VOCABULARIO

1. **Acceptance criteria:** Conditions that a system must meet to be accepted by a user or stakeholders.
2. **Acceptance tests:** Confirm that system meets user requirements before final deployment.
3. **Access Control:** Managing and regulating access to resources or systems.
4. **Acts:** Individual commands in a script or program.
5. **Adaptive design:** Adjusts website layout to different screen sizes.
6. **Adjustment:** A minor change made to a system or process.
7. **Agile methodology:** An iterative and flexible approach to software development.
8. **Agile software development:** Iterative and flexible approach to software development.
9. **AI:** Artificial Intelligence, a branch of computer science that seeks to create intelligent systems.
10. **AJAX:** Technique for asynchronous web page updates.
11. **Algorithm:** Set of instructions for solving a problem.
12. **Algorithmic solution:** A set of step-by-step instructions to solve a specific problem.
13. **Analysis:** The process of examining and breaking down a complex system or problem into smaller parts to understand its components and relationships.
14. **Android:** Mobile operating system developed by Google.
15. **API integration:** Connecting different software components to enable communication.
16. **API:** Interface for software components to interact.
17. **Architecture:** The high-level structure or design of a system, defining its components and their interactions to achieve specific objectives.
18. **ASCII:** Code that numerically represents a character, such as "a" or "@.".
19. **Asynchronous:** A technique where tasks operate independently and don't wait for each other to complete.
20. **Authentication:** Verifying the identity of a user or system to ensure secure access.
21. **Authorization:** Granting appropriate access rights to users or systems based on their authenticated identity.
22. **Automated test:** Test run automatically by software.
23. **Backdoor:** Unauthorized access point to a system.
24. **Backend Development:** Writing server-side logic and handling database interactions to support the functionality of a software application.
25. **Backup:** Copy of data used for recovery in case of loss.
26. **Bandwidth:** Amount of data transferrable in a given time.
27. **Batch:** It is a file that chains commands that are carried out by the PC when it is invoked.
28. **BCC:** BCC is the acronym for "Blind Carbon Copy."
29. **Benchmarking:** Comparing performance of different systems or software.
30. **Big data analytics:** Analyzing large and complex datasets for insights.
31. **Big data:** Large and complex datasets requiring specialized processing.
32. **Binary code:** Machine-readable code using 0s and 1s.
33. **Binary digit (Bit):** The smallest unit of data storage, either 0 or 1.
34. **Binary large object (BLOB):** Storing data without a specific format.
35. **Binary representation:** Binary representation is a way of representing data using only two states or values: 0s and 1s. It is used by computers and other digital systems to encapsulate and manipulate numerical values.
36. **BIOS initialization:** Basic Input/Output System starting hardware during computer startup.
37. **BIOS:** Basic input/output system, initializes hardware during startup.
38. **Bitbucket:** Software development platform similar to GitHub.
39. **BLOB:** Binary large object, data stored without specific format.
40. **Blockchain technology:** A secure digital ledger using cryptographic techniques.
41. **Blockchain:** Digital ledger of transactions secured through cryptography.
42. **Boot:** Starting up a computer or device.
43. **Bootstrap:** Front-end framework for web development.
44. **Bootstrapping:** Self-starting a computer or device.
45. **Botnet network:** A network of compromised computers controlled remotely.
46. **Botnet:** Network of compromised computers controlled remotely.
47. **Branch:** Different versions of code within a single project.
48. **Breakpoint:** Pause execution of program to debug.
49. **Browser:** Software used to access websites.
50. **Bug:** Error in software code causing issues.
51. **Business analysis:** Analyzing business needs and translating them into technical requirements.
52. **Business Continuity:** Planning and processes to ensure that critical business functions can continue in the face of disruptions.
53. **Byte storage:** the basic unit of information in computer storage and processing. A byte consists of 8 adjacent binary digits (bits), each of which consists of a 0 or 1.
54. **Byte:** Group of 8 bits, basic unit of memory and storage.
55. **C#:** Object-oriented programming language developed by Microsoft.
56. **C:** General-purpose programming language.
57. **C++:** General-purpose programming language based on C.
58. **Cache:** Copy that a computer maintains of the recently visited web pages, if the user requests them, they are read from the hard drive without having to connect to the network again; thus, achieving a very noticeable improvement in speed.
59. **Captcha:** Completely Automated Public Turing, or Completely Automated Public Turing. It is a small test that helps determine whether a user is a computer or a human.
60. **Carriers:** Telecommunications operators who own the Internet backbone networks and are responsible for data transport.
61. **Cascading Style Sheets (CSS):** Defines the visual style of web pages.
62. **CD:** Compact disc for storing data.
63. **CD-RW:** Rewritable compact disc, which can be written, erased and rewritten over.
64. **Central processing unit (CPU):** The main brain of a computer.
65. **Client-Side Scripting:** Executing scripts on the user's browser to enhance interactivity.
66. **Cloud computing:** Accessing services over the internet instead of locally.
67. **Cloud service:** Services accessed over the internet instead of being hosted locally.
68. **Cluster:** Hard drive storage unit.
69. **Code branching:** Creating different versions of code within a single project.
70. **Code compilation:** Translating programming code into machine-readable instructions.
71. **Code declaration:** Specifying a variable, function, or other entity in code.
72. **Code Refactoring:** Restructuring existing code to improve its readability, maintainability, and efficiency.
73. **Code Review:** The systematic examination of source code to find and fix issues, ensure quality, and share knowledge among team members.
74. **Code Smells:** Indicators of potential issues in code, suggesting the need for refactoring.
75. **Code:** Set of instructions for a computer to execute.
76. **Command execution:** Issuing instructions to a system or program.
77. **Command:** Instruction given to a system or program.
78. **Commit:** Submitting changes to a version control system like Git.
79. **Compact disc storage (CD):** Storage medium for data.
80. **Compatibility testing:** Ensuring software works on different systems.
81. **Compiler:** Translates programming code into machine-readable instructions.
82. **Compliance Audits:** Evaluations to ensure that software processes and practices adhere to established standards and regulations.
83. **Compliance:** Adhering to laws, regulations, and industry standards relevant to software development and operation.
84. **Component tests:** Tests individual parts of a larger software system.
85. **Compress:** Reduce the size of a file to save space or to transmit it at a faster speed.
86. **Condition:** Statement defining if a specific situation exists.
87. **Conditional code:** Code that executes based on specific conditions.
88. **Conditional statement:** A statement that defines if a specific situation exists.
89. **Connector:** Links different parts of a system or program.
90. **Console:** Text-based interface for interacting with a computer.
91. **Containers:** Packaging of software with all its dependencies for easy deployment.
92. **Continuous Delivery:** Automating the process of delivering software changes to a staging or production environment.
93. **Continuous Deployment:** Automating the release and deployment of software changes to production.
94. **Continuous Integration:** The practice of regularly integrating code changes into a shared repository, ensuring early detection of integration issues in the development process.
95. **Cookies:** Small files stored on a user's device by websites.
96. **CPU:** Central processing unit, main brain of a computer.
97. **Cross-Browser Compatibility:** Ensuring that web applications function consistently across different web browsers.
98. **Cross-Platform Development:** Creating software that can run on multiple operating systems or platforms.
99. **CRUD:** Create, Read, Update, Delete basic operations on data.
100. **Cryptography:** The study and application of techniques for secure communication and data protection.
101. **Cyber Threat Intelligence:** Information about potential cyber threats to inform decision-making and response.
102. **Cybersecurity Frameworks:** Comprehensive structures outlining best practices for managing cybersecurity risks.
103. **Cybersecurity measures:** Actions taken to protect against cyberattacks.
104. **Cybersecurity:** Measures taken to protect against cyberattacks.
105. **Daemon:** UNIX application that is permanently on alert on a server in order to perform certain tasks.
106. **Data Analysis:** Examining and interpreting data to extract meaningful insights.
107. **Data backup:** Copying and storing data for recovery in case of loss.
108. **Data center:** Place to place large quantities of servers and communication equipment; It has all the facilities of bandwidth, physical security, 24-hour air conditioning, etc.
109. **Data Encryption:** Converting data into a secure form to prevent unauthorized access.
110. **Data Integrity:** Ensuring the accuracy and consistency of data throughout its lifecycle.
111. **Data Privacy:** Protecting the confidentiality and integrity of personal or sensitive information.
112. **Data shape:** Structure of data, including its organization and types.
113. **Data structure:** The organization and types of data.
114. **Data Structures:** Organized formats for storing and managing data, such as arrays, lists, and trees.
115. **Database administrator (DBA):** Manages and maintains databases.
116. **Database management:** Organized collection of data, usually accessed electronically.
117. **DDL (Data Definition Language):** A subset of SQL that defines and manages database structures.
118. **DDoS:** Distributed Denial of Service. It is an attack with a multitude of systems that have been "hacked" directed at a particular target.
119. **Debugging breakpoint:** Pausing program execution to identify and fix errors.
120. **Debugging process:** Finding and fixing errors in software code.
121. **Debugging Tools:** Software tools used by developers to identify and fix errors or issues in their code.
122. **Debugging:** Process of finding and fixing errors in software code.
123. **Decision:** Choice made between different options.
124. **Decision-making:** The process of choosing between different options.
125. **Declaration:** Specifies a variable, function, or other entity in code.
126. **Dependency Management:** Handling and organizing the external libraries and components that a software project relies on.
127. **Deployment:** The process of releasing and installing a software application for use in a specific environment.
128. **Design Patterns:** Reusable solutions to common problems in software design, providing templates for solving issues in a flexible and efficient way.
129. **Designate:** Assign or label something with a specific name or purpose.
130. **Development Environment:** The set of tools, processes, and configurations used by developers to write, test, and debug software.
131. **Device:** Electronic equipment with specific functions.
132. **DevOps:** Culture and practice of collaboration between development and operations teams.
133. **DevSecOps:** Integrating security practices into the DevOps (Development and Operations) process.
134. **Disaster Recovery Planning:** Developing strategies to recover IT systems after a catastrophic event.
135. **Disaster Recovery:** Planning and procedures for restoring and recovering systems after a catastrophic event.
136. **Display:** Screen or other device for showing visual information.
137. **Django:** High-level web framework written in Python.
138. **DML (Data Manipulation Language):** A subset of SQL that manipulates and queries data within a database.
139. **Document:** A file containing information, usually in electronic format.
140. **Documentation:** A set of documents that describe a system, software, or process.
141. **DOM:** Document Object Model, a tree-like representation of the content of a web page.
142. **Domain:** A unique name that identifies a website on the internet.
143. **Drag:** To move an object on a touchscreen or with a mouse.
144. **Encryption:** The process of converting information into a secure code to protect it from unauthorized access.
145. **Error Handling:** Managing and addressing errors that occur during the execution of a program.
146. **Event:** An action that occurs in a system or program.
147. **Exception Handling:** The process of managing and responding to unexpected events or errors in software.
148. **Extract:** To obtain data from one source and store it in another location.
149. **Firewall Configuration:** Setting up security measures to control incoming and outgoing network traffic.
150. **Flesh out:** To expand or complete details of a plan or idea.
151. **Floppy disk:** An outdated portable magnetic storage device.
152. **Flowchart:** A graphical representation of the steps in a process.
153. **Fluid design:** A web design that automatically adapts to different screen sizes.
154. **Framework:** A set of tools and libraries for developing software.
155. **Frameworks:** Pre-built structures and tools that provide a foundation for developing software applications.
156. **Frontend Development:** Creating the user interface and user experience components of a software application.
157. **Frontend:** The part of a website or application that is visible and interactive for the user.
158. **Full-Stack:** Software development that encompasses both the frontend and backend.
159. **Function:** A block of code that performs a specific task.
160. **Functional Programming:** A programming paradigm where computation is treated as the evaluation of mathematical functions, avoiding changing state and mutable data.
161. **Git:** A distributed version control system for tracking changes in code.
162. **Github:** A hosting platform for Git projects.
163. **Gitlab:** Similar to Github, a platform for hosting Git projects.
164. **Go:** A modern programming language developed by Google.
165. **Hacker:** A person with computer skills who uses them for unauthorized purposes.
166. **Hacking:** The attempt to gain unauthorized access to a computer system or network.
167. **Hard Drive:** A magnetic storage device that stores data on a spinning disk.
168. **Hardware:** The physical components of a computer system.
169. **HDMI:** High-Definition Multimedia Interface. It is a digital interface to transmit audio and video data on the same cable. Most HDTVs, DVDs, Blu-rays, cable boxes, and game consoles have HDMI support.
170. **Headphones:** An audio device that is worn over the ears.
171. **Highlight:** To emphasize or make something more visible.
172. **Hosting:** A service that provides space on a server to store web files.
173. **HTML:** HyperText Markup Language defines the structure and content of web pages.
174. **HTTP:** HyperText Transfer Protocol, the protocol for transferring data on the web.
175. **Hyperlink:** A link in an electronic document that takes you to another location.
176. **IaaS:** Infrastructure as a Service, a cloud computing model that provides infrastructure as a service.
177. **IDE:** Integrated Development Environment, an environment for creating software.
178. **Incident Management:** The process of identifying, responding to, and resolving incidents or issues in a system.
179. **Incident Response:** Coordinated actions taken in response to a security incident or breach.
180. **Infrastructure:** The physical and virtual components that form the basis of a computer system.
181. **Input device:** A device used to enter data into a computer system.
182. **Integration test:** A test that verifies how different components of a system interact.
183. **Internal storage:** The storage space on an electronic device.
184. **Intrusion Detection:** Detecting and responding to unauthorized access or security breaches.
185. **Involve:** To include or make someone participate in something.
186. **Java:** A general-purpose object-oriented programming language.
187. **JavaScript:** A programming language that allows user interaction on web pages.
188. **JSON:** A lightweight data-interchange format used for easy data exchange between applications.
189. **Keyboard:** An input device that allows you to type letters, numbers, and symbols.
190. **Library:** A collection of reusable code for performing specific tasks.
191. **Load Balancing:** Distributing network traffic evenly across multiple servers to ensure optimal performance.
192. **Load tests:** Tests that verify the performance of a system under load.
193. **Log:** A file that records events and activities in a system.
194. **Logging:** Recording events, activities, or errors in a system for analysis and troubleshooting.
195. **Loops:** Sections of code that execute repeatedly until a condition is met.
196. **Manual input:** The process of entering data into a computer system manually.
197. **Manual operation:** The process of performing a task in a computer system manually.
198. **Merge:** To integrate two or more things into one.
199. **Microphone:** A device that converts sound into an electrical signal.
200. **Microservices:** An architectural style where an application is composed of small, independent services.
201. **Middleware:** Software that acts as an intermediary between different software components.
202. **Mixin:** A reusable piece of code that can be included in other classes.
203. **Mobile Applications:** Software designed to operate on mobile devices such as smartphones and tablets.
204. **Mobile development:** Creating apps for smartphones, tablets, and wearables.
205. **Mobile OS:** Operating system designed for mobile devices, e.g., Android.
206. **Modem:** Modulates/demodulates signals for communication.
207. **Monitor:** Display to visualize computer output.
208. **Monitoring:** Continuously observing and measuring the performance and behavior of a system.
209. **Monolith:** Large application with all functionalities.
210. **Mouse:** Device to control the cursor on the screen.
211. **MVP:** Minimum viable product to validate an idea.
212. **Network bandwidth:** The amount of data that can be transmitted in a given time.
213. **Network Security:** Protecting a computer network infrastructure against unauthorized access and attacks.
214. **NoSQL database:** Stores data in a flexible format.
215. **Objective-C:** Programming language for iOS and macOS.
216. **Object-Oriented Programming (OOP):** A programming paradigm that uses objects, which encapsulate data and behavior, for software development.
217. **OEM:** Original Equipment Manufacturer, it refers to companies that manufacture and package products for others, and also to companies that acquire a product or component and reuse it or incorporate it into another new product with their own brand.
218. **Open source:** Accessible and modifiable source code.
219. **Operators:** Companies that provide telecommunications services.
220. **Organization chart:** A diagram that shows the hierarchical structure of an organization.
221. **Outcome:** Consequence of an action or process.
222. **Output:** Information or data produced by a system.
223. **PaaS:** Platform to develop, deploy, and manage applications.
224. **Penetration Testing:** Simulating cyber-attacks to identify vulnerabilities and weaknesses in a system.
225. **Performance benchmark:** Comparing the performance of different systems or software.
226. **Performance Monitoring:** Tracking and analyzing the performance of a system during its operation.
227. **Performance Optimization:** Improving the efficiency and speed of a software application.
228. **Performance test:** Performance measurement under real load.
229. **Perl:** Versatile language for web development and system administration.
230. **PHP:** Language for developing dynamic web applications.
231. **Plugin:** Module that adds functionalities to a program.
232. **Plugs:** Connectors to connect devices.
233. **Port:** Logical connection for communication between devices.
234. **Predefined process:** A set of steps established beforehand to guide a particular action or task.
235. **Preparation:** The act of getting ready for something, often involving gathering necessary materials and information.
236. **Printed document:** A physical copy of a document with text and/or images, created using a printer.
237. **Printer:** A device that produces printed documents from digital files.
238. **Privilege Management:** Controlling and assigning specific privileges or permissions to users.
239. **Process:** A series of actions or steps taken to achieve a specific outcome.
240. **Program:** A set of instructions that tells a computer how to perform a specific task.
241. **Programming language:** A system of communication used to write programs for computers.
242. **Programming paradigm:** A fundamental approach to software development, such as object-oriented or functional programming.
243. **Programming:** The process of creating programs using programming languages.
244. **Progressive Web App (PWA):** A website that offers app-like features and functionality, accessible even offline.
245. **Project Manager:** A person responsible for planning, coordinating, and executing projects to achieve specific goals.
246. **Pseudocode:** An informal, human-readable description of an algorithm, used for planning and understanding before writing actual code.
247. **Python:** A general-purpose, interpreted programming language known for its readability and versatility.
248. **Quality Assurance:** The systematic process of ensuring that a product or system meets specified requirements and quality standards.
249. **R:** A programming language and software environment for statistical computing and graphics.
250. **Recursion:** A programming technique where a function calls itself within its own body.
251. **Refactoring Techniques:** Methods for restructuring code to enhance its design and maintainability.
252. **Refactoring:** The process of improving the structure and design of existing code without changing its functionality.
253. **Regression Testing:** The process of retesting previously working code after changes are made to ensure it still functions correctly.
254. **Regulations:** Official rules or directives governing the development and use of software.
255. **Regulatory Compliance:** Ensuring that software development and operations comply with relevant laws and regulations.
256. **Relational database:** A type of database that organizes data in tables with rows and columns, where rows represent records and columns represent attributes of those records.
257. **Repeat:** The act of performing an action multiple times, either a specific number of times or until a certain condition is met.
258. **Repository:** A storage location for version-controlled code and related assets.
259. **Responsive Design:** Designing applications or websites that adapt to different screen sizes and devices.
260. **Responsive design:** Designing websites to adapt and display properly on various screen sizes.
261. **Responsive design:** Websites that adapt their layout and content to different screen sizes and devices.
262. **REST:** An architectural style for web APIs based on HTTP requests and responses.
263. **Risk Assessment:** Evaluating potential risks and vulnerabilities to determine their impact.
264. **Risk Management:** Developing strategies to mitigate and control risks within an organization.
265. **Ruby:** A general-purpose, interpreted programming language known for its elegance and readability.
266. **SaaS (Software as a Service):** A cloud-based software delivery model where users access applications over the internet instead of installing them locally.
267. **SASS:** A preprocessor for writing CSS stylesheets, making them more maintainable and readable.
268. **Scala:** A general-purpose, object-functional programming language known for its scalability and expressiveness.
269. **Scalability:** The ability of a system or application to handle increased load or demand.
270. **Scanner:** A device that captures images or data electronically, often used for documents or barcodes.
271. **Scripting:** Writing code to automate tasks or processes in a script or program.
272. **Scroll:** The act of moving vertically or horizontally within a window or on a screen.
273. **Scrum Master:** A person who facilitates and guides a Scrum team, ensuring they follow the process effectively.
274. **Scrum:** An agile project management framework that uses short sprints, daily stand-ups, and retrospectives.
275. **SD card (Secure Digital card):** A removable flash memory card used for storing data in devices like cameras and smartphones.
276. **Secure Coding Practices:** Adhering to coding practices that prioritize security and reduce vulnerabilities.
277. **Secure Coding:** Writing code with security considerations to prevent vulnerabilities.
278. **Secure Communication:** Ensuring that data transmission between systems is confidential and tamper-proof.
279. **Secure Configuration Management:** Ensuring that systems and software are configured securely to minimize risks.
280. **Secure Deployment:** Implementing security measures when deploying software to production.
281. **Secure Design Principles:** Fundamental principles for designing secure software applications.
282. **Secure Development Frameworks:** Established structures and tools that prioritize security in software development.
283. **Secure DevOps:** Integrating security practices into the DevOps process to create a secure software development pipeline.
284. **Secure Software Development Lifecycle (SDLC):** Integrating security into every phase of the software development process.
285. **Security Measures:** Strategies and implementations to safeguard software and data from unauthorized access and threats.
286. **Security Operations Center (SOC):** A centralized unit responsible for monitoring and managing security incidents.
287. **Security Policies:** Guidelines and rules defining the organization's approach to information security.
288. **Security Standards:** Established criteria and benchmarks for maintaining security in software and systems.
289. **Security Testing:** Assessing the security aspects of a system to identify vulnerabilities and weaknesses.
290. **Sequencing:** The act of arranging things in a specific order or sequence.
291. **Server:** A computer that provides services to other computers over a network, such as websites, emails, and files.
292. **Server-Side Scripting:** Executing scripts on the server to generate dynamic web content.
293. **Setting up:** The process of configuring or preparing something for use.
294. **Shell:** A command-line interface for interacting with a computer operating system.
295. **Smoke test:** A basic test to quickly verify if a system or application is functioning at a minimum level.
296. **SOAP (Simple Object Access Protocol):** A protocol for exchanging information between applications using XML and web services.
297. **Software bug:** An error in software code causing issues.
298. **Software containers:** Packaging software with all dependencies for easy deployment.
299. **Software Engineering:** The systematic application of engineering principles to the design, development, testing, and maintenance of software.
300. **Software testing:** The process of evaluating and validating software to ensure it meets requirements and operates as expected.
301. **Software:** A collection of instructions that tell a computer how to perform specific tasks.
302. **SOLID principles:** A set of object-oriented design principles for creating maintainable and flexible software.
303. **Sort:** The process of arranging items in a specific order, such as numerically or alphabetically.
304. **Source code:** Human-readable instructions written by a programmer.
305. **Source:** The origin of something, such as data, code, or information.
306. **Speakers:** Devices that produce sound, often used for listening to music, movies, or other audio content.
307. **SQL (Structured Query Language):** A language for interacting with relational databases.
308. **SSL (Secure Sockets Layer):** A security protocol that encrypts data transmitted over the internet, protecting it from eavesdropping and tampering.
309. **Steps:** A sequence of actions taken to achieve a goal.
310. **Storage device:** A device used to store data, such as a hard drive, SSD, or USB drive.
311. **Storage:** The act of saving or keeping data for future use.
312. **Stored data:** Data that has been saved for future use.
313. **Stress tests:** Tests that are designed to push a system or application to its limits to identify potential failures.
314. **Subversion:** A version control system that allows users to track changes to files and folders.
315. **SysAdmin:** A system administrator, responsible for the maintenance and operation of computer systems.
316. **System connector:** A component linking different parts of a system or program.
317. **Task:** A unit of work that needs to be completed.
318. **Technical Debt:** Accumulated compromises in code quality that may require future attention.
319. **Terminator:** A program that terminates other programs.
320. **Test Automation:** Running tests automatically using software.
321. **Test Cases:** Specific conditions or scenarios designed to verify the functionality of a software application.
322. **Test Coverage:** The extent to which a system has been tested, measured by the percentage of code or functionality covered.
323. **Test Execution:** Running the tests to evaluate the behavior and performance of a software application.
324. **Test Metrics:** Quantifiable measures used to assess the effectiveness and progress of testing activities.
325. **Test Plan:** A comprehensive document outlining the testing approach, scope, resources, and schedule.
326. **Test Results:** The outcomes and findings from the execution of software tests.
327. **Test Strategy:** The overall plan for testing, including the scope, resources, and testing methods.
328. **Test Suites:** A collection of test cases that are executed together to validate different aspects of a system.
329. **Tester:** A person who tests software to ensure it meets requirements and operates as expected.
330. **Text editor:** A program used to create and edit text files.
331. **Text-based interface:** Interaction with a computer through text.
332. **The call:** The act of making a phone call or using a communication service.
333. **Threat Intelligence:** Gathering and analyzing information about potential cybersecurity threats.
334. **Threat Modeling:** Identifying potential security threats and planning countermeasures during the design phase.
335. **UI (User Interface):** The part of a software application that users interact with.
336. **Unauthorized access:** Illegitimate entry into a system or network.
337. **Unit tests:** Tests that are designed to test individual units of code, such as functions or methods.
338. **Until:** A conjunction used to indicate a point in time before which something will happen.
339. **Usability:** The ease with which a user can use a product or service.
340. **USB Flash Drive:** A portable storage device that uses flash memory.
341. **User Authentication:** Verifying the identity of users accessing a system.
342. **User Authorization:** Granting or restricting access rights to users based on their roles or permissions.
343. **UX (User Experience):** The overall experience of using a product or service.
344. **Variables:** Named locations in memory that store data.
345. **Version control commit:** Submitting changes to a version control system like Git.
346. **Version Control:** Managing changes in software code over time.
347. **Visual Basic .Net:** A programming language that allows users to create Windows applications.
348. **Vulnerability Management:** Identifying, prioritizing, and addressing vulnerabilities in a systematic way.
349. **Vulnerability Scanning:** Identifying and assessing potential vulnerabilities in a system.
350. **Web Applications:** Applications accessed and operated through web browsers.
351. **Web browser:** Software used to access and navigate websites.
352. **Webhook:** A way for an app to provide other applications with real-time information.
353. **While loop:** A programming loop that repeats a block of code until a condition is met.