Task #5 (ALL GROUPS)

Use language and platform **FOR YOU GROUP**:

- JavaScript or TypeScript, React, Express, PostgreSQL or MySQL or any database),
- C#, .NET, some kind ASP.NET, SQL Server or PostgreSQL or MySQL or any database.

THE FIRST REQUIREMENT: YOU NEED TO CREATE A UNIQUE INDEX IN DATABASE.

THE SECOND REQUIREMENT: YOUR TABLE SHOULD LOOK LIKE TABLE AND YOUR TOOLBAR SHOULD LOOK LIKE TOOLBAR.

THE THIRD REQUIREMENT: DATA IN THE TABLE SHOULD BE SORTED (E.G., BY THE LAST LOGIN TIME).

THE FOURTH REQUIREMENT: MULTIPLE SELECTION SHOULD BE IMPLEMENTED USING CHECKBOXES (SELECT ALL/DELECT ALL IS ALSO A CHECKBOX).

THE FIFTH REQUIREMENT: BEFORE EACH REQUEST EXCEPT FOR REGISTRATION OR LOGIN, SERVER SHOULD CHECK IF USER EXISTS AND ISN'T BLOCKED (if user account is blocked or deleted, any next user's request should redirect to the login page).

Create a *working and deployed (remotely accessible*). Web application with user registration and authentication.

Non-authenticated users should not have access to the user management (admin panel). They have only access to login form or registration form.

Only authenticated non-blocked users should have access the user management **table** with at least the following fields: (selection checkbox), name, e-mail, last login time (or the last "activity" time or both), status (unverified/active/blocked). You also may add registration time, some sparklines for activity, etc. (but that's optional).

The leftmost column of the table should contains checkboxes without labels for multiple selection (table header contains *only checkbox without label that selects or deselects all the records*).

There must be a **toolbar** over the table with the following actions: Block (button with text), Unblock (icon), Delete (icon), Delete unverified (icon). NO BUTTONS IN EACH ROW. Toolbar shouldn't appear/disappear, but may change enabled/disable status.

You have to use any **CSS framework** (Bootstrap is recommended, but you can choose any CSS framework).

All users should be able to block or delete *themselves* or any other user.

User can use any non-empty password (even one character). User can login even if e-mail is unverified.

Deleted users should be deleted, not "marked".

Users are registered right away and the corresponding message is show after the registration; the confirmation e-mail should be sent asynchronously. Clicking the link in the e-mail changes the status from "unverified" to "active" ("blocked" stays "blocked").

Blocked user should not be able to login, deleted user can re-register.

JUST IN CASE: YOU HAVE TO CREATE A UNIQUE INDEX IN THE DATABASE. NOT TO CHECK WITH YOU CODE FOR UNIQUENESS, BUT CREATE THE INDEX.

YOU STORAGE SHOULD GUARANTEE **E-MAIL** UNIQUENESS INDEPENDENTLY OF HOW MANY SOURCES PUSH DATA INTO IT SIMULTANEOUSLY.

Generally, you can use any "architecture". It's not important in this task. Generally, it's the most simple and straightforward to use a relational DB, e.g. PostgreSQL or MySQL. You can, if you wish, use anything else. E.g., MongoDB, but it really won't give you any advantage in this task (you just deprive yourself a very major skill). But you can. Or you can even try use some SaaS like Firebase. But be careful, you easily will get into some troubles. E.g., if you decide to use "out-of-the-box" users, it may be problematic to delete them. If you decide to use custom "users", you may have some problems with maintaining uniqueness of e-mails, etc. I have to remind, that you code can contain only error processing and management of uniqueness have to be performed on the storage level, but you code shouldn't contain checks whether the e-mail already exists. Arguably, the simplest solution is to use a classic boring "trivial" relational database. But even in Firebase you may try to create documents that uses e-mails as keys (again, it limits you ability to change e-mail by user request, etc.).

NO WALLPAPERS UNDER THE TABLE. NO ANIMATIONS. NO BROWSER ALERTS. NO BUTTONS IN THE DATA ROWS.

The application should work properly in a any browser and on any resolution (desktop/mobile) and look like a business-oriented professional app (boring, but consistent and accurate). Well, you have to get that automagically by properly using CSS framework like Bootstrap.

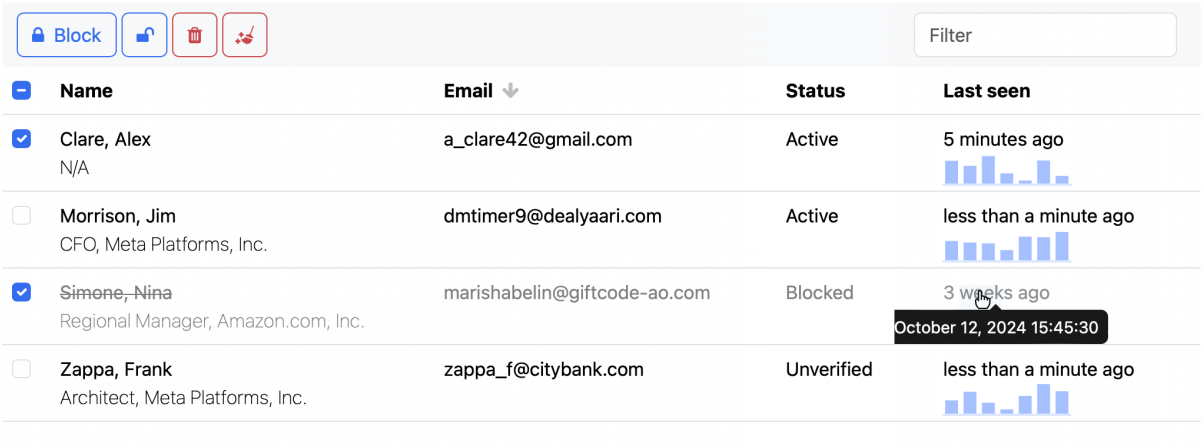
DON'T INVENT ANYTHING, USE LIBRARIES FOR EVERYTHING.

Implement:
- adequate error messages;
- tooltips;
- status messages (informing user about successful or failed operation results).

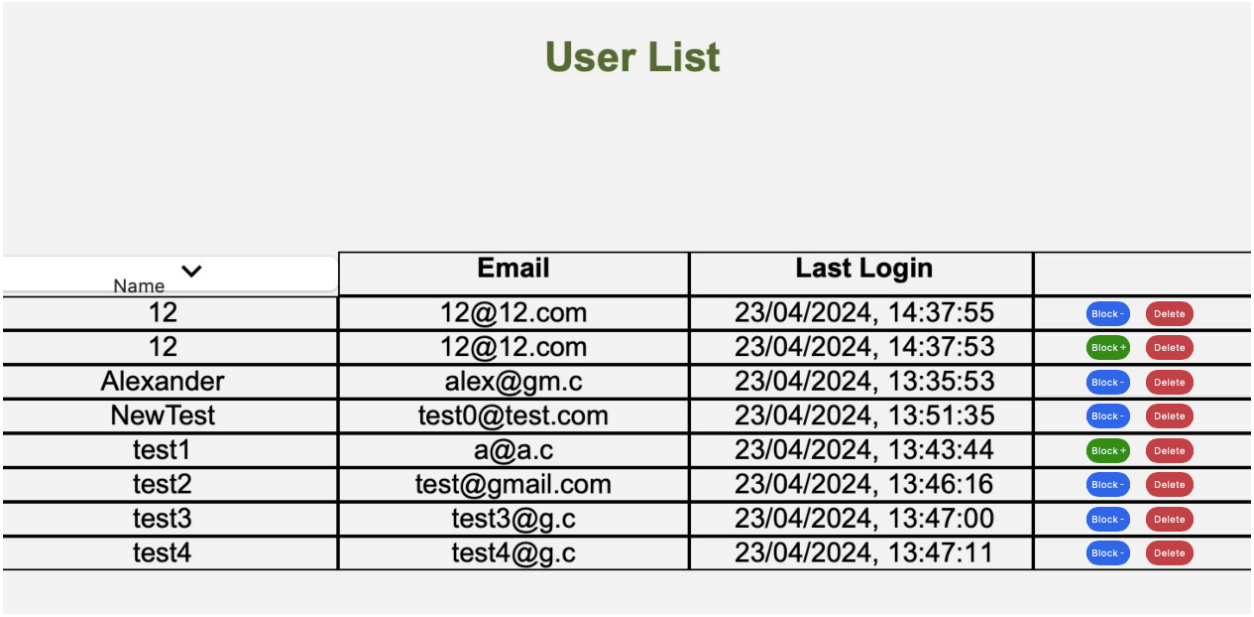
If you use N buttons (separate set of buttons for each row), the grade will be reduced by 20%. You need to implement selection and a toolbar. And align text properly in the cells (check the screenshot below).

Several technical notes:
- You may use anything to send e-mails, even access your Gmail account via code, it's OK for testing purposes.
- Note that a *unique index* is not the same as a *primary key* (which you should have too): https://en.wikipedia.org/wiki/Database_index

This is table with toolbar (well, *you are not required to show sparklines* with previous user activity, ***it's definitely optional***; also, this is only the table, you'll need to implement the standard navigation header, logout, etc.):



That's not:



This is a login form:

THE APP

Start your journey

Sign In to The App

E-mail
test@example.com

Password
••••••••

Remember me

Sign In

Don't have an account? Sign up

Forgot password?

That's not:

Login

E-mail:

Password:

Sign In | Register