Asian College of Technology

College of Computer Studies

APPDEV

System Documentation

For

BMI Calculator Mobile Application

GROUP 6

Jose Aristotle Morpheous A. Belarmino

John Chris Pardo

Britni P. Uy

Edrian Visagas

1. **Overview:**

Body Mass Index (BMI) is a measure of body fat based on an individual's height and weight. It provides a numerical value that helps assess whether a person is underweight, normal weight, overweight, or obese. BMI is important because it serves as a simple and widely used screening tool for identifying potential health risks associated with weight-related issues such as cardiovascular diseases, diabetes, and certain cancers. It helps individuals, healthcare professionals, and public health organizations monitor and manage weight-related health concerns, promoting overall well-being and disease prevention.

Creating a BMI calculator mobile app involves the seamless integration of design, functionality, and user experience. It is important to spend time in the first step outlining the needs and essential features of the app. This includes determining the necessary input fields for height and weight. When the requirements are apparent, the design process starts, during which the UI/UX is created to be both aesthetically pleasing and intuitive. Users may easily enter their data and understand the results thanks to straightforward navigation and feedback features.By prioritizing accuracy, simplicity, and user-centric design, a BMI calculator app can become a valuable tool for individuals seeking to monitor their health and fitness journey.

**1.1 Objective**

1. Develop a user-friendly and informative BMI calculator app for Android using Xamarin.
2. Allow users to:
   1. Calculate BMI based on height and weight.
   2. View their BMI category and a corresponding description.
   3. Access detailed information about BMI and its interpretation.
   4. Optionally (using Firebase):
      1. Create an account to save and track their BMI history.
      2. View historical BMI calculations and trends.

**Stakeholders:**

* Development Team: Responsible for building and implementing the app functionality.
* Project Manager: Oversees the development process, manages resources, and ensures project timelines are met.
* Users: Individuals looking to calculate and track their BMI for health and fitness goals.

**Resources:**

* Visual Studio with Xamarin: Integrated development environment (IDE) for building cross-platform mobile apps.
* Firebase: Google's cloud platform for mobile app development, offering services like authentication and database management (if user accounts and data storage are desired).
* Xamarin.Essentials: Provides access to cross-platform APIs for common functionalities like device preferences and secure storage (useful for storing user settings or recent calculations locally).

**1.2 Scope and Limitation**

The scope of a BMI calculator mobile app outlines the functionalities and features it encompasses, while the limitations highlight constraints or boundaries that may affect its performance or usability. Here's a breakdown:

**Scopes:**

1. BMI Calculation: Provide a tool for users to calculate their Body Mass Index (BMI) based on inputted height and weight.
2. User Interface: Design an intuitive interface for easy data input and clear presentation of BMI results.

**Limitations:**

1. Accuracy: BMI calculation may not fully represent individual health factors like muscle mass or body composition.
2. Privacy: Ensure strict data privacy measures to protect users' personal information.
3. Health Disclaimer: Include a disclaimer stating the app's informational nature and the recommendation to consult healthcare professionals for personalized advice.
4. Device Compatibility: App functionality may vary based on users' device specifications and capabilities.
5. Variability in Results: Results may vary based on factors such as different BMI calculation formulas or rounding methods. The app should aim for consistency and transparency in how BMI is calculated and presented to users.

**II. Analysis Requirements**

## Functional and Non-Functional Requirements for BMI Calculator App. Here's a breakdown for the BMI Calculator App:

Functional Requirements:

* User Registration and Login:
  + Users can register and Login using their credentials.
* User Inputs:
  + Users can enter height, weight, age (optional), and gender (optional).
* BMI Calculation:
  + The app calculates BMI based on user input values.
* Results Display:
  + Display BMI result and category (Underweight, Healthy, Overweight, Obese).
  + Users can navigate to a details page for more information about their BMI.
* Navigation and Other Features:
  + Users can sign out (if user accounts are implemented).
  + The app offers navigation to other informative sections:
    - About Us
    - Rate Us
* Track and Display User's BMI History
* Deletion of Specific History Entries

Non-Functional Requirements:

* Usability and User-Friendliness:
  + The app should be responsive and user-friendly with an intuitive interface.
  + Responsive design to adapt to different screen sizes.
  + Clear instructions and visually appealing design.
* Security and Data Handling:
  + Secure user data and handle it appropriately (if user accounts are implemented).
  + Implement strong password hashing and encryption.
  + Be transparent about data collection and usage through a privacy policy.
* Performance:
  + The app should be performant with minimal lag or delay.
  + Fast BMI calculations and optimized code.
  + Consider enabling offline BMI calculations (optional).

**III. Design:**

This section dives into the system architecture and UI design for the BMI calculator app.

System Architecture:

* Frontend:
  + Xamarin.Forms: Used to create a cross-platform UI that adapts seamlessly to different Android devices.
  + Activities and Fragments:
    - Activities manage the app's lifecycle and handle overall screen navigation.
    - Fragments manage smaller, modular UI components within an Activity (useful for Login Page and potentially Details Page).
* Backend:
  + Firebase Authentication: (if user accounts are implemented) Provides user registration, login, and authentication functionalities.
  + Data Storage:
    - Firebase: (if user accounts are implemented) Firebase Realtime Database or Cloud Firestore can store user data (BMI history with timestamps).
    - Local Storage: (alternative to Firebase) Use Xamarin.Essentials' secure storage to save BMI results locally on the device (without user accounts).

UI Design:

* Login Page (Optional):
  + Text fields for email and password entry.
  + Login button and potentially a "Sign Up" button (if user registration is included).
  + "Forgot Password" functionality (consider integrating with Firebase password reset).
* Main Page:
  + Navigation bar or buttons for accessing different sections:
    - Calculate BMI
    - About Us
    - Rate Us
    - Settings (optional: manage units, privacy settings)
* BMI Calculation Page:
  + Input fields for height, weight, age (optional), and gender (optional).
  + Unit selection toggle for metric and imperial units.
  + "Calculate" button to trigger BMI calculation.
* Result Page:
  + Display calculated BMI value.
  + Visually represent BMI category (e.g., color-coded text or progress bar) with a corresponding description.
* Track Page (History)
  + Display the history a user taken a calculated BMI value.
* Details Page:
  + Comprehensive information about BMI, its limitations, and interpretation.
  + Links to credible resources for further reading.

Database Design:

* Firebase Data Structure (Optional):
  + Users collection: Stores user information (email, password hash).
* Local Storage Structure (Alternative to Firebase):
  + Secure storage on the device can hold user preferences (units) and BMI history entries (timestamps, BMI values) in a structured format (e.g., JSON).

## Database Model

### Firestore Collections and Documents

#### Users Collection

* Document ID: User ID
  + Fields:
    - fullName: String
    - email: String
    - password: String (hashed)

#### user\_results Collection (Sub-Collection of Users)

* Document ID: Auto-generated
  + Fields:
    - Height: Double
    - Weight: Double
    - Age: Integer
    - Gender: String
    - BMI: Double
    - Date: String
    - BMICategory: String

IV. **Implementation**

This section outlines the development tools, coding standards, and key modules involved in building the BMI calculator app.

**Development Tools:**

* Visual Studio with Xamarin
* Firebase SDK for Xamarin

**Coding Standards:**

* Consistent Naming Conventions
* Commenting and Documentation for code readability.

**Modules Developed:**

* Authentication Module
* BMI Calculation Module
* UI Components
* Navigation and Data Passing between activities.

V.  **Testing**

* Types of Testing:

Unit Testing: Test individual components like BMI calculation logic.

Integration Testing: Ensure different parts of the app work together.

User Acceptance Testing (UAT): Gather feedback from potential users.

* Testing Tools:

Xamarin Test Cloud

Firebase Test Lab

* Test Cases:

Validating correct BMI calculation.

Ensuring data is correctly passed between activities.

Verifying user authentication flow.

Checking UI responsiveness and layout on different devices.

**VI. Deployment**

Deployment Environment:

- Google Play Store for Android distribution.

- Internal testing track for beta testing.

Deployment Steps:

- Create a signed APK.

- Upload the APK to Google Play Console.

- Set up app listing and distribution settings.

- Submit for review and publish.

Post-Deployment Monitoring:

- Monitor user feedback and crash reports using Firebase Crashlytics.

- Track user engagement and usage analytics.

**VII. Maintenance**

Types of Maintenance:

Corrective Maintenance: Fixing bugs and issues reported by users.

Adaptive Maintenance: Updating the app to ensure compatibility with new Android versions and devices.

Perfective Maintenance: Enhancing the app with new features based on user feedback.

Maintenance Process:

Regularly check for and respond to user feedback.

- Schedule periodic updates to address any discovered issues or enhancements.

- Ensure continuous integration and testing with updates.