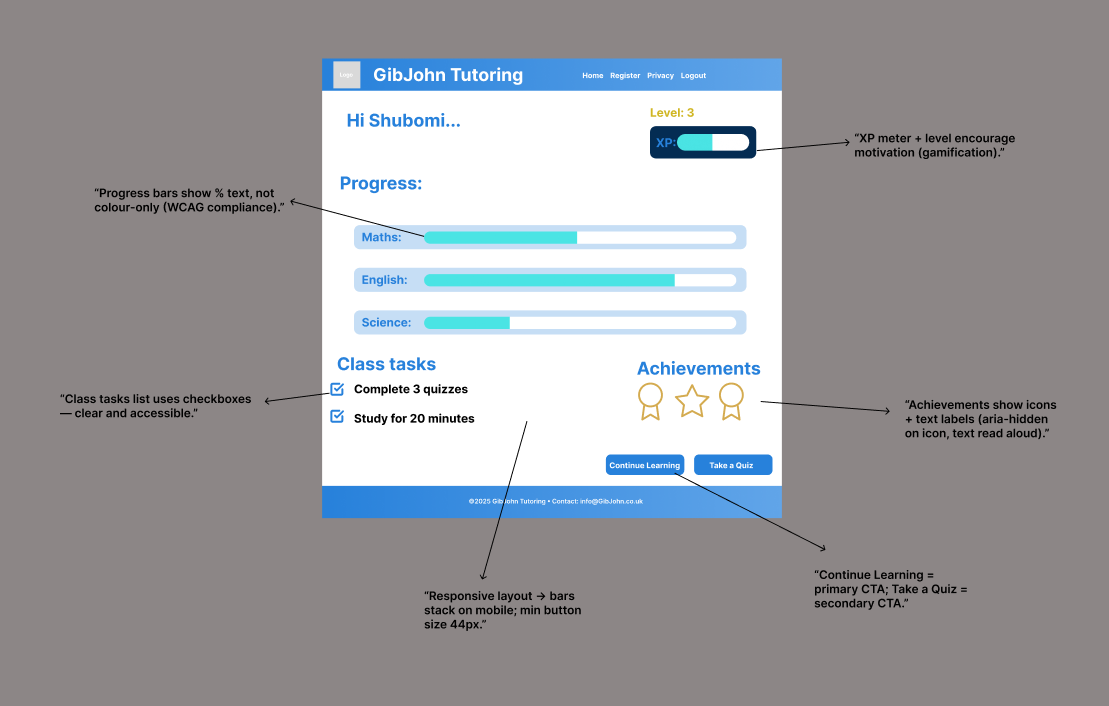
**t**

**Part B –**

1. **Design**

****

**Login**

Purpose: sign in quickly and securely.

Elements

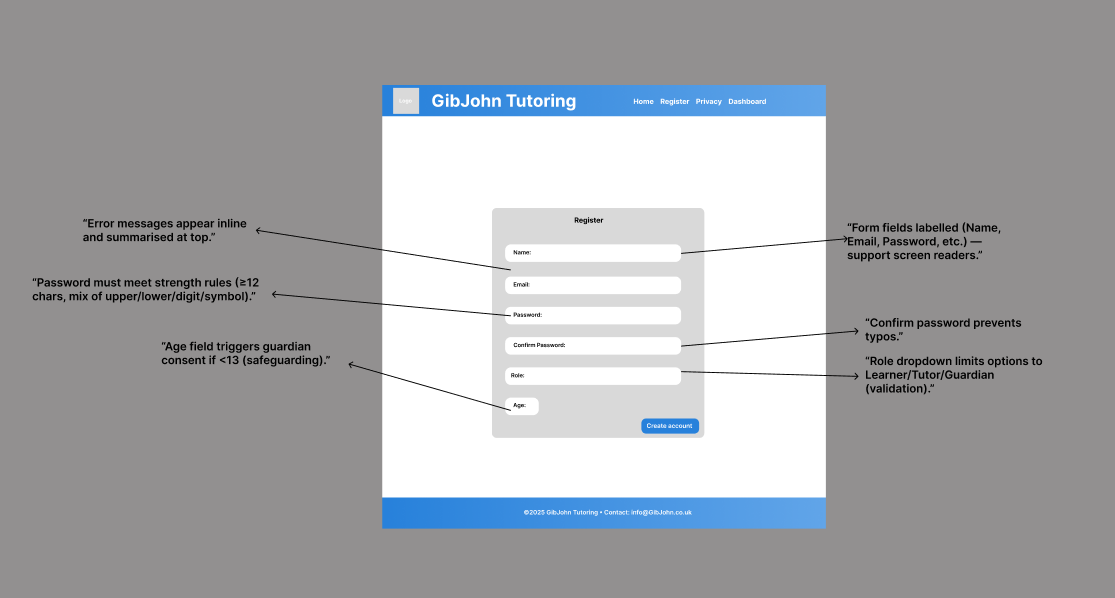
* Email + Password; “Show password;” “Remember me;” “Forgot password?”
* Secondary link to Register.

**UX & accessibility**

* On error, say “Email or password incorrect” (don’t reveal which).
* Keyboard shortcuts: Enter to submit; logical tab order.
* Focus returns to the first invalid field with aria-describedby message.

Security

* Brute-force protection; lockout with gentle messaging; generic error responses.
* Session cookie HttpOnly, Secure, SameSite=Lax.

****

**Register**

Purpose: create account with minimum friction and strong validation.

Fields & rules

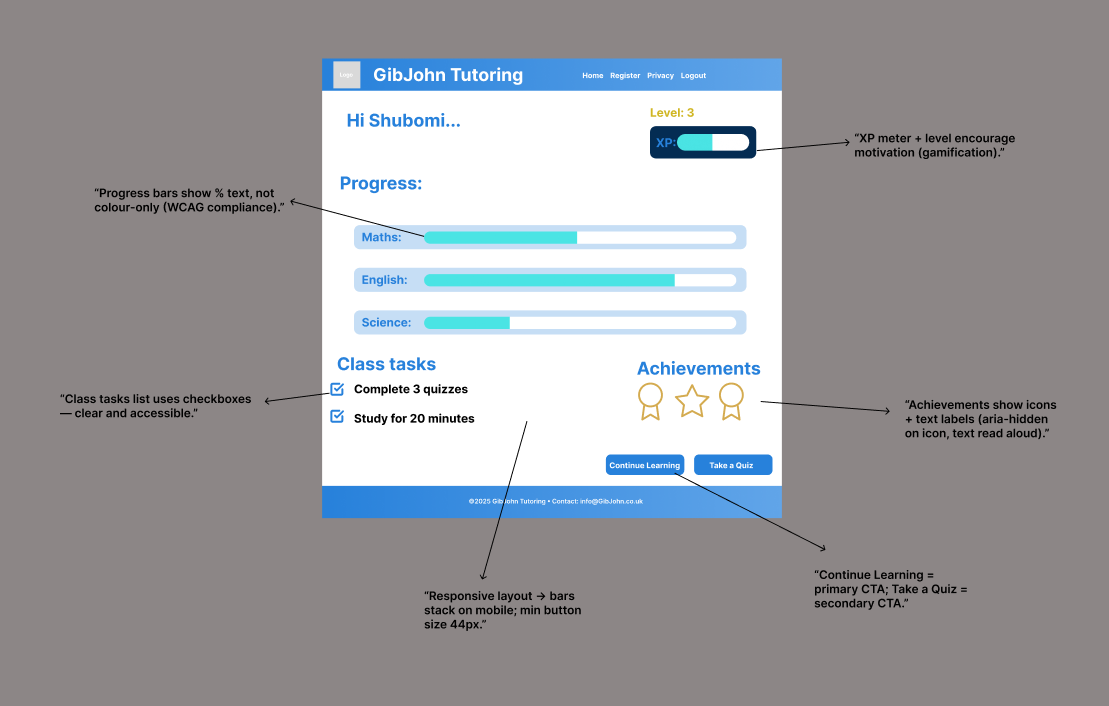
* Name (letters, spaces, hyphen, 2–50 chars).
* Email (format + uniqueness check).
* Password (≥12 chars, 1 upper, 1 lower, 1 digit, 1 symbol).
* Confirm password (match).
* Role (Learner / Tutor / Guardian).
* Age (integer; if <13 then request guardian consent).

**UX & accessibility**

* Inline validation messages; show requirements before typing.
* Password reveal toggle; strength meter with text (e.g., “Strong”).
* Error summaries above form; labels tied to inputs; autocomplete hints.

Security & privacy

* CSRF token; rate-limit; no password in URL; link to concise Privacy Policy.
* Consent checkboxes (privacy/terms) with clear language.

****

**Learner Dashboard**

Purpose: show progress by subject, XP/Level, class tasks, achievements, main actions.

Layout & hierarchy

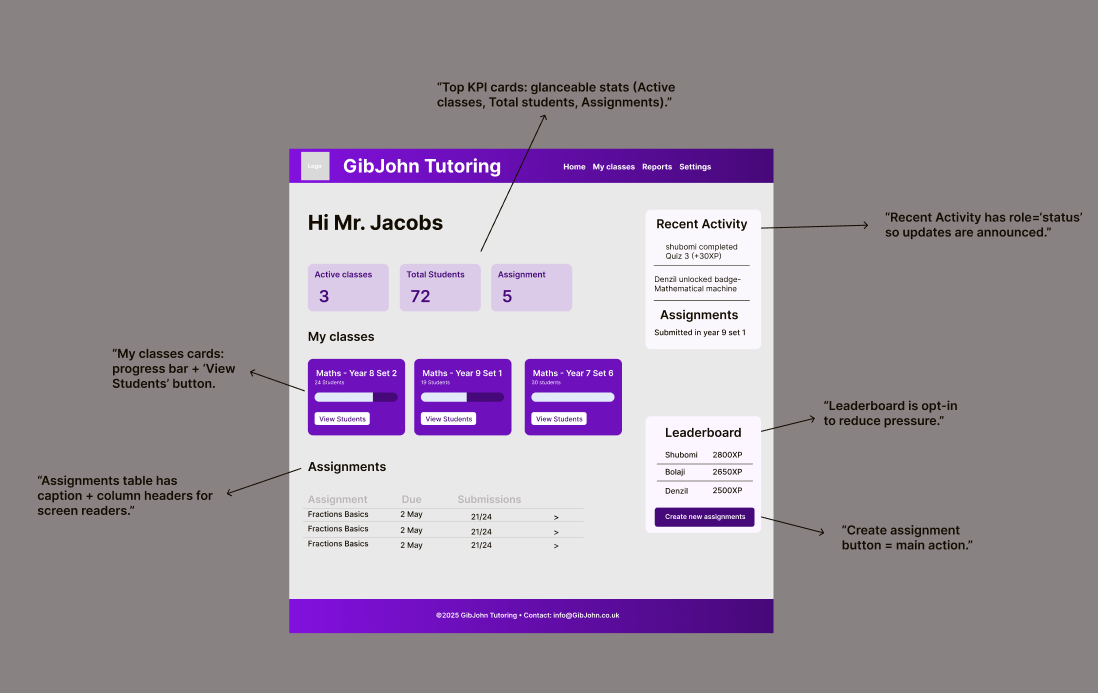
* Header (logo + Home, Register, Privacy, Logout).
* H1 greeting (“Hi Shubomi…”).
* Progress bars (Maths, English, Science) with labels and numeric % (not colour alone).
* XP meter + Level (badge).
* “Class tasks” checklist; “Achievements” icons.
* Primary CTAs: “Continue learning,” “Take a Quiz.”

**Accessibility**

* Progress bars: include aria-valuenow/min/max, and % text.
* Icons have aria-hidden="true" with text labels adjacent.
* Clear focus outlines; button sizes ≥ 44px target.
* Responsive: bars become vertical or full-width stacked on mobile.

**Motivation & ethics**

* Achievements reinforce effort, not only score; allow opt-out of leaderboard.

****

**Tutor Dashboard**

Purpose: quick class health, recent activity, assignments, leaderboard.  
Layout & hierarchy

* Header: logo (top-left), primary nav (Home, My classes, Reports, Settings).
* H1 page title: “Hi Mr. Jacobs” (clearest heading).
* “KPI cards” (Active classes / Total students / Assignments) = primary glanceable stats.
* “My classes” cards with progress bars + “View students” CTAs.
* Right rail: “Recent Activity” (alerts), “Leaderboard” (motivation), “Create new assignments” CTA.
* Table: “Assignments” with Due / Submissions / chevron.

**Conventions & patterns**

* Cards for metrics; table for submissions; chevrons (>) indicate drill-in.
* Link styling for “View students;” button styling for “Create new assignments.”

**Accessibility (WCAG 2.1 AA)**

* Colour contrast ≥ 4.5:1 for text on p purple; focus states on all links/buttons; skip-to-content link.
* ARIA: role="status" for Recent Activity; table has <caption> and < scope="col">.
* Keyboard: tab order left→right, top→bottom; no keyboard traps.
* Responsive: cards wrap to 2-up/1-up; right rail stacks below main content on mobile.

Content checks

* Microcopy: clear labels (“Assignments,” “View students”).
* Feedback: empty states (e.g., “No recent activity yet”).
* Privacy: do not show full student names in public areas.

1. **Data Design**

The data structure for the applications database is an important consideration that must be considered carefully, this is due to both technical and legal reasons. For technical it must be simple to adapt and expand upon in the future to allow for new features to be added, along with being well designed to allow for replication/database operation over multiple servers. For legal we must know what data is collected, and how that data is stored so we can apply appropriate

security measures and give information to the end-user about what data we collect. To easily visualize the data structure for the application and the underlying system, an Entity Relationship Diagram (ERD) has been made and attached below. This table contains the tables required to implement the login and role system, along with the features requested by the client. To understand this diagram, please note that any text that is specified as a `PK` is a primary key, whereas `FK` is a foreign key. Any text within [square] brackets are specific data types, to understand what this type means see the table below which explains each datatype along with giving an example. Any text within {curly} brackets are specific rules that must be applied to this data, such as a primary key being unique.

A screenshot of a computer

AI-generated content may be incorrect.

| **Entity** | **Attributes (with keys & types)** | **Notes / Constraints** |
| --- | --- | --- |
| **User** | user\_id **PK** (INT), role (ENUM: learner/tutor/guardian/admin), email (VARCHAR UNIQUE), password\_hash (TEXT), full\_name (VARCHAR), dob (DATE), created\_at (DATETIME), last\_login (DATETIME) | Email must be unique; password stored as hash; role restricted to enum values. |
| **Class** | class\_id **PK** (INT), title (VARCHAR), subject (VARCHAR), year\_group (INT), tutor\_id **FK→User.user\_id** | Each class linked to one tutor (User). |
| **ClassEnrollment** | class\_id **FK→Class.class\_id**, user\_id **FK→User.user\_id**, enrolled\_at (DATETIME), **PK(class\_id, user\_id)** | Junction table for many-to-many (learners in classes). |
| **Resource** | resource\_id **PK** (INT), title (VARCHAR), subject (VARCHAR), type (ENUM: video/doc/quiz), url (TEXT), owner\_id **FK→User.user\_id**, created\_at (DATETIME) | Type restricted to enum; owned by a tutor/admin. |
| **Assignment** | assignment\_id **PK** (INT), class\_id **FK→Class.class\_id**, resource\_id **FK→Resource.resource\_id**, due\_date (DATE), points (INT) | Due date must be in the future; points ≥ 0. |
| **Submission** | submission\_id **PK** (INT), assignment\_id **FK→Assignment.assignment\_id**, user\_id **FK→User.user\_id**, score (INT), status (ENUM: not\_started/submitted/marked), submitted\_at (DATETIME) | Score must be 0–100; one submission per assignment per user. |
| **Progress** | progress\_id **PK** (INT), user\_id **FK→User.user\_id**, subject (VARCHAR), best\_score (INT), attempts (INT), last\_accessed (DATETIME) | Tracks per-subject performance; score range 0–100. |
| **Reward** | reward\_id **PK** (INT), name (VARCHAR), description (TEXT), criteria (JSON/TEXT) | Criteria stored as JSON for flexibility. |
| **UserReward** | user\_id **FK→User.user\_id**, reward\_id **FK→Reward.reward\_id**, awarded\_at (DATETIME), **PK(user\_id, reward\_id)** | Junction table for many-to-many (users can earn multiple rewards). |
| **ActivityLog** | activity\_id **PK** (INT), user\_id **FK→User.user\_id**, action (VARCHAR), details (JSON/TEXT), created\_at (DATETIME) | Audit trail; logs key events (register, login, quiz submission, reward unlocked). |

1. **Algorithm Design**

**Registration (validation + hashing)**

**Pseudocode**

INPUT: name, email, pw, confirm\_pw, role, age

IF !valid\_name(name) OR !valid\_email(email) THEN error("Invalid details")

ELSE IF pw != confirm\_pw OR !strong(pw) THEN error("Password rules")

ELSE IF exists(User.email=email) THEN error("Account exists")

ELSE

hash = bcrypt\_hash(pw)

create User(name, email, hash, role, age)

log(ActivityLog, user=email, action="register")

redirect("login", flash="Account created")

END IF

[Start]

|

v

[Collect form]

|

v

{Valid name/email?}--No-->[Show error]-->[End]

|

Yes

v

{PW match & strong?}--No-->[Show error]-->[End]

|

Yes

v

{Email exists?}--Yes-->[Show error]-->[End]

|

No

v

[Hash PW -> Save User]

|

v

[Log "register"] -> [Redirect to login] -> [End]

**Login**

INPUT: email, pw

user = find\_user(email)

IF user is None THEN error("Email or password incorrect")

ELSE IF !bcrypt\_check(pw, user.password\_hash) THEN error("Email or password incorrect")

ELSE

create\_session(user\_id=user.id)

user.last\_login = now()

log(ActivityLog, user=user.id, action="login")

redirect("dashboard")

END IF

[Start]→[Read email/pw]→{User found?}--No-->[Generic error]→[End]

|

Yes

v

{Password valid?}--No-->[Generic error]→[End]

|

Yes

v

[Session + last\_login]

|

V [Redirect]→[End]

**Submit Quiz → Update Progress**

INPUT: user\_id, resource\_id, score

REQUIRE 0 <= score <= 100

p = get\_progress(user\_id, subject\_of(resource\_id))

IF p exists THEN

p.attempts += 1

p.best\_score = max(p.best\_score, score)

p.last\_accessed = now()

ELSE

create Progress(user\_id, subject, best\_score=score, attempts=1)

END IF

log(ActivityLog, user=user\_id, action="submit\_quiz", details={score})

maybe\_award(user\_id)

return success

[Start]→[Receive score]

|

v

{0<=score<=100?}--No-->[400 error]→[End]

|

Yes

v

{Progress exists?}--No-->[Create Progress]

| |

Yes v

v [Set attempts=1, best=score]

[Update attempts++, best=max]

|

v

[Log activity]→[maybe\_award]→[Success]→[End]

**Reward Engine (criteria met?)**

INPUT: user\_id

FOR reward IN Rewards:

IF not has\_reward(user\_id, reward) AND criteria\_met(user\_id, reward.criteria):

grant\_reward(user\_id, reward)

log(ActivityLog, user=user\_id, action="award", details={reward})

END FOR

[Start]→[For each reward]

|

v

{Already owned?}--Yes-->[Next reward]

|

No

v

{Criteria met?}--No-->[Next reward]

|

Yes

v

[Grant + Log]→[Next reward]→[End]

**Tutor: Create Assignment**

INPUT: class\_id, resource\_id, due\_date, points

REQUIRE user.role == 'tutor'

IF due\_date < today THEN error("Due date must be in the future")

create Assignment(class\_id, resource\_id, due\_date, points>=0)

notify\_enrolled\_students(class\_id)

log(ActivityLog, user=tutor\_id, action="create\_assignment")

return success

[Start]→[Tutor form]

|

v

{Role=tutor?}--No-->[403 Forbidden]→[End]

|

Yes

v

{Due date ≥ today?}--No-->[Show error]→[End]

|

Yes

v

[Create assignment]→[Notify class]→[Log]→[Success]→[End]

**System Flow Diagram**

[Start: Home/Login]

|

v

[Register] -----> [Login]

| |

v v

[Learner Dashboard] <---- [Tutor Dashboard]

| |

v v

[View Content/Take Quiz] [Upload Resources/Create Assignment]

| |

v v

[Progress Tracking] [View Learner Progress]

|

v

[Rewards System]

|

v

[Logout]

**4. Test strategy**

**1. Scope & Objectives**

scope = Home, Login, Register, Dashboard, Progress, Rewards, Assignments.

Objectives = Make sure features actually work (functional), and make sure they’re fast, secure, and usable (non-functional).

**2. Test Levels & Order**

Unit tests → does my email validator reject abc.com?

Integration tests → Register form saves user data and login accesses user

System tests → user logs in → takes quiz → progress bar updates

Acceptance tests (UACs) →

* Learner registration & login are functional
* Tutor uploads are functional and dummy resources are active
* Progress tracking dashboards actively update
* Reward system gives badges

**3. Environments**

**Local** = my laptop (e.g., Flask + SQLite).

**Staging** = a copy of the system online (e.g., on Heroku) with fake data. It’s safe to break things here before going live.

**Browsers:** Chromium, Firefox, Safari (latest), mobile viewport checks.

1. **Test Types & Techniques**

| **Type** | **Goal** | **Techniques you’ll use** | **Example in your project** |
| --- | --- | --- | --- |
| Functional | Features work as intended | Black-box (spec-based), happy/edge paths, CRUD checks | Login accepts valid creds; Register creates a user; Submit quiz updates Progress |
| Integration | Parts work together | API/DB integration, fixture data, contract checks | Register → DB row created → Login uses same row; Quiz submit → Progress + ActivityLog update |
| System / End-to-End | Whole user journeys succeed | Scenario scripts, data seeding, realistic flows | Learner logs in → takes quiz → sees updated dashboard and achievement |
| Acceptance (UACs) | Meets user/client expectations | UAC checklist, demo scripts, pass/fail criteria | “Dashboard shows subject progress bars with % text” passes for client |
| Security | Protect data & sessions | Input validation, authz checks, rate-limit, cookie flags | Wrong pw lockout; no PII in error; HttpOnly/Secure/SameSite session cookie |
| Accessibility (a11y) | Usable for all users | Keyboard-only nav, screen reader labels, colour contrast, focus order | Tab through pages; visible focus; aria labels on form fields; contrast ≥ 4.5:1 |
| Usability | Clear, efficient UX | Heuristic review, first-click test, microcopy checks | Error messages near fields; “Continue learning” primary CTA; empty states |
| Performance | Fast enough | Simple page-load timing, throttle network, image sizing | Dashboard FCP < 2.5s; total load < 5s on Fast 3G |
| Compatibility/Responsive | Works on devices/browsers | Viewport breakpoints (320–1440px), cross-browser smoke tests | Mobile nav collapses; tables scroll on small screens; Chrome/Firefox/Safari |
| Data Quality/Validation | Clean, valid data | Boundary values, invalid/absent inputs, type checks | Email regex, password rules, score 0–100, required fields blocking submit |

**5. Representative Test Data**

**Normal:** send a basic name email and password through the register page and see if it is accessible at login

**Erroneous:** incorrect number of characters/symbols to create user password

**Boundary:** Enter just the right number of characters/symbols to create user password

**Absent inputs:** Don’t enter anything and click submit on quiz, login register

**6. Example Test Cases**

| **Test Case** | **Input** | **Expected Output** |
| --- | --- | --- |
| Register Name valid | 3+ characters 1 number 1 symbol | Account Registers |
| password Invalid | 8 characters 1 number 0 symbol | Error message |
| Login valid | Correct email & password | Dashboard loads |
| Login invalid | Incorrect email & correct password | Error message |
| Add resource | Upload PDF | Resource visible |
| Progress update | Complete quiz | Dashboard shows new score |
| Invalidemail | Enter ada@ as email with valid other fields | Error message shown; account not created |
| Login lockout | Wrong password entered 5 times quickly | Account temporarily locked; generic error shown |
| Keyboard only journey | Navigate login → dashboard → quiz using only Tab + Enter | All controls accessible, no traps, journey comp |

**7. Traceability to KPIs & UACs**

* **Test IDs** = my drills / checks.
* **Req IDs** = my promises (FR/NFR/UAC).
* **Mapped Tests** = which drills prove which promises.
* **Pass criteria** = how you’ll know the test worked.

**Test IDs :**

**FR (Functional Requirements):** FR-1 Login, FR-2 Register, FR-3 Submit Quiz, FR-4 Progress Dashboard, FR-5 Rewards, FR-6 Tutor Upload/Assignment.

**NFR (Non-Functional Requirements):** NFR-1 Security, NFR-2 Accessibility (WCAG 2.1 AA), NFR-3 Performance (<5s), NFR-4 Reliability/Uptime (host), NFR-5 Usability.

**UAC (User Acceptance Criteria):** UAC-1 Successful Login, UAC-2 Registration Validation, UAC-3 Dashboard Shows % with text, UAC-4 Reward Appears After Criteria, UAC-5 Tutor Can Create Assignment.

**KPI (Key Performance Indicators):** KPI-1 Page load < 5s, KPI-2 99.9% uptime (host target), KPI-3 80% weekly active (tracking hook), KPI-4 90% UAT satisfaction.

**test IDs**

* **Auth-01 Login success**
* **Auth-02 Login failure/lockout**
* **Reg-03 Unique email**
* **Prog-10 Submit quiz updates progress**
* **Reward-1 Award rule**
* **A11y-20 Keyboard & focus order**
* **Perf-25 Load time**
* **Tutor-30 Create assignment**

| **Req ID** | **Requirement (summary)** | **Mapped Tests** | **Pass criteria** |
| --- | --- | --- | --- |
| FR-1 | System provides **Login** | Auth-01, Auth-02 | Success redirects to dashboard; failure shows generic error; session cookie set |
| FR-2 | System provides **Registration** | Reg-03 | New user persisted; duplicate email blocked with clear message |
| FR-3 | Learner can **Submit Quiz** | Prog-10 | Progress row upserts; attempts++ ; best\_score updates; activity logged |
| FR-4 | **Dashboard** shows subject progress | Prog-10, A11y-20 | Bars show numeric % (not color-only); values reflect latest quiz |
| FR-5 | **Rewards** granted on criteria | Reward-1 | Correct badge appears once; visible on dashboard; logged to ActivityLog |
| FR-6 | **Tutor** can create assignment | Tutor-30 | Valid form creates Assignment; enrolled learners notified (record/log) |
| NFR-1 | **Security** (validation, sessions) | Auth-02, Reg-03 | Lockout after N attempts; no PII in errors; secure cookies |
| NFR-2 | **Accessibility** WCAG 2.1 AA | A11y-20 | Keyboard complete journey; focus visible; labels/aria present; contrast passes |
| NFR-3 | **Performance** < 5s load | Perf-25 | Dashboard total load < 5s; FCP < 2.5s (Fast 3G throttle) |
| NFR-4 | **Reliability** (host uptime) | (Staging smoke), Perf-25 | Smoke tests pass on staging; uptime metric recorded from host |
| NFR-5 | **Usability** (clear copy/flows) | Auth-01, Reg-03, A11y-20 | Error messages near fields; primary CTA obvious; empty states present |
| UAC-1 | “User can log in successfully” | Auth-01 | Demonstrated in acceptance demo/script |
| UAC-2 | “Registration enforces rules” | Reg-03 | Password rules + unique email enforced with helpful guidance |
| UAC-3 | “Dashboard shows % text” | Prog-10, A11y-20 | % visible as text; screen reader announces value |
| UAC-4 | “Reward shown after criteria” | Reward-1 | Reward visible immediately after conditions met |
| UAC-5 | “Tutor creates assignment” | Tutor-30 | Assignment visible to class with due date & points |
| KPI-1 | Page load < 5s | Perf-25 | Measured under throttled network ≤ 5s |
| KPI-2 | 99.9% uptime | (Host metric) | Host dashboard screenshot / reference |
| KPI-3 | 80% weekly active | (Analytics hook) | Event tracking present; not fully assessed in prototype |
| KPI-4 | 90% UAT satisfaction | (UAT survey) | Survey summary ≥ 90% (Task 3 evidence) |