Project Requirements:

1. The whole application should only be accessible to authenticated users, all unauthenticated requests should redirect the browser to a login page.
    a. Users need to enter an email/password combination to log in
    b. In the event that a user has entered incorrect credentials when logging in, the platform should display a message informing them of the error
        i. The form should also pre populate the email address after an error, but should not keep the password
    c. In the event that a user tries to log in, but does not have their email address verified a corresponding message should be displayed, while also showing a link to resend their activation email
    d. In the event that a user has unsuccessfully tried to login 10 times, they should be blocked from logging in for 30 seconds.
        i. They should be informed of this block by an error message
        ii. This action should be logged by the application as a warning. The message should contain a description as well as the IP of the request
    e. Once a user has successfully logged in they should arrive on the page they originally tried to access (before they got redirected to the login page)
    f. The login page needs have a link to the registration page
2. Users should be able to 'Log out' of the application
3. Users need to be able to register on the registration page
    a. Users should enter the following information when registering:
        i. Email
        ii. Name
        iii. Password
            1. Needs to be hashed in the database
        iv. Phone Number
        v. Accept our Terms of Service
            1. In the database the date and time that the user accepted the terms should be saved
            2. The 'Terms of Service' text needs to be a link to a page where this can be viewed (this page can be viewed by any user, even unauthenticated ones)
    b. The registrations should perform validation:
        i. Validation rules:
            1. Email: required, valid email, unique
            2. Name: required
            3. Password: required, minimum 8 characters, minimum 1 digit, minimum 1 symbol, minimum 1 lowercase letter, minimum 1 uppercase letter
                a. Password confirmation should also be asked in another input
            4. Terms accepted

Commented [1]: They should also add maximum character validation, so the DB doesn't give an error on too long strings

    ii. The form should pre populate the email and name fields after a validation error
  c. After registration the system should send the user a validation email
  d. After the user validates their email they should be redirected to the dashboard
4. After logging in users need to be able to see the dashboard:
  a. Dashboard should contain a table listing all users in the system
    i. Each row should contain the users ID, email and phone number
    ii. Each row should have an edit, delete and an unverify button
  b. Users need to be editable
    i. All the same validation rules apply as at registration
      1. Form should be repopulated with old data in case of error
    ii. If email is updated, user should receive a new verification email, same as at registration
  c. User can be deleted
    i. In case of delete, page should not reload, DELETE action should be done with AJAX, and on success row removed from HTML with fade effect
  d. User can be unverified
    i. This should simply unset the users email verified status
  e. Dashboard needs to contain a search field at the top to filter users
    i. Search should be case insensitive
    ii. Live search is required, page should not reload, but request new data from server via AJAX, and reload the list of users accordingly
    iii. Search should search in email, name and phone number fields with mySQL LIKE
5. The application should have an artisan command that can be used to unverify a users email based on their ID. This should use the same functionality as when clicking on the unverify button on the dashboard
6. The application should have a terms page
  a. This page should have a list showing all previous version of the sites ‘Terms of Service’
  b. Each term object needs to have the following fields:
    i. Administrative name
      1. Only for administrative purposes
    ii. Content
      1. A text field containing the terms
    iii. Publication date
  c. You should be able to create new term objects
  d. Unpublished terms (No publication date) should be publishable
    i. The term receives the current date and time as it’s publication date
    ii. All users in the system receive an email informing them that we have updated our terms. The email should contain a link to our ‘Terms of Service’ page which always shows the term object with the most recent Publication date
  e. Unpublished terms should be editable

        i. Only the name and content can be edited
        ii. Published terms can not be edited any more, and their publication date is also locked
    f. Unpublished terms can be deleted
        i. Published terms can not be deleted
7. User terms acceptance
    a. In the event that a user has not accepted our latest terms (their accepted field is older then the newest publication date in our terms) then the user should see a small message in the header informing them of this
        i. This message should also have a button they can click on to 'Accept'
        ii. This message should also have a link to the 'Terms of Service' page
        iii. This message should have a link to 'Currently Accepted Terms' which will show the user the last terms object that he has accepted
            1. Keep in mind that this could be one very much in the past if the user has not been online for a while and several new term objects have been published since

Extras

1. Please create an artisan command that deletes all published terms that are not the "Currently Accepted Terms" of any users.
2. Create Tests for you code

Additional Requirements:

- The application should be created with the newest version of Laravel
- All database tables should be created with migrations
- A few users and at least 1 published and 1 unpublished term object should be created as seeds
- Please create a document detailing the finished product and describing the steps needed to install your website on a server, specific requirements like PHP, Nginx/Apache, etc..
    - This document should be created as if it were being sent to a client, it should be clear and easily readable.

Notes:

- If a requirement is proving hard to perfect, you can skip it. The point of this exercise is not to get everything 100% right, but to touch many ideas so we get better understanding of how you work.
- Treat this project as if you were building something that you will also have to maintain and extend in the future. Try to follow good coding standards and principles so the code is easily readable and understandable, and robust in the face of possible changes