

10 · Code and platforms

Digital Platform Specification Template

A template for specifying a platform before building it: roles, permissions matrix, entities, workflows, states, MVP scope, risks and backlog, with an example.

For: Business owners, product managers and associations planning a platform with users in different roles, before handing it to a development team or building it with AI help.

Level: Intermediate Version 1.0 · 2026-09-21

A website shows information; a platform manages people, data and decisions. Once you have users in different roles (who registers, who approves, who sees what), any ambiguity in the spec later becomes a permissions bug or a data leak. AI tools can now write much of the code, but they cannot decide for you who may delete a registration or see a phone number. This template forces you to make those decisions in writing before the first line of code.

Real requirements before tools: do not start with "which tool or framework?" Start with "who uses the platform, what do they do, and what must they not be able to do?" The technology choice comes after those answers are written, not before.

How to use the template

1. Copy the template into a shared document and fill the sections in order; each feeds the next.
2. Write only what you know. What you do not know goes into "open questions" at the end, not into a guess.
3. Review the permissions matrix with someone who represents each role; they know best what they need and what they should not see.
4. Cut the first version ruthlessly: whatever does not serve the core workflow moves to "later".
5. Ask an AI model to critique the spec (prompt below), not to write it for you.
6. Freeze a version signed off by the decision-maker before development starts, and log every later change in the decisions log.

The template

Platform specification template

1) Overview

Problem: [what happens today and why it is costly, slow or unfair]

For whom: [roles in brief]

Success after 3 months means: [observable outcomes]

Not a goal: [what the platform will not do]

2) Roles

For each role: name | who they are | what they want |
how they get the role (self sign-up, invitation, assignment)

3) Permissions matrix

Rows = actions on each entity, columns = roles

Values: yes | no | own only | with approval

4) Entities and data

For each entity: fields | required? | sensitive? | owner |
relation to other entities | retention period

5) Workflows

For each workflow: trigger | steps in order | who performs each step |
notifications | what if it fails

6) States and transitions

For each entity with a lifecycle: states | allowed transitions |
who triggers them | what happens

7) First-version scope

Must have now | Later | Will not do

8) Integrations

Service | purpose | data direction | account owner | what if it stops

9) Non-functional requirements

Languages and direction | accessibility | security and sign-in | backups
| audit log | expected performance | devices

10) Risks

Risk | likelihood | impact | mitigation | owner

11) Backlog

As a [role] I want [action] so that [benefit]
+ acceptance criteria + priority + stage

12) Open questions and decisions log

Question | who answers | due date | decision | date

The hardest sections, explained

Roles

A role is a set of permissions, not a person. One person can hold two roles (coordinator and trainer, for example). Start with as few roles as possible; every extra role multiplies testing. For each role ask: how does a user get it? Self sign-up suits a participant, but an admin role must be assigned, not requested.

Permissions matrix

The most common security flaws in platforms are not sophisticated attacks but permissions nobody thought about: a user changes a number in the URL and sees another user's data. The matrix turns every action on every entity into a written decision. Pay special attention to "own only": it must be checked on the server, not just in the interface.

States

Every entity that moves through stages (registration: pending, confirmed, cancelled...) needs a state diagram: which states, which transitions are allowed, and who triggers them. Unwritten transitions become bugs: can a cancelled registration be confirmed? Can a certificate be issued to someone who never attended?

First-version scope

The first version should serve the core workflow end to end, even in its simplest form, rather than half-serve five workflows. Anything that can be done manually at first (a sheet, a message, a phone call) does not need automation in version one.

Risks

Write your project's real risks, not a generic list: sensitive data, dependence on one person, an external service whose terms may change, users without smartphones, AI-generated code nobody reviewed.

A prompt to critique the spec

Critical review prompt

You are a strict systems analyst.

Below is the spec for a platform [short description].

Do not add features or suggest technologies. Only list:

- 1) Contradictions between roles, the permissions matrix and the workflows.
- 2) Missing or dangerous state transitions (e.g. what if cancelled after confirmation?).
- 3) Data collected without a clear reason, or sensitive data with no stated protection.
- 4) Workflow actions that no role has permission to perform.
- 5) Five questions the project owner must answer before starting.

Order findings from most to least serious. Do not flatter.

Spec: [paste text after removing any personal data or keys]

Completed example

Illustrative example

Warsha Platform is a fictional platform for the **Al-Ofok Training Centre** in Tripoli, used to publish workshops, take registrations, record attendance and issue certificates. Today all of this runs through scattered forms, WhatsApp messages and a manual sheet.

1. Overview

Problem: registrations get lost in messages, seats are double-booked, and certificates are written by hand and arrive weeks late. **Success after 3 months:** all registrations go through the platform, no double booking, and certificates are issued within a week of the last session. **Not a goal:** online payment, remote learning, a mobile app.

2. Roles

Role	Who	How they get it
Visitor	Anyone browsing workshops	No account
Participant	Someone who registers for a workshop	Self sign-up with phone and email
Trainer	The person delivering the workshop	Assigned by a coordinator
Coordinator	Centre staff who run workshops	Assigned by the admin
Admin	Centre director	Created at launch; cannot be requested

3. Permissions matrix

Action	Visitor	Participant	Trainer	Coordinator	Admin
View published workshops	Yes	Yes	Yes	Yes	Yes
Create or edit a workshop	No	No	No	Yes	Yes
Publish a workshop	No	No	No	Yes	Yes
Register for a workshop	No	Yes	No	On a participant's behalf	On a participant's behalf
Cancel a registration	No	Own only	No	Yes	Yes
View registrant list	No	No	Own workshops, names without phones	Yes	Yes
Record attendance	No	No	Own workshops	Yes	Yes
Issue certificates	No	No	No	Yes	Yes

Action	Visitor	Participant	Trainer	Coordinator	Admin
Download a certificate	No	Own only	No	Yes	Yes
Manage roles	No	No	No	Assign trainer only	Yes
Export data	No	No	No	No	Yes, recorded in the audit log

4. Entities

Entity	Key fields	Sensitive?	Relations
User	Name, phone, email, role	Yes (phone and email)	Has registrations; may train workshops
Workshop	Title, description, trainer, seats, sessions, status	No	Has sessions and registrations
Session	Date, time, place	No	Belongs to a workshop; has attendance records
Registration	User, workshop, status, date, manual payment status	Medium	Links a user to a workshop
Attendance	Registration, session, present or absent, recorded by	No	Links a registration to a session
Certificate	Registration, verification code, issue date, issued by	No	Belongs to one registration

Retention: contact data for inactive participants is reviewed yearly and deleted when no longer needed; certificates are kept because they are used for verification.

5. Core workflow: from publishing to certificate

1. The coordinator creates the workshop and its sessions, assigns the trainer, then publishes it.
2. A participant registers; if a seat is free the registration becomes "pending" until manual payment is confirmed, otherwise it joins the waitlist.
3. The coordinator confirms the registration once the fee is received at the centre; the participant gets a confirmation notice.
4. At each session the trainer records attendance from a phone.
5. After the last session, attendance against the required rate is calculated; the coordinator reviews the list and issues certificates.
6. The participant receives a download link; the certificate carries a verification code that can be checked on a public page showing only the name, workshop title and date.

On failure: if a notice does not arrive, the participant can always see the registration status in their account; notifications help but are not the source of truth.

6. Registration states

From	To	Triggered by	What happens
(new)	Pending	Participant	A seat is held for 72 hours
(new)	Waitlisted	System (no seats)	Notice with waitlist position
Pending	Confirmed	Coordinator	Confirmation notice
Pending	Expired	System after 72 hours	Seat released and offered to the first waitlisted person
Pending or confirmed	Cancelled	Participant or coordinator	Seat released
Confirmed	Completed	System after last session, attendance rate met	Eligible for certificate
Confirmed	Incomplete	System after last session	No certificate

Forbidden: moving from "cancelled" or "expired" straight to "confirmed" (a new registration is required); issuing a certificate in any state other than "completed".

7. First-version scope

Must have now	Later	Will not do
Publishing workshops, registration, manual confirmation, waitlist, attendance, certificates with verification codes, email notices	WhatsApp notices, post-workshop feedback, director reports, English interface	Online payment, live streaming, mobile app

8. Integrations

An email service for notices (account in the centre's name); no other integrations in version one. If email stops, statuses remain visible in accounts.

9. Non-functional requirements

Arabic first, right-to-left; works on an average phone; one-time-code sign-in; daily backup with a monthly restore test; audit log for every role change and every data export; permissions checked on the server for every request.

10. Risks

Risk	Likelihood	Impact	Mitigation	Owner
A trainer sees participants' phone numbers and uses them outside the centre	Medium	High	Trainers see names only; contact goes through the coordinator	Admin
AI-generated code leaves "own only" unchecked on the server	Medium	High	Code review with a checklist and a manual test of every matrix row	Developer
The only coordinator is absent and confirmations stop	High	Medium	Assign a backup coordinator	Admin

Risk	Likelihood	Impact	Mitigation	Owner
Participants without email	Medium	Medium	Sign-up by phone; confirmation can happen by phone and be recorded manually	Coordinator

11. Backlog (excerpt)

Story	Acceptance criterion	Priority
As a participant I want to register for a workshop so that I hold a seat	If seats are full I join the waitlist and see my position	Now
As a coordinator I want to confirm pending registrations so that seats are secured	An expired or cancelled registration cannot be confirmed	Now
As a trainer I want to record attendance on my phone so that I stop using paper	I see only my workshops and no phone numbers	Now
As a participant I want to download my certificate so that I can attach it to my CV	I cannot download another participant's certificate even by editing the URL	Now
As the admin I want a monthly attendance report so that I can evaluate workshops	The report contains no contact data	Later

12. Open questions

- What attendance rate is required for a certificate? Does it vary by workshop? (Answers: admin, before development starts)
- Can people under 16 register? If so, what parental consent is required? (Answers: admin)

Common mistakes

- **Mistake:** starting by choosing the tool or framework. **Fix:** roles, permissions and workflows first; the technology is chosen to serve them.
- **Mistake:** permissions enforced only in the interface (hiding a button). **Fix:** every matrix cell is checked on the server and tested by editing the URL or ID by hand.
- **Mistake:** states without written forbidden transitions. **Fix:** write down what is explicitly forbidden; it is exactly what someone will try.
- **Mistake:** a first version that tries to serve everyone. **Fix:** one complete core workflow; the rest goes to "later".
- **Mistake:** letting the model write the requirements. **Fix:** use it as a critic to find gaps; decisions belong to you and the people who represent each role.
- **Mistake:** nobody "owns" the spec. **Fix:** one decision-maker signs off the version, and a decisions log records every change.

Completion checklist

- ☐ Problem, success and non-goals are written in the overview.
- ☐ Every role has a description and a way of being obtained.
- ☐ The permissions matrix covers every action on every entity and was reviewed with role representatives.
- ☐ Entities are documented with fields, sensitivity and retention.
- ☐ The core workflow is written end to end, including the failure case.
- ☐ The state diagram defines what is allowed and what is forbidden.
- ☐ First-version scope is split into now, later and will not do.
- ☐ Risks are specific to the project, each with an owner.
- ☐ User stories have testable acceptance criteria.
- ☐ Open questions have someone to answer them and a date.
- ☐ The spec has had a critical review and is signed off by the decision-maker.

About the author — Mohamad Charafeddine is an entrepreneur from Tripoli, Lebanon: founder of the Passion digital marketing agency, president of Takaful Youth Association, and founder of digital platforms. He trains individuals and teams in applied AI and marketing.

Examples in this file are fictional. Free to use in your work and organisation; credit the source if you republish.
mohamadcharafeddine.com/en/resources/platform-spec-template/