Batch: A3 Roll No: 16010121051

Exp No: 2

Title: Identification of data and hardware software Requirement (Analysis phase).

Objective: To understand & identify hardware and software, usage of Tools needed to meet the desired specification of the project.

Expected Outcome of Experiment:

Course Outcome	**After successful completion of the course students should be able to**
CO 2	Identify the various hardware and software, usage of Tools needed to meet the desired specification.

Books/ Journals/ Websites referred:
1.judge0
2.www.projectdoc.in

Describe the need of this stage in project:

Project Requirements specify what features a product should include and how those features should work. They help to define the test criteria, which is vital for verification and validation.

At the beginning of the software development process, collating and analysing the requirements is one of the first tasks to be done. This usually takes place between the client, software engineer, and tester. All three stakeholders, must come to an agreement as to what the final product should do and how. In addition, to understanding how the product will be verified and validated according to the requirements set out.

Types of requirements:
There are several types of requirements that should be considered when it comes to designing or modifying software. They each depend on what the product is. These are: business requirements, software requirements, and hardware requirements.

Business requirements:
These outline the business case and reason for making the product in the first place. If a company doesn’t have a business case that stacks up then why is the company investing money and time into taking the project on further.

Also known as stakeholder requirements, they describe the characteristics of the proposed product from the point of view of the end user. They don’t define what the system should do or how it should be done, but instead they define the ‘why’ requirements. For example, why does this customer need your product? It sets the tone and vision for the product.

Software requirements:
These can be categorised into **Non-Functional and Functional requirements**. The difference is as follows:

Functional requirements are the ‘what’ requirements. They outline what is the system designed to do. For example, say you were a medical company looking to develop a COVID testing device. The types of functional requirements, which might exist could be that the product must be able to identify a COVID test or the product must display a positive or negative test result to the user.

Non-Functional requirements are the ‘how’ requirements. They outline how the system will do what it is designed to do. For example, the product will run a COVID test within 20 minutes and the device will send results via Bluetooth to the end user’s mobile phone. Both set of requirements are important as they provide input into what will be required for development and testing.

Hardware requirements:
These are the requirements that a required to develop and create a hardware device. Sometimes a product is completely software-based and so they may not be needed. But if your company is making a physical product then there are many aspects to consider before designing and making the device.

There are many aspects such as what processor speed is required, how much memory is needed, what shape will the product take that all need to be reviewed. Hardware and software requirements need to be considered together to determine the software compatibility for the system being designed.

Answer the following based on your case study:

1. **Identify data:**

User data: We would need to collect and store user data, such as names, email addresses, and login credentials, to create and manage user accounts.

Code and project data: We would need to store and manage user code and project data, including files, folders, and version history, in a structured and secure manner.

Performance and usage data: We would need to track performance metrics such as page load times, user traffic, and usage patterns to optimize the platform's performance and user experience.

Learning and assessment data: We would need to collect and manage data related to user engagement with learning resources and assessment features, such as quizzes and code analysis reports.

Collaboration and feedback data: We would need to store and manage data related to user collaboration and feedback features, such as code reviews and comments.

Security and privacy data:We would need to collect and manage data related to user privacy and security, such as authentication logs, access controls, and encryption keys.

2. **Identify software and hardware needed.**

Software:
Code editor: We would need a code editor that supports syntax highlighting, code completion, and code formatting for multiple programming languages.
Web development framework: we would need a web development framework to develop the platform's front-end and back-end components.
Database management system:We would need a database management system to store and manage user data, project data, and other platform data.
Version control system: We would need a version control system to track changes and collaborate on coding projects.
Collaboration tools: We would need collaboration tools such as chat, video conferencing, and project management tools to facilitate teamwork among users.

Hardware:

Web server: We would need a web server to host the online coding platform and serve user requests.
Storage device: We would need a storage device to store user data, project data, and other platform data.
Network infrastructure: We would need a reliable network infrastructure to ensure that users can access the platform quickly and reliably.
Backup and disaster recovery infrastructure: We would need backup and disaster recovery infrastructure to ensure that user data is secure and protected in case of a disaster or hardware failure.

3. Prepare E-R schema or equivalent description of data.

E-R diagram that describes the core entities and relationships for the project:

Entities:

User: Represents a registered user of the platform. Contains attributes such as user_id, name, email, password, and profile information.
Project: Represents a coding project created by a user. Contains attributes such as project_id, name, description, and creation time.
File: Represents a file within a project, such as a source code file or a README file. contains attributes such as file_id, name, content, and creation time.
Version: Represents a version of a file within a project, such as a specific commit or snapshot. Contains attributes such as version_id, content, and creation time.
Language: Represents a programming language supported by the platform. Contains attributes such as language_id, name, and syntax highlighting rules.

Relationships:

User can create multiple projects. One-to-many relationship between User and Project.
Project can contain multiple files. One-to-many relationship between Project and File.
File can have multiple versions. One-to-many relationship between File and Version.
Each project can use a specific programming language. One-to-one relationship between Project and Language.

4. **Identify usage of Tools, library functions, APIs/packages, applicable to solve the problem**

Code editor: You could use existing code editors such as CodeMirror or Ace Editor, which are open-source and support syntax highlighting, code completion, and code formatting for multiple programming languages.
Web development framework: You could use popular web development frameworks such as React,js to develop the platform's front-end and back-end components.
Database management system: You could use relational database management systems such as MySQL to store and manage user data, project data, and other platform data.
Version control system: You could use Git, which is a popular version control system, to track changes and collaborate on coding projects.
Collaboration tools: Github is one of the famous
Cloud services: You could use cloud services such as Amazon Web Services (AWS) or Microsoft Azure to host the platform and provide scalable infrastructure.
Terminal screen: You could use existing libraries such as xterm.js to provide a terminal screen that displays errors and debugging information.
File management: You could use existing libraries such as Dropzone.js or React Dropzone to provide a file management system that allows users to save and access their code and projects.
Learning resources: You could use e-learning platforms such as Coursera, Udemy or youtube to provide structured learning courses that guide users from beginner to advanced programming concepts, presented in an engaging format that includes interactive lessons and quizzes.

<u>Post Lab Activities (with reference to your tool):</u>

1. What tools are used in analysis phase of the Software development life cycle?

Requirements gathering tools: These tools help capture and manage requirements from stakeholders, including business analysts, end-users, and other project team members. Some examples include Jira, Trello, and Asana.

Use case modeling tools: These tools help model user requirements and define how users interact with the system. Some examples include Visual Paradigm, Enterprise Architect, and Lucidchart.