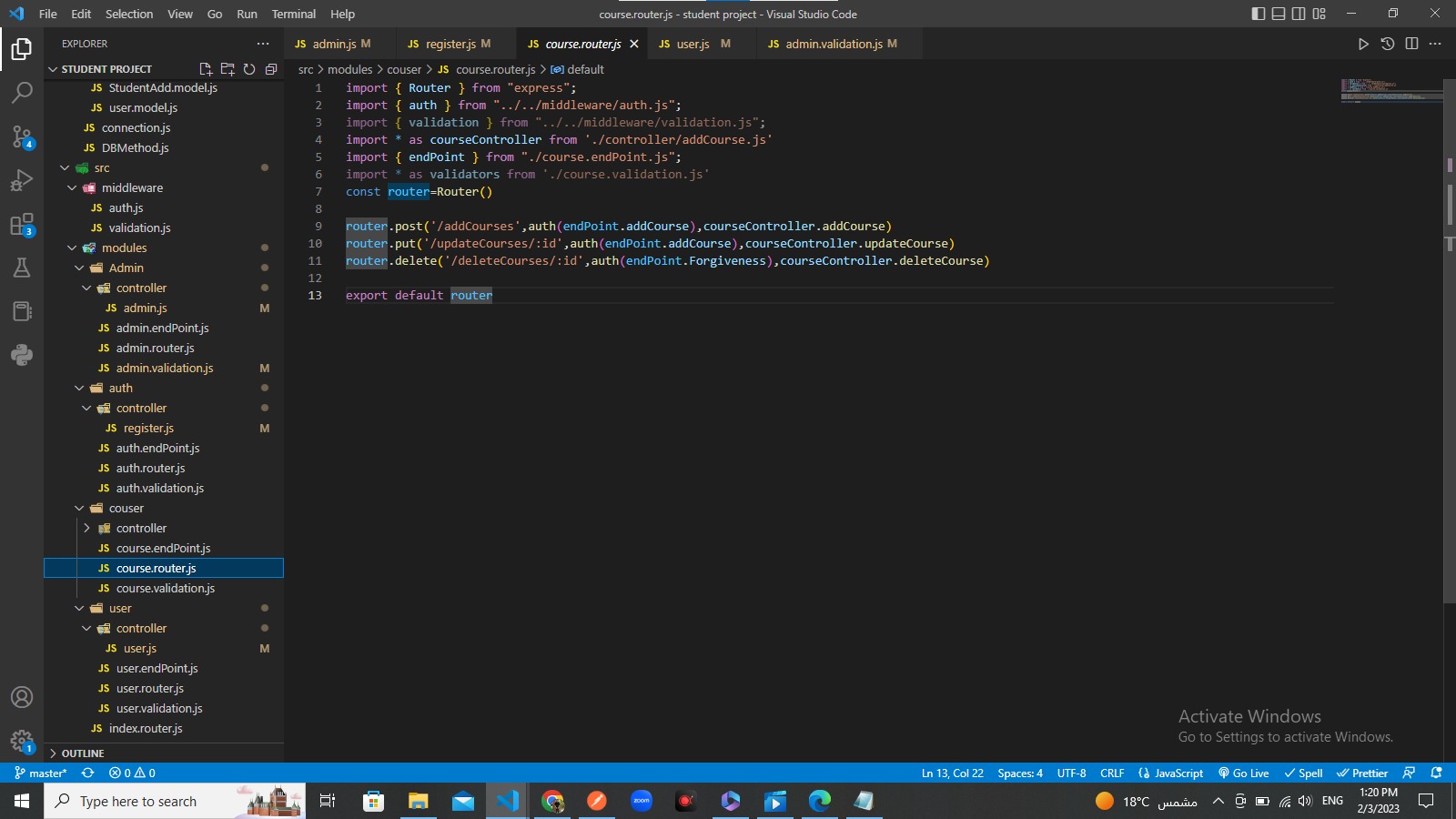
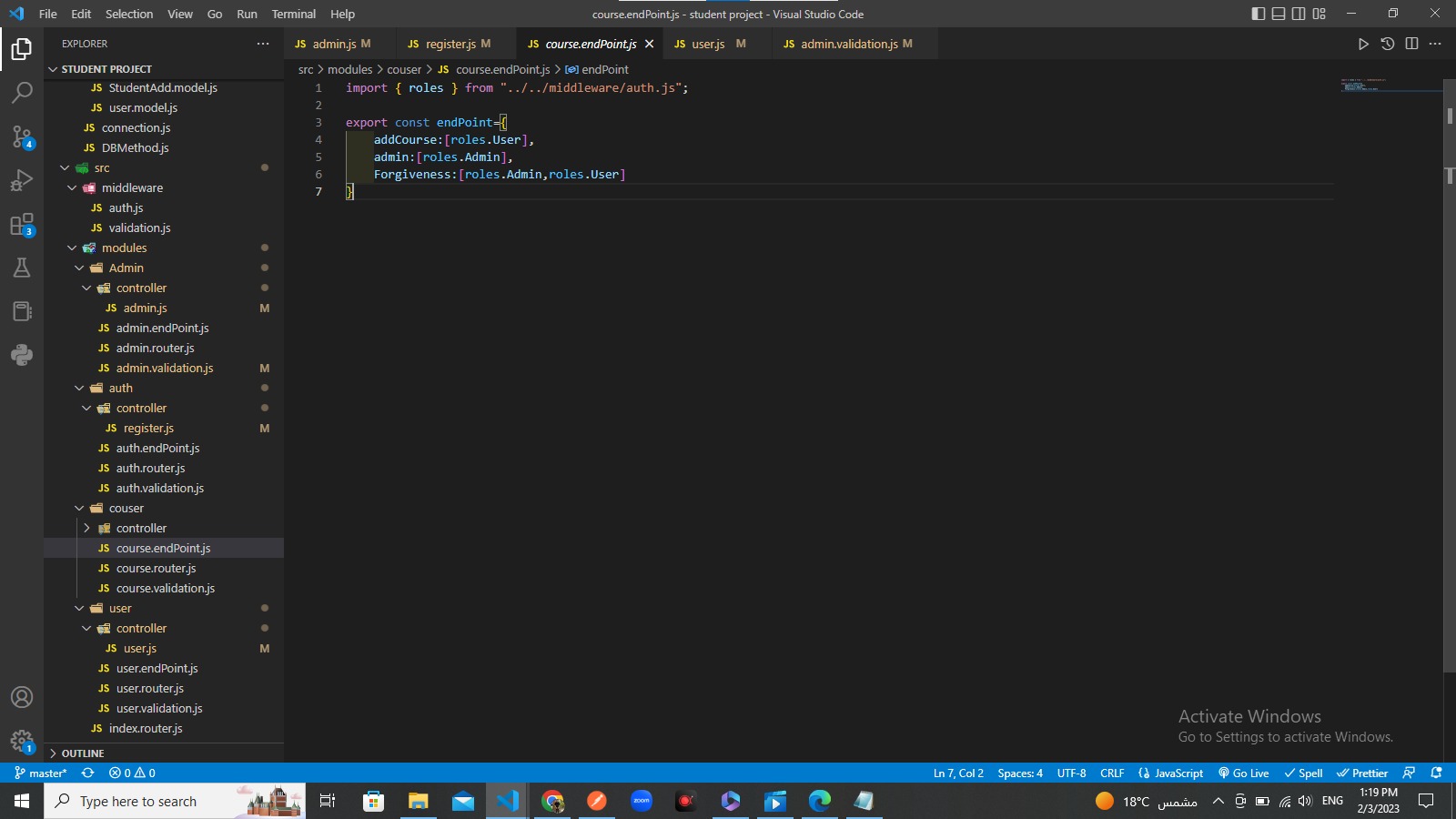
**What Is Authentication?**

At a basic level, authentication refers to identifying an user to confirm they are who they claim to be they are before being granted access to information resources. A common way to authenticate a user is by using a combination of a username and password. Authentication proves the user is whom they say they are — only the user is supposed to know his or her exclusive username and password combination.

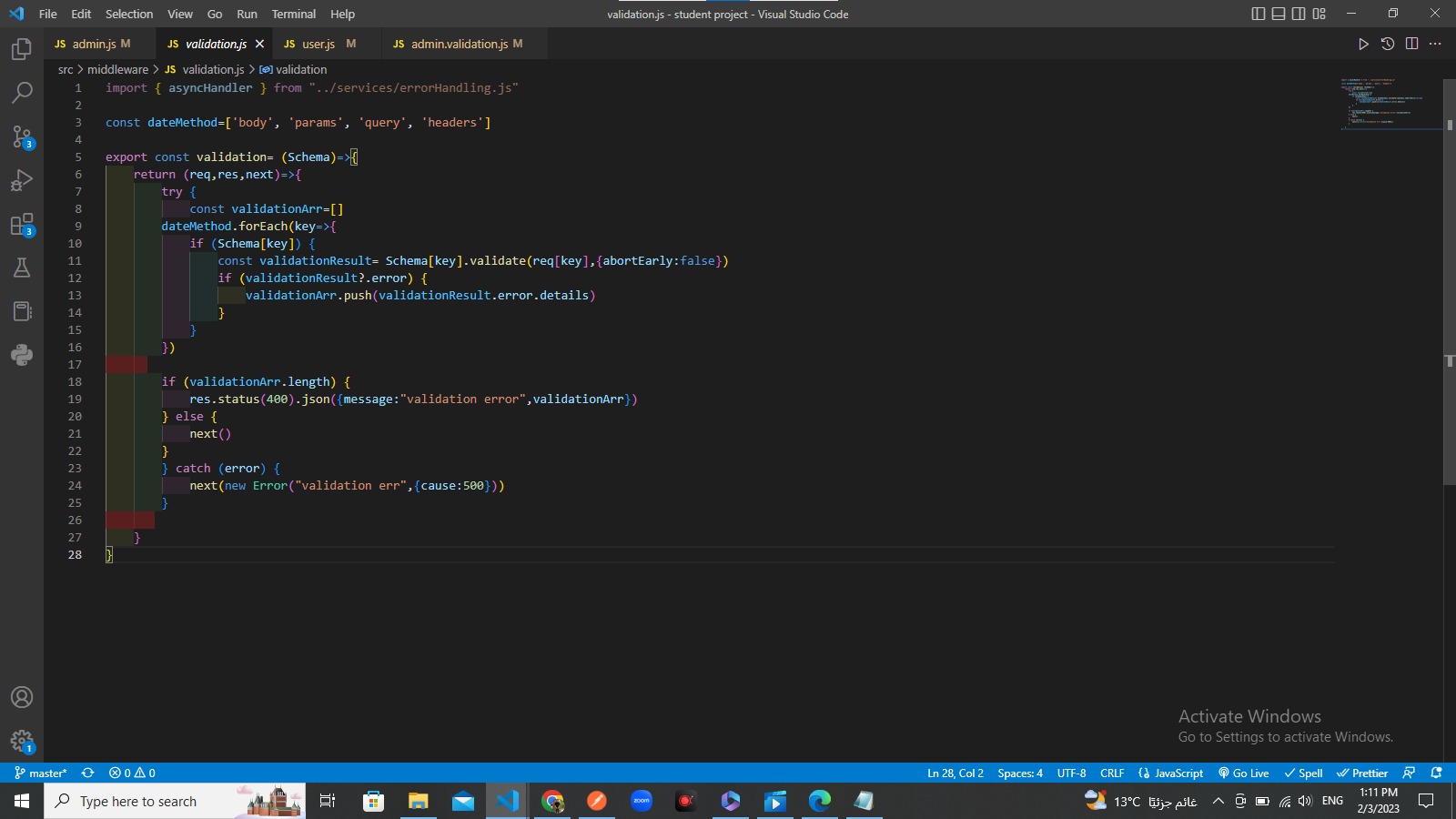
Although the traditional username and password combination — also known as user credentials — is still the primary form of authentication, it's just one piece in a larger puzzle of digital security. Secure authentication systems now feature designs to authenticate users using multiple factors, including one-time passwords or biometric markers. This strategy makes it challenging for bad actors to impersonate employees and steal important information.





**Validation**

 validate the form on the client-side so data processing will be faster than server-side validation. It is preferred by most of the web developers.



**Key Takeaways**

* Authentication is an essential information security process requiring employees to verify their identities before gaining access to systems, applications or other information.
* Usernames and passwords have been a primary form of authentication. New forms of authentication supplement user credentials with other factors.
* By switching to more progressive forms of authentication, businesses can mitigate the effects of credential theft and protect their most critical data.

**Authentication Explained**

What do you mean by authentication? Authentication verifies a user's credentials. In most cases during onboarding, employees create a username and password to prove their identity and access tools and data they are authorized to use. Providing access control in the form of authentication is the acritical step in information security. While the most basic form is a username and password, further efforts can include multi-factor authentication, or even biometric authentication.

Authentication is a way to keep your proprietary and private information secure. Everything from access to programs and applications to confidential customer information, and even details on pricing and suppliers.

**How Authentication Works**

There are different approaches to authentication — traditional user ID and password authentication, multifactor authentication, public key authentication and more. Each type has advantages and disadvantages, and they all have some common attributes.

First, let's talk about the basic setup. When users want to log in to an application or computer system, they input their user ID and password into the requisite form. These details then send to an application server, which matches the input credentials against the stored credentials connected to the username. If the two inputs match precisely, the user is free to access the services they're authorized to use.

Sometimes, attackers target authentication servers specifically, which might allow them to intercept passwords as users enter them. To mitigate against this possibility, some system administrators use public key authentication. Here, the user generates a key pair — one public key shared with many services and private key users keep secret. Using the private key, the user generates signatures. When the user logs in, a server uses the public key to validate these signatures and authenticate the user.

While public key authentication is secure, it can be inconvenient. And if the key is stored unprotected on your computer and that computer is stolen, the thief can impersonate the user. In general, authentication methods should make it easy for users to log in and difficult for bad actors to log in. They should also avoid single points of failure.

There are several different approaches to authentication, and public key is just one. But there's no silver bullet — the type of authentication should fit the characteristics of an organization as well as the kind of information it's trying to protect.

## Why Is Authentication Important?

Authentication is important for many reasons, security being foremost. Without authentication, strangers from the public internet would be able to access and modify, copy or delete a company's corporate data and applications. Authentication is one of the processes forming the "perimeter" of the corporate network — it separates proprietary computer systems from the internet at large.

What is the purpose of authentication? Authentication is also necessary because it triggers the authorization process. Authentication establishes the identity of the user, while authorization establishes the user's privileges. When "Bob Smith" authenticates with his credentials, the authorization system links his identity with the role of a junior salesperson, allowing him to access a VoIP application, an email client and the CRM. Bob can't access the firewall, network monitoring controls or security logs, even though he established his identity.

Authentication is also important because it can include measures designed to mitigate identity theft. Let's say Bob steals credentials from the administrator Jane Doe. Some authentication systems would let Bob validate using Jane's credentials, allowing Bob to access secure systems and data. More progressive authentication systems could use a device fingerprinting technique to detect anomalies with the login — for example, Bob might be using a Windows PC, while Jane typically uses a Mac. That would prompt the authentication system to ask for an additional authentication factor, preventing Bob from logging in with Jane's credentials.

## How Authentication Is Used

Authentication is applied when someone needs to use a computer system containing information or applications not meant for the public. Typically, this means logging in with a user ID and password, but there are other ways to authenticate. For example, a user might log in with a user ID, a password and a biometric identifier like a retina scan. At a banking website, a user might log in with their bank account number and password. Users might also be asked to authenticate themselves using information such as their Social Security number or driver's license number.

In general, [authentication establishes the identity of the user](https://www.netsuite.com/blog/establishing-trust-and-identity-in-the-cloud). And the more sensitive the information under guard, the more thorough the authentication process. If people are only trying to access a social media website, for example, they'll probably just need their user ID and password. If they're trying to access their bank account, they might need to enter a one-time pass code sent to a verified phone number. If they're trying to access something like legal documents, then authentication might require a strongly identifying token such as a Social Security number.

## Authentication Methods & Levels

Which authentication method is best may depend on a few factors, but some authentication types are much more secure than others. The average user can encounter three types of authentication. What are the three types of authentication? Password-based authentication, two-factor authentication and multifactor authentication. Methods of authentication under this umbrella include biometric authentication, mobile authentication and API-based authentication. What does each of these involve?

### Traditional authentication

All users need to provide are usernames and passwords they memorize. There aren't any hedges against password theft — if someone steals the password, he or she can easily impersonate the user.

### Password-based authentication

This type is an improvement on traditional authentication. Here, the password scrambles using one-way encryption called a hash, and the hash stores on an authentication server. When users input their passwords, it's hashed, and the input hash is compared to the stored hash on the server. Even if an attacker steals the list of hashes, it's much more complicated (if not impossible) for them to discover the user's password from this theft.

### Two-factor authentication

Here, users enter their passwords and one additional factor. This type usually takes the form of a one-time password that gets texted to the user's device. By entering this password, the user proves they're in control of the device used to authenticate.

### Multifactor authentication

Here, the user must enter an additional factor, but the factor can take many forms. It might be a one-time password, but it could also be a secure USB key, an ID card or a biometric marker.

### One-time password

This type is a temporary password. It is used only once and then deleted. These passwords are usually texted or emailed to a user to prove they control the device they're using to log in. Also, these passwords can be created using random-number generator apps. Keep in mind SMS and email are vulnerable to interception.

### Three-factor authentication

Unlike multifactor authentication, three-factor authentication requires users to provide two other factors in addition to their password. This might include a one-time password and the last four digits of a social security number. Three-factor authentication is used for highly sensitive data and systems.

### Biometrics

These are immutable factors serving to distinctively identify a human being — fingerprints, faces and retina scans are all distinct at an individual level and can be used to authenticate a user.

### Mobile authentication

This authentication method uses a mobile application to generate a random number which serves as a one-time password.

### Continuous authentication

Typically, a user authenticates once and is then authorized. With continuous authentication, a system performs identity checks — such as device fingerprinting — throughout the user's session, ensuring an attacker can't take it over.

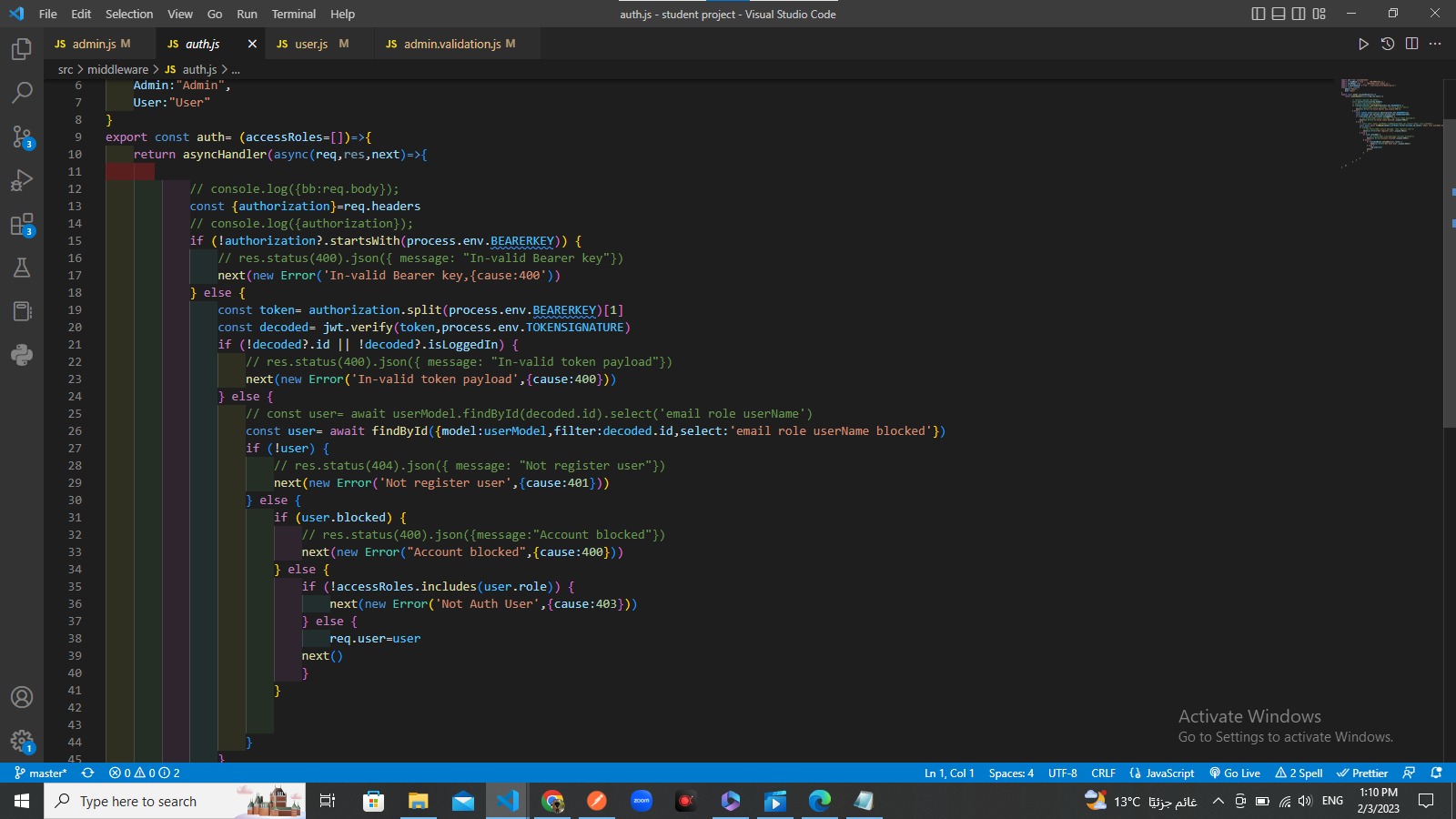
### API authentication

An application programming interface (API) is a way for two software programs to communicate. API authentication helps prevent attackers from impersonating software programs. API authentication methods include:

* + **HTTP basic**  
    Here, a client application is simply assigned a username and password. These factors are encoded using an HTTP header and automatically transmitted to an authentication server before the client makes a request.
  + **API key**  
    API keys are based on public key authentication. The client application has a private key that generates signatures and a public API key shared with servers. The client adds their signature to the request header, and the server authenticates their identity using its stored public key.
  + **OAuth**  
    Lastly, OAuth lets applications authenticate using a token instead of a password or an API key. That means applications don't have to share any credentials vulnerable to theft.

## What Is Authorization?

What do you mean by authorization? Authorization determines the level and type of access to resources that a user has. It answers the questions who can do what with your data and applications. Once a user is authenticated (more on this later) with their user credentials such as username and password, their Authorization will determine the predetermined menu of operating systems, applications, functionality and the level and ability to make changes to underlying data. Collectively, these permissions are known as client privileges.



In most organizations, the authorization structure is not flat. Authorization does not grant access to every resource. Instead, different employees have access to different resources. Companies use various processes to determine which employees can access a resource based on what they need to do their jobs. By restricting access to certain materials, businesses have a way to protect their most critical data – including intellectual property, consumers' identity, medical records, payroll information and more. Additionally, it can be a tool to help employees find the resources and information they need more quickly without having to wade through every company document and folder. Generally, organizations follow a principle of least authority when granting access to information. This approach ensures that users and employee only have access to information that is required for their role and nothing more than that. The damage a potential bad actor can do is minimized.

**Key Takeaways**

* Authorization is an essential best practice for both security and compliance.
* Not only does authorization protect your information, but it also preserves the privacy of consumers and can shelter companies from fines resulting from compliance violations.
* Although authorization is not necessarily easy in large organizations, the proper access control process can help mitigate this difficulty and protect users, data and customers.
* Follow principle of least authority when establishing authorization parameters.

**Authorization Explained**

Why do we need authorization? There are two primary reasons. First, you want to protect your business information from bad actors both inside and outside the company. If an employee misplaces or has credentials stolen, controlling what data may be vulnerable is important. One goal of authorization is to ensure that each employee has access only to systems and information they need to do their jobs, and not to all business data. This way, if thieves manage to steal credentials, they'll be limited in how much information they can access.

The second reason deals with compliance. For example, the Health Insurance Portability and Accountability Act (HIPAA) requires that documents such as medical records need to be kept private. Not all employees should have access to patient information, and those who do will need to undergo regular trainings and that information will need added layers of security.

If the U.S. Department of Health & Human Services (HHS) finds employees or others have improperly accessed or exposed patient records, HHS can levy fines against the organization for this compliance breach. By limiting access, providing training and closely monitoring access to confidential information, health care organizations can protect patients while also avoiding fines and legal liability. Many other organizations, including those outside the health care space, are subject to similar controls. For example, schools and companies that work with students are governed by similar rules under Family Educational Rights and Privacy Act (FERPA), which protects the privacy of student records. Other information you need to protect includes private employee information, such as social security numbers, and customer information, such as credit card numbers.

**How Authorization Works**

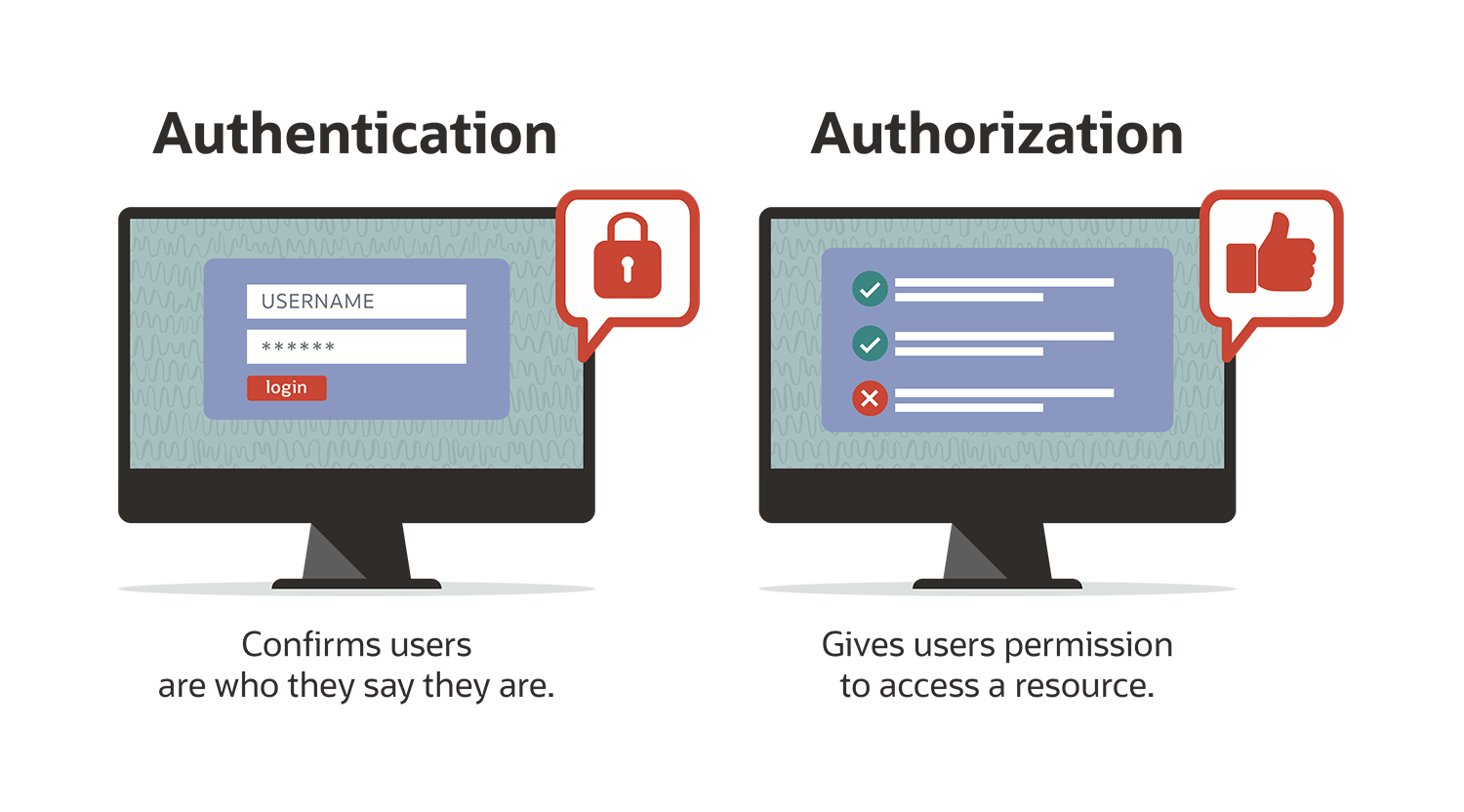
At its most basic level, authorization is allowing access to certain applications and information with a username and password. A user might be authorized to access a word processor, an email client, a CRM and more. With basic authorization, users would have a distinct user ID and password for each system. For example, your employees might need one log in for the CRM, another for their email, another to access a server, and so on. When they need access to new systems or information, they send requests to an administrator, who reviews the request and then provides an additional login credential.

Basic authorization has some challenges.

* **It doesn't scale.** As the company grows, manually tracking which employees have access to programs and information becomes an unsustainable burden for administrators to do manually.
* **It's not convenient.** Without a single log in service, your employees have to recall multiple separate usernames and passwords just to go about their work. This can also result in insecure behaviors such as writing down passwords on post-it notes.
* **It's not secure.** Suppose a user needs to make an access request to access a single financial record. With basic authorization, the administrator may have to turn over access to the entire database to that employee and then remember to revoke access later. This can pose security and compliance issues, depending on the information that’s stored.

Robust authorization protocols have systems to quickly and automatically generate client privileges for employees, single sign-on systems and offboarding processes that automatically terminate temporary access to essential systems once it is no longer needed.

**Authorization vs. Authentication**

[](https://www.netsuite.com/portal/assets/img/business-articles/erp/infographic-bsa-erp-authorization.png)Authentication and authorization are often used interchangeably — but there are some important differences. Authentication is a way for users to verify they are who they say they are. This can be done through usernames and passwords. And increasingly popular is two-factor authentication. This requires not just a username and password, but also another layer of security through another device, most often via a phone app or a texted code to a user’s phone. Authorization follows authentication. Just because users are authenticated, this doesn't mean they receive access to all applications, services and data within the corporate network. Instead, they only get to access the resources that they are authorized to use.

The two steps of access control, authorization and authentication, are both necessary for a user to interact with a corporate network. If users aren’t authenticated, they can't use any services because they can't get into the network. If a user isn't authorized, they won't be able to use any services even if they're authenticated.

## Why Use Authorization?

Authorization is used for security, compliance and operational reasons.

It’s vital to information security because it represents one of the innermost components of a concept known as "defense in depth." The idea of defense in depth is that a company should place many information security applications — firewalls, monitoring tools and identity management, to name a few — around the corporate network's perimeter. If an attacker evades one application, other applications still have a good chance of mitigating their attack. However, even defense in depth can be thwarted if employee credentials are stolen. But if each employee only has access to a small and relatively unimportant amount of data, an attacker needs to steal multiple credentials to pull off a meaningful attack. That gives them a much greater likelihood of being caught and stopped.

One of the best illustrations of authorization for compliance reasons is HIPAA, which is meant to protect private patient data. But other industries need to maintain compliance in protecting confidential information, such as customer and employee data. If things like credit card numbers or personal employee information are left vulnerable, your company could be held liable for damages, not to mention potentially losing valuable team members or customers and bad PR.

Authorization is also a key component of operations for many workplaces. Making sure your employees have enough information to get their jobs done, but aren’t bogged down with the entire company data sources, can help improve worker efficiency. And a single sign-on can make it easier for them to access necessary information and resources, as well as make the authorization more secure.

## Authorization Use Cases

Preventing unauthorized employees from looking at critical data is another reason why authorization is essential. HIPAA is an excellent example of why it is critical, but let's look at a few more examples of what the process looks like in the context of compliance.

### Offboarding former employees

Not every employee leaves on good terms with an organization. If employees think they've been terminated unjustly, they might try to steal valuable information on their way out the door. Businesses can use authorization processes to automatically terminate corporate account access after parting ways with personnel.

### Working with vendors and contractors

Businesses often need to share information with third parties. For example, a business may be working with a managed service provider to migrate its applications to the cloud. Here, the MSP would need to access their applications and data. With authorization, businesses can assign client privileges to vendors, allowing them to view this information and move it around but preventing them from editing or deleting their data.

### Diminishing privilege creep

Privilege creep means employees are gradually accruing permissions they don't need. For example, suppose an employee transfers from sales to accounts payable while retaining her sales permissions. This employee may still have access to the CRM, even if she no longer needs it. That makes the employee a potential security risk – if her credentials are stolen, it opens the business up to even more vulnerabilities. Using up-to-date processes, administrators can detect if employees have permissions they don't need and then roll them back.

### Authorization Approaches and Methods

Authorization is used in many contexts to defend businesses and their critical information, but how is it accomplished? There are several methods, each with its advantages. The overriding goal is to make it easy to manage enterprise-wide authorization while making it difficult for employees to accumulate permissions that they don't need.

### Token-based authorization

One of the primary concerns of managing authorization without a single sign-on is that employees often find it inconvenient, and they'll think of ways to work around it when necessary. For example, imagine employees needed to re-enter one of several different passwords every time they left their computer or navigated away from a website. They'd probably find their productivity to be slow, and they'd be tempted to resort to insecure practices.

With token-based authorization, the user only needs to log into an application once. In exchange for their credentials, they receive a token from the system. The token takes the form of a small text file that's stored in the user's browser. As long as the user continues to store that token, they can stay logged into the service without re-authenticating.

### Role-based access control (RBAC)

Token-based authorization works well, but it's usually used in conjunction with other systems. How does an administrator understand which systems to grant an authorization token? Often, they do this using role-based access control (RBAC).

RBAC rapidly assigns permissions to new and existing employees based on their role in an organization. With RBAC, you start by creating a predetermined list of every position within the company and then deciding what resources this position should access. There's no need to spend time wondering which permissions an individual employee should have once they're hired – that decision has already been made.

This method is common among companies with large numbers of employees, but it can have drawbacks. For example, if an employee needs access to a new resource, the administrator needs to handle that request individually. It isn't easy for a large company to process these requests promptly and then revoke access once it's no longer needed.

### Access control lists (ACLs)

RBAC works by determining which applications and files a given role should have access to and then placing users within these roles. Access control lists (ACLs) work in reverse. Instead of starting with a role and then appending a list of applications that the role can access, ACLs start with the applications and files.

Let's say that a user — call him Bob Smith — wants to log into a database. After Bob authenticates his credentials, the application checks its embedded access control list to see if Bob's name is there. If Bob's name is on the list, then he can access the application.

In some cases, RBAC is somewhat easier to use than ACLs because it can be challenging to go to every application in an organization and add a user's name to a list if he or she is allowed to use it.

ACLs have a security use case when it comes to the network perimeter. Since ACLs can be applied to infrastructure, such as routers, administrators can place ACLs on routers controlling the network's edge. Here, the ACLs don't just control which credentials are authorized to access the network – they also act as traffic filters. Based on characteristics such as the IP address and port number of incoming packets, the ACL can accept or deny specific traffic, adding to the network's security as a whole.

## Authorization Examples

What is authorization? Here are a few examples. These include somewhat rare methodologies compared to token-based authorization, RBAC or ACLs, but they may become more prevalent in the coming years as companies look for better authorization methods to improve security.

### Attribute-based access control (ABAC)

With this process, users are directly associated with an attribute that grants them access to a resource. One commonly used example is a secure USB key. Possessing a secure USB key is an attribute, and having one can grant access to sensitive files and applications within an organization.

### Mobile access control

This control is much like ABAC. Except, instead of possessing any attribute, the attribute in question is possessing a smartphone. Specifically, the user downloads a mobile credential application and then uses their smartphone to authenticate. Their authorizations store on their device as well. An example of this would be using a phone to make a mobile payment — tapping a phone on the pin pad authenticates the users’ credentials, which then authorizes them to exchange money for goods and services.

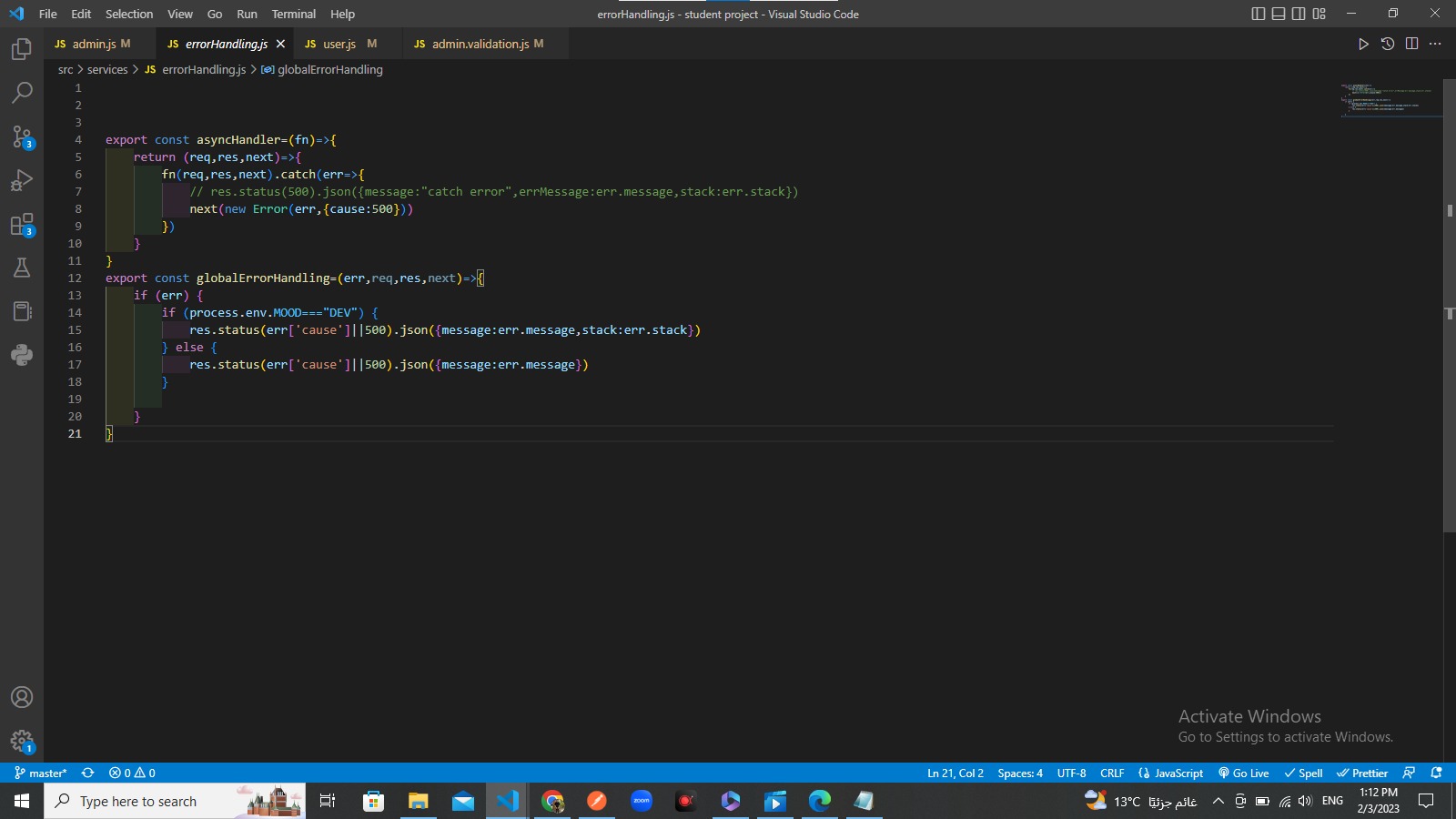
### Graph-based access control (GBAC)

Here, instead of configuring access by employee or role, permission is configured at the objects level such as files and applications. Access rights generate using a query language instead of exhaustively listing the permissions assigned to each role, which reduces workloads.

NetSuite offers a flexible and highly customizable access control system compatible with whatever authorization process administrators have in place. This system promotes security and compliance, allowing you to focus on core competencies while protecting users and customers.

## What Does Error Handling Mean?

Error handling refers to the response and recovery procedures from error conditions present in a software application. In other words, it is the process comprised of anticipation, detection and resolution of application errors, programming errors or communication errors. Error handling helps in maintaining the normal flow of program execution. In fact, many applications face numerous design challenges when considering error-handling techniques.



**Techopedia Explains Error Handling**

Error handling helps in handling both hardware and software errors gracefully and helps execution to resume when interrupted. When it comes to error handling in software, either the programmer develops the necessary codes to handle errors or makes use of software tools to handle the errors. In cases where errors cannot be classified, error handling is usually done with returning special error codes. Special applications known as error handlers are available for certain applications to help in error handling. These applications can anticipate errors, thereby helping in recovering without actual termination of application.

There are four main categories of errors:

* Logical errors
* Generated errors
* Compile-time errors
* Runtime errors

Error-handling techniques for development errors include rigorous proofreading. Error-handling techniques for logic errors or bugs is usually by meticulous application debugging or troubleshooting. Error-handling applications can resolve runtime errors or have their impact minimized by adopting reasonable countermeasures depending on the environment. Most hardware applications include an error-handling mechanism which allows them to recover gracefully from unexpected errors.

As errors could be fatal, error handling is one of the crucial areas for application designers and developers, regardless of the application developed or programming languages used. In worst-case scenarios, the error handling mechanisms force the application to log the user off and shut down the system.