Programming Challenge

Introduction

We're building a simple tool for an international spy agency. The agency conducts planned assassinations and has done so for decades. But it needs a new system to assign assassinations ("hits").

The general user is a hitman. He can be assigned a hit and see it on his list of upcoming work. Typically, it succeeds and is closed out. But occasionally things don't work out and the target lives. In those cases, we assume the target hires security and thus the case is forever closed, as a failed mission.

Like everyone else, hitmen have bosses. These are directly in charge of a group of hitmen and can create hits and assign them. But they can only assign hits to the hitmen they manage.

Finally, there's the big boss of the agency - Giuseppi . He does *not* manage managers directly. Rather, he just has free access to assign hits to anyone in the system. Even managers. But when a new employee comes into the spy agency, he need not be added to The Boss' list. It's automatic.

The boss is also in charge of assigning hitmen to managers. Only he can do that.

For simplicity, the boss is always the first user in the database. No special indications need to be added to flag the user as the boss.

Sadly, our hitmen do occasionally die in the field. Or worse, retire from the industry. In this case, they can no longer be assigned hits and can no longer use the system. Managers and The Boss *can* however, still check old assignments for these hitmen.

Stack

You can use any stack you want. Anything goes.

Only one rule: you must use a relational database and referential integrity must exist.

Goal

This is a *very* subjective exercise. But overall, the goal is to build a platform that has the highest developer performance as possible. What is developer performance? It's subtle, and hard to define. But I know it when I see it. Think ergonomics. Think productivity.

Think about what is the best architecture that achieves a good balance between:

1. Ease of adding new features,
2. Ease of maintaining existing features.
3. Speed of delivering the above (time to market).
4. Ease of adding testing, both manual and automated, in isolation.
5. Conforming https://12factor.net/
6. Ease of deployment.
7. Ease of refactor.
8. Ease of monitoring.
9. Ease of distributing development load.

There's no right answer, but a combination of the above should be a good start. You'll notice a common word though: ease.

Acceptance criteria

1. All data must be persisted accurately.
2. Styling does *not* matter, but it should be very functional and easy to use. Think admin tool for internal users.
3. Errors should show up inline in forms as appropriate. They do not have to be real time, but after submission of each screen.
4. novalidate should be used for forms. Turn *off* browser validation.
5. All screens should be valid and linkable. Assuming I have access to a screen, I should be able to share a URL with any other user with access to it. If I don't have access, a 404 or 403 can be shown.
6. Generally, confirmation messages should appear when an editing action is taken. Example: "Hitman updated" or "Hit updated" or "Hit created".

Initial data

1. A userbase must be seeded. Create 9 hitmen, 3 managers, and 1 big boss (uid=1).

Screens

URLs given are for UI and what the browser address bar would show. Whatever happens behind the scenes doesn't matter.

1. Login. Email based login.

Url: `/`
If already logged in, this redirects to /hits/.

2. Logout. Not really a screen, but a redirect view.

Url: `/logout/`
Redirects to / when done.

3. Registration page: no send-an-email-to-validate flow. Just enter (valid) email and password and sign up. Disregard security and feel free to say an email is already taken. Enforce password rules.

Url: `/register/`

All other views require authentication.

4. List of hits

Url: `/hits/`

Show every hit in the system for that logged in user. Cases:

1. Hitmen: Hits assigned to him.
2. Manager: Hits assigned to him or to his lackeys.
3. The Boss: all hits.

5. Hit detail

Url: `/hits/<id>/`

Access rules are just like #3. This screen shows details about a hit. A few notes:

A hit has:

1. The assignee.
2. A brief description.
3. The name of the target.
4. A finite state machine for the status: Assigned -> Failed. Assigned -> Completed.
5. The creator of the assignment.

Rules:

1. If the user is an end hitman, the screen is read only in general. He *can* mark the case as Failed or Completed.
2. A manager can change the assignee for open cases. For closed cases, either failed or succeeded, everything is read only.
3. The Boss has the same rules as the manager.
4. Here's a very subtle point: a hit cannot be assigned to an inactive user. *But* if it was already assigned to an inactive user, it should be allowed to stay as is but updated other items. Essentially, the form should be save-able as it was when it was loaded. Feel free to ask more about this.

6. Create hit

Url: `/hits/create/`

The contents of this page are similar to a hit detail. Except there are no transition buttons, only a button to create.

Rules:

1. End hitmen cannot see this screen at all.
2. Both the manager and The Boss can create hits.
3. Under assignee, the manager can only assign to his lackeys. The boss can assign to anyone.
4. Neither can assign to themselves.
5. Neither the boss nor the manager can assign to inactive users.

7. List of hitmen

Url: `/hitmen/`

Rules:

1. This screen cannot be reached by end hitmen.
2. Managers can see anyone they manage here, including inactive users.
3. The boss can see everyone here, including inactive users.

7. User detail

Url: `/hitmen/<id>/`

Rules: same access as the hitmen list, but an individual hitman view.

A hitman has:

1. A name
2. An email.
   3 A description.
3. A finite state machine for Active -> Inactive. Editable but only in Active -> Inactive. Hitmen cannot rejoin.
4. A list of other hitmen they manage.

Only the boss can edit #4. In fact, only the boss can even see #4.

8: Hit reassignment (nice to have)

Url: `/hits/bulk/`

Occasionally, managers (or The Boss) need to reassign a ton of hits. It's just the nature of the business.
This view shows all *active* hits and all information about the hits. Only the assignee is editable.
Same rules apply as in the individual detail.