**Create diabetes prediction system using python**

**Phase 1: Problem Definition and Design Thinking**

**Problem Definition:** The problem.Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Your body breaks down most of the food you eat into sugar (glucose) and releases it into your bloodstream. When your blood sugar goes up, it signals your pancreas to release insulin.

**Design Thinking:**

1. Functionality: With diabetes, your body either doesn't make enough insulin or can't use it as well as it should. Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Your body breaks down most of the food you eat into sugar (glucose) and releases it into your bloodstream.
2. User Interface: The kernel of the GUI is a registry database located on a server accessible to both patients and their providers. The patient-GUI includes the resources of the glucose predicting engine and user-friendly, intuitive means to enter body weight and all home-monitored blood glucose levels. In response, means to modify medication dosages (dosing decision support) and modify planned diet and physical activity (lifestyle decision support) are afforded the user. Each action is animated so that the patient can visually see the impact of his or her changes on predicted glucose outcomes and the pending risks of hypoglycemia.
3. Natural Language Processing (NLP): Real-world evidence research plays an increasingly important role in diabetes care. However, a large fraction of real-world data are “locked” in narrative format. Natural language processing (NLP) technology offers a solution for analysis of narrative electronic data.
4. Responses: With diabetes, your body doesn't make enough insulin or can't use it as well as it should. When there isn't enough insulin or cells stop responding to insulin, too much blood sugar stays in your bloodstream.
5. Integration: Integrating care across disciplines and organisations around the needs of the person with diabetes has been proposed as an approach that could improve care while reducing cost- but has it and can it? Integrated Diabetes Care- A Multidisciplinary Approach collates evidence of worldwide approaches to both horizontal integration (across disciplines) and vertical integration (across organizations) in diabetes care and describe what was done, what worked and what appeared to be the barriers to achieving the goals of the programmes. Evidence is sought from groups who have developed different approaches to integrating diabetes care in different health systems (eg insurance vs tax payer funded, single vs multiple organization, published vs unpublished). A final chapter brings the evidence together for a final discussion about what seems to work and what does not.
6. Testing and Improvement: Improvement of glucose levels into the normal range can occur in some people living with diabetes, either spontaneously or after medical interventions, and in some cases can persist after withdrawal of glucose-lowering pharmacotherapy.

**Steps Involved:**

1. Design: Diabetes occurs in four stages: Insulin resistance, prediabetes, type 2 diabetes, and type 2 diabetes with vascular complications. You are at higher risk for these conditions if you are older than 45, have close biological relatives with diabetes, are physically inactive, or have extra weight.
2. Applicability: The “stepwise approach”' is usually adopted to manage glycemic control in type 2 diabetes. On diagnosis, lifestyle modification is initiated, followed by treatment with a single oral antidiabetic agent, which is often up-titrated to maximal recommended doses before combination therapy is introduced.
3. Technology: Diabetes is a chronic, metabolic disease characterized by elevated levels of blood glucose (or blood sugar), which leads over time to serious damage to the heart, blood vessels, eyes, kidneys and nerves.
4. Coding: E08, Diabetes mellitus due to underlying condition. E09, Drug or chemical induced diabetes mellitus. E10, Type 1 diabetes mellitus. E11, Type 2 diabetes mellitus.
5. Architecture: The algorithms like K nearest neighbour, Logistic Regression, Random forest, Support vector machine and Decision tree are used. The accuracy of the model using each of the algorithms is calculated. Then the one with a good accuracy is taken as the model for predicting the diabetes.
6. Transformation: Some people who don't know much about type 1 or type 2 diabetes might question whether it's transferrable from person-to-person through sexual contact, saliva, or blood. Science has confirmed that diabetes is a non-communicable disease, so it's not contagious — nor is a diagnosis your fault.
7. Real-World Analogy: Even children with type 1 diabetes can understand the analogy of a “diabetic car.” A type 1 diabetic car has no insulin (key) to open the gas tank so the tank stays empty, with no fuel for energy. Without being able to open the lid, a meal is eaten but no gas enters the tank and the car doesn't function correctly.

**INSTRUCTIONS:**

**Step 1: Download and Install Git**

1. Visit the official Git website: [https://git-scm.com/](https://git-scm.com/" \t "[object Object])

2. Download the appropriate version of Git for your operating system (Windows, macOS, or Linux).

3. Run the installer and follow the on-screen instructions to complete the installation.

4. Open a terminal or command prompt and verify the installation by typing git−−version.

**Step 2: Download and Install Visual Studio Code**

1. Go to the Visual Studio Code website: [https://code.visualstudio.com/](https://code.visualstudio.com/" \t "[object Object])

2. Download the installer for your operating system (Windows, macOS, or Linux).

3. Run the installer and follow the installation prompts.

4. Launch Visual Studio Code.

**Step 3: Create a GitHub Account**

1. Open a web browser and go to [https://github.com/](https://github.com/" \t "[object Object])

2. Click on the "Sign up" button.

3. Follow the registration process, providing your username, email address, and password.

4. Complete the verification process if prompted.

**Step 4: Create a GitHub Repository**

1. Log in to your GitHub account.

2. Click on your profile icon in the upper right corner and select "Your repositories" from the dropdown menu.

3. On the "Repositories" page, click the green "New" button.

4. Fill in the required information for your new repository, including the repository name, description, visibility, and other settings.

5. Optionally, you can choose to initialize the repository with a README file or add a .gitignore file for your specific project.

6. Click the green "Create repository" button to create your GitHub repository.

**Step 5: Create a Local Folder**

1. Minimize any open windows on your computer to see your desktop.

2. Right-click on an empty area of your desktop.

3. Hover over "New" in the context menu.

4. Click on "Folder" to create a new folder.

5. Give your folder a meaningful name, like "MyProject."

**Step 6: Open the Folder in Visual Studio Code**

1. Launch Visual Studio Code.

2. Click on "File" in the top-left corner.

3. Select "Open Folder" from the dropdown menu.

4. Browse to your desktop and select the folder you created in Step 5 (e.g., "MyProject").

5. Click the "Open" button to open the folder in Visual Studio Code.

**Step 7: Clone Your GitHub Repository**

1. In Visual Studio Code, open the integrated terminal by clicking on "View" in the top menu and selecting "Terminal" or using the keyboard shortcut (**Ctrl+** on Windows/Linux or **Cmd+** on macOS).

2. Use the **git clone** command to clone your GitHub repository by pasting the HTTPS URL of your repository. Replace **repository\_url** with the actual URL.

   git clone <repository\_url>

3. Navigate to the newly created repository folder using the cd command:

   cd <repository\_name>

**Step 8: Check Git Status**

1. To check the status of your local repository, enter the following command:

   git status

**Step 9: Modify the README File**

1. Open the README file in your repository folder using Visual Studio Code.

2. Make the desired modifications to the README file.

**Step 10: Check Git Status Again**

1. Return to the terminal in Visual Studio Code.

2. Use the gitstatus command again to see the changes you made:

   git status

**Step 11: Add Modifications to Staging Area**

1. To stage your changes for a commit, use the gitadd command:

   git add README.md

**Step 12: Commit Your Changes**

1. Commit your staged changes with a descriptive message:

   git commit -m "Updated README file"

**Step 13: Push Changes to GitHub**

1. Push your committed changes to your GitHub repository:

   git push

**Step 14: Create a New Branch**

1. To create a new branch, use the gitbranch command followed by the desired branch name:

   git branch branch\_name

**Step 15: Switch to the New Branch**

1. To switch to the newly created branch, use the gitcheckout command:

   git checkout branch\_name

**Step 16: Check Your Current Branch**

1. To confirm the branch you're currently working on, use the gitbranch command:

  git branch

**Report:**

All the above instructions are installed and executed successfully.

**Project by:**

**Name: k. Brundha**

**Dept:** CSE III YEAR

**Reg No:** 420121104014

**College code:** 4201

**Group: APGVB\_5**