Access control-related requirements are necessarily often written by people who don't know a lot about security, because few organizations have a security expert on hand or can justify the expense of engaging one just to help write a few requirements. Fortunately, the requirements don't need to grapple with the intricacies of the subject or worry about the implementation challenges, which are considerable. As a result, neither do the requirement patterns here: they don't attempt to say what to do to make a system secure. So if you're going to bring in an expert, it can be more effective to do so *after* specifying the requirements. Anything to do with security is harder to achieve than it appears—because it doesn't merely have to function; it must also remain steadfast under attack, leaving no holes. Any bricklayer can build a wall, but one that resists determined assault is another matter.

11.1 User Registration Requirement Pattern

Basic Details

Related patterns:	User authentication, user authorization, data structure
Anticipated frequency:	One or two requirements
Pattern classifications:	Functional: Yes; Affects database: Yes

Applicability

Use the user registration requirement pattern to specify how new users are registered (set up in the system), with emphasis on capturing those details by which a user can later be authenticated (log in).

Discussion

Users are special. They are people, and they drive the bulk of what happens in a typical commercial system, so it's critical to know who each user is—to a greater or lesser degree of confidence depending on what we let them do. User registration, then, is more important than just another data maintenance activity. The primary aim of user registration is to record enough information to facilitate user authentication (login); everything else is secondary, though there could be quite a lot else to record.

Letting a user perform their own registration is the most convenient way, but anything they enter must be regarded as suspect (untrusted). They could supply false details (such as lying about their age), or they could enter details about someone else (which is, in some circumstances, a form of identify fraud). Alternatively, a user could be registered by a person who's already trusted, which is usually done when that person is in a position to attest to the prospective user's identity (such as if they're physically present). This is inherently much more dependable, so use it whenever it's practical (for example, for registering new employees). There is scope for more complex registration processes: to have some details entered by the user and some by someone else, or to check the veracity of certain information supplied by the user (against electoral rolls, credit reference agencies, the telephone directory, and such). This might sound invasive, but it depends on the nature of your system, the type of user, and how much faith you need in the accuracy of what the user tells you about their identity. These steps can raise the trustworthiness of the information in question, but they involve extra effort—both in operation and in development.

Content

A user registration requirement should contain:

1. **Class of users** This identifies the type of user that is created by the registration process described by this requirement. If all users are registered in the same way, but the system treats different classes of users in different ways, explain how the user class is determined.
2. **User details** What information do we want to record about each user? You can either list every item individually, or refer to a data structure (as per the data structure requirement pattern) that defines them, or a mixture of both. Alternatively, some details can be defined in extra requirements. Indicate which information is mandatory and which is optional. Not all information about a user needs to be entered when they register: concentrate on what information the system must be able to handle, rather than on which items must be entered during registration. It's useful to divide this information into the following five categories:
   a. **Identification details**, such as user ID, name, and email address–things used to distinguish this person from someone else. Sometimes other facts about a person are used to *assist* identification (for example date of birth), but they should be regarded as identification details in their own right only if they are essential to uniquely identify a particular person.
   b. **Authentication information**, most commonly a password. It might be augmented by further information that's called upon if the user forgets their password (such as a "secret phrase" or a few questions and answers that an impostor is unlikely to know). These details indicate that a person who knows them when logging in is the same person who registered.

      Criteria for an acceptable password can be stated here, or they can be the subject of a separate requirement. See the "Password Format" subsection of the "Extra Requirements" section of this pattern for suggestions on what to say.
   c. **Facts** about the person (that aren't used for identification), such as date of birth, title, gender, address, and phone number(s).

      If you use a biometric reader when authenticating a user, allow anyone who's physically incapable of using such a reader to be identified. (See the "Accessibility" point in the "Extra Requirements" section of the user authentication requirement pattern for more explanation.)
   d. **Preferences**. Define a user preference for each aspect of how the system behaves that the user is able to control. We needn't force a user to express their preferences when they register, but it's a good opportunity to let them do so.
   e. **Access privileges**, that define what this user is permitted to do and see. One way is for the user registration requirement to mention that one or more "roles" can be assigned to a user. This subject is covered in greater depth in the user authorization requirement patterns–though it is the registration process that assigns at least some initial access privileges to the user.
3. **Registration process** This can be as simple as entering information about the user, or it can involve additional steps. The latter are typically to check the correctness of what we know about this user, including obtaining evidence that the user who wants to be registered is actually the person whose details have been provided. (That is, they're not an impostor.)

Extra registration steps usually involve interacting with external systems (such as a credit reference agency) or manual activities (such as examining a faxed copy of an identity document). A commonly-used step is to send an email to the user's email address and ask for a response; this merely proves that this person receives the mail sent to that address. Describe whatever you need in your case.

Even one class of users could have more than one **level** of registration. You might have rudimentary registration for basic access, but ask for more (and check more thoroughly) if a user wants to do more (that is, to be trusted more). A user's access privileges can thus be affected by which stage a non-trivial registration process has reached.

Template(s)

Summary	Definition
«*User class*» self-registration	A person shall be able to self-register as a «*User class*», by «*Registration process description*». They shall be asked to enter the following personal information: ■ «*User detail 1*». ■ «*User detail 2*». ■ ...
«*User class*» registration	It shall be possible to register a person as a «*User class*», by «*Registration process description*». The following information shall be entered about them: ■ «*User detail 1*». ■ «*User detail 2*». ■ ...

Example(s)

Summary	Definition
Customer registration	A visitor to the Web site shall be able to self-register as a customer, by entering the following details: ■ Chosen user ID * ■ Chosen password (to be entered twice) * ■ Name details, as described in requirement «*Req't ID*» * ■ Address * ■ Email address * ■ Contact telephone number ■ Gender ■ Preferred language * All items flagged with an asterisk are mandatory.

Summary	Definition
Employee registration	It shall be possible for an authorized user to register a person as an employee, by entering the following details about them: ■ Full name (first, middle, and family names) ■ Diminutive name (for example, "Bill") ■ Job title ■ Company email address ■ Company telephone extension number ■ Home address ■ Home telephone number * ■ Mobile telephone number * ■ Gender ■ Role(s) the employee is authorized to perform All this information is mandatory except those flagged with an asterisk (*). The employee shall automatically be allocated the next available employee number, which shall be displayed to the user. An initial password shall be generated for the new employee based on the details known about them (including their employee number). The *form* of this password shall be sufficiently simple to be easily communicated to the employee without divulging its actual value, though a specific actual form is not mandated by this requirement. (For example, it could be "family name plus employee number.") When logging in for the first time, the employee shall be forced to change their password.

Extra Requirements

A user registration requirement can be accompanied by various kinds of extra requirements, most of which relate to the ongoing protection and maintenance of information initially provided at user registration. Consider which of the following areas you feel warrant requirements. Each area is discussed further in its own subsection that follows:

1. **Password format** What criteria must a password satisfy to be acceptable?
2. **Changing password** How and when?
3. **User de-registration** If users can be created, it must also be possible to remove them.
4. **User data protection** How well must we protect personal and private information about people? Are there any data protection laws we must comply with?
5. **Good security practices** Are there any steps we want taken to encourage protecting user information? Are there any particular bad practices that we want to guard against?
6. **Special processes for new users** Is there any more to be said about the user registration process? Are there any situations that arise with new users (such as how to tell them their initial password)?
7. **Day one considerations** Who'll be able to use the system when it's first installed? Make sure *someone* can—and that in doing so, no security hole is left open.

Any of these things that affect users should be documented, so they can read what they're supposed to do. The documentation requirement pattern has an example requirement for a security procedures manual.

Password Format Settling on criteria for an acceptable password involves making compromises. The best balance depends on the nature of your environment: how frequently users log in (and thus how likely they are to forget their password and need to write it down), the potential damage if a password is discovered, how likely an interested party is to discover a password that's written down, and so on. It can make sense to have separate rules for different classes of users: one for employees who use a system all day, every day, and another for visitors to a Web site. Making passwords convoluted makes them harder to guess, but also much harder to remember–and thus makes it much more likely that users will write them down. (*Long* passwords aren't necessarily unmemorable, though: a password made up of several words can be hard to crack, especially when accompanied by deviant spelling and with an unusual symbol or two thrown in.) Many apparently sensible steps that look like they improve security can actually have the opposite effect, especially when taken to extremes. Specify whatever password criteria you wish, but don't go overboard.

Summary	Definition
Customer password format	Customer passwords shall: ■ Be at least six characters long. ■ Contain at least one character from each of at least three of the following four categories: i. Lowercase alphabetic (for example, a-z, ä, θ). All characters in languages that do not recognize case shall be regarded as lowercase for the purposes of this requirement. ii. Uppercase alphabetic (for example, A-Z, Ě, Φ). iii. Numeric (0-9). iv. Any other character (not in the first three categories). ■ Not contain a string of three or more consecutive characters that can be found in any part of the user's name or user ID (when considered case-insensitively). All passwords shall be regarded as being case-sensitive. Thus typing "open1sesame" isn't good enough if the password is "Open1Sesame." These criteria apply to every function that can set or change a password. This requirement is based on the Windows Vista password complexity requirements as stated at *http://www.microsoft.com/technet/windowsvista/security/security_group_policy_settings.mspx*.

Any of these things that affect users should be documented, so they can read what they're supposed to do. The documentation requirement pattern has an example requirement for a security procedures manual.

Password Format Settling on criteria for an acceptable password involves making compromises. The best balance depends on the nature of your environment: how frequently users log in (and thus how likely they are to forget their password and need to write it down), the potential damage if a password is discovered, how likely an interested party is to discover a password that's written down, and so on. It can make sense to have separate rules for different classes of users: one for employees who use a system all day, every day, and another for visitors to a Web site. Making passwords convoluted makes them harder to guess, but also much harder to remember–and thus makes it much more likely that users will write them down. (*Long* passwords aren't necessarily unmemorable, though: a password made up of several words can be hard to crack, especially when accompanied by deviant spelling and with an unusual symbol or two thrown in.) Many apparently sensible steps that look like they improve security can actually have the opposite effect, especially when taken to extremes. Specify whatever password criteria you wish, but don't go overboard.

Summary	Definition
Customer password format	Customer passwords shall: ■ Be at least six characters long. ■ Contain at least one character from each of at least three of the following four categories: i. Lowercase alphabetic (for example, a-z, ä, θ). All characters in languages that do not recognize case shall be regarded as lowercase for the purposes of this requirement. ii. Uppercase alphabetic (for example, A-Z, Ě, Φ). iii. Numeric (0-9). iv. Any other character (not in the first three categories). ■ Not contain a string of three or more consecutive characters that can be found in any part of the user's name or user ID (when considered case-insensitively). All passwords shall be regarded as being case-sensitive. Thus typing "open1sesame" isn't good enough if the password is "Open1Sesame." These criteria apply to every function that can set or change a password. This requirement is based on the Windows Vista password complexity requirements as stated at *http://www.microsoft.com/technet/windowsvista/security/security_group_policy_settings.mspx*.

Summary	Definition
Passwords difficult to guess	There shall be a table of values (dictionary of words) that may not be used as passwords. Any complete string in a password (that is, any complete sequence of alphabetic characters, but not substrings within an alphabetic sequence) that is an entry in this table shall not be allowed.

For good measure, you could ask for a utility to hunt for poorly-chosen passwords:

Summary	Definition
Password guesser	There shall be a "password guesser" utility that can be run periodically to create a list of all users whose password it was able to guess.

Changing Password Users should be able to change their password whenever they want–for example, if they suspect that it's been compromised (because someone else might have discovered it). But unless you specify a requirement to this effect, the system is acceptable if delivered without a change password function. You might also want to force users to change their password from time to time or to restrict what they can change it to (such as not allowing the same value every time). The situation by which a user forgetting their password eventually leads to the password being changed is discussed in the user authentication requirement pattern's "Extra Requirements" section.

Summary	Definition
Change password	A user shall be able to change their own password. When changing their password, the user must enter: ■ Their current password. ■ Their new password, twice. (The second time is to guard against mis-keying, resulting in a password the user does not know.)
User password expiration	A user shall be forced to change their password the next time they log in if they have not changed it for a (configurable) given length of time.
Force user password change	There shall be a means to force a nominated user to change their password, such that the next time they attempt to log in, they shall be unable to do so until they successfully change their password to a different value.
Employee cannot reuse password	When changing their password, an employee cannot choose a password that is the same as one of their last few passwords (where the actual number is configurable). Use of this feature shall be configurable; that is, it shall be possible to switch it off.
Change employee password	There shall be a function that permits an employee to change the password of another employee. It is strongly recommended that access to this function be restricted to a very small number of users.

User De-Registration For each function that creates users, there needs to be at least one function that removes them. Normally it should be possible to de-register them in the same manner as they were registered–which means that if a user can self-register, they can de-register on their own initiative; and if a user is added by someone else, then someone else must be able to de-register them. It might be worth writing a separate de-registration requirement for each relevant class of user, as in the following two examples:

Summary	Definition
Customer de-registration	A customer shall, provided they have no outstanding business, be able to de-register, after which they shall never be able to use the system again (unless they re-register). A customer is deemed to have outstanding business if their account has a non-zero balance or if they have one or more incomplete orders. When a customer is de-registered, all information of a personal nature held about them shall be deleted (except insofar as it is needed in order to comply with other legal or regulatory demands). Contact details (such as physical address or email address) and transaction details shall not be regarded as being of a personal nature for the purposes of this requirement.
Employee de-registration	It shall be possible to de-register an employee, after which they shall never be able to use the system again (unless they are re-registered).

User Data Protection We return to the theme of users being special, this time not because they drive the system, but because they are people who are sensitive about what is known about them and about how information about them is used (and not abused). A well-specified system (and a responsibly-run business) will provide reasonable protection for user information, but that might not be enough. You need to worry about data protection and privacy laws that make further demands. These vary considerably around the world, so don't assume that what's acceptable where you live is good enough everywhere. Europe is generally stricter than the rest of the world, for instance.

Data protection is a large topic in its own right, deserving of more than this backwater subsection. But there isn't space in this book to do it justice, so we'll confine ourselves to suggestions on how to tackle this subject, list a few general guidelines, and give a few example requirements.

Here's a basic high-level process for specifying data protection requirements:

1. Find out which countries' data protection laws apply to the system. Obviously this includes the country (or all countries) in which the system is to be installed, but that might not be all. For instance, European Union data protection provisions extend to data recorded about EU residents by systems outside the EU.
2. Investigate the implications of all the applicable data protection laws, and write requirements for them. It's not good enough simply to write requirements that say, "such-and-such a

law shall be complied with"—because that encumbers someone else with figuring out what it means. (See the comply-with-standard requirement pattern in Chapter 5, "Fundamental Requirement Patterns," for more.)

3. Ask whether all demands of all data protection laws must apply from the first day the system operates, or you can obtain some leeway and postpone delivery of some features. Assign priorities to requirements accordingly.
4. Think beyond strict legal demands. Are there any practices that could cause your company embarrassment or loss of business if they became known? If so, write requirements that prevent these practices or render them unacceptable by the system. Laws that are tougher than any you need to comply with can be a source of ideas: they do, after all, exist to protect people. A list of a few key provisions follows. What's popularly regarded as acceptable can (and does) vary from one country to another, too.
5. If, along the way, you spot that a business activity that your organization is contemplating (or conceivably might contemplate in the future) could be illegal in some jurisdiction, don't let it pass. For example, you might be concerned that your company could be tempted to earn extra revenue by selling customer data. Deal with it either via a requirement, an informal note in the specification, or by bringing it the attention of senior management (depending on how likely or unlikely you feel it is to happen).

Some of the key data protection provisions are:

A. Capture only those personal details that you actually need and use.
B. Don't capture sensitive data about any person (unless it's vital to that person's use of the system, and they agree to it).
C. Don't use personal information for any other purpose than that for which it was gathered.
D. Let people know how information about them is used. In general, when a person supplies personal information, they are implicitly giving their approval for it to be used thus. But it might be appropriate for you to go further and ask them to express their consent. If the information is to be used in multiple ways, consider whether to let the user to choose which they accept. For example, you could let them opt out of receiving emails of various kinds.
E. Delete personal information as soon as it's no longer needed (including out-of-date and incorrect—and subsequently corrected—information). Because there could be other laws that mandate that we retain financial information for several years, there might be some data that we **must** keep and some that we **must not**.
F. Make provision for a person to see what information is held about them.
G. Store information about people only in the country in which the main system resides; don't move it to any other country.

A system that's built sensibly shouldn't have difficulty complying with these provisions. Don't regard this list as complete, though. Here are example requirements covering a few of these steps:

Summary	Definition
No sensitive information about any person	The system shall not record any sensitive information about any person. For the purposes of this requirement, "sensitive information" shall include all of the following: a. Racial or ethnic origin b. Political opinions c. Religious beliefs or other beliefs of a similar nature d. Membership of a trade union e. Physical or mental health or condition f. Sexual life g. The commission or alleged commission of any offense h. Proceedings for any offense committed or alleged to have been committed, the disposal of such proceedings, or the sentence of any court in such proceedings These provisions are in accordance with the U.K. Data Protection Act (1998), which is in line with European Union Directive 96/46/EC on data privacy. See *http://www.dataprotection.gov.uk*.
Explain use of personal information	There shall be a Web page that describes how the system uses the information it knows about a person, and the ways in which personal information is protected.
Let a person see information about them	It shall be possible to print out all the information the system stores about a selected person (except secret items used expressly for authentication, such as their password). The motivation for this requirement is to be able to respond to a person's request to see what information is held about them. Manual processes are needed to verify that the request comes from that person and to then furnish the print-out to them.

Good Security Practices First of all, we want passwords to be handled securely. Requirements can't guarantee this, but you might want to force developers to make some basic precautions, as these two examples do:

Summary	Definition
Passwords never displayed	Passwords shall never be displayed on the screen when they are entered or at any other time.
Passwords undecipherable	Passwords shall be translated into an undecipherable form as close as technically possible to the point in the system at which they are entered. Any occurrence of the clear password shall be discarded immediately and never recorded in any kind of permanent storage. This requirement applies to every function that asks for entry of a password. It is strongly recommended that a one-way hashing algorithm be used for translation into an undecipherable form, because there is then no technical way to retrieve the original password from it.

Then you might want to outlaw a few bad practices such as loopholes, ways to bypass security precautions, and proliferation of software that asks users to enter secret information (such as their password). There are situations in which developers might be tempted–for honest reasons–to do something that weakens security. A few examples are:

Summary	Definition
No special user IDs	There shall be no user IDs that are treated differently by the software itself. That is, it is unacceptable for any piece of software to condition any action on the value of the user ID itself or any other user information (such as their name or address). The intent of this requirement is to prevent "back door" access of any kind, for any reason. This includes (but is certainly not limited to): user accounts for "demonstration" purposes; features to assist testing, debugging, problem investigation, or administration; and different handling of this user at login (such as use of a fixed password or other differences in authentication).
No access via system-generated passwords	No user shall be allowed to use a system-generated password as their normal password. The only acceptable use of a system-generated password is to give the user first-time access to the system; the user must be forced to change the password the first time they log in. A system-generated password is any password computed by the system itself or hard-coded within its software (though use of the latter in any circumstances is frowned upon).
Application software must not ask for password	No application software itself shall ask a user to enter their password unless that part of the software has been explicitly designated as being for security purposes. The intent of this requirement is to keep to a minimum the number of pieces of software that ask a user for their password, because each represents a risk—something that could be subverted to capture the password and pass it on for improper use.

Special Processes for New Users Whenever a user is set up by someone else, we have the pesky problem of how to assign a password that only the new user knows, without the risk of someone else logging in as that new user first. Options include:

Option 1 Have the system automatically generate a password for the new user, which is then told to them. One way is to use a simple algorithm for the new password (as in the second preceding example user registration requirement), and for the new user to be told that algorithm. Always insist that a different password be generated for each user, rather than a standard password being used for everyone–because of the risk of the latter becoming widely-known. A second way to tell the user their password is to display the new password to the person setting up the account, who then tells the new user.

Option 2 Ask the new user to choose a password and type it in when their account is being set up. This insists that they be physically present, and runs the risk of the other person watching them enter their password–but at no time is it possible to log in as the new user with an artificial password.

Option 3 Ask the user to type their chosen password into a special preregistration function, which makes it available for the registration function to pick up later. Such a function must comply with all requirements for the secure handling of passwords (such as storing them in an undecipherable form).

The second option doesn't need extra requirements. Here are a couple for the first option:

Summary	Definition
New employee must change password	The first time a new employee uses the system, they must change their password.
Block inactive new employee	If a new employee has not logged in within a given length of time (say, three hours) after being set up, their access to the system shall be blocked. The purpose of this requirement is to limit the risk of another person discovering the absence of the new employee, being able to guess the initial password generated for them, and effectively taking over the new employee's account.

Day One Considerations More than a few systems have been installed and started up on their first day–only for the project team's pride to be replaced by red faces when it's realized that no one can log in and no one can register a new user. Since it's a one-off occurrence, this might sound like it doesn't deserve a place in the requirements. But solving it might leave an ongoing security flaw, so it's best that everyone be made aware of the potential risk. We don't want a surreptitious utility to be built that can set up new users and then hangs around in the hands of one or two developers or systems administrators. If such a utility is needed, make sure it's known about and kept under strict control. Here's an example requirement to prevent one cause of a bad first day:

Summary	Definition
New system usable	When the system is freshly installed, it shall be usable by someone, at least to the extent that they can set up a new user and grant them access to system functions. The purpose of this requirement is to prevent a newly-installed system from being unusable by anyone. Any special processing needed to render the system usable before any real users have been registered shall be permanently disabled or removed from the system as soon as at least one real user has been registered.

Considerations for Development

The range of topics raised in this requirement pattern makes it clear that there is much to think about. The best advice is simply to take user registration seriously and not to cut corners. Always handle secret values (such as passwords) securely: at the first opportunity, render them indecipherable, and destroy the clear form.

Considerations for Testing

A user registration function is basically a data entry function and should be tested as such. Test the entry and validation of all values. Test that validation is not performed only on an untrusted client device (a remote user's PC). Test that any values that are intended to be secret (such as passwords) are treated as such, and cannot be viewed using the system (such as via inquiries or reports or obtained from logs or from querying the database).

Testing that a user registration process is secure cannot be done solely by *using* the system. It's a specialized activity (as discussed in the user authentication requirement pattern).

11.2 User Authentication Requirement Pattern

Basic Details

Related patterns:	Accessibility
Anticipated frequency:	One or two requirements
Pattern classifications:	Functional: Yes

Applicability

Use the user authentication requirement pattern to specify that a person must make their identity known to the system before they can access anything non-public or anything for which they cannot remain anonymous (in short, anything for which they must log in).

Discussion

Authentication is the process by which a registered user declares who they are and proves this assertion to the system's satisfaction. This requirement pattern says little on the subject beyond what is of interest to requirements; it doesn't worry about the considerable challenges of implementing a secure system. The most widespread form of authentication in commercial systems is just asking the user to enter their user ID and password, and that's what this requirement pattern concentrates on. The common name for user authentication is "logging in." (This book prefers logging *in* to logging *on*—on the basis that *entering* a system is a better metaphor than *climbing atop* it. Let's also point out that, as a verb, to "log in" is two words: "to log someone in" sounds fine, and is only possible if there is a gap in which to squeeze "someone.")

There are three accepted ways to check that a person is who they claim to be: something they **know** (such as a password), something they **have** (such as an ID card), and something they **are** (using a part of their body that can be distinguished from that of everyone else—such as a finger or an eye). Depending on how secure you want your authentication process to be, you can ask for more than one (a password *and* an ID card *and* a fingerprint, if you insist). There are various factors that affect what's best in a particular situation, including:

Factor 1: The potential damage that an impostor could cause This depends on what the user has access to—not just which functions but also other assets (for example, a customer who has a large sum of money in their account), so you might want to consider stronger authentication for users who have the greatest powers (or valuables) in the system. Also consider possible *indirect* damage, such as to the company's reputation if unauthorized access became public knowledge.