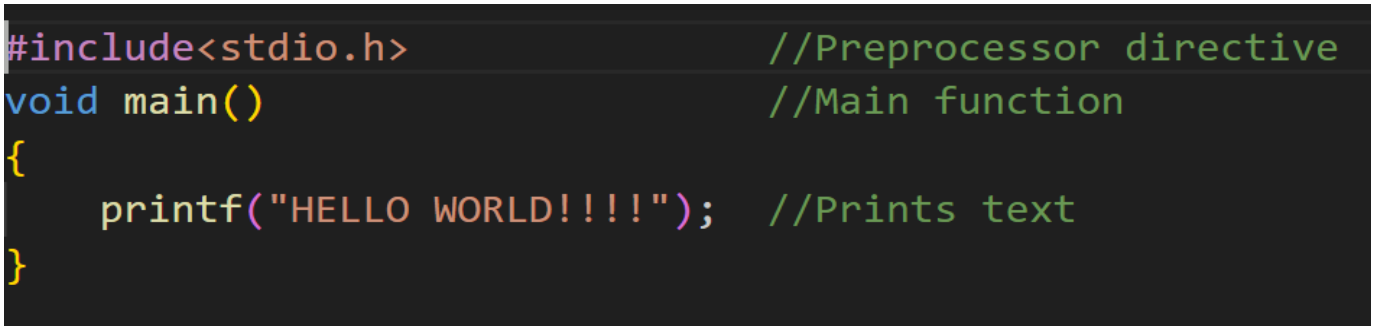
MODULE 1 – OVERVIEW OF IT INDUSTRY

1. Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

- *C Program:*



```c
#include<stdio.h>                //Preprocessor directive
void main()                      //Main function
{
    printf("HELLO WORLD!!!!");   //Prints text
}
```

- *Python Program:*

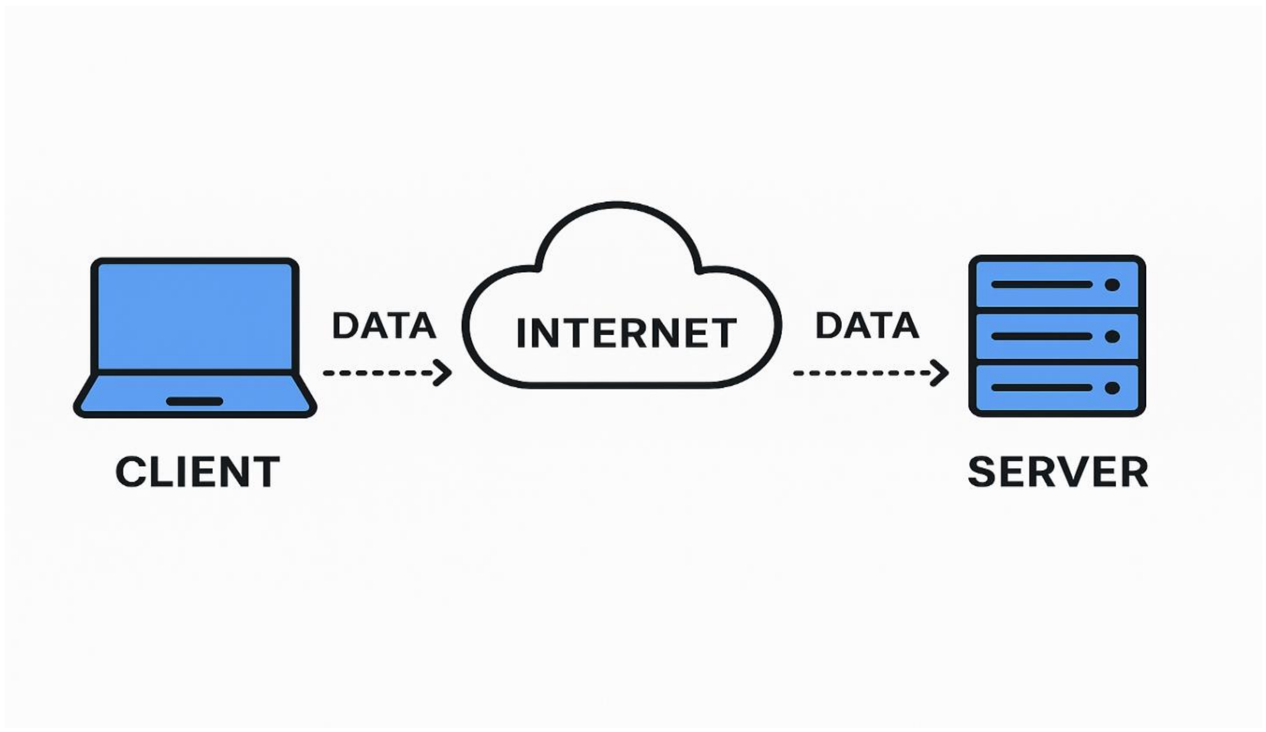
```python
print("Hello World")         # Prints text to the console
```

- *Comparison of Structure & Syntax*

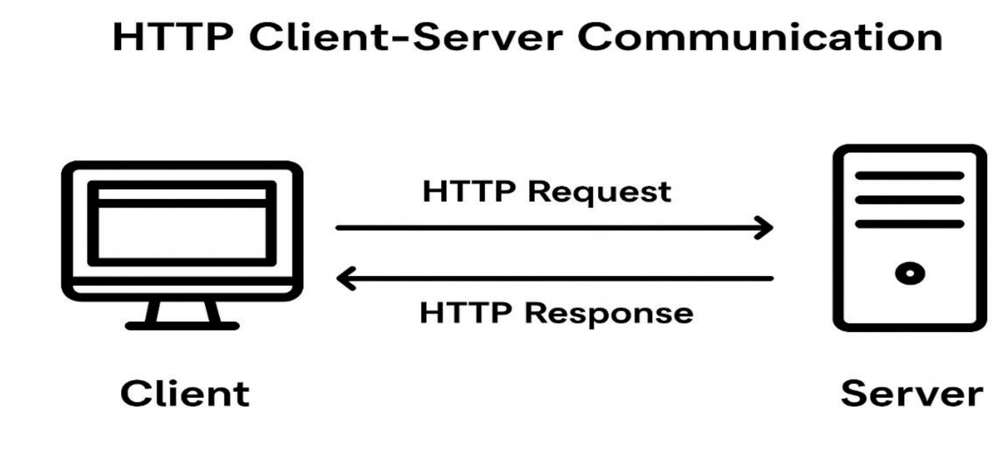
Aspect	C	Python
File setup	Needs #include for standard I/O	No setup needed for simple printing
Main function	Execution starts in main() function	No main() function required
Semicolons	Required at the end of statements	Not required
Curly braces {}	Used to define code blocks	Uses indentation for blocks
Data types	Must specify data types (e.g., int)	Dynamically typed (no declaration)
Line count	Longer (more boilerplate code)	Very short and concise
Compilation	Needs to be compiled before execution	Interpreted directly by Python

2. Research and create a diagram of how data is transmitted from a client to a server over the internet.

i. You send a request
ii. Example: You type a website name and press Enter.
iii. Data is split into packets
iv. Your request is broken into small pieces called *packets*.
v. Packets travel through the internet
vi. They pass through routers and networks until they reach the server.
vii. Server processes the request
viii. The server puts packets together, understands what you want, and finds the data.
ix. Response is sent back
x. The server sends the answer in packets back to you the same way.
xi. You see the result
xii. Your device reassembles the packets and shows the webpage or data.



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



- **Client**: Sends a request to the server asking for data (e.g., a web page).
- **Server**: Receives the request, processes it, and sends back a response (e.g., HTML page).
- **HTTP Request**: "Give me this page or data."
- **HTTP Response**: "Here's the data you asked for."

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

Internet Type	Description	Pros	Cons
Broadband (DSL / Cable)	High-speed internet via telephone lines (DSL) or TV coaxial cables (Cable)	Widely available, affordable, always-on connection	Speed drops with distance (DSL), cable slows at peak times, lower upload speeds
Fiber-Optic	Uses glass fiber cables to transmit data as light pulses	Very high speeds, low latency, not affected by electrical interference	Limited availability, higher installation cost
Satellite	Connects via satellites orbiting Earth	Available almost anywhere, good for remote areas	High latency, weather affects performance, data caps, high cost
Mobile Data (3G/4G/5G)	Wireless internet via cellular networks	Portable, easy setup, 5G offers high speed	Data caps, speed depends on signal, expensive for heavy use

Internet Type	Description	Pros	Cons
Fixed Wireless	Uses radio signals from local tower to antenna	No cables needed, faster than satellite, lower latency	Needs line-of-sight, weather and obstruction issues, limited coverage

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

- Curl, short for "Client for URLs", is a command line tool for transferring data using various protocols.

1. HTTP Example

a) GET Request

Command:

```
curl https://httpbin.org/get
```

Purpose: Fetches webpage data in JSON format, showing request details like IP and headers.

---

b) POST Request

Command:

```
curl -X POST -d "name=Shifa&course=IT"
https://httpbin.org/post
```

Purpose: Sends form data to the server. The server responds with the exact data sent in JSON format.

---

2. FTP Example

a) List Files

Command:

```
curl ftp://speedtest.tele2.net/
```

Purpose: Lists files available on the public FTP server.

b) Download a File

Command:

```
curl -O ftp://speedtest.tele2.net/1KB.zip
```

Purpose: Downloads a small 1KB test file to the current folder.

- Conclusion:

We successfully simulated HTTP and FTP requests using `curl` commands, retrieved data, and downloaded files from a public FTP server.

6. Identify and explain three common application security vulnerabilities. Suggest possible solutions.

1. SQL Injection (SQLi)

- Happens when attackers insert malicious SQL commands into input fields.
- Can lead to unauthorized data access, modification, or deletion.
- **Solution:** Use prepared statements, sanitize user input, and apply least-privilege access to databases.

2. Cross-Site Scripting (XSS)

- Attackers inject malicious scripts into web pages.
- These scripts run in the victim's browser, stealing data or hijacking sessions.
- **Solution:** Escape output, sanitize HTML inputs, and use Content Security Policy (CSP).

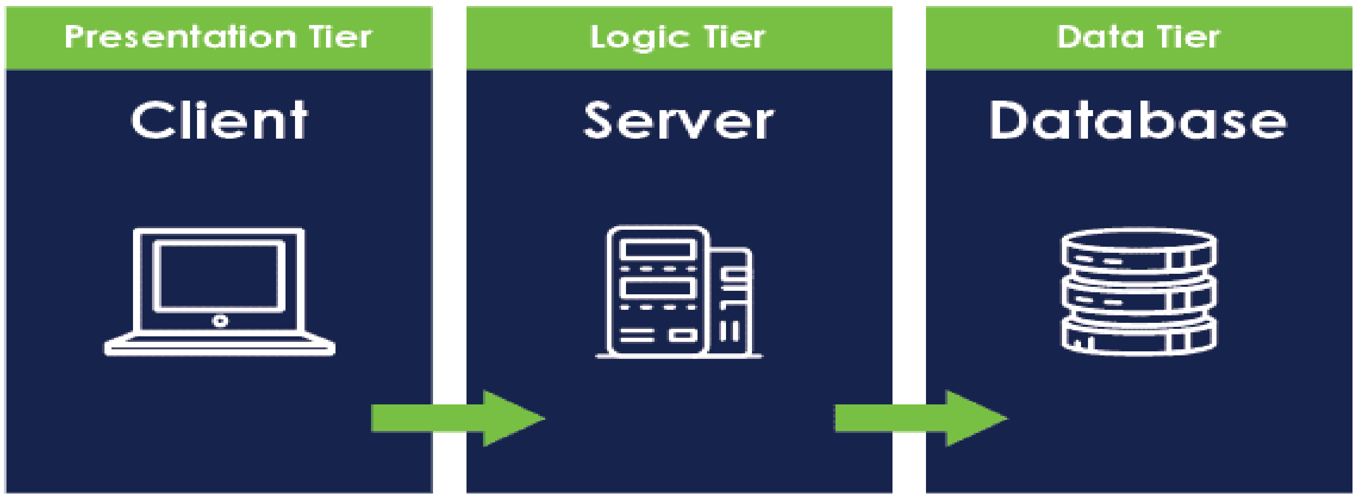
3. Cross-Site Request Forgery (CSRF)

- Tricks a logged-in user into performing unwanted actions without knowing.
- Can be used for unauthorized transactions or settings changes.
- **Solution:** Use anti-CSRF tokens, Same Site cookies, and re-confirm sensitive actions.

7. Identify and classify 5 applications you use daily as either system software or application software.

Application Name	**Type**	**Reason**
Google Chrome	Application Software	Used to browse the internet and interact with web content.
Microsoft Word	Application Software	Used for creating and editing documents.
WhatsApp	Application Software	Messaging and calling tool for communication.
Windows 11 (OS)	System Software	Manages hardware and runs other programs.
Antivirus (e.g., Avast)	System Software	Protects the operating system from malware and threats.

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



9. Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

- Case Study – Online Food Ordering App
  - Presentation Layer (UI)

    - Shows menus, prices, order status.
    - Example: Mobile app screen or website.

  - Business Logic Layer

    - Calculates bill, applies discounts, processes orders.
    - Example: Server code that checks if delivery is free.

  - Data Access Layer

    - Stores and retrieves data from database.
    - Example**:** Saves user profile, fetches menu items.

<u>Flow:</u>

User selects food $\Rightarrow$ App calculates bill $\Rightarrow$ Data is stored in database $\Rightarrow$ Order status is shown to user.

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

Environment	**Purpose**	**Key Features**
Development	Where developers write and test new code.	Has debugging tools, code editors, local databases; may have mock data instead of real data.
Testing (or Staging)	Where QA/testers check if the software works correctly.	Mirrors production as closely as possible; uses real or close-to-real data; automated/manual testing is done here.
Production	The live environment where users interact with the software.	Fully optimized, stable, secure, and connected to real databases and users.

- Basic VM Setup

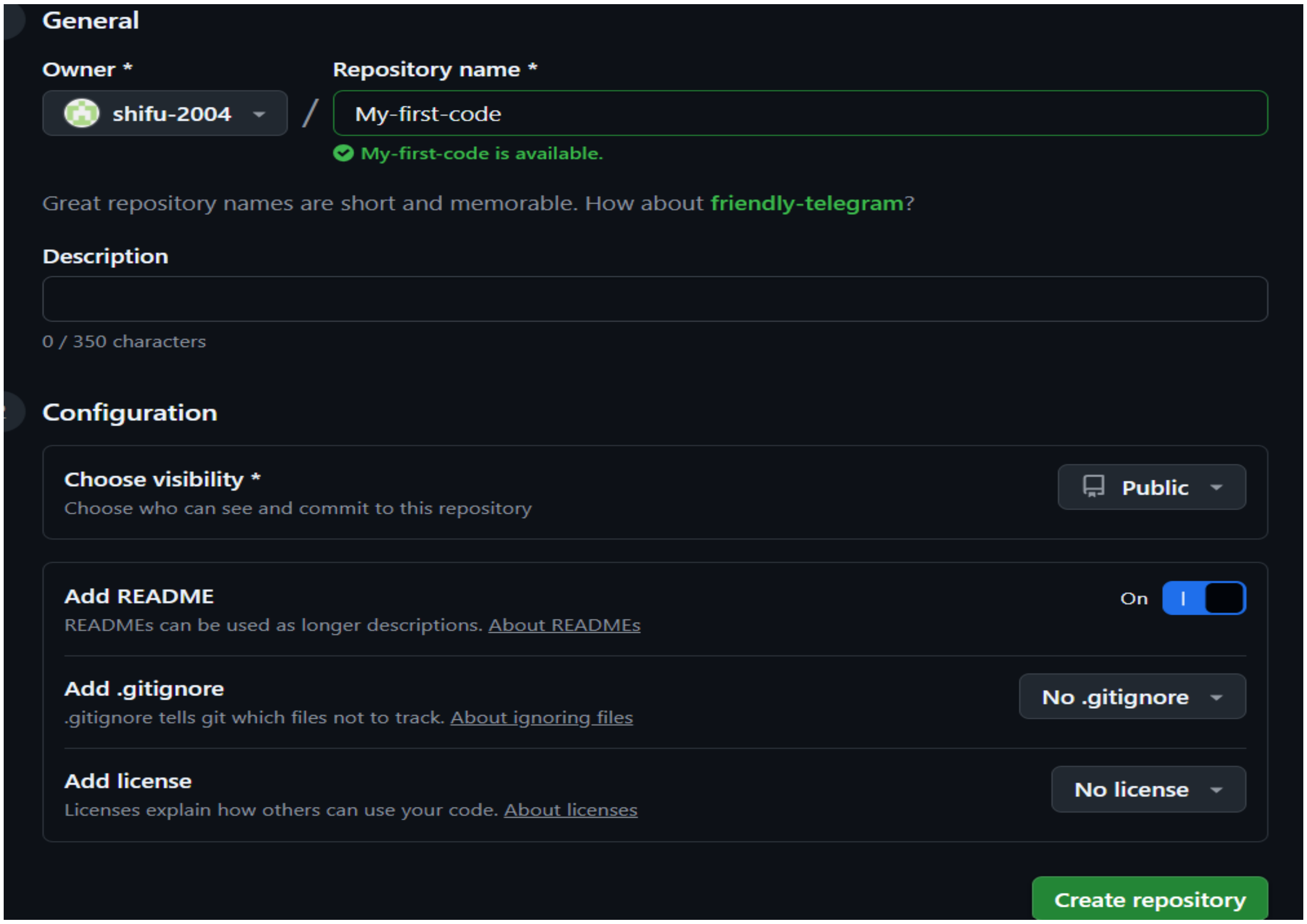
- Get Virtual Box (install it).
- Download Ubuntu ISO**.**
- Create VM → choose Linux, Ubuntu, set RAM (2–4 GB) & Disk (20 GB).
- Start VM → install Ubuntu (default settings).
- Install tools inside Ubuntu:
  - bash
  - CopyEdit
  - sudo apt update
  - sudo apt install git
- Done – your basic environment is ready.

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

1. Create a GitHub account **→** github.com.
2. Make a new repository → Click **+ →** new repository → Create**.**
3. Write your code →Example:

   C programming>>>>>
   ```
   #include <stdio.h>
   int main() {
      printf("Hello, GitHub!\n");
      return 0;
   }
   ```

4. Upload **→** In your repo, click Add file → Upload files → Choose file → Commit changes**.**



General

Owner * shifu-2004 / Repository name * My-first-code

My-first-code is available.

Great repository names are short and memorable. How about friendly-telegram?

Description

0 / 350 characters

Configuration

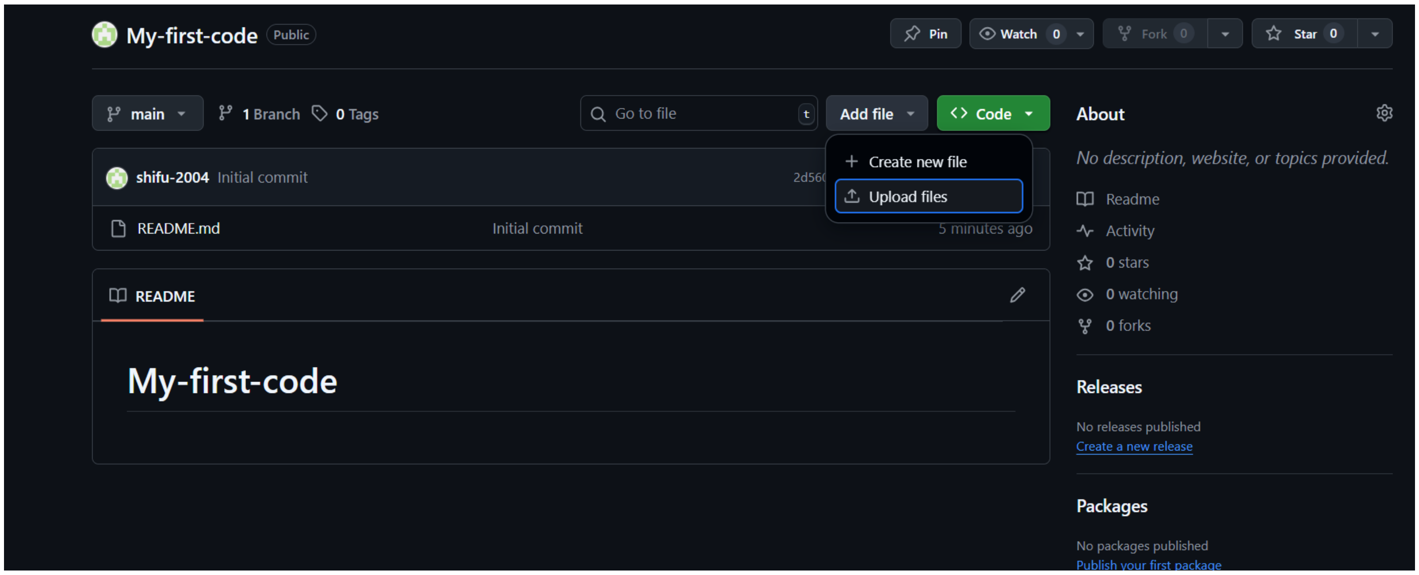
Choose visibility * — Choose who can see and commit to this repository — Public

Add README — READMEs can be used as longer descriptions. About READMEs — On

Add .gitignore — .gitignore tells git which files not to track. About ignoring files — No .gitignore

Add license — Licenses explain how others can use your code. About licenses — No license

Create repository



My-first-code Public

Pin · Watch 0 · Fork 0 · Star 0

main · 1 Branch · 0 Tags · Go to file · Add file · Code

Create new file
Upload files

shifu-2004 Initial commit 2d56

README.md — Initial commit — 5 minutes ago

README

My-first-code

About

No description, website, or topics provided.

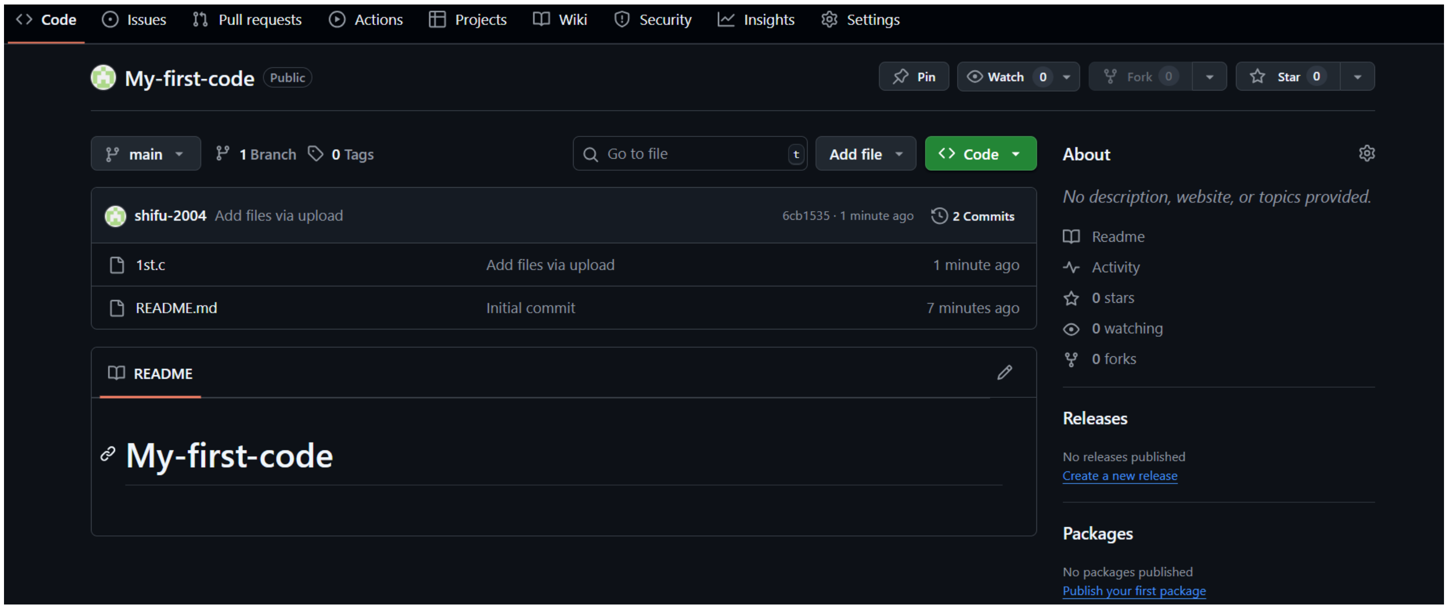
Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package



My-first-code Public

1 Branch 0 Tags

shifu-2004 Add files via upload 6cb1535 · 1 minute ago 2 Commits

1st.c	Add files via upload	1 minute ago
README.md	Initial commit	7 minutes ago

My-first-code

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

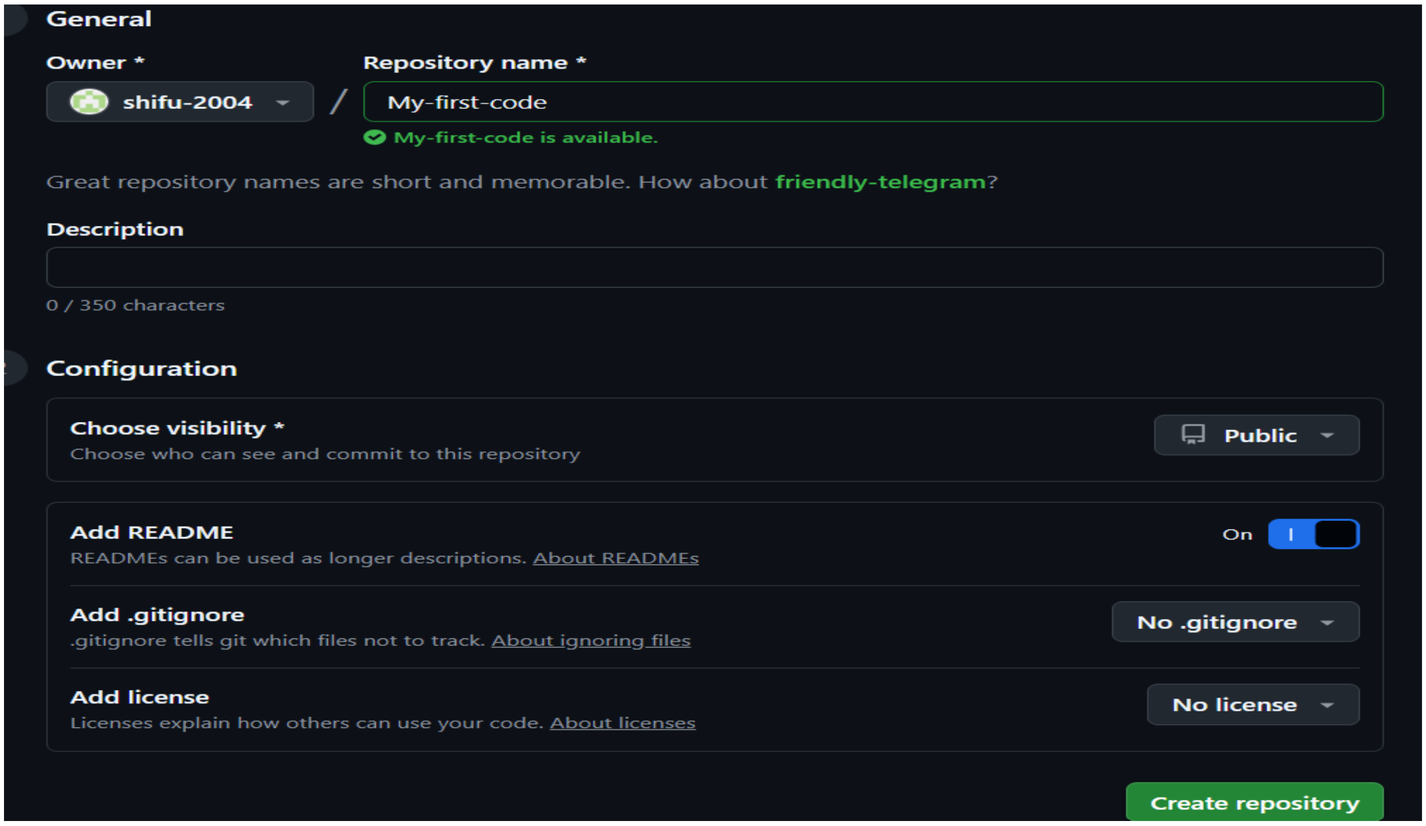
Packages

No packages published
Publish your first package

✓ Done — your code is now on GitHub!

12. Create a Github repository and document how to commit and push code changes.

- <u>Create Github Repository:</u>



General

Owner * shifu-2004 / Repository name * My-first-code

My-first-code is available.

Great repository names are short and memorable. How about friendly-telegram?

Description

0 / 350 characters

Configuration

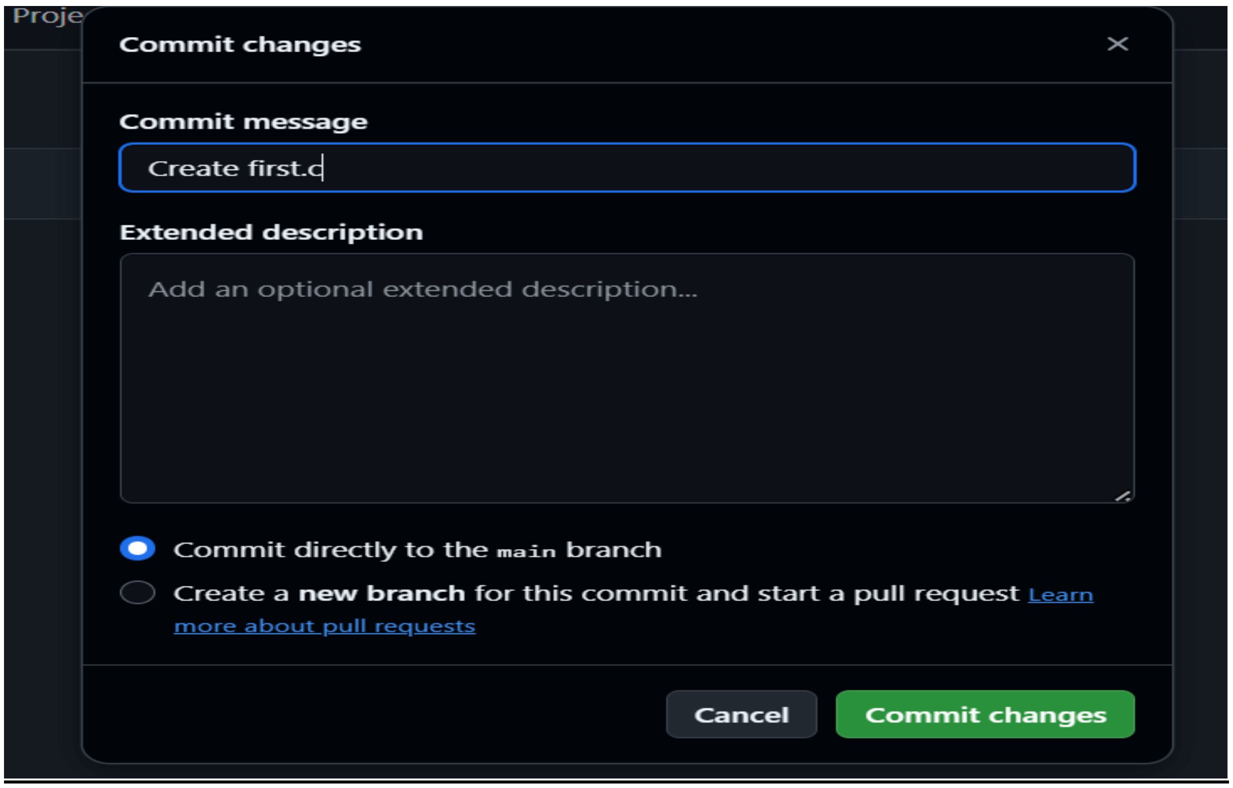
Choose visibility *
Choose who can see and commit to this repository — Public

Add README
READMEs can be used as longer descriptions. About READMEs — On

Add .gitignore
.gitignore tells git which files not to track. About ignoring files — No .gitignore

Add license
Licenses explain how others can use your code. About licenses — No license

- ***Commit and push code:***



Commit changes

Commit message

Create first.c

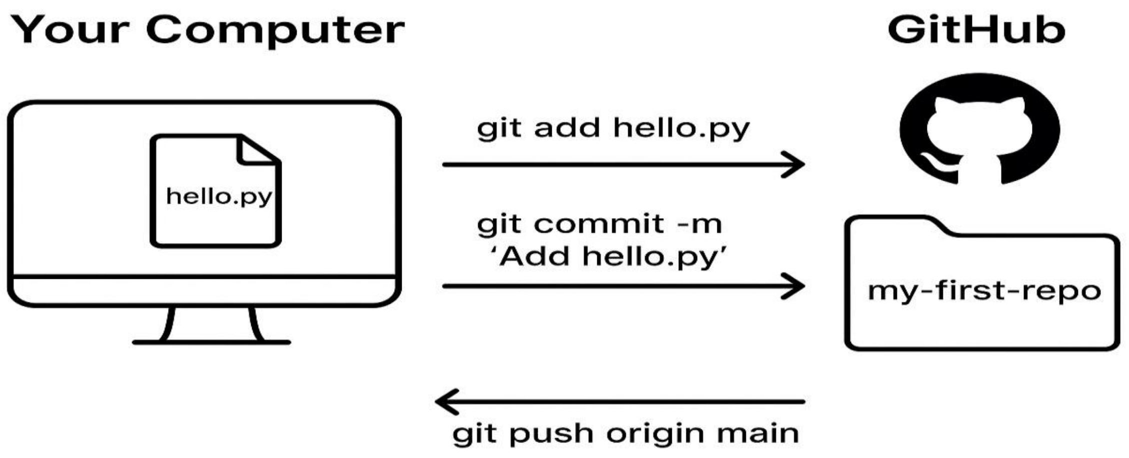
Extended description

Add an optional extended description...

[✓ Commit directly to the `main` branch]

Create a **new branch** for this commit and start a pull request Learn more about pull requests

Cancel | Commit changes



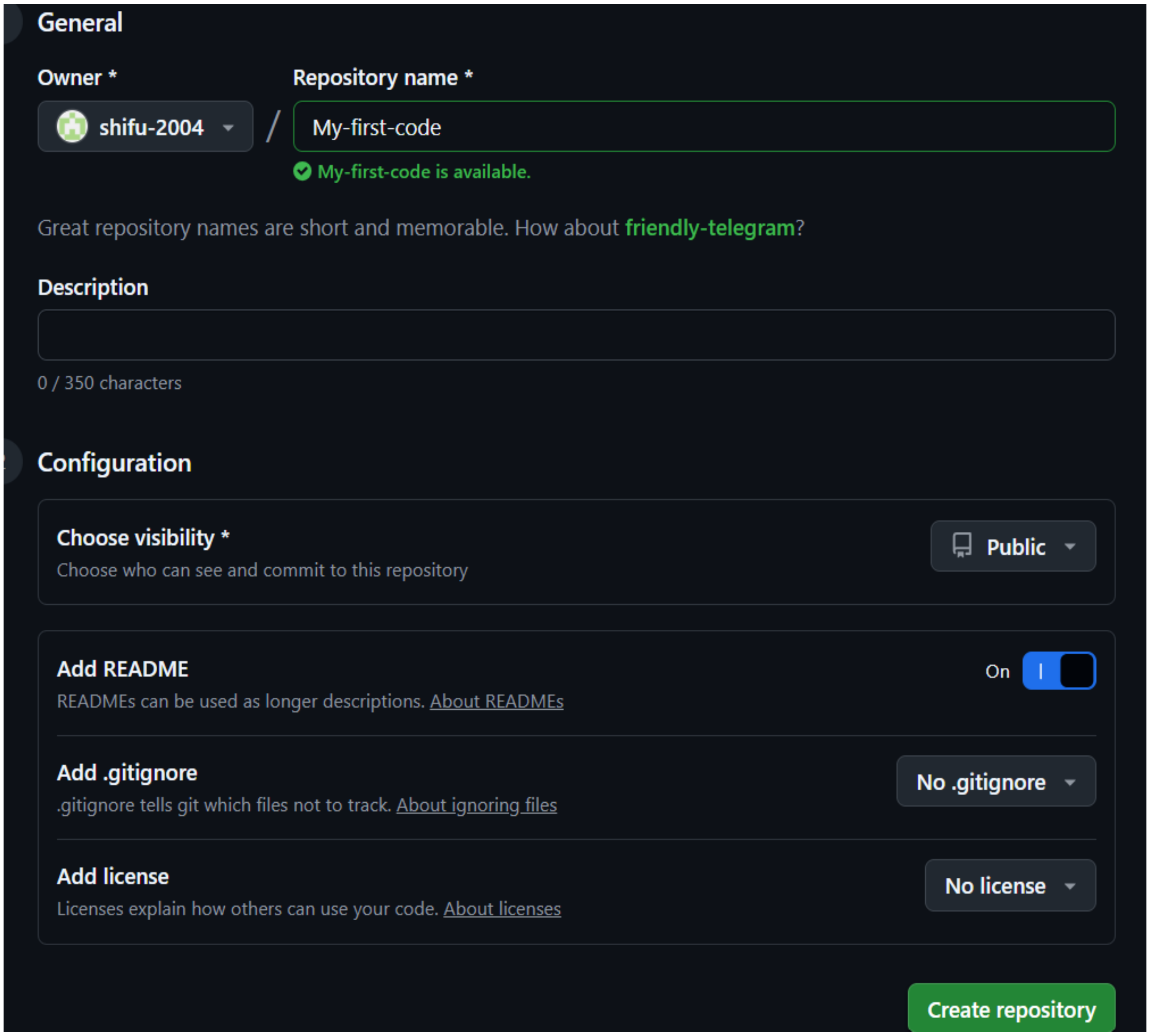
13. Create a student account on Github and collaborate on a small project with a classmate.

➢ Steps to Create a Student Account on GitHub & Collaborate

1. **Create Account**

- Go to github.com.
- Click **Sign up** → Enter **username, email, password**.
- Verify your email.

2. **Create a Repository (Project Space)**



General

Owner * shifu-2004 / Repository name * My-first-code

My-first-code is available.

Great repository names are short and memorable. How about **friendly-telegram**?

Description

0 / 350 characters

Configuration

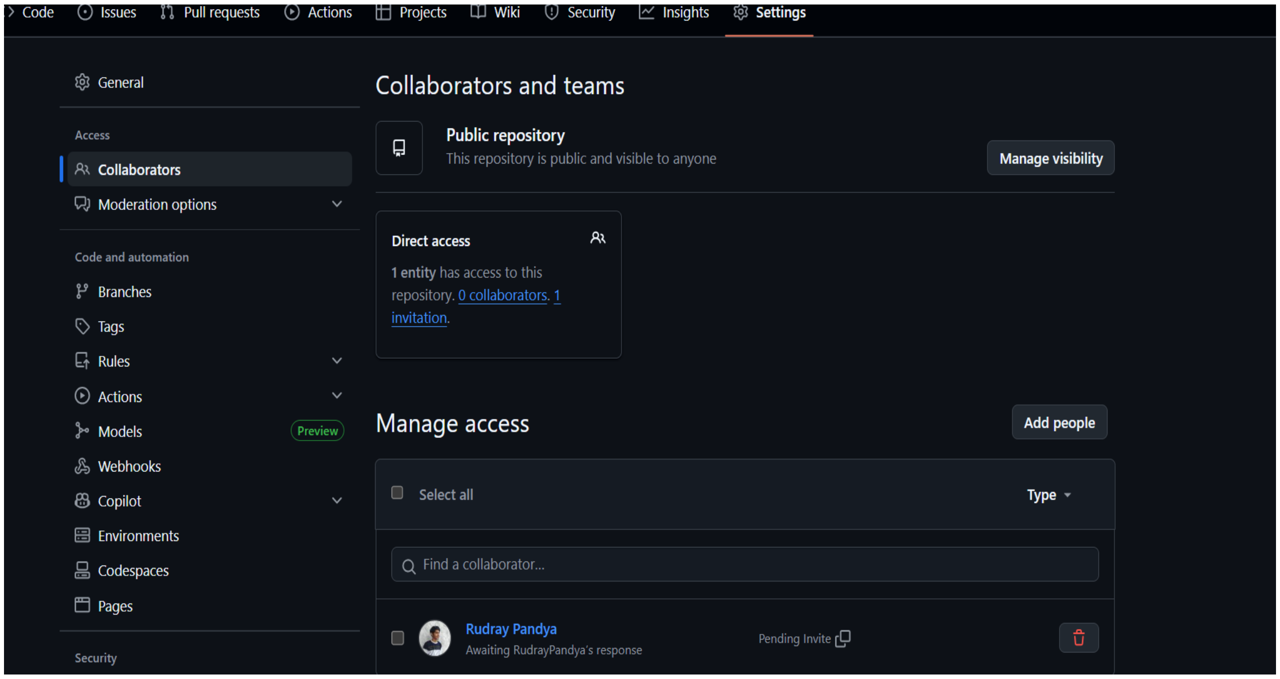
Choose visibility * — Public
Choose who can see and commit to this repository

Add README — On
READMEs can be used as longer descriptions. About READMEs

Add .gitignore — No .gitignore
.gitignore tells git which files not to track. About ignoring files

Add license — No license
Licenses explain how others can use your code. About licenses

3. **Invite Classmate**
   - Open your repository → **Settings → Collaborators**.
   - Click **Add collaborator** → Enter your classmate's GitHub username.
   - They accept the invite.



Collaborators and teams

Public repository
This repository is public and visible to anyone

Manage visibility

Direct access
1 entity has access to this repository. 0 collaborators. 1 invitation.

Manage access

Add people

Select all

Type

Find a collaborator...

Rudray Pandya
Awaiting RudrayPandya's response
Pending Invite

✓ Done! You and your classmate are now collaborating on the same GitHub project.

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

➢ System Software

(Manages and controls computer hardware and provides a platform for application software.)

- Windows 10 / 11 (Operating System)
- Linux Ubuntu (Operating System)
- macOS (if you use Apple devices)
- Device Drivers (e.g., printer driver, graphics driver)

➢ Application Software

(Programs designed for end users to perform specific tasks.)

- Microsoft Word (word processing)
- Google Chrome (web browsing)
- Zoom (video conferencing)
- VS Code (code editor)
- Spotify (music streaming)

- WhatsApp Desktop (messaging)

- ➢ Utility Software

(Helps maintain, analyze, and optimize computer performance.)

- Antivirus (e.g., Windows Defender, Avast)
- WinRAR / 7-Zip (file compression)
- CCleaner (system cleaning)
- Disk Management Tool (built into Windows)
- Backup Tool (e.g., Google Drive, OneDrive sync)

15. Follow a GIT tutorial to practice cloning, branching, and merging repositories.

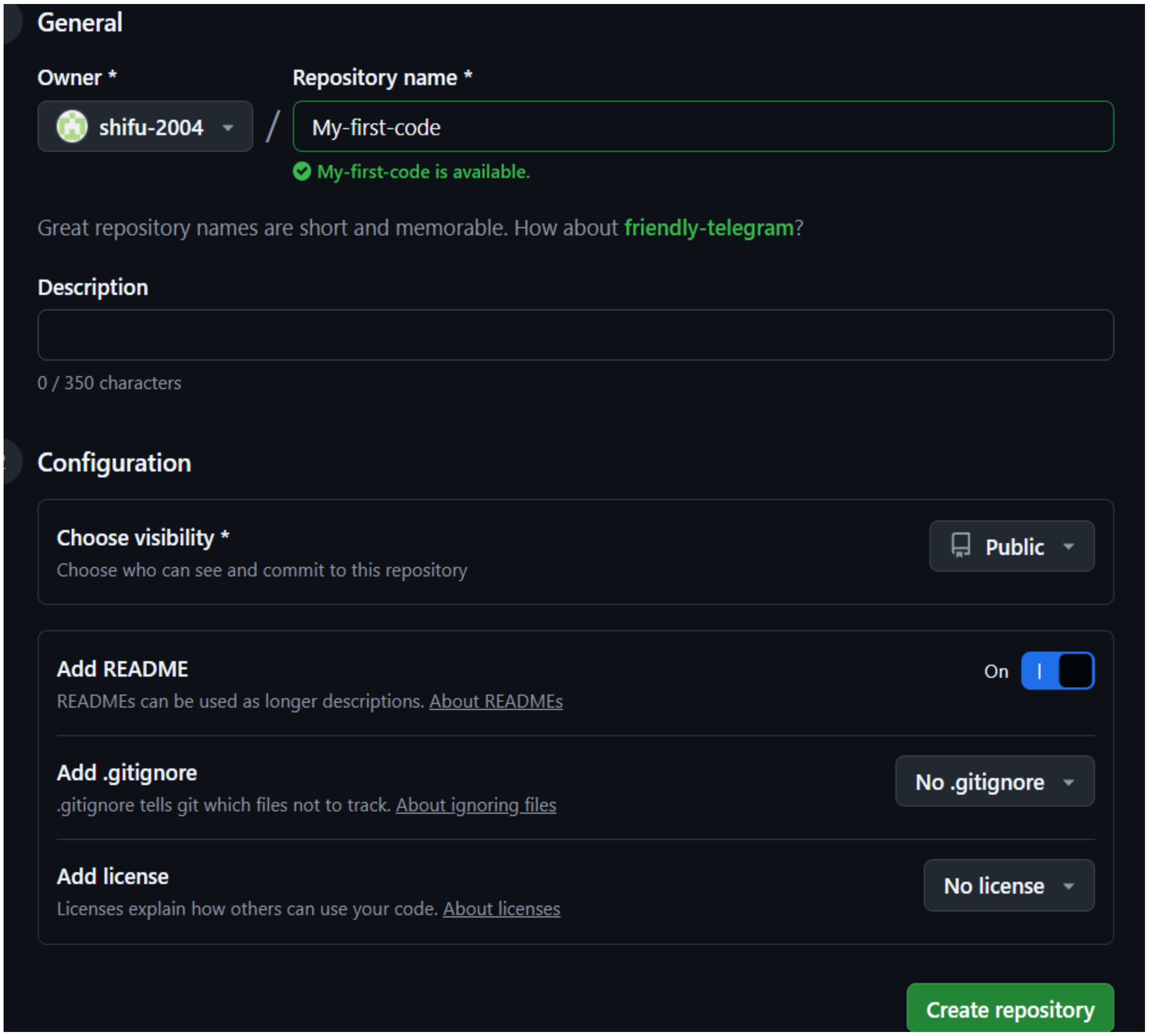
➢ Prerequisites

- Install Git:
  - **Windows**: Install from git-scm.com
  - **Mac**: Comes pre-installed
  - **Linux**:
  - sudo apt install git
- Have a GitHub account.

---

2. Create a Repository on GitHub

1. Go to GitHub → Click **New**.
2. Name it git-practice.
3. Check **Add a README**.
4. Click **Create repository**.


General

Owner * shifu-2004 / Repository name * My-first-code

My-first-code is available.

Great repository names are short and memorable. How about friendly-telegram?

Description

0 / 350 characters

Configuration

Choose visibility *
Choose who can see and commit to this repository — Public

Add README
READMEs can be used as longer descriptions. About READMEs — On

Add .gitignore
.gitignore tells git which files not to track. About ignoring files — No .gitignore

Add license
Licenses explain how others can use your code. About licenses — No license

Create repository

---

3. Clone the Repository

This downloads the repo from GitHub to your computer.

```
#  Go to a folder where you want the project
cd Desktop

# Clone from GitHub
git clone https://github.com/<your-username>/git-practice.git

# Enter the repo folder
cd git-practice
```

---

4. Create and Switch to a New Branch

Branches let you work separately without changing the main branch.

```
# Create and switch in one command
git checkout -b feature1
```

---

5. Make Changes in the Branch

Example: Create a file

```
echo "This is feature 1" > feature1.txt
```

Stage and commit changes:

```
git add feature1.txt
git commit -m "Added feature1 file"
```

---

6. Switch Back to Main and Merge

```
git checkout main
git merge feature1
```

---

7. Push Changes to GitHub

```
git push origin main
```

---

8. (Optional) Delete the Branch

```
git branch -d feature1
git push origin --delete feature1
```

16. Write a report on the various types of application software and how they improve productivity.

- ➢ <u>Types of Application Software</u>

o Word Processing Software

- Examples**:** Microsoft Word, Google Docs, LibreOffice Writer
- Purpose**:** Create, edit, and format text documents.

- Productivity Benefits:
    - Speeds up document creation with templates.
    - Enables easy editing, formatting, and sharing.
    - Provides spelling and grammar checks to reduce errors.

- Spreadsheet Software

- Examples**:** Microsoft Excel, Google Sheets
- Purpose**:** Organize, calculate, and analyze numerical data.
- Productivity Benefits:
    - Automates calculations with formulas.
    - Enables data visualization through charts and graphs.
    - Facilitates financial planning, reporting, and decision-making.

- Presentation Software

- Examples**:** Microsoft PowerPoint, Google Slides, Canva
- Purpose**:** Create visual presentations for meetings, lectures, and events.
- Productivity Benefits**:**
    - Enhances communication with visuals and animations.
    - Saves time with pre-designed templates.
    - Supports collaboration for team presentations.

- Database Management Software (DBMS)

- Examples**:** Microsoft Access, MySQL, Oracle Database
- Purpose**:** Store, manage, and retrieve structured data.
- Productivity Benefits**:**
    - Centralizes information for quick access.
    - Reduces duplication and improves data accuracy.
    - Supports large-scale business operations efficiently.

- Communication Software

- Examples**:** Microsoft Teams, Slack, Zoom, WhatsApp
- Purpose**:** Facilitate real-time communication and collaboration.
- Productivity Benefits**:**
    - Reduces delays in information sharing.

    - Enables remote work and global collaboration.
    - Integrates chat, voice, and video in one platform.

- Graphics and Multimedia Software

- Examples**:** Adobe Photoshop, CorelDRAW, Canva, Audacity
- Purpose**:** Create and edit images, videos, and audio.
- Productivity Benefits**:**
    - Allows quick production of professional designs.
    - Improves brand communication with high-quality visuals.
    - Supports creative industries and marketing teams.

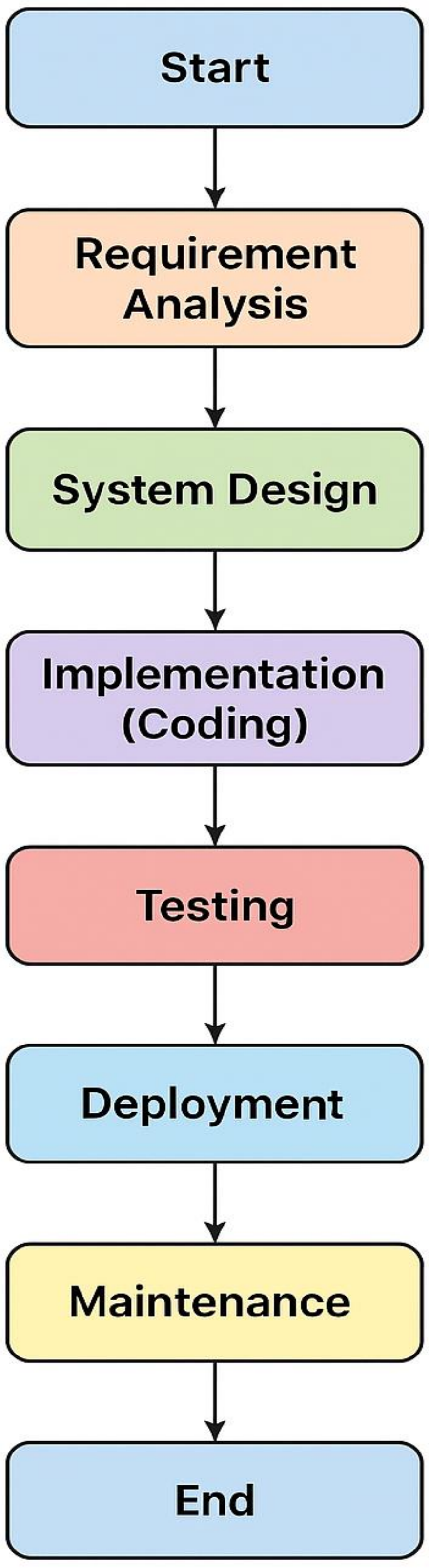
- Project Management Software

- Examples**:** Trello, Asana, Microsoft Project, Notion
- Purpose**:** Plan, track, and manage projects.
- Productivity Benefits**:**
    - Improves task organization and scheduling.
    - Increases accountability through progress tracking.
    - Reduces project delays and resource waste.

- Specialized/Custom Software

- Examples**:** AutoCAD (engineering), Tally (accounting), MATLAB (scientific computing)
- Purpose**:** Perform niche tasks for specific industries.
- Productivity Benefits:
    - Automates industry-specific processes.
    - Reduces the need for manual calculations and repetitive work.
    - Increases accuracy and compliance with industry standards.

17. Create a flowchart representing the Software Development Life Cycle (SDLC).



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

- Library Management System – Requirements:

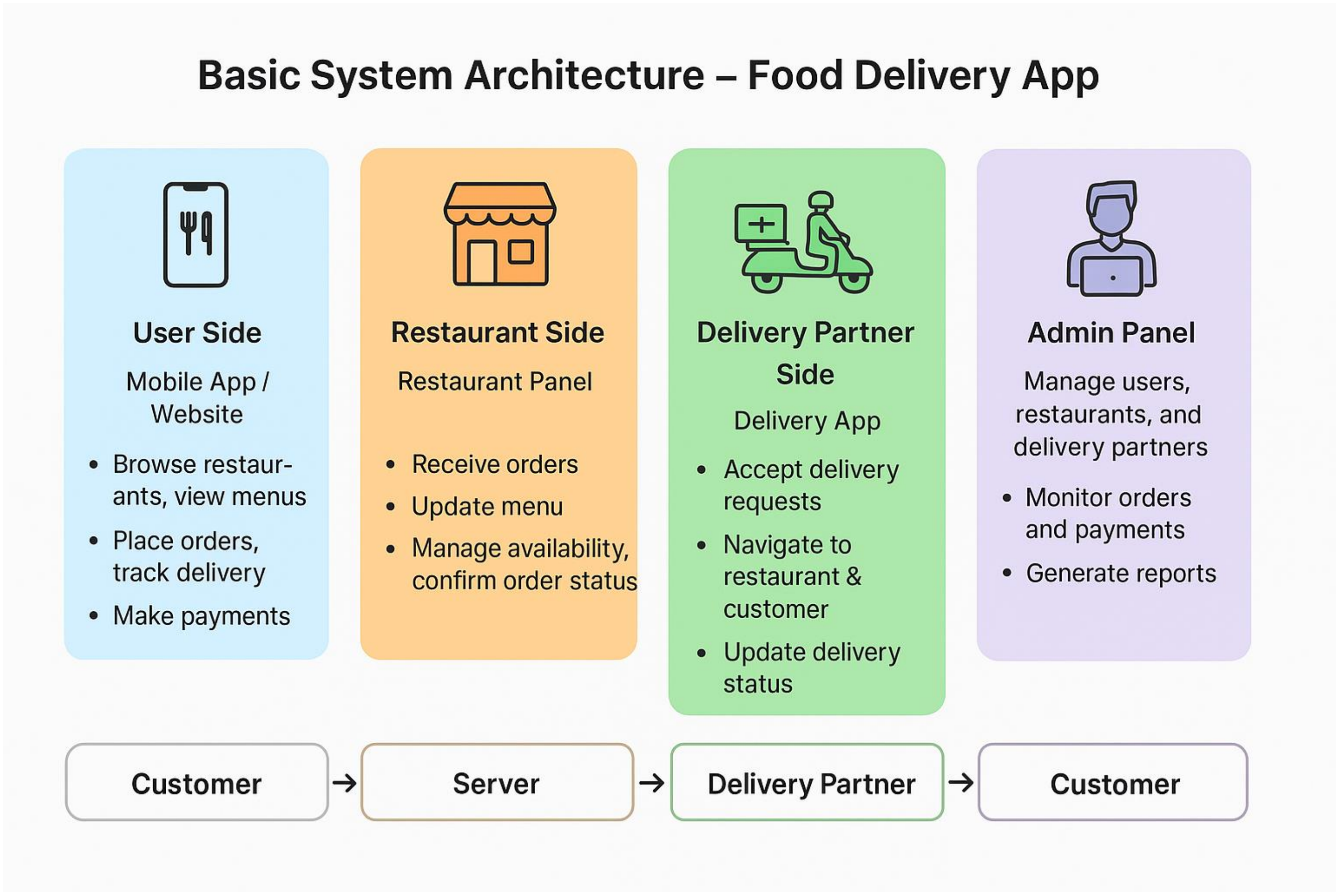
i. <u>Add, edit, delete books</u> with details like title, author, ISBN, category.
ii. <u>Register members</u> with name, contact and member ship ID.
iii. <u>Issue books</u> to members and record issue/return dates.
iv. <u>Return books and</u> update availability status.
v. <u>Calculate fines</u> for late returns.
vi. <u>Search books</u> by title, author, or category.
vii. <u>View reports</u> – available books, issued books, overdue books, member history.
viii. <u>User roles</u> – Librarian (full access) and Member (view/search only).
ix. <u>Secure login</u> for librarians.
x. <u>Simple interface</u> for easy use.

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

- Functional Analysis – Online Shopping System

1. <u>User Functions</u>
   - Create account, login/logout
   - Search & browse products
   - Add to cart / wishlist
   - Checkout & make payment
   - Track orders
   - Give reviews
2. <u>Admin Functions</u>
   - Manage users
   - Add/update/remove products
   - Manage orders
   - Check payments & reports
3. <u>System Functions</u>
   - Secure login
   - Payment gateway
   - Send notifications
   - Protect data

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


Basic System Architecture – Food Delivery App
User Side
Mobile App / Website
Browse restaurants, view menus
Place orders, track delivery
Make payments
Restaurant Side
Restaurant Panel
Receive orders
Update menu
Manage availability, confirm order status
Delivery Partner Side
Delivery App
Accept delivery requests
Navigate to restaurant & customer
Update delivery status
Admin Panel
Manage users, restaurants, and delivery partners
Monitor orders and payments
Generate reports
Customer
Server
Delivery Partner
Customer

21. Develop test cases for a simple calculator program.

Test Case ID	Description	Input	Expected Output	Remarks
TC01	Add two positive numbers	5 + 3	8	Pass if correct
TC02	Add positive and negative number	10 + (-4)	6	Pass if correct
TC03	Add two negative numbers	-7 + (-2)	-9	Pass if correct
TC04	Subtract smaller from larger	9 - 4	5	Pass if correct
TC05	Subtract larger from smaller	4 - 9	-5	Pass if correct
TC06	Multiply two positive numbers	6 × 3	18	Pass if correct
TC07	Multiply positive and negative	-8 × 5	-40	Pass if correct
TC08	Multiply two negative numbers	-4 × -7	28	Pass if correct
TC09	Divide two positive numbers	8 ÷ 2	4	Pass if correct
TC10	Divide positive by negative	10 ÷ -2	-5	Pass if correct

Test Case ID	Description	Input	Expected Output	Remarks
TC11	Divide negative by negative	-12 ÷ -3	4	Pass if correct
TC12	Division by zero	5 ÷ 0	Error/Infinity	Must handle gracefully
TC13	Zero divided by number	0 ÷ 7	0	Pass if correct
TC14	Large number addition	999999 + 1	1000000	Pass if correct
TC15	Decimal addition	2.5 + 1.2	3.7	Pass if correct
TC16	Decimal multiplication	1.5 × 2	3.0	Pass if correct

22. Document a real-world case where a software application required critical maintenance.

Real-World Case of Critical Software Maintenance

➢ **Case:** The WannaCry Ransomware Attack (2017)

➢ **Background:**

- In May 2017, a global ransomware attack called **WannaCry** spread rapidly.
- It exploited a vulnerability in Microsoft Windows operating systems.

➢ **Problem:**

- Computers in more than **150 countries** were affected.
- Important organizations like the **UK's National Health Service (NHS)**, banks, and transport systems were disrupted.
- Hospitals could not access patient data, delaying treatments.

➢ **Critical Maintenance Action:**

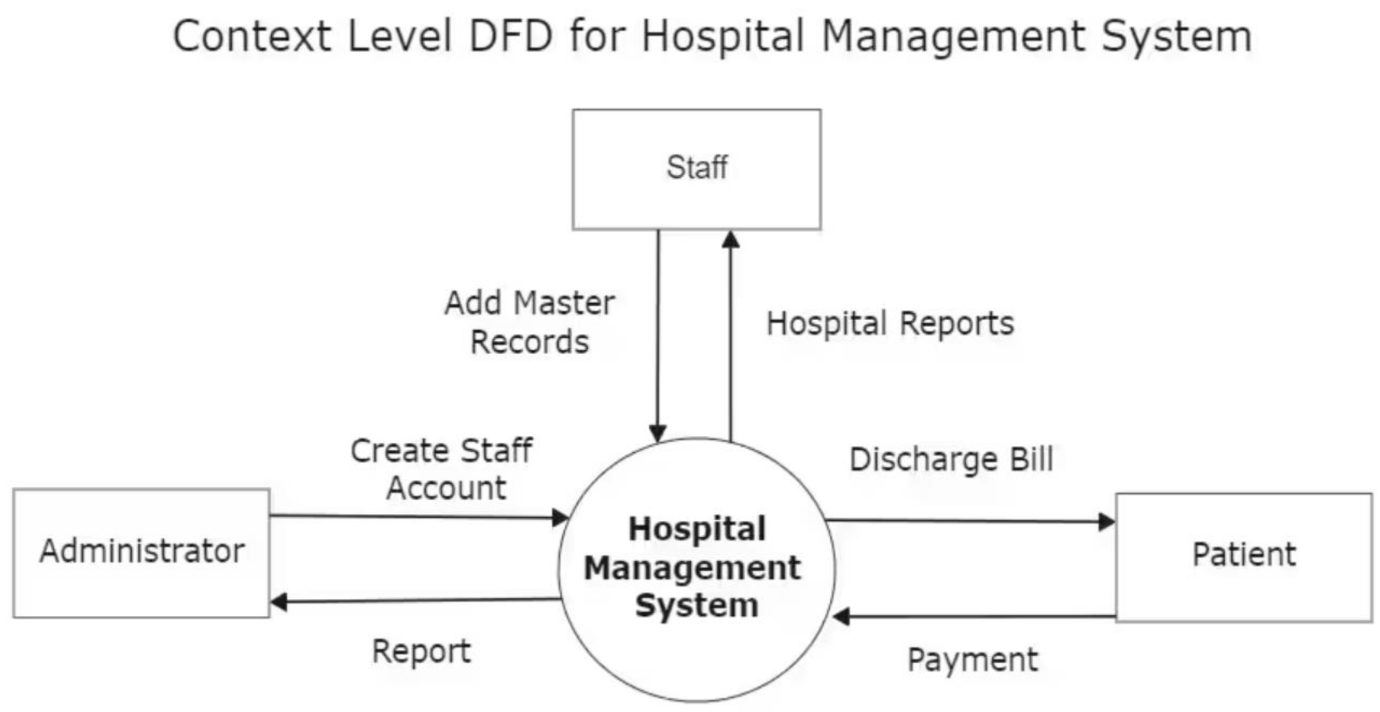
- Microsoft released an **emergency security patch** to fix the vulnerability.
- Even unsupported versions of Windows (like Windows XP) received urgent updates.
- IT teams worldwide had to **quickly install the patches** and remove the malware.

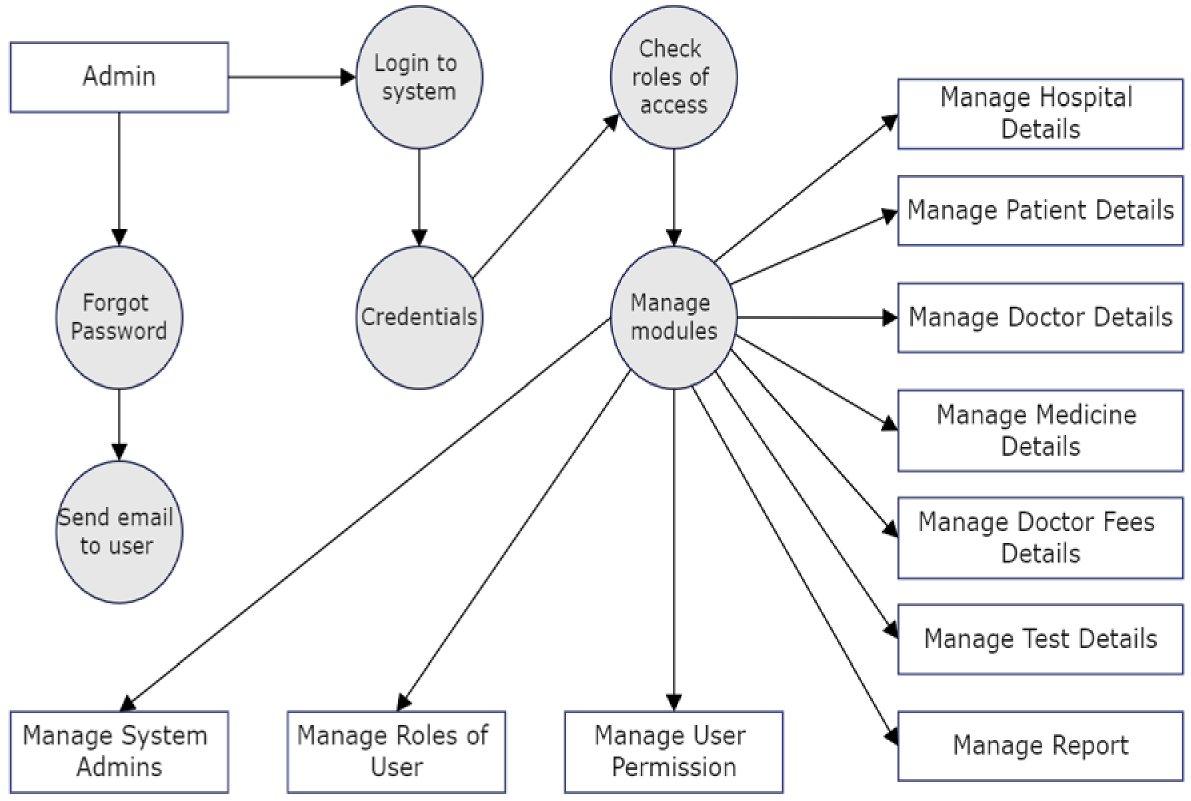
➢ **Outcome:**

- Systems were restored after applying patches.
- The attack highlighted the importance of **regular software updates and maintenance** to prevent such failures.

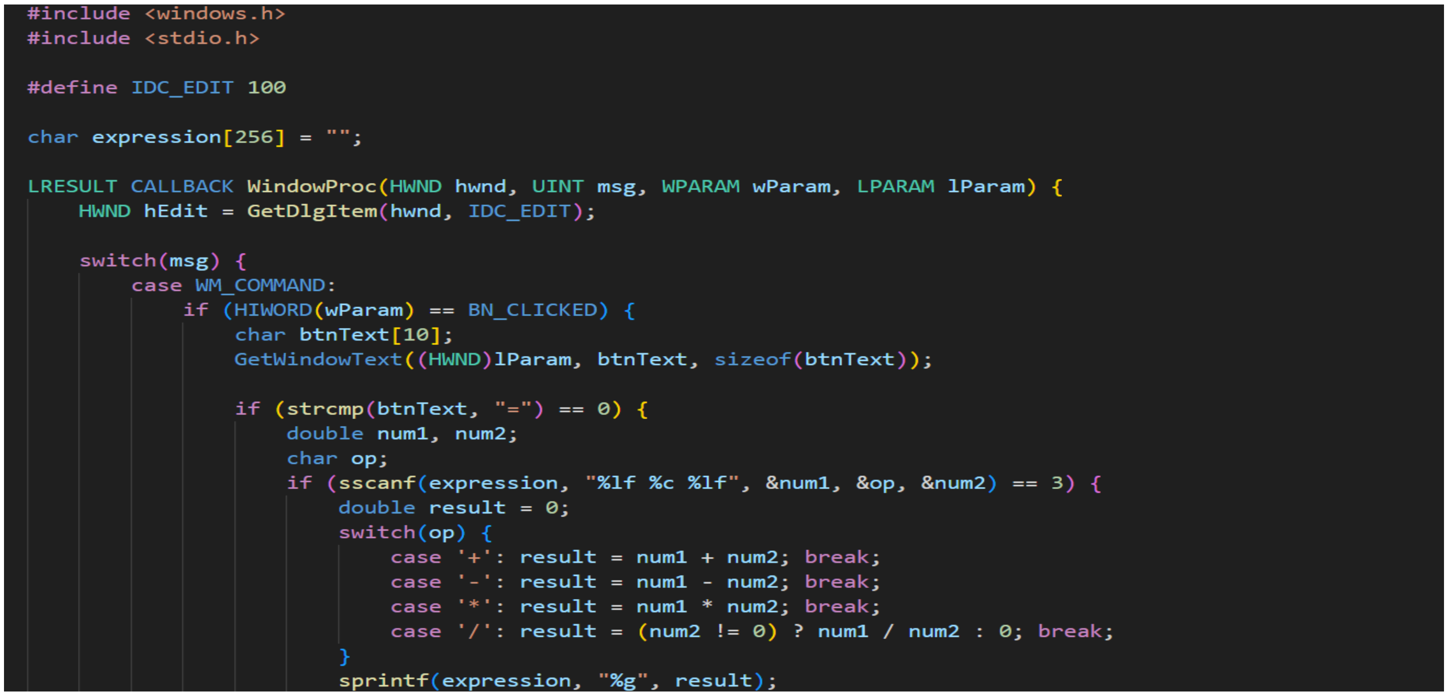
- **Conclusion:** This case shows that **critical maintenance** (like urgent security patching) is necessary to protect software systems from major failures and cyberattacks.

23. Create a DFD for a hospital management system.

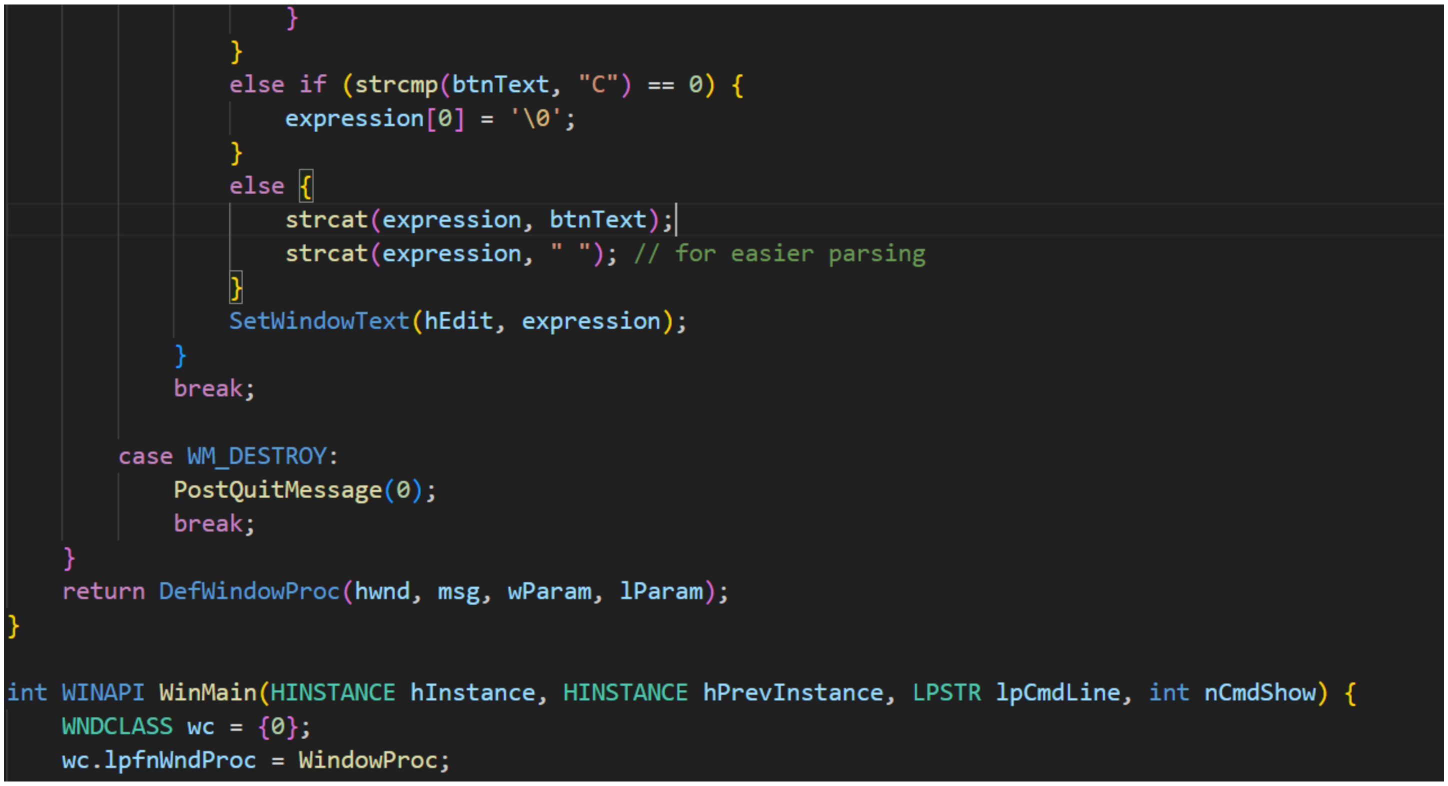




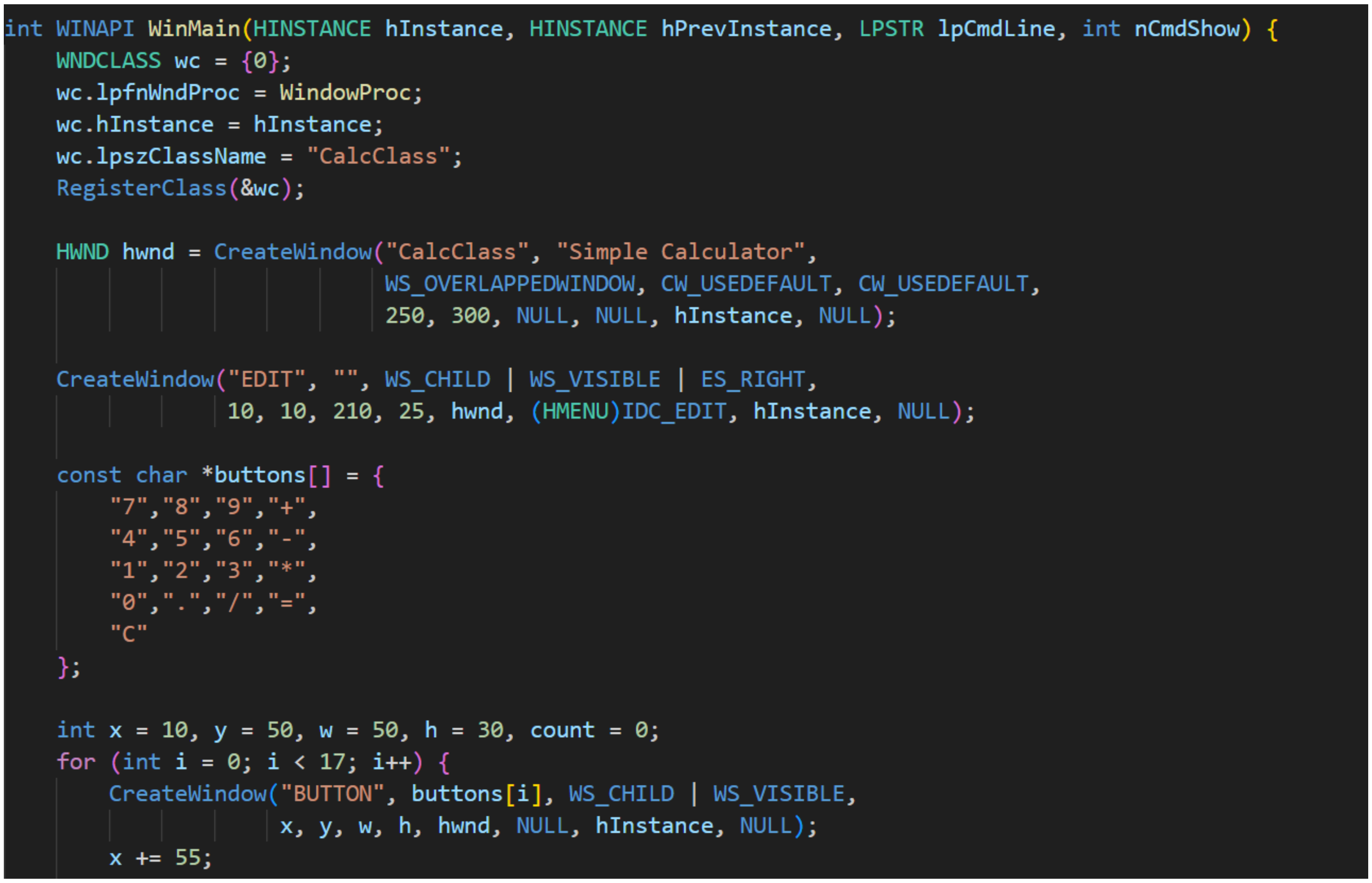
24. Build a simple desktop calculator application using a GUI library.



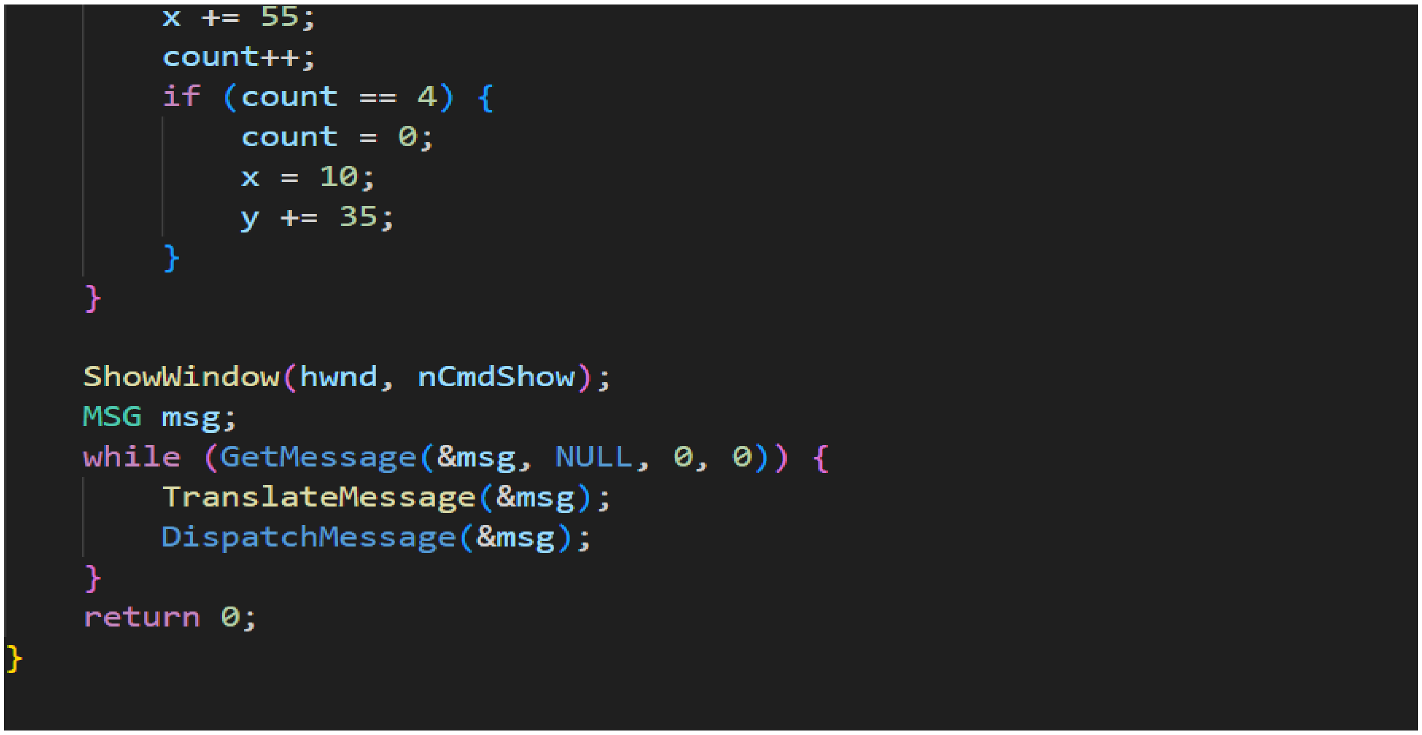
```
#include <windows.h>
#include <stdio.h>

#define IDC_EDIT 100

char expression[256] = "";

LRESULT CALLBACK WindowProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {
    HWND hEdit = GetDlgItem(hwnd, IDC_EDIT);

    switch(msg) {
        case WM_COMMAND:
            if (HIWORD(wParam) == BN_CLICKED) {
                char btnText[10];
                GetWindowText((HWND)lParam, btnText, sizeof(btnText));

                if (strcmp(btnText, "=") == 0) {
                    double num1, num2;
                    char op;
                    if (sscanf(expression, "%lf %c %lf", &num1, &op, &num2) == 3) {
                        double result = 0;
                        switch(op) {
                            case '+': result = num1 + num2; break;
                            case '-': result = num1 - num2; break;
                            case '*': result = num1 * num2; break;
                            case '/': result = (num2 != 0) ? num1 / num2 : 0; break;
                        }
                        sprintf(expression, "%g", result);
```



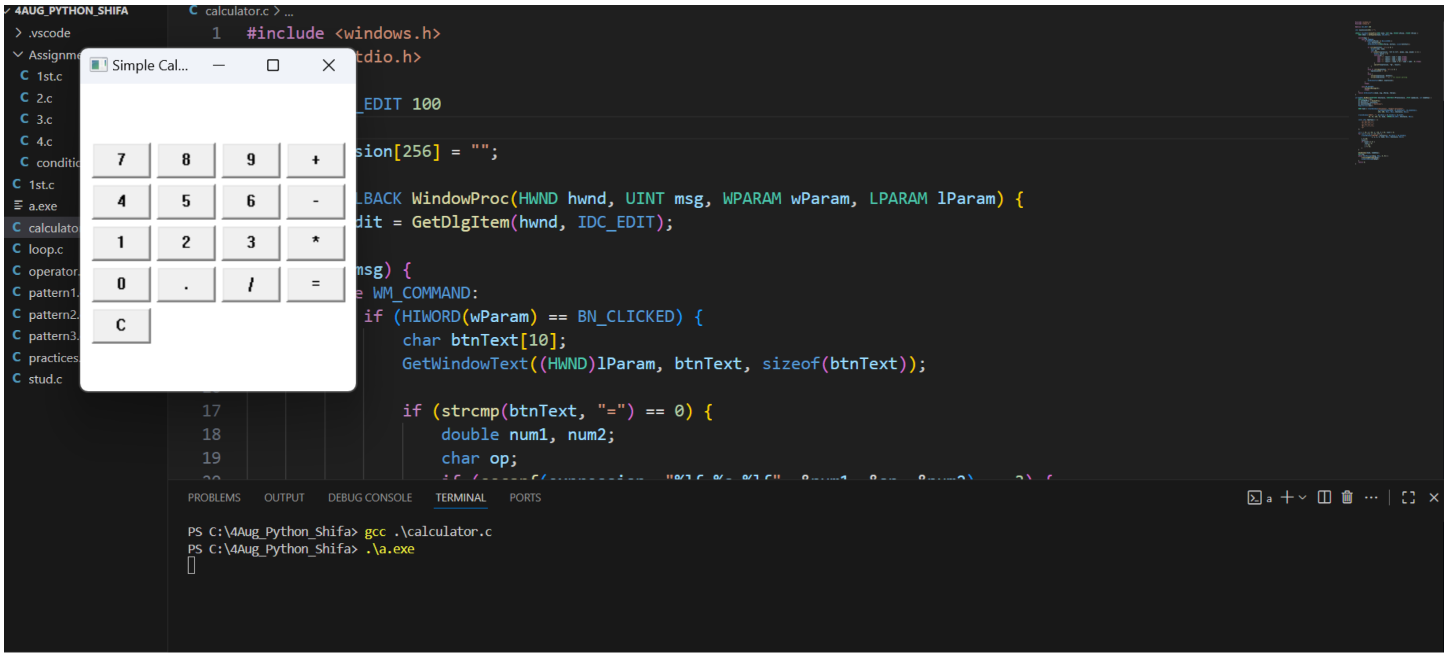
```c
                }
            }
            else if (strcmp(btnText, "C") == 0) {
                expression[0] = '\0';
            }
            else {
                strcat(expression, btnText);
                strcat(expression, " "); // for easier parsing
            }
            SetWindowText(hEdit, expression);
        }
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASS wc = {0};
    wc.lpfnWndProc = WindowProc;
```



```c
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASS wc = {0};
    wc.lpfnWndProc = WindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = "CalcClass";
    RegisterClass(&wc);

    HWND hwnd = CreateWindow("CalcClass", "Simple Calculator",
                             WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
                             250, 300, NULL, NULL, hInstance, NULL);

    CreateWindow("EDIT", "", WS_CHILD | WS_VISIBLE | ES_RIGHT,
                 10, 10, 210, 25, hwnd, (HMENU)IDC_EDIT, hInstance, NULL);

    const char *buttons[] = {
        "7","8","9","+",
        "4","5","6","-",
        "1","2","3","*",
        "0",".","/","=",
        "C"
    };

    int x = 10, y = 50, w = 50, h = 30, count = 0;
    for (int i = 0; i < 17; i++) {
        CreateWindow("BUTTON", buttons[i], WS_CHILD | WS_VISIBLE,
                     x, y, w, h, hwnd, NULL, hInstance, NULL);
        x += 55;
```



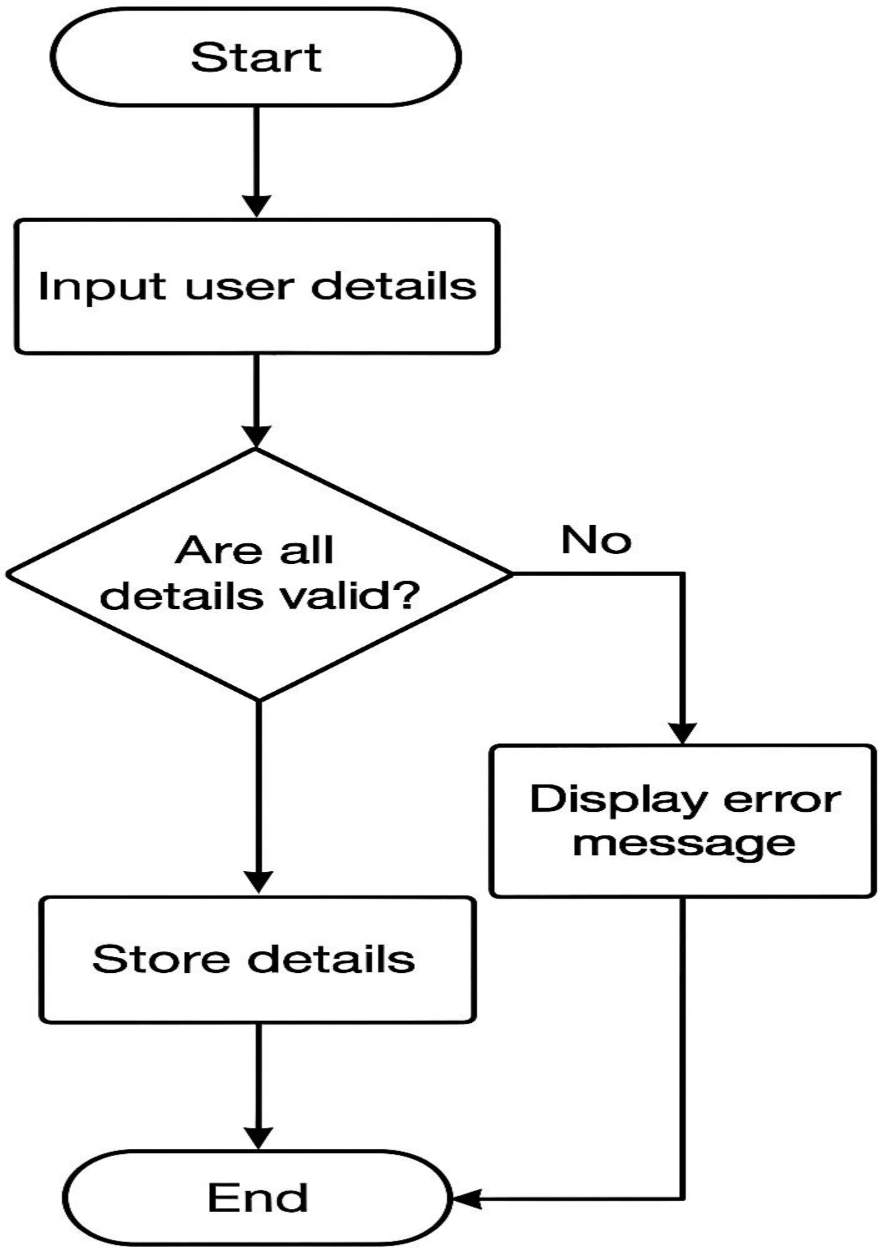
```
        x += 55;
        count++;
        if (count == 4) {
            count = 0;
            x = 10;
            y += 35;
        }
    }

    ShowWindow(hwnd, nCmdShow);
    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}
```



```
#include <windows.h>
tdio.h>

_EDIT 100

sion[256] = "";

LBACK WindowProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {
dit = GetDlgItem(hwnd, IDC_EDIT);

msg) {
e WM_COMMAND:
 if (HIWORD(wParam) == BN_CLICKED) {
     char btnText[10];
     GetWindowText((HWND)lParam, btnText, sizeof(btnText));

     if (strcmp(btnText, "=") == 0) {
         double num1, num2;
         char op;
```

```
PS C:\4Aug_Python_Shifa> gcc .\calculator.c
PS C:\4Aug_Python_Shifa> .\a.exe
```

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


Start
Input user details
Are all details valid?
No
Display error message
Store details
End