Project Documentation: BMI Calculator

1. Cover Page

Project Title: BMI Calculator
Subtitle: A Health Monitoring Web Tool
Date: November 24, 2025
Tech Stack: Python, Flask, HTML, CSS
Deployment: Vercel

2. Introduction

The **BMI Calculator** is a web-based application designed to help users monitor their physical health by calculating their Body Mass Index (BMI). By inputting weight and height, users receive an immediate calculation along with a health categorization (Underweight, Normal, Overweight, or Obese).

The project is built using **Flask**, demonstrating the integration of Python backend logic with a responsive HTML/CSS frontend. It also features a modern UI with a dark mode toggle.

3. Problem Statement

Maintaining a healthy weight is crucial for general well-being, but calculating BMI manually can be tedious and prone to error. People often lack a quick, accessible tool to verify their weight category according to standard health guidelines.

BMI Calculator solves this by providing:

1. An instant calculation engine.
2. Clear, color-coded health categories.
3. A user-friendly interface accessible from any device.

4. Functional Requirements

The system provides the following core functionalities:

- **Calculate BMI:** Accepts user input for Height (cm) and Weight (kg) to compute the BMI value.
- **Health Categorization:** Automatically classifies the result into standard categories:
  - Underweight (BMI < 18.5)
  - Normal (18.5 ≤ BMI < 24.9)
  - Overweight (25 ≤ BMI < 29.9)
  - Obese (BMI ≥ 30)
- **Input Validation:** Prevents calculation with zero or empty values to avoid errors.
- **Reset Functionality:** A button to clear all inputs and results for a new calculation.

- **Dark Mode:** A toggle switch to change the visual theme of the application for better visibility in low light.

5. Non-functional Requirements

- **Responsiveness:** The layout adjusts seamlessly to mobile and desktop screens using standard CSS media queries.
- **Performance:** Lightweight execution ensures calculations happen instantly.
- **Usability:** High contrast text and clear input fields ensure accessibility.
- **Aesthetics:** Uses a clean, modern design with a background image for visual appeal.

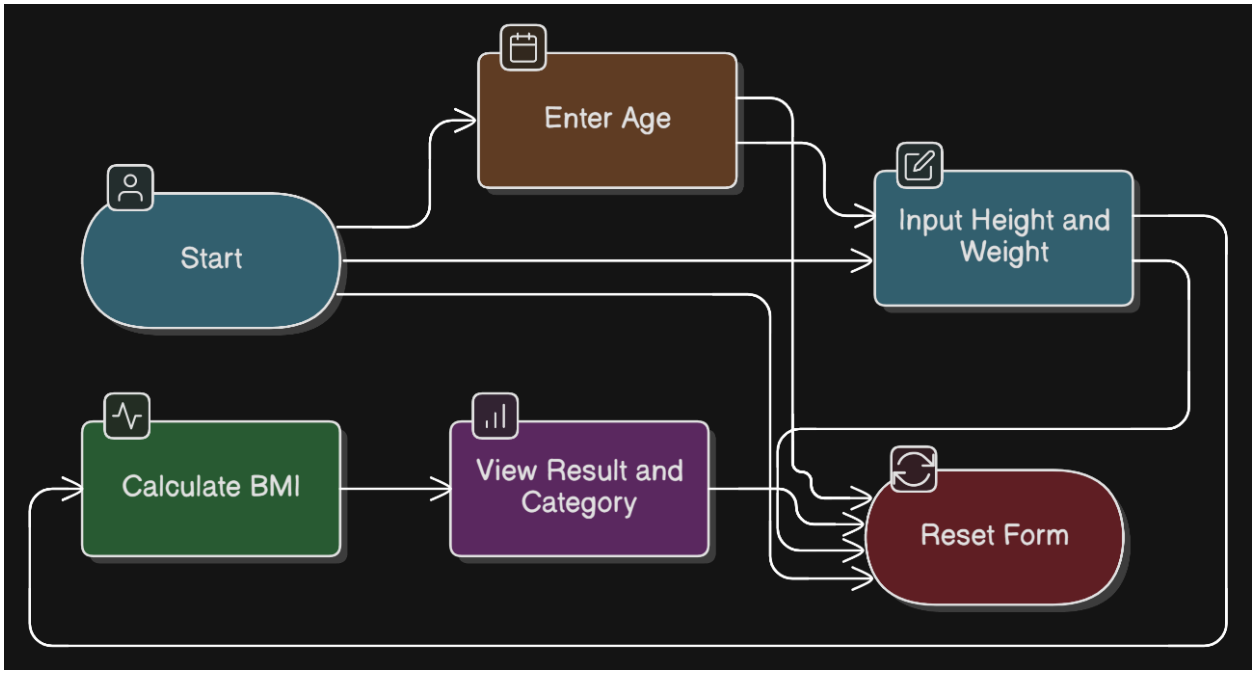
6. System Architecture

The project follows the **Model-View-Controller (MVC)** pattern via Flask:

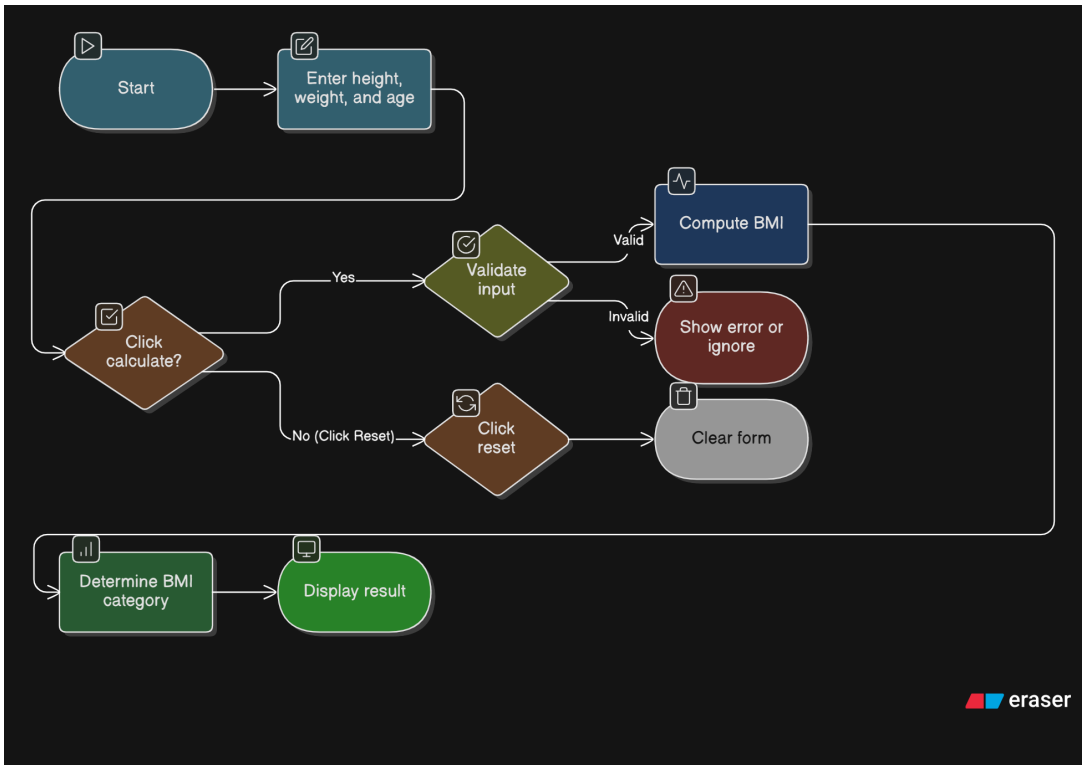
- **Controller (Backend):** bmi_app.py handles the HTTP requests, performs the BMI math, and determines the category.
- **View (Frontend):** templates/index.html renders the form and displays results using Jinja2. static/style.css handles the presentation.
- **Model (Logic):** The calculation logic resides within the route function in bmi_app.py.

7. Design Diagrams

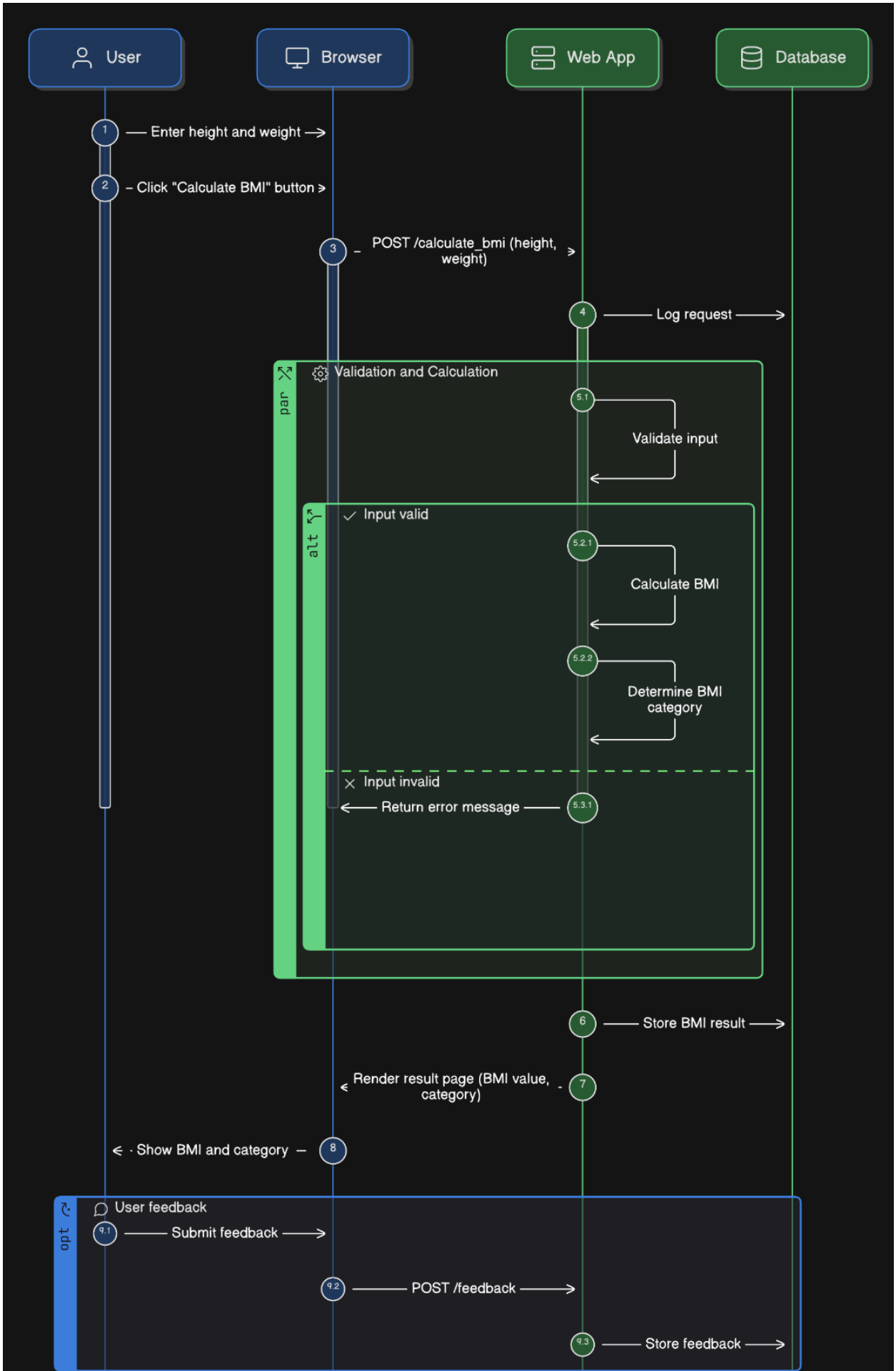
7.1 Use Case Diagram



7.2 Workflow Diagram (Activity)



7.3 Sequence Diagram (Calculation)


User
Browser
Web App
Database
1
Enter height and weight
2
Click "Calculate BMI" button
3
POST /calculate_bmi (height, weight)
4
Log request
par
Validation and Calculation
5.1
Validate input
alt
Input valid
5.2.1
Calculate BMI
5.2.2
Determine BMI category
Input invalid
Return error message
5.3.1
6
Store BMI result
Render result page (BMI value, category)
7
Show BMI and category
8
opt
User feedback
9.1
Submit feedback
9.2
POST /feedback
9.3
Store feedback

8. Design Decisions & Rationale

1. **Framework: Flask**: Selected for its lightweight nature, making it ideal for a single-page calculation app.
2. **Metric System**: Height in Centimeters and Weight in Kilograms were chosen as they are the standard scientific units for BMI.
3. **Server-Side Calculation**: Logic is handled in Python (bmi_app.py) rather than JavaScript to demonstrate backend processing capabilities.
4. **CSS Custom Properties**: Used for the Dark Mode implementation (--text-color, --bg-color) to allow efficient theming without duplicate CSS.

9. Implementation Details

Backend Logic (bmi_app.py)

The core logic converts height from centimeters to meters before the standard BMI formula application:

```
height_m = height_cm / 100
bmi = weight_kg / (height_m ** 2)
```

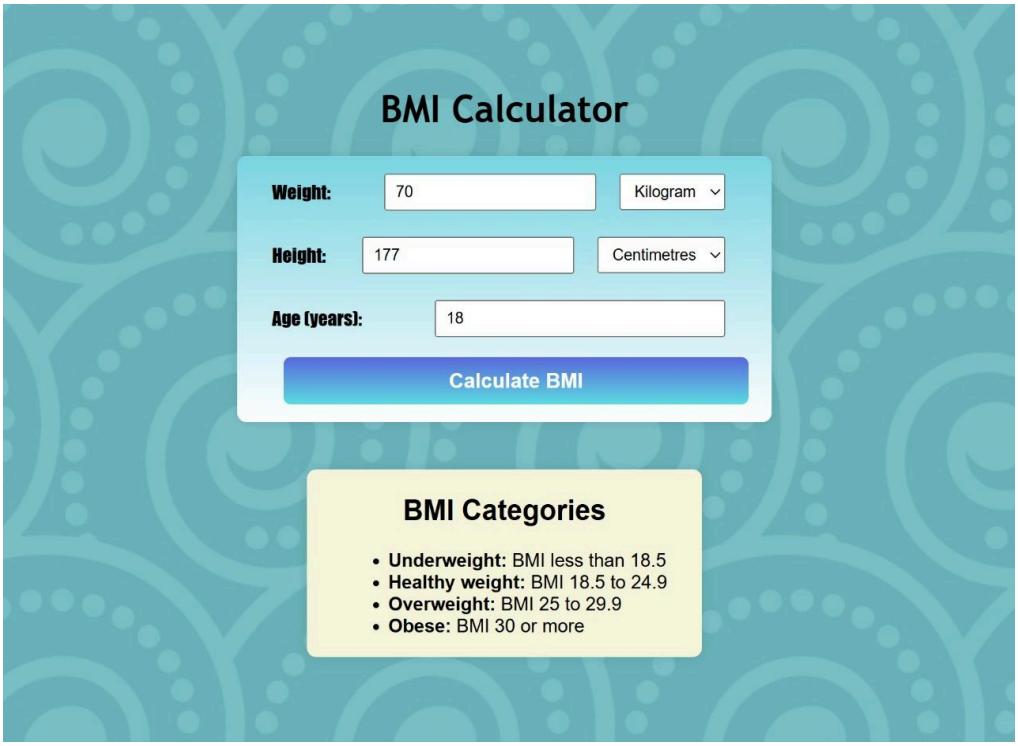
It then uses conditional statements (if/elif/else) to assign the string category (e.g., "Normal", "Obese").

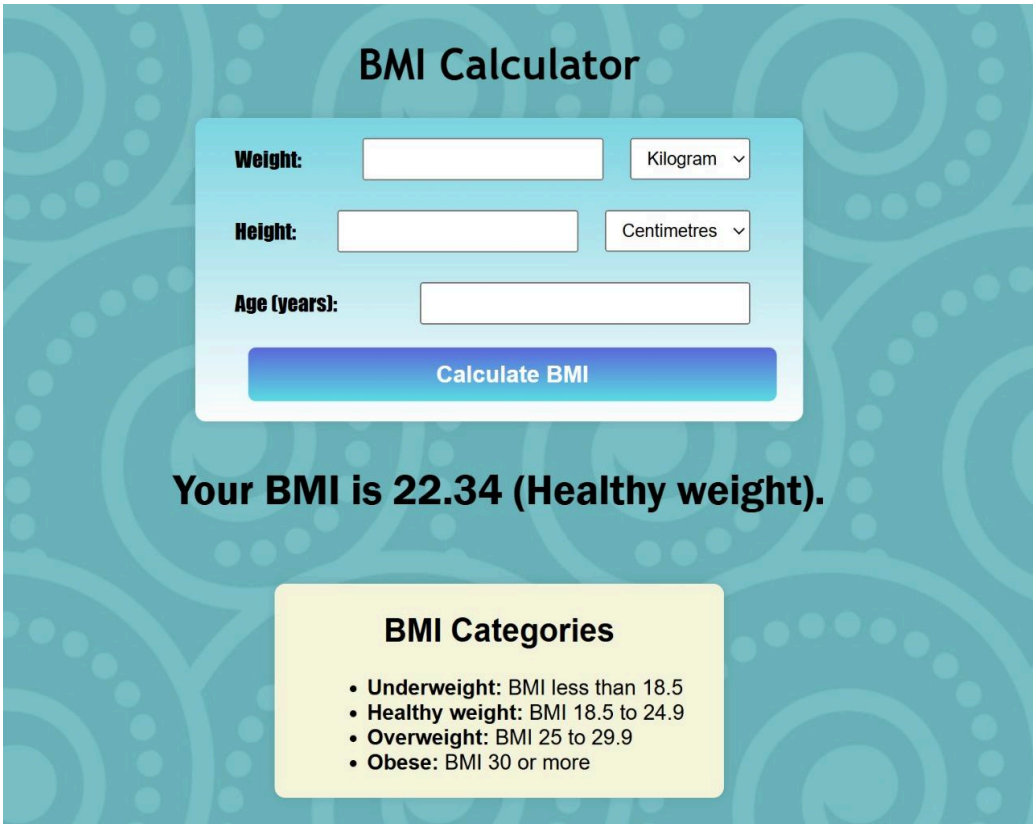
Frontend Styling (style.css)

- **Glassmorphism**: The container uses blue-white background to blend with the background image.

10. Screenshots / Results

1. **Input Screen**: Displays fields for Height and Weight, along with "Calculate" and "Reset" buttons.
2. **Result Display**: Shows the calculated BMI number formatted to two decimal places and the corresponding health category.


BMI Calculator
Weight:
70
Kilogram
Height:
177
Centimetres
Age (years):
18
Calculate BMI
BMI Categories
Underweight: BMI less than 18.5
Healthy weight: BMI 18.5 to 24.9
Overweight: BMI 25 to 29.9
Obese: BMI 30 or more


BMI Calculator
Weight:
Kilogram
Height:
Centimetres
Age (years):
Calculate BMI
Your BMI is 22.34 (Healthy weight).
BMI Categories
Underweight: BMI less than 18.5
Healthy weight: BMI 18.5 to 24.9
Overweight: BMI 25 to 29.9
Obese: BMI 30 or more

11. Testing Approach

- **Unit Testing (Manual):**
  - **Standard Case:** 180cm / 75kg -> Expected: ~23.15 (Normal).
  - **Boundary Case:** 180cm / 40kg -> Expected: Underweight.
  - **Error Case:** Entering 0 height -> Logic handles division by zero or input validation ensures > 0.
- **UI Testing:** Verified that the "Reset" button clears the result text and input fields. Checked Dark Mode persistence/toggling.

12. Challenges Faced

- **Zero Division Error**: Ensuring the app doesn't crash if a user enters 0 for height.
- **CSS Specificity**: Making sure the dark mode styles correctly overrode the default styles for all elements, including inputs and buttons.

13. Learnings & Key Takeaways

- **Flask Form Handling**: Learned how to retrieve data using request.form.get().
- **Jinja2 Templating**: Mastered the use of {% if bmi %} blocks to conditionally render results only after a calculation is performed.
- **Deployment Config**: Learned to configure vercel.json for deploying Python apps to a serverless environment.

14. Future Enhancements

1. **Imperial Units**: Add a switch to calculate using Feet/Inches and Pounds.
2. **Health Tips**: Display dynamic diet or exercise advice based on the calculated category.
3. **History Log**: Use local storage or a database to save previous calculations for progress tracking.

15. References

1. **Flask Documentation**: https://flask.palletsprojects.com/
2. **BMI Formula Source**: World Health Organization (WHO).
3. **MDN Web Docs (CSS/HTML)**: https://developer.mozilla.org/