**Module 1 :- Practical Assignment**

**Run Program in Online Editor**

1. LAB EXERCISE: Write a simple "Hello World" program in two different programming languages

**C – Language**

Ans. #include <stdio.h>

int main() {

printf("Hello World");

return 0;

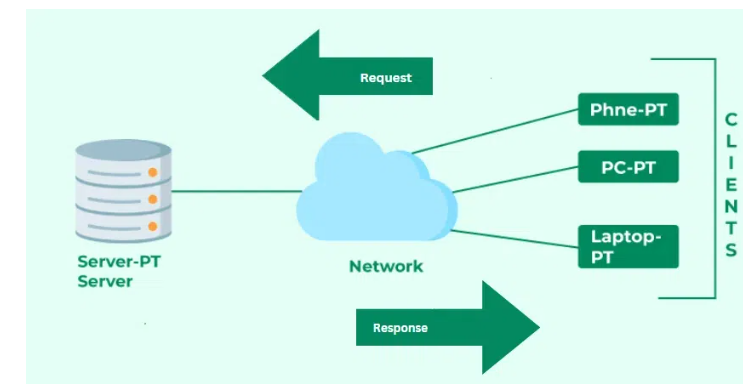
}

**Python- Language**

print("Hello World")

1. Research and create a diagram of how data is transmitted from a client to a serverover the internet.

**Ans.**  it is the Clientrequesting something and the Server serving it as long as it is in the database.



A diagram of a server

AI-generated content may be incorrect.

1. Design a simple HTTP client-server communication in any language.

**Ans.**

#include <netinet/in.h> //structure for storing address information

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h> //for socket APIs

#include <sys/types.h>

int main(int argc, char const\* argv[])

{

int sockD = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in servAddr;

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port

= htons(9001); // use some unused port number

servAddr.sin\_addr.s\_addr = INADDR\_ANY;

int connectStatus

= connect(sockD, (struct sockaddr\*)&servAddr,

sizeof(servAddr));

if (connectStatus == -1) {

printf("Error...\n");

}

else {

char strData[255];

recv(sockD, strData, sizeof(strData), 0);

printf("Message: %s\n", strData);

}

return 0;

}

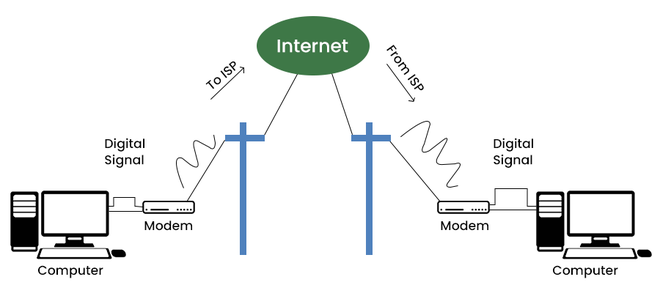
1. Research different types of internet connections (e.g., broadband, fiber,satellite)and list their pros and cons

**Ans.**

**1. Dial-Up Connection**

A [dial-up connection](https://www.geeksforgeeks.org/difference-between-dialup-and-broadband-connection) is established between your computer and the ISP server using a modem. A dial-up Connection is a cheap and traditional connection that is not preferred these days as this type of connection is very slow.

To access the internet connection in the dial-up connection we need to dial a phone number on the computer and that’s why it requires a telephone connection. It requires a modem to set up a dial-up connection, which works as interference between your computer and the telephone line. In this connection, we can use either an internet connection or a telephone at a time.

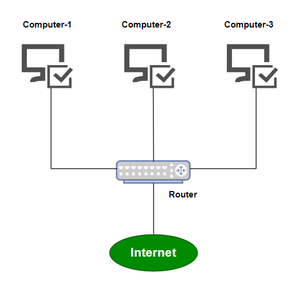


*Dial Up Connection*

**2. Broadband Connection**

Broadband refers to high-speed internet access that is faster than traditional dial-up access. It is provided through either cable or telephone composition. It does not require any telephone connection that’s why here we can use telephone and internet connection simultaneously. In this connection, more than one person can access the internet connection simultaneously.

It is a wide bandwidth data transmission that transports several signals and traffic types. In this connection, the medium used is [coaxial cable](https://www.geeksforgeeks.org/what-is-coaxial-cable/), [optical fiber cable](https://www.geeksforgeeks.org/optical-fibre-cable/), radio, or [twisted pair cable](https://www.geeksforgeeks.org/twisted-pair-cable/).

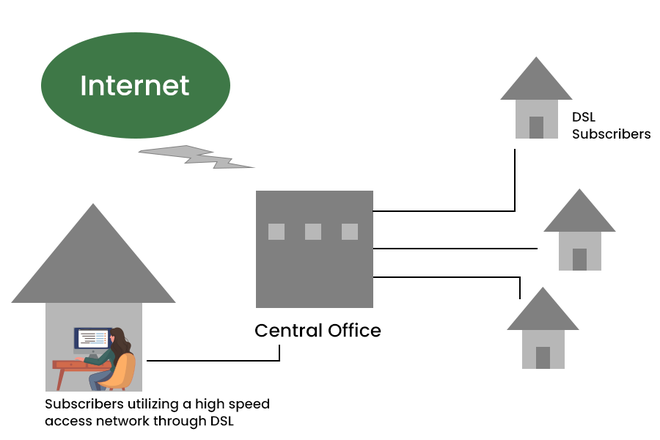


*Broadband-Connection*

**3. DSL (Digital Subscriber Line)**

DSL stands for [Digital Subscriber Line](https://www.geeksforgeeks.org/digital-subscriber-line-dsl). It provides an internet connection through the telephone line(network). DSL is a form of broadband communication that is always on, there is no need to dial a phone number to connect. DSL connection uses a router to transport data and the speed of this connection range between 128k to 8Mbps depending on the service offered. A DSL connection can translate data at 5 million bytes per second, or 5mbps.

DSL service can be delivered simultaneously with wired telephone service on the same telephone line due to high-frequency bands for data.

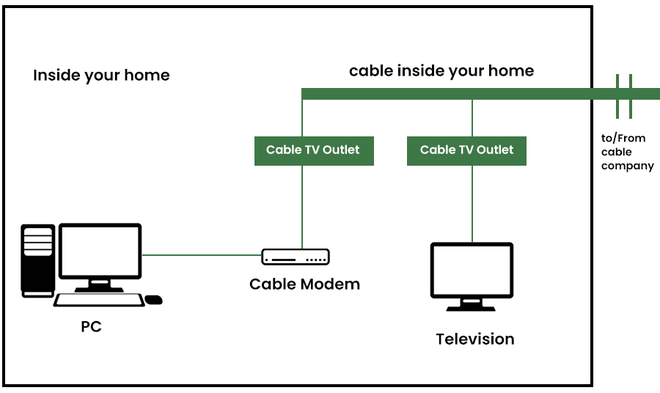


*DSL*

**4. Cable**

It is a form of broadband access cable [modem](https://www.geeksforgeeks.org/what-is-modem/) that can provide extremely fast access to the internet. The speed of this connection varies which can be different for uploading data transmission or downloading.

It uses a cable modem to provide an internet connection and operates over cable TV lines. The speed of cable connection ranges from 512k to 20Mbps.h

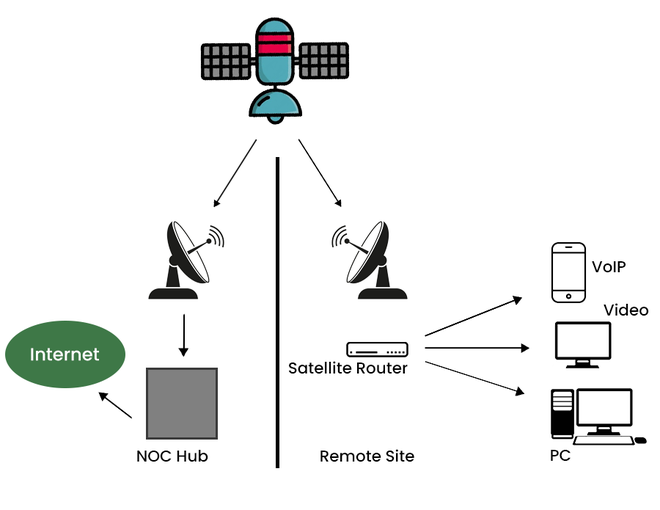


*Cable*

**5. Satellite Connection**

This type of connection is provided mainly in rural areas where a broadband connection is not yet offered. It accesses the internet via a satellite that is in Earth’s orbit.

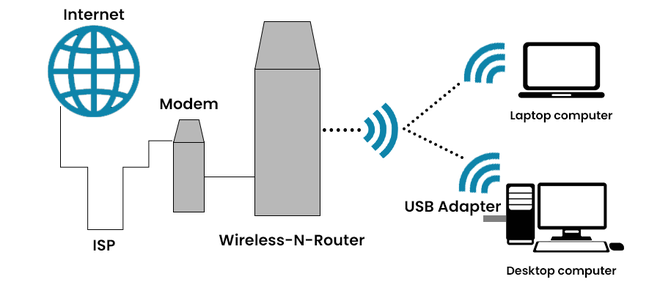
The signal travels from a long distance that is from earth to satellite and back again which provides a delayed connection. Satellite connection speeds range from 512k to 2.0Mbps.



*Satellite Connection*

**6. Wireless Connection**

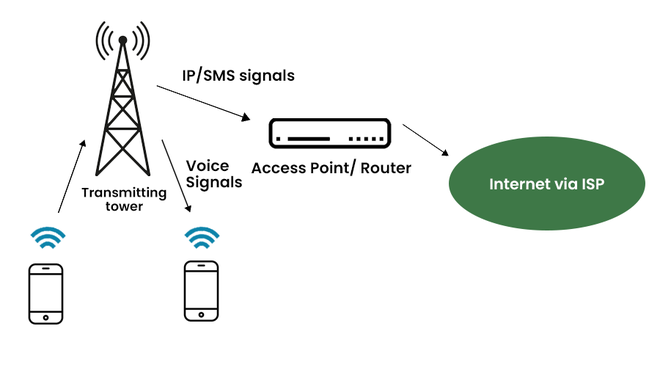
As the name suggests[wireless connection](https://www.geeksforgeeks.org/wired-and-wireless-networking/)does not use telephone lines or cables to connect to the internet. The wireless connection uses a radio frequency band to connect to the internet. It is also an always-on connection and this connection can be accessed from anywhere and speed may vary for different locations. It ranges from 5Mbps to 20Mbps.



*Wireless Connection*

**7. Cellular**

[Cellular technology](https://www.geeksforgeeks.org/mobile-technologies-definition-types-uses-advantages) provides wireless Internet access through cell phones. Speed may vary depending on the service provider. The most common are 3G and 4G which means from 3rd generation and 4th generation respectively. The speed of the 3G cellular network is around 2.0Mbps and the 4G cellular network is around 21Mbps the goal of the 4G network is to achieve peak mobile speeds of 100Mbps but the current speed of the 4G network is about 21Mbps.

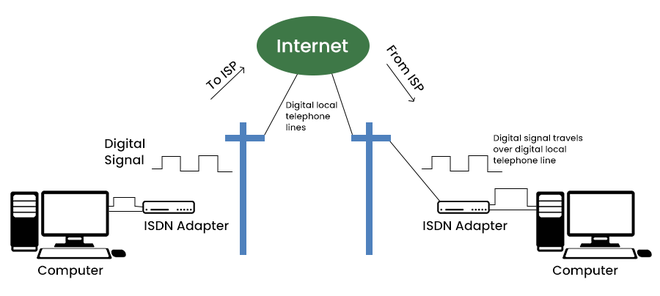


*Cellular*

**8. ISDN (Integrated Service Digital Network)**

ISDN stands for [Integrated Service Digital Network](https://www.geeksforgeeks.org/integrated-services-digital-network-isdn) and it is a [circuit-switched](https://www.geeksforgeeks.org/circuit-switching-in-computer-network/) telephone network system, but it also provides access to [packet-switched](https://www.geeksforgeeks.org/packet-switched-network-psn-in-networking/) networks that transmits both voice and data over a digital line. It provides a packet-switched connection for data in increments of 64 kilobit/s.

ISDN connection provides better speeds and higher quality than traditional connections. It provided a maximum of 128kbit/s bandwidth in both upstream and downstream directions.



1. Simulate HTTP and FTP requests using command line tools (e.g., curl)

**Ans.** Telnet www.example.com 80

1. Identify and explain three common application security vulnerabilities. Suggestpossible solutions.

**Ans.** 1. Injection Vulnerabilities

What it is:

Attackers exploit flaws in how applications handle user input by injecting malicious code into queries, commands, or other parts of the application's logic.

Example:

SQL injection occurs when an attacker inserts malicious SQL code into a query, tricking the application into executing unintended commands.

2. Broken Authentication

What it is:

Weak or improperly implemented authentication mechanisms allow attackers to bypass login systems and gain unauthorized access to applications.

Example:

A weak password policy or lack of multi-factor authentication (MFA) can be exploited by attackers.

3. Cross-Site Scripting (XSS)

What it is:

Attackers inject malicious scripts into web pages viewed by other users, potentially leading to data breaches or other malicious activities.

Example:

An attacker might inject a script into a comment section that, when accessed by another user, steals their cookies or redirects them to a malicious site.

1. Identify and classify 5 applications you use daily as either system software orapplication software.
2. Ans. 1. Operating System (OS):

(e.g., Windows, macOS, Android) - System Software. The OS manages hardware and software resources, allowing other applications to run.

1. 2. Web Browser:

(e.g., Chrome, Firefox) - Application Software. Used for browsing the web and accessing online information.

1. 3. Word Processor:

(e.g., Microsoft Word, Google Docs) - Application Software. Used for creating and editing text documents.

1. 4. Email Client:

(e.g., Outlook, Gmail) - Application Software. Used for sending and receiving emails.

1. 5. File Manager:

(e.g., File Explorer, Finder) - System Software. Allows users to organize, access, and manage files and folders on their computer.

1. Design a basic three-tier software architecture diagram for a web application.

**Ans.** The Three-Tier Client-Server Architecture divides systems into presentation, application, and data layers, increasing scalability, maintainability, and efficiency. By separating the concerns, this model optimizes resource management and allows for independent scaling and updates, making it a popular choice for complex distributed systems.

A computer and a calculator

AI-generated content may be incorrect.

1. Create a case study on the functionality of the presentation, business logic, and dataaccess layers of a given software system.

**Ans.** The Business-Logic Layer (BLL) is a component of a software architecture that is responsible for implementing the business logic of an application. It sits between the presentation layer (e.g., the user interface) and the data access layer (e.g., the database), and is responsible for processing and manipulating data before it is presented to the user or stored in the database.

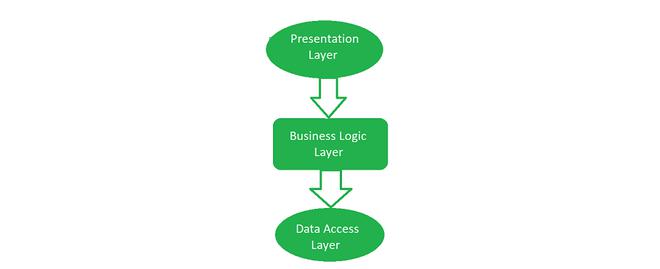
The BLL is responsible for performing tasks such as:  
-Validating input data to ensure that it meets the necessary business rules and constraints.  
-Performing calculations and transformations on data, as required by the business logic.  
-Enforcing business rules and policies, such as access control and security.  
-Communicating with the data access layer to retrieve and store data.  
-Handling errors and exceptions.

The BLL is designed to be reusable and independent of the user interface and data storage implementation. This allows the application to be easily modified or extended without affecting the underlying business logic.

The BLL is also responsible for managing the workflows and the use cases of the application, by handling the communication between the different layers, and by implementing the rules and constraints of the business.

In summary, The Business-Logic Layer (BLL) is a component of a software architecture that is responsible for implementing the business logic of an application. It sits between the presentation layer and the data access layer, and is responsible for processing and manipulating data before it is presented to the user or stored in the database. It also manages the workflows and the use cases of the application, and it is designed to be reusable and independent of the user interface and data storage implementation.

* Presentation Layer: The layer at which users interact with the application and the final data will be visible to the users at this interface. It acts as an interface between the user and the application.
* Business Logic Layer: It acts as an intermediate between the Presentation and the Data Access Layer.
* Data Access Layer: The layer at which the data is managed.

****

1. Explore different types of software environments (development, testing,production).Set up a basic environment in a virtual machine.

**Ans.**

Setting up a Basic Environment in a Virtual Machine:

1. 1. Install Virtualization Software:

Tools like VirtualBox or VMware are used to create and manage virtual machines.

1. 2. Create a Virtual Machine:

Choose a virtual machine operating system (e.g., Linux, Windows) and allocate resources (memory, storage) as needed.

1. 3. Install Necessary Software:

Install the required software packages, including development tools, testing frameworks, databases, etc., based on the specific environment's requirements.

1. 4. Configure the Environment:

Set up network connections, firewall rules, and any other configurations needed for the environment to function correctly.

1. 5. Test the Environment:

Verify that the virtual machine is running, the software is installed, and the environment is configured as expected.

1. Write and upload your first source code file to Github.

**Ans.**

* Apply version control when you make edits to the files, so your project's history is protected and manageable.
* Back up your work, because your files are now stored in the cloud.
* Pin the repository to your personal profile, so that others can see your work.
* Share and discuss your work with others, either publicly or privately.

If you're already familiar with Git, and you're looking for information on how to upload a locally-stored Git repository to GitHub, see [Adding locally hosted code to GitHub](https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github#adding-a-local-repository-to-github-using-git).

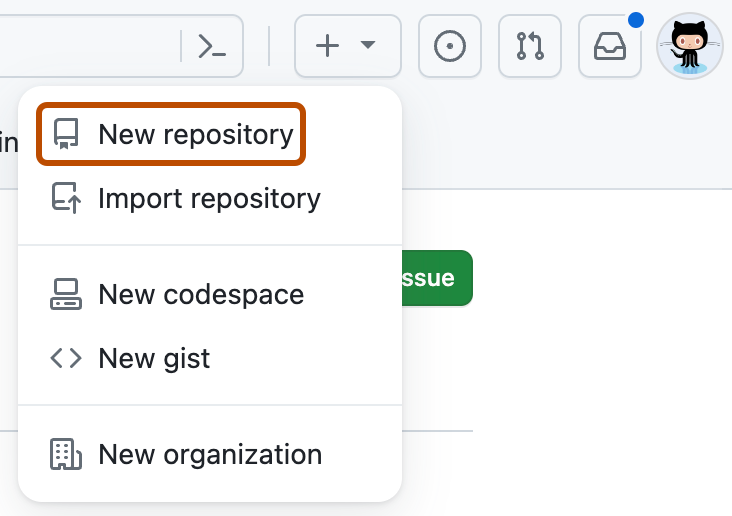
[Prerequisites](https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github#prerequisites)

* You must have a GitHub account. For more information, see [Creating an account on GitHub](https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github).
* You should have a group of files you'd like to upload.

[Step 1: Create a new repository for your project](https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github#step-1-create-a-new-repository-for-your-project)

It's a good idea to create a new repository for each individual project you're working on. If you're writing a software project, grouping all the related files in a new repository makes it easier to maintain and manage the codebase over time.

1. In the upper-right corner of any page, select , then click New repository.



1. In the "Repository name" box, type a name for your project. For example, type "my-first-project."
2. In the "Description" box, type a short description. For example, type "This is my first project on GitHub."
3. Select whether your repository will be Public or Private. Select "Public" if you want others to be able to see your project.
4. Select Add a README file. You will edit this file in a later step.
5. Click Create repository.

[Step 2: Upload files to your project's repository](https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github#step-2-upload-files-to-your-projects-repository)

So far, you should only see one file listed in the repository, the README.md file you created when you initialized the repository. Now, we'll upload some of your own files.

1. To the right of the page, select the Add file dropdown menu.
2. From the dropdown menu, click Upload files.
3. On your computer, open the folder containing your work, then drag and drop all files and folders into the browser.
4. At the bottom of the page, under "Commit changes", select "Commit directly to the main branch, then click Commit changes.

[Step 3: Edit the README file for your project's repository](https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github#step-3-edit-the-readme-file-for-your-projects-repository)

Your repository's README file is typically the first item someone will see when visiting your repository. It usually contains information on what your project is about and why your project is useful.

As we learned in the [Hello World](https://docs.github.com/en/get-started/start-your-journey/hello-world) tutorial, the README file (README.md) is written in Markdown syntax. Markdown is an easy-to-read, easy-to-write language for formatting plain text.

In this step, we'll edit your project's README.md using Markdown so that it includes some basic information about your project.

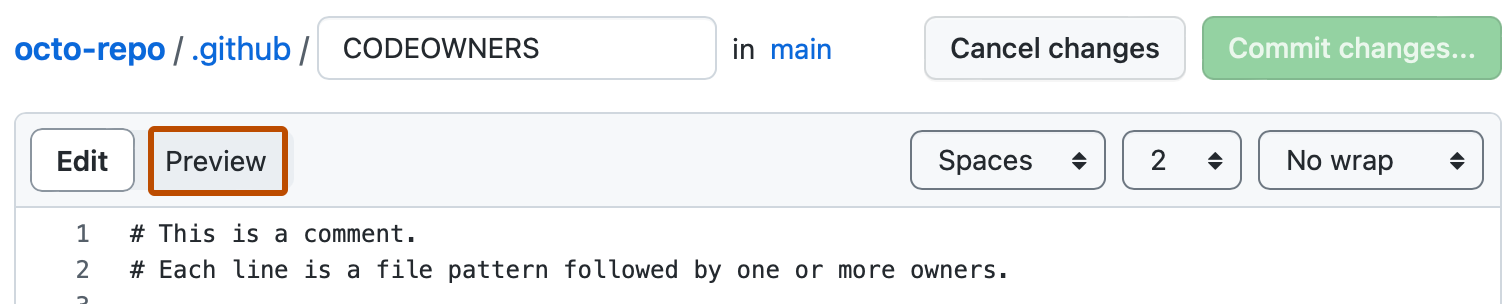
1. From the list of files, click README.md to view the file.
2. In the upper right corner of the file view, click  to open the file editor.
   * You will see that some information about your project has been pre-filled for you. For example, you should see the repository name and repository description you completed in Step 1 displayed on line 1 and line 2.
3. Delete the existing text apart from #, then type a proper title for your project.
   * Example: # About my first project on GitHub.
4. Next, add some information about your project, such as a description of the project's purpose or its main features.

Note

If you're not sure what to write, take a look at other repositories on GitHub to see how other people describe their projects.

To apply more sophisticated formatting, such as adding images, links, and footnotes, see [Basic writing and formatting syntax](https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax).

1. Above the new content, click Preview.

****

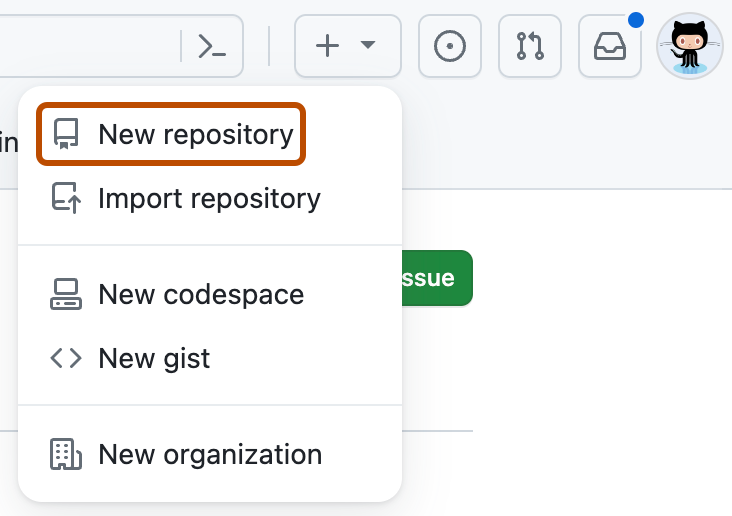
1. Take a look at how the file will render once we save our changes, then toggle back to "Edit".
2. Continue to edit and preview the text until you're happy with the content of your README.
3. In the top right, click Commit changes.
4. In the dialog box that opens, a commit message has been pre-filled for you ("Update README.md") and, by default, the option to "Commit directly to the main branch" has been selected. Leave these options as they are and go ahead and click Commit changes
5. Create a Github repository and document how to commit and

push code changes

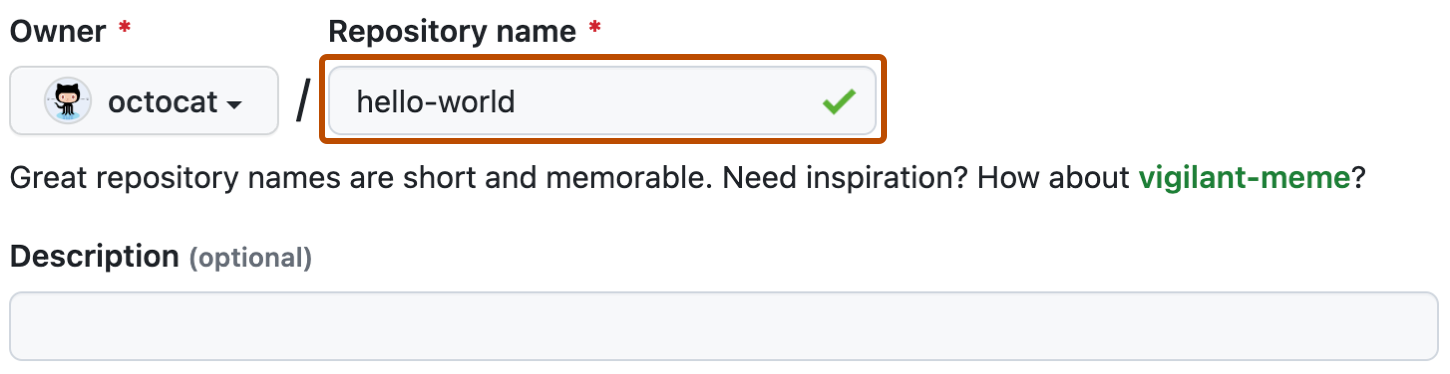
**Ans.**

GitHub repositories store a variety of projects. In this guide, you'll create a repository and commit your first change.

1. In the upper-right corner of any page, select , then click New repository.



1. Type a short, memorable name for your repository. For example, "hello-world".



1. Optionally, add a description of your repository. For example, "My first repository on GitHub."
2. Choose a repository visibility. For more information, see [About repositories](https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories#about-repository-visibility).
3. Select Initialize this repository with a README.
4. Click Create repository.

Congratulations! You've successfully created your first repository, and initialized it with a *README* file.

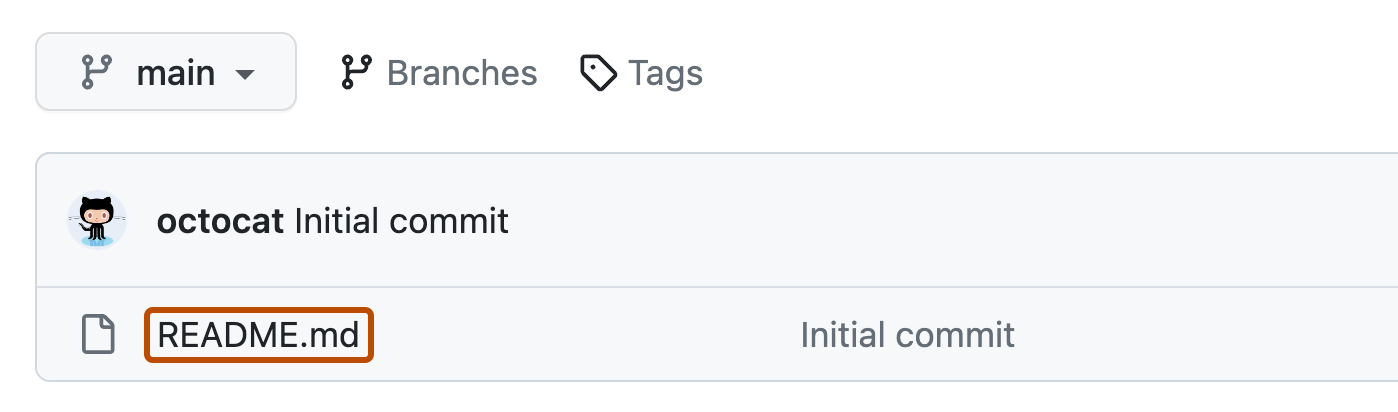
[Commit your first change](https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories#commit-your-first-change)

A [commit](https://docs.github.com/en/get-started/learning-about-github/github-glossary#commit) is like a snapshot of all the files in your project at a particular point in time.

When you created your new repository, you initialized it with a *README* file. *README* files are a great place to describe your project in more detail, or add some documentation such as how to install or use your project. The contents of your *README* file are automatically shown on the front page of your repository.

Let's commit a change to the README file.

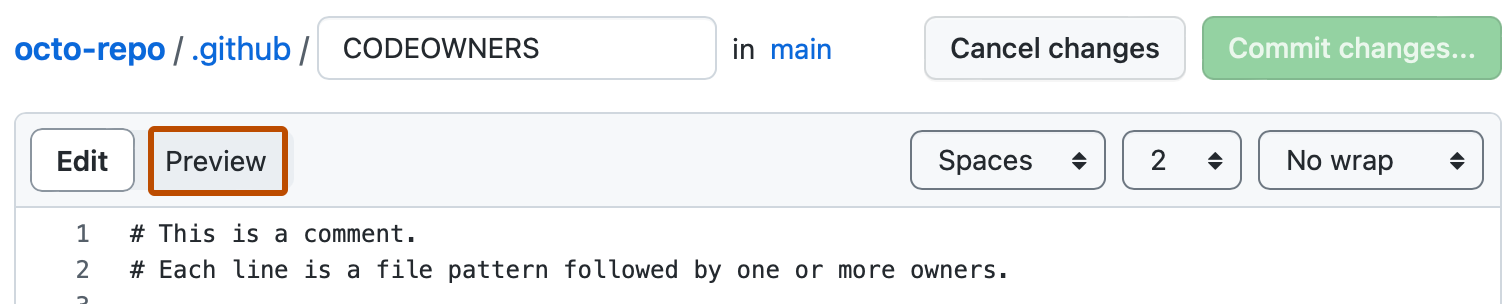
1. In your repository's list of files, select README.md.



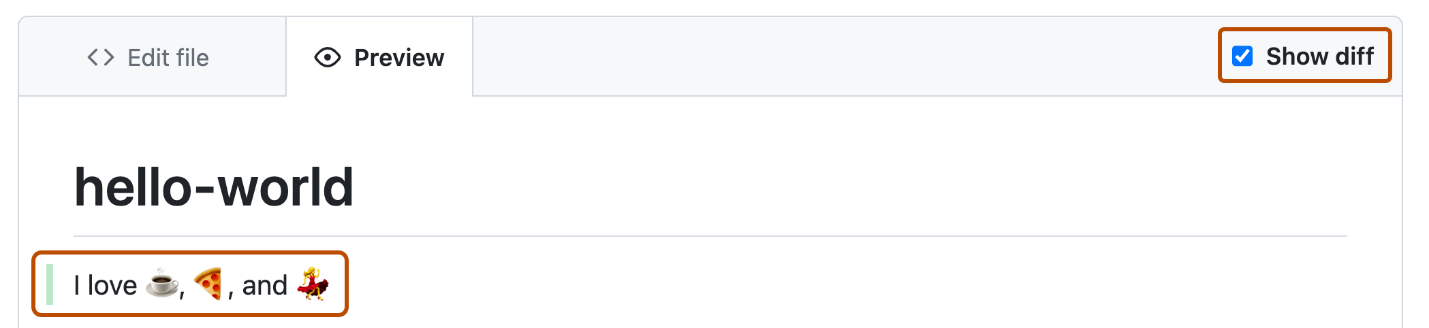
1. In the upper right corner of the file view, click  to open the file editor.



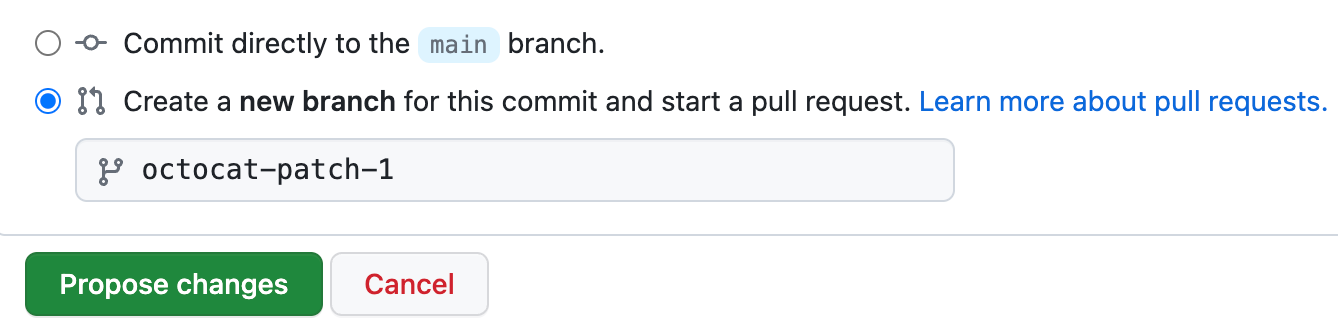
1. In the text box, type some information about yourself.
2. Above the new content, click Preview.



1. Review the changes you made to the file. If you select Show diff, you will see the new content in green.



1. Click Commit changes...
2. In the "Commit message" field, type a short, meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message. For more information, see [Creating a commit with multiple authors](https://docs.github.com/en/pull-requests/committing-changes-to-your-project/creating-and-editing-commits/creating-a-commit-with-multiple-authors).
3. Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is the default branch, you should choose to create a new branch for your commit and then create a pull request. For more information, see [Creating a pull request](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request).



1. Click Commit changes or Propose changes.

[Next steps](https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories#next-steps)

You have now created a repository, including a *README* file, and created your first commit on GitHub.

* You can now clone a GitHub repository to create a local copy on your computer. From your local repository you can commit, and create a pull request to update the changes in the upstream repository. For more information, see [Cloning a repository](https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository) and [Set up Git](https://docs.github.com/en/get-started/git-basics/set-up-git).
* Secure your repository using GitHub's available security features. For more information, see [Quickstart for securing your repository](https://docs.github.com/en/code-security/getting-started/quickstart-for-securing-your-repository).
* You can find interesting projects and repositories on GitHub and make changes to them by creating a fork of the repository. Forking a repository will allow you to make changes to another repository without affecting the original. For more information, see [Fork a repository](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo).
* Each repository on GitHub is owned by a person or an organization. You can interact with the people, repositories, and organizations by connecting and following them on GitHub. For more information, see [Finding inspiration on GitHub](https://docs.github.com/en/get-started/start-your-journey/finding-inspiration-on-github).
* GitHub has a great support community where you can ask for help and talk to people from around the world. Join the conversation on [GitHub Community](https://github.com/orgs/community/discussions).

1. Create a list of software you use regularly and classify them into the following categories: system,application, and utility software.

**Ans.**

1. System Software

System software oversees the core components of a computer system. It allows hardware and other software to work in harmony. This software handles file management, program execution, and system resource allocation. Let's find out the types of system software we come across every day:

* Operating Systems (OS): Your computer's most crucial software is the operating system. It manages all hardware components such as CPU, RAM hard drive, and screen. It also enables other software to run without issues. Windows macOS, Linux, and Android serve as examples.
* Device Drivers: Device drivers are compact programs that help hardware interact with the operating system. To illustrate, a printer requires a printer driver. A keyboard needs a keyboard driver. Each device relies on its own driver to communicate with the computer.
* Utility Software: Utility software keeps your computer in good shape. It has an impact on tasks like virus scanning, junk file removal, and backup management. Some examples include antivirus programs, disk cleanup tools, and file compressors such as WinZip.

2. Application Software

Application software is designed to perform specific jobs like writing, painting, studying, or gaming. End users are its main focus. Let’s explore the types of applications software use every day:

* Productivity Software: These are tools made to speed up work. Word processors, spreadsheets, and presentation tools – these help you make and handle content. Think Microsoft Office, Google Workspace, or Notion.
* Multimedia Software: Multimedia tools deal with audio, video, graphics, and animations. Apps like Adobe Photoshop or VLC Media Player fit this group. DevOps teams use multimedia software to record processes and create training modules.
* Web Browsers: Web browsers are digital doors. They allow users to reach websites and online platforms. Examples include Chrome, Firefox, and Safari. They run web-based DevOps dashboards like Jenkins UI and Grafana panels.
* Communication Software: These apps keep teams connected. Tools like Zoom, Slack, and Microsoft Teams help with instant messaging, video calling, and teamwork. In agile and DevOps workflows, these tools are crucial for smooth communication during sprints.

3. Programming Software

Programming software allows coders to write and test programs. It's used to create other software.[](https://squareboat.com/?ref=blog-post)

* Code Editors: Code editors are basic text environments used to write source code. Examples include Notepad++, Sublime Text, and Visual Studio Code. In DevOps, VS Code extensions support Docker, YAML, GitOps, and Terraform scripting.
* Compilers and Interpreters: These tools change human-written code into a language machines can understand. Compilers process entire programs at once, while interpreters run one line at a time. Languages like Python and Java need these to run.
* Debuggers: Debuggers help find and fix bugs in code. They follow errors, look at logs, and test how code runs. DevOps teams often use debugging tools during Continuous Testing in their pipelines.

4. DevOps and Automation Software

DevOps software brings development and operations teams together. It makes processes automatic, connects tools, and cuts down on deployment mess.

* Configuration Management Tools: These tools have an influence on infrastructure management and control. Ansible, Puppet, and Chef are some examples. They maintain server synchronization, make OS updates automatic, and guarantee consistency across different environments.
* CI/CD Tools: CI/CD tools such as Jenkins, GitLab CI, and CircleCI make build, test, and deployment cycles automatic. They help teams send code to production more without causing issues.
* Containerization and Orchestration Tools: Docker and Kubernetes operate applications in containers and coordinate how they grow. Kubernetes makes load balancing, pod scaling, and rollout strategies automatic in a cluster setting.
* Monitoring and Alerting Tools: Software like Prometheus, Datadog, and New Relic keeps an eye on system health and performance. It alerts users right away if something breaks or doesn't work well.

5. Embedded Software

Embedded software runs in machines we don't think of as computers. You'll find it in IoT gadgets medical gear, and factory robots. It comes pre-loaded, works with tiny controllers, and does things in real-time.

* Firmware: Firmware is a special kind of embedded code stored in hardware chips. It helps the device start-up and talk to other software layers. Think about your router, washing machine, or smart thermostat.
* Real-Time Operating Systems (RTOS): RTOS manages time-critical tasks in systems like pacemakers or car braking systems. It handles commands giving minimal delay and top accuracy.

6. Open Source vs Proprietary Software

This group looks at who owns and can access software.

* Open Source Software: Open Source tools are free for anyone to use, change, and help improve. Linux, Apache, and Git are some examples. DevOps works well with open-source platforms because they're flexible, clear about how they work, and have lots of add-ons.
* Proprietary Software: Companies own proprietary tools. You need to buy a license to use them. Windows OS and Adobe Creative Suite are examples. They often come with special support and extra features.

**14 .**  Write a report on the various types of application software and how they improveproductivity.

**Ans.**

Application software (App) is a kind of software that performs specific functions for the end user by interacting directly with it. The sole purpose of application software is to aid the user in doing specified tasks.

Types of Application Software

Application software can also be classified depending on how much it costs and how easily it can be accessed. Here are some examples of application software:

1. Word Processing Software

Word Processing refers to the act of using a personal computer (PC) or laptop to create, edit, save and print documents which can be performed only with specialized software known as a Word Processor. One example of a Word Processor is Microsoft Word which is widely used by all professionals.

2. Spreadsheet Software

Spreadsheet software is a type of computer program that enables a user to perform numerical functions and explore numbers through an automated version of an accounting worksheet. Best example of spreadsheet software is Microsoft Excel.

3. Presentation Software

Presentation software also commonly known as presentation graphics is a particular category of application program used to construct sequences of words and a series of pictures that tell a story or help support a speech or public presentation of any type of information or a launch of new products or services.

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack[Explore Program](https://www.simplilearn.com/full-stack-developer-course-mern-certification-training?source=GhPreviewCTABanner)



4. Multimedia Software

Multimedia software can be described as the combination of text, audio, images, animation, or video to produce a wide scope of interactive content for both professional and personal use. You can easily learn about media players, file formats, and how to operate audio and video software on the whole.

5. Web Browsers

A web browser can take you all over the internet. It retrieves data from other parts of the web and shows it on your desktop or mobile device for your viewing. The data is transmitted using the Hypertext Transfer Protocol, which describes how text, images, and video are shared on the World Wide Web.

6. Educational Software

Educational software refers to any computer software designed solely for educational reasons. It includes a wide range of software, including language learning software, classroom management software (CMS), and reference software for students and other professionals.

7. Graphics Software

Graphics software can rework with bitmap and/or vector graphics and can be utilized to create label templates. Graphics software generally includes Canva, Adobe Illustrator, Photoshop, InDesign, CorelDraw, Inkscape, Microsoft Paint, and Paint.Net.

8. Freeware

Freeware is typically marketed for profit but might be allocated specifically for a business or commercial purpose with the aim to expand the market share of any newly launched premium product. Some of the widespread examples of closed-source freeware include Adobe Reader, Free Studio, and Skype.

9. Shareware

Shareware is software that is supplied for free on a trial basis in order for the user to test or use the programme for a specific amount of days with the understanding that the user may need or want to pay for it later if they are satisfied with the product usage. Some software manufacturers provide a shareware edition of their product with an expiration date built in, such that after 30 days, the user or customer will no longer be able to access the application for further use.

10. Simulation Software

Simulation software authorizes engineers to evaluate, optimize, and compare product designs with other similar software by modeling real-world events in a computer-generated environment.

11. Open Source

Open source software is a specific code designed to be publicly accessible so that anyone can see, modify, and distribute the code as they see which fits the purpose. It is designed in a decentralized and coordinated way, depending on peer assessment and community production.

12. Closed Source

Closed source software is where the source code is not freely accessible. It is developed and delivered to the customer as a fully compiled, executable set of files. The developer often provides aid to users after purchase and ensures that the software works as foreseen by the creator.

**Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software.** SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step. The **goal of the SDLC life cycle model** is to deliver high-quality, maintainable software that meets the user’s requirements. SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users requirements. In this article we will see Software Development Life Cycle (SDLC) in detail.

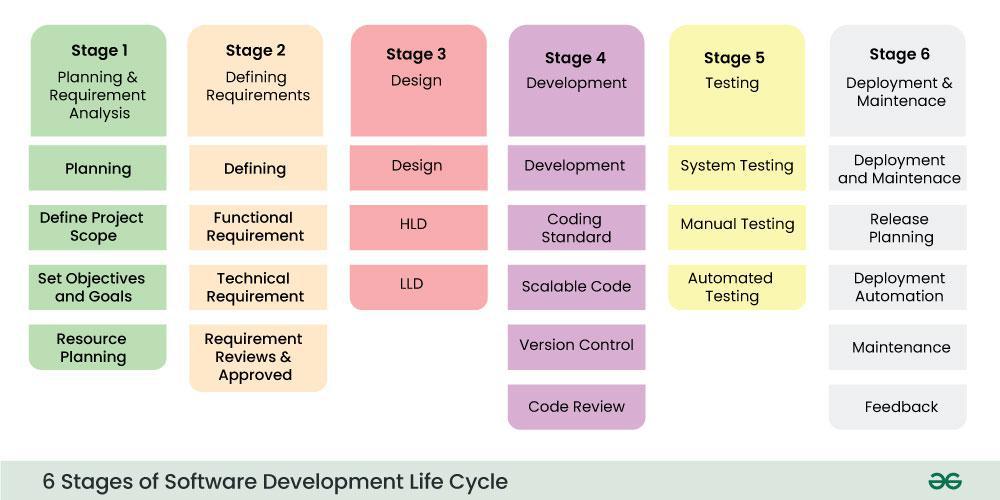
**What is the Software Development Life Cycle (SDLC)?**

**SDLC is a process followed for software building within a software organization.**SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software. The life cycle defines a method for improving the quality of software and the all-around development process.



**Stages of the Software Development Life Cycle**

SDLC specifies the task(s) to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer’s expectations and fits within the overall budget. Hence, it’s vital for a software developer to have prior knowledge of this software development process. SDLC is a collection of these six stages, and the stages of SDLC are as follows:



*Software Development Life Cycle Model SDLC Stages*

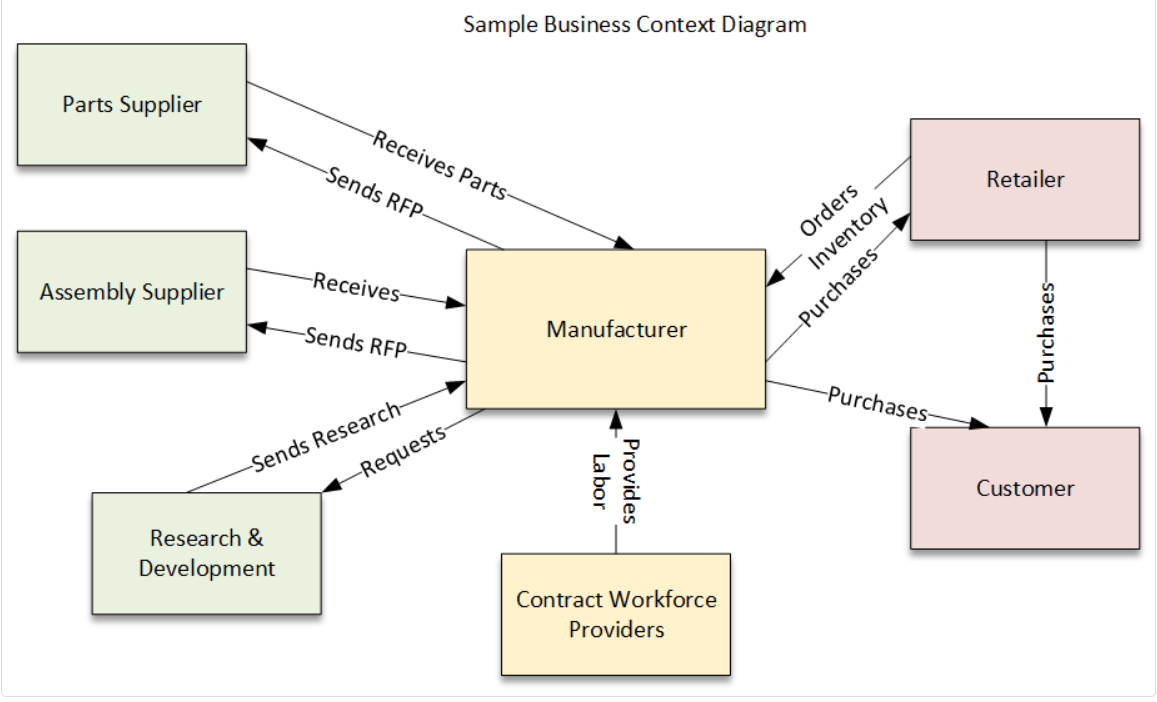
The [**SDLC Model**](https://www.geeksforgeeks.org/sdlc-models-types-phases-use)involves six phases or stages while developing any software.

1. Write a requirement specification for a simple library management system.

**Ans.** Software Requirements Specification for Library Management System Table of Contents: 1.0 Introduction 1.1 Purpose 1.2 Scope of Product 1.3 Definitions, Acronyms, and Abbreviations 1.4 References 1.5 Overview 2.0 General Description 2.1 Product Perspective 2.2 Product Functions 2.3 User Characteristics 2.4 General Constraints 2.5 Assumptions and Dependencies 3.0 Specific Requirements 3.1 Functional Requirements 3.1.1 User Interface 3.1.1.1 Library user account manage system 3.1.1.2 Book borrowing system 3.1.1.3 Book recall system 3.1.1.4 Search book record 3.1.2 Data Entry 3.1.2.1 Library user account manage system 3.1.2.2 Book borrowing system 3.1.2.3 Book recall system 3.1.2.4 Search book record 3.1.2.5 Update book database 3.1.3 Report Generation 3.2 Performance Requirements 3.3 Design Constraints 3.4 Security Requirements 3.5 Reliability 1.0 Introduction 1.1 Purpose This is the Software Requirements Specification (SRS) for the Library Management System. The purpose of this document is to convey information about the application's requirements, both functional and non-functional, to the reader. This document provides (a) a description of the environment in which the application is expected to operate, (b) a definition of the application's capabilities, and (c) a specification of the application's functional and nonfunctional requirements. The document is intended to serve several groups of audiences: First, it is anticipated that the SRS will be used by the application designers. Designers will use the information recorded here as the basis for creating the application's design. Second, the client for the project, the library manager in our case, is expected to review this document. The SRS will serve to establish a basis for agreement between the client and development team about the functionality to be provided by the application. Third, the application maintainers will review the document to clarity their understanding of what the application does. Fourth, test planners will use this document to derive test plans and test cases. Finally, the project manager will use this document during project planning and monitoring. 1.2 Scope of Product The purpose of this software development project is to create a new application called: Library Management System. The client for this project wishes to enter the PC-based LAN environment. The Library Management System will be PC-base with a LAN, allowing library users to search for books and library staff members to manage the book inventory and user database. The application will provide the following capabilities: The application will be access via a LAN on a PC terminal in the Library Library staff will be able to manage library user accounts including remove, change, and add. Library staff will be able to manage the book inventory database including remove, change, and add. The application will record all books that are checked out, checked in, and recalled. The application will generate reports for administrative purposes. The application will provide search function on books based on ISBN, subject, title, or author. The project's client has determined that this application will provide the following benefits: Provide additional flexibility and convenience to the library users. Provide better reliability and security of the library information. Provide a more productive environment for the library staff member. Reduce the cost of the library operations. 1.3 Definitions, Acronyms, and Abbreviations SRS - Software Requirements Specification. PC - Personal Computer. LAN - Local Area Network. IEEE - Institute of Electrical and Electronics Engineers. ISBN - International Standardized Bin Number. 1.4 References 1. Merlin Dorfman, Richard H. Thayer, "Standards, Guidelines, and Examples on System and software Requirements Engineering", IEEE, 1990. 1.5 Overview A brief description of the content of each chapter is given below. 1.0 Introduction Provides an overview of the project. Summarizes the major capabilities of the product. 2.0 General Description Presents the environment in which the application is expected to operate, provides an overview of the system requirements, describes assumptions about possible users of the application, possible constraints on the project, and the underlying assumptions that on which the requirements analysis is based. 3.0 Specific Requirements The specification of requirements. Contains a complete description of the application's requirements, both functional and non-functional. 2.0 General Description 2.1 Product Perspective The Library Book System is used for Library Manager, Librarian, and Library User. The system is self-contained. However, it is possible to exchange data with other system through external interface if required. The following is a typical system diagram: 2.2 Product Functions The high level summary of functions in Library Book System is described in the following concept map. Detail functional requirements will be described in section 3. 2.3 User Characteristics The three types of user for the Library Book System are: Library Manager Librarian Library User The following table describe general users characteristics that will affect the functionality of the software product. Type of User User Characteristic User Technical Expertise How the user characteristic and technical expertise affect Library Book System functionality Library Manager Good understanding to library operation Responsible for library operation as a whole. Responsible for library staff managing Average in technical proficiency Used text type terminal in the old library book system User interface with less input steps. Easy to learn. Librarian Good understanding to library operation Responsible for library operation. Average in technical proficiency Used text type terminal in the old library book system User interface with less input steps. Easy to learn. (Diverse user characteristic) Younger generation tends to accept and learn new thing (i.e. computer system) easier than older generation Younger generation GUI interface may be easier to learn than text interface. Library User Older generation may like the new system be similar to the old one in terms of user interfaces and functionality. Will not have any formal training to use the system. has a lot of exposure to Windows type application Provide system help Provide appropriate error messages for invalid user inputs. 2.4 General Constraints The current hardware for the existing Library Book System is mainframe with text type terminal. Therefore, if the new system is PC based, there will be a need to replace with PC hardware and new network facility. The Library Book System can potentially have hundreds of users. It is unrealistic to provide training for everyone. Therefore, the system should be designed for easy to use, providing help instructions, and appropriate error messages for invalid user inputs. Security is important to library operation. Library user is allowed to use the Library Book System only for searching book records. User should never be able to break into the system and to perform any modification. Reliability is vital to library operation. The Library Book System should not have any unscheduled down time during library operation hours. Any down time in operation hours has significant impact to the operation and cause inconvenience to everyone in library. 2.5 Assumptions and Dependencies The following is a list of assumptions and dependencies that would affect the software requirements if they turned out to be false: Users have basic understanding to PC and Windows. There is a method to convert all book records and library user records from the existing system into the Library Book System. 3.0 Specific Requirements: This section contains the detailed requirements. In this section, the users of "Search Book Record" are refereed to librarians and patrons (library users). Users of other sections are only refereed to the librarian card holder (librarians and library managers.) 3.1 Functional Requirements 3.1.1 User Interface The user interface requirements are concerned with the user interface and how information is presented to the user. 3.1.1.1 Library user account manage system SRS-001: The system shall display the user account information including user ID, last and first name, and user position, privilege. SRS-002: The system shall use a graphic user interface which allows librarians to choice actions including removing, changing and adding user account and account information.. 3.1.1.2 Book borrowing system. SRS-003: When check out the books, the system shall show all the book borrowing information about a particular user including: the first and last name of the user the library card number the library card expiring date if the user has unpaid penalty the number of books the user has not return SRS-004: When check out the books, when required by librarians, the system shall show the information about books which is borrowed before and not returned yet including: the ISBN of the book the title of the book the borrowing and due date of the book the branch which the book is checked out SRS-005: When check out the books, the system shall display the information of the book which is just being checking out including: ISBN, title, due date. SRS-006: When check in the books, the system shall show the title, ISBN of the book which is being checked in. When check in is finished, a "check in" stamp shall be seen. 3.1.1.3 Book recall system SRS-007: When recalling the book, the system shall display a list of all the copies of the book which is borrowed out ordered by book lend out time. SRS-008: When the recalled book is arrived, the system shall display the last and first name, the recall date and the arriving date of the book. If only one copy of book is arrived and more than one user are waiting, users shall be displayed ordered by recalling time. When check in recall book is finished, a "check in recall" stamp shall be seen. 3.1.1.4 Search book record SRS-009: The system shall display a list of books which are match the search criteria sorted by book titles including: the category the ISBN the title the author SRS-008: When required by users, the system shall display the information about a particular book including: the category the title the ISBN the publisher the brief description of the book (if any stored in database) the location in library 3.1.2 Data Entry The data entry requirements are concerned with how data is entered and validated. SRS-010: The system shall allow a user to enter his/her data via a keyboard or choose an item via a mouse. SRS-011: Whenever the "date" data is needed, it shall be entered only by choose date from a online calendar. SRS-012: The system shall allow the user to enter the library card number and ISBN both by typing or scanning. SRS-013: The system shall allow the user to enter book borrowing, recalling data as frequently as required. 3.1.2.1 Library user account manage system SRS-014: The system shall allow the user to attach notes to each account SRS-015: The system shall allow the user to add or change information in an account including: last name, first name, user ID, user position, user privilege. SRS-016: the system shall allow the user to delete an entire account. 3.1.2.2 Book borrowing system. SRS-017: The system shall allow the user to specify a checking out book using its ISBN. SRS-018: The system shall allow the user to specify a patron by the library card number. SRS-019: The system shall allow the user to specify a checking in book using its ISBN. SRS-020: The system shall allow the user to specify that a penalty is paid. SRS-021: The system shall check and show the number of books which are checked out and if the number is exceeded the limitation for patrons except for librarian card holders. SRS-022: The system shall check and show if the book can only be used in library SRS-023: The system shall let the librarian card holders to check out books which can only be used in library. SRS-024: The system shall commit the check in and check out data to the database as soon as the data is entered. 3.1.2.3 Book recall system SRS-025: The system shall allow the user to specify a recalled book by book ISBN SRS-026: The system shall allow the user to choose one copy of book to recall from a list of copies which are borrowed out. SRS-027: The system shall allow the user to record the record notification send out date, the book arrive date, the pick-up notification send out date. 3.1.2.4 Search book record SRS-028: The system shall allow the user typing in search criteria including book title, key word in title, ISBN, subject, category. SRS-029: The system shall allow the user choose language option which the searched book is used including English, Spanish and French. SRS-030: If the search result are a list of books, the system shall allow the user to choose any one of them to see the details. 3.1.2.5 Update book database SRS-031: The system shall allow the user to add or change the record information including: the category the title the ISBN the publisher the brief description of the book the location in library the purchase date the price SRS-032: the system shall allow the user to put "delete" stamp for a existing book and specify the deleting reason. 3.1.3 Report Generation the report generation requirements are concerned with the report generation capabilities of the Library system. SRS-034: The system shall have a report feature that will allow the user to generate a report showing the information of all the sign out book in a time period which is the search criteria input by user. The information includes the number of books, the time period and the information is grouped by book categories. SRS-035: The system shall have a report feature that will allow the user to generate a report showing the information about all the users who have overdue books and penalty. SRS-036: The system shall have a report feature that will allow the user to generate a report showing the information of a particular patron. SRS-037: The system shall have a report feature that will allow the user to generate a report showing the information of book purchase information in a period including the book titles, category, the author, the publisher, the price. It also shall give statistic data about the total number of books purchased, the money paid by category. SRS-038: The system shall be generate those reports to the display, a file or a printer which is linked to the system. 3.2 Performance Requirements SRS-039: The check in and check out system shall respond to the user no more than 5 seconds. The search function shall respond to the user no more than 9 seconds. 3.3 Design Constraints SRS-040: The system shall be installed in a windows-NT network. 3.4 Security Requirements SRS-041: The account management system shall only be used by managers or users with defined privileges. SRS-042: The check-in, check-out and recall system shall only be used by users who have librarian ID. SRS-043: The Patron information report shall be generated by users who have librarian ID. SRS-044: The book sign out report or book purchase report shall only be generated by managers or users with defined privileges. SRS-045: Database update data shall be committed to the database only after the managers have approved. 3.5 Reliability SRS-041: The system shall be recovered within 10 minutes if it is down. SRS-042: The system shall be recovered without intervention at user terminal if it is down. SRS-043: The system shall show appropriate messages at terminal when system is down. SRS-044: The system shall have 99% reliability during library operating hours. SRS-045: Scheduled down time after library operating hours shall not be more than 1 hour per day. SRS-046: The system shall generate error messages when the user attempts to enter invalid data. See also: Requirement Specification Document Last modified: Oct. 21, 97

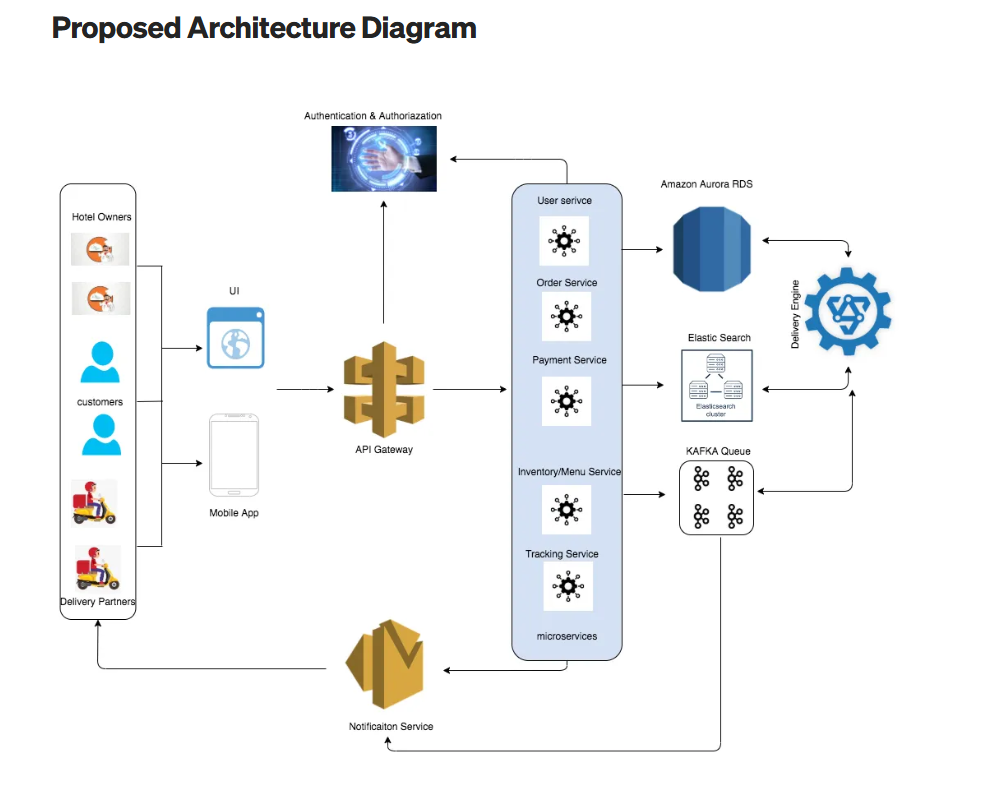
1. Perform a functional analysis for an online shopping system.

**Ans.** The business context diagram is a block diagram that outlines all the major internal entities in the organization and the relationships between them, together with the various external entities that the organization interacts with and the relationships and information flows that occur.

****

1. Design a basic system architecture for a food delivery app

**Ans.**

****

1. Develop test cases for a simple calculator program.

**Ans.**

C Program to make a Simple Calculator using

switch-case statements

#include <stdio.h>

#include <float.h>

int main() {

char op;

double a, b, res;

// Read the operator

printf("Enter an operator (+, -, \*, /): ");

scanf("%c", &op);

// Read the two numbers

printf("Enter two operands: ");

scanf("%lf %lf", &a, &b);

// Define all four operations in the corresponding

// switch-case

switch (op) {

case '+':

res = a + b;

break;

case '-':

res = a - b;

break;

case '\*':

res = a \* b;

break;

case '/':

res = a / b;

break;

default:

printf("Error! Incorrect Operator Value\n");

res = -DBL\_MAX;

}

if(res!=-DBL\_MAX)

printf("%.2lf", res);

return 0;

}

1. Document a real-world case where a software application required criticalmaintenance.

**Ans.**

A critical maintenance scenario could involve a bank's online banking platform experiencing a widespread outage due to a security vulnerability, leading to a complete failure of user authentication and transaction processing. This would necessitate immediate corrective maintenance to patch the security flaw and restore functionality.

Elaboration:

Imagine a scenario where a major bank, like a hypothetical "MegaBank", utilizes a complex web-based banking platform to manage its users' accounts and transactions. This platform relies on a sophisticated authentication system, a robust database for storing user information, and a secure network infrastructure to handle financial transactions.

The Problem:

Suddenly, a major security vulnerability emerges in the authentication module of MegaBank's platform. This vulnerability could be a flaw in how user credentials are handled, potentially allowing malicious actors to gain unauthorized access to user accounts. As a result, the authentication system begins to fail, leading to a complete outage where users cannot log in or perform any transactions.

The Impact:

This outage has severe consequences:

* **User Frustration:**

Users cannot access their accounts, leading to widespread frustration and panic as they try to manage their finances.

* **Financial Loss:**

The inability to perform transactions could lead to missed payments, overdrafts, and financial losses for users.

* **Reputational Damage:**

The outage can severely damage MegaBank's reputation and erode user trust, potentially leading to a loss of customers.

* **Legal Ramifications:**

The outage could lead to legal repercussions, especially if user data is compromised or if the bank is deemed negligent in securing its platform.

The Solution:

In this situation, MegaBank's IT team must immediately implement a critical maintenance plan. This plan would involve the following steps:

1. **1. Identify the Vulnerability:**

The root cause of the outage needs to be pinpointed. This involves analyzing system logs, network traffic, and other relevant data to identify the specific security flaw.

1. **2. Develop a Patch:**

A temporary workaround or a permanent fix needs to be developed to address the vulnerability. This may involve modifying the authentication module's code, updating libraries, or implementing new security measures.

1. **3. Implement the Patch:**

The patch needs to be deployed to the affected systems in a controlled manner, minimizing downtime and ensuring that the changes are successful.

1. **4. Restore Functionality:**

Once the patch is deployed, the authentication system needs to be tested to ensure that it is working correctly and that users can log in and perform transactions.

1. **5. Communicate with Users:**

MegaBank needs to communicate with its users, informing them about the outage, the reason for it, and the steps being taken to resolve the issue.

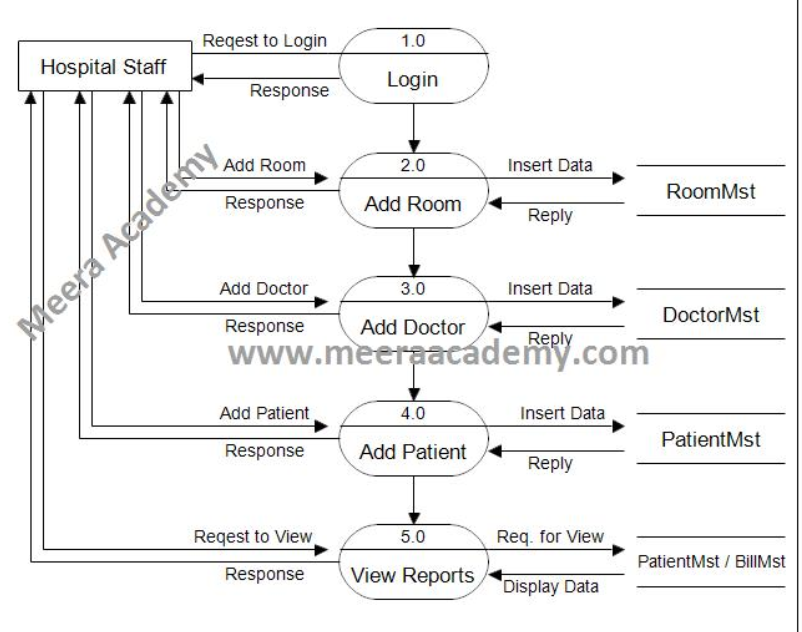
1. **6. Monitor and Review:**

After the issue is resolved, the system needs to be monitored to ensure that it remains stable and that no new vulnerabilities are introduced.

In summary, this real-world case highlights the importance of critical maintenance in addressing unexpected events and maintaining the integrity of software systems. It emphasizes the need for a robust security framework, proactive monitoring, and a well-defined maintenance plan to ensure that software applications can withstand unexpected challenges and continue to meet the needs of their users.

1. Create a DFD for a hospital management system.

**Ans.**



1. Build a simple desktop calculator application using a GUI library.

**Ans.**

# Python program to create a simple GUI

# import everything from tkinter module

from tkinter import \*

# globally declare the expression variable

expression = ""

# Function to update expression

# in the text entry box

def press(num):

# point out the global expression variable

global expression

# concatenation of string

expression = expression + str(num)

# update the expression by using set method

equation.set(expression)

# Function to evaluate the final expression

def equalpress():

# Try and except statement is used

# for handling the errors like zero

# division error etc.

# Put that code inside the try block

# which may generate the error

try:

global expression

# eval function evaluate the expression

# and str function convert the result

# into string

total = str(eval(expression))

equation.set(total)

# initialize the expression variable

# by empty string

expression = ""

# if error is generate then handle

# by the except block

except:

equation.set(" error ")

expression = ""

# Function to clear the contents

# of text entry box

def clear():

global expression

expression = ""

equation.set("")

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

# create a GUI window

gui = Tk()

# set the background colour of GUI window

gui.configure(background="light green")

# set the title of GUI window

gui.title("Simple Calculator")

# set the configuration of GUI window

gui.geometry("270x150")

# StringVar() is the variable class

# we create an instance of this class

equation = StringVar()

# create the text entry box for

# showing the expression .

expression\_field = Entry(gui, textvariable=equation)

# grid method is used for placing

# the widgets at respective positions

# in table like structure .

expression\_field.grid(columnspan=4, ipadx=70)

# create a Buttons and place at a particular

# location inside the root window .

# when user press the button, the command or

# function affiliated to that button is executed .

button1 = Button(gui, text=' 1 ', fg='black', bg='red',

command=lambda: press(1), height=1, width=7)

button1.grid(row=2, column=0)

button2 = Button(gui, text=' 2 ', fg='black', bg='red',

command=lambda: press(2), height=1, width=7)

button2.grid(row=2, column=1)

button3 = Button(gui, text=' 3 ', fg='black', bg='red',

command=lambda: press(3), height=1, width=7)

button3.grid(row=2, column=2)

button4 = Button(gui, text=' 4 ', fg='black', bg='red',

command=lambda: press(4), height=1, width=7)

button4.grid(row=3, column=0)

button5 = Button(gui, text=' 5 ', fg='black', bg='red',

command=lambda: press(5), height=1, width=7)

button5.grid(row=3, column=1)

button6 = Button(gui, text=' 6 ', fg='black', bg='red',

command=lambda: press(6), height=1, width=7)

button6.grid(row=3, column=2)

button7 = Button(gui, text=' 7 ', fg='black', bg='red',

command=lambda: press(7), height=1, width=7)

button7.grid(row=4, column=0)

button8 = Button(gui, text=' 8 ', fg='black', bg='red',

command=lambda: press(8), height=1, width=7)

button8.grid(row=4, column=1)

button9 = Button(gui, text=' 9 ', fg='black', bg='red',

command=lambda: press(9), height=1, width=7)

button9.grid(row=4, column=2)

button0 = Button(gui, text=' 0 ', fg='black', bg='red',

command=lambda: press(0), height=1, width=7)

button0.grid(row=5, column=0)

plus = Button(gui, text=' + ', fg='black', bg='red',

command=lambda: press("+"), height=1, width=7)

plus.grid(row=2, column=3)

minus = Button(gui, text=' - ', fg='black', bg='red',

command=lambda: press("-"), height=1, width=7)

minus.grid(row=3, column=3)

multiply = Button(gui, text=' \* ', fg='black', bg='red',

command=lambda: press("\*"), height=1, width=7)

multiply.grid(row=4, column=3)

divide = Button(gui, text=' / ', fg='black', bg='red',

command=lambda: press("/"), height=1, width=7)

divide.grid(row=5, column=3)

equal = Button(gui, text=' = ', fg='black', bg='red',

command=equalpress, height=1, width=7)

equal.grid(row=5, column=2)

clear = Button(gui, text='Clear', fg='black', bg='red',

command=clear, height=1, width=7)

clear.grid(row=5, column='1')

Decimal= Button(gui, text='.', fg='black', bg='red',

command=lambda: press('.'), height=1, width=7)

Decimal.grid(row=6, column=0)

# start the GUI

gui.mainloop()

21. Draw a flowchart representing the logic of a basic online registration system.

**Ans.**

**A diagram of a flowchart

AI-generated content may be incorrect.**