**Medical Condition Checker Web Application Documentation1. Overview:**This documentation provides an overview of a web application called the Medical Condition Checker. The application allows users to enter their personal information, including name, gender, and blood group, and provides them with information about the medical conditions associated with their blood group. The application is built using Python with the Flask web framework for the backend, HTML for the frontend interface, and CSS for styling.**2. Application Components:**The Medical Condition Checker web application consists of the following components:**HTML Form (index.html):** This component contains an HTML form where users can enter their name, select their gender, and choose their blood group.**CSS Styles (styles.css):** This component provides styling for the HTML elements, ensuring a visually appealing and user-friendly interface.**Flask Application (route.py):** This component implements the server-side logic of the application using the Flask web framework. It handles form submissions, calculates the associated medical condition based on the user's blood group, and stores the entered data in CSV and JSON files.**CSV and JSON Files (form\_data.csv, form\_data.json):** These files store the data entered by users, including their name, gender, and blood group, for future reference.**3. Application Workflow:**The workflow of the Medical Condition Checker web application is as follows:**User Interface**: Users access the application through a web browser. They are presented with a simple interface containing a form to enter their personal information.**Form Submission:** Users enter their name, select their gender, and choose their blood group using the form. Upon submission, the form data is sent to the Flask backend for processing.**Server-Side Processing:** The Flask backend receives the form data and calculates the associated medical condition based on the user's blood group using a predefined mapping. It then stores the entered data in CSV and JSON files for future reference.**Response:** After processing the form data, the server sends a response back to the user's browser. The response includes the user's entered information along with the calculated medical condition, which is displayed on the interface.**4. Implementation Details:**The implementation of the Medical Condition Checker web application is divided into the following sections:**HTML Form (index.html):** The HTML form contains input fields for the user's name, gender (radio buttons), and blood group (dropdown list). It also includes a submit button to submit the form data.**CSS Styles (styles.css):** The CSS file provides styling for the HTML elements, including centering the form within a container using flexbox layout.**Flask Application (route.py):** The Flask application defines routes to handle HTTP requests. The / route renders the HTML form, while the /submit route handles form submissions. Upon form submission, the server processes the data, calculates the medical condition, and renders a result template with the user's information and medical condition.**Data Storage (CSV and JSON Files):** The entered data is stored in separate CSV and JSON files (form\_data.csv and form\_data.json). These files are created and updated dynamically by the Flask application as users submit the form.**5. Deployment and Usage:**To deploy and use the Medical Condition Checker web application:Ensure you have Python and Flask installed on your system.Clone or download the application code from the repository.Navigate to the project directory in the terminal.Run the Flask application by executing the command: python app.py.Access the application in your web browser by visiting the specified URL (typically http://localhost:5001).Enter your personal information in the form and submit it to view the associated medical condition.

**6. Conclusion:**

The Medical Condition Checker web application provides users with valuable information about the medical conditions associated with their blood group. By leveraging Flask for server-side processing and HTML/CSS for the frontend interface, the application offers a simple and intuitive user experience. With its ability to store user data in CSV and JSON files, the application ensures data persistence and allows for future analysis and reference.