**CHAPTER-1**

**INTRODUCTION**

**INTRODUCTION:**

The healthcare industry has been continuously seeking innovative ways to improve the overall fitness and health of individuals. One such solution is the development of a mobile app that focuses on the prevention of conditions through improved fitness. This app utilizes cutting-edge technology and data analysis to provide personalized exercise and nutrition plans, helping individuals take control of their health and reduce the risk of developing conditions. With the increasing prevalence of chronic diseases, the implementation of such a solution is more important than ever, and the pledge to estimate its impact on healthcare is a crucial step in promoting the widespread adoption of this innovative approach. The increasing incidence of chronic diseases has become a major concern for healthcare professionals worldwide. These conditions, such as obesity, cardiovascular diseases, and diabetes, often result from unhealthy lifestyles and lack of physical activity. To address this issue, there is a growing need for innovative solutions that can help individuals take control of their health and improve their overall fitness.

One such solution is the development of a mobile app that is designed to promote health and fitness by preventing the onset of conditions. This app uses cutting-edge technology to provide personalized exercise and nutrition plans based on an individual's unique health profile. It also features real-time monitoring and tracking of progress, as well as motivational tools to help users stick to their fitness goals.

The pledge to estimate the impact of this app on healthcare is a crucial step in promoting its widespread adoption. By quantifying the positive effects of this innovative approach, healthcare professionals and organizations can make informed decisions about incorporating the app into their existing programs and initiatives. Additionally, this pledge serves as a commitment to continuously improving and refining the app to provide the best possible outcomes for individuals seeking to improve their health and fitness.

ALGORITHM

Predictive modeling: This technique uses past data to make predictions about future outcomes. For example, the app might use data on an individual's exercise and nutrition habits to predict the risk of developing a certain condition.

Clustering: This algorithm is used to group individuals with similar characteristics, such as fitness levels, based on data analysis. This can be used to provide targeted recommendations and track progress.

Decision trees: This type of algorithm uses a tree-like structure to make decisions based on certain conditions. For example, the app might use a decision tree to determine the best exercise plan for an individual based on their fitness level and other factors.

Recommender systems: This algorithm uses data analysis to recommend items or actions that are likely to be of interest to an individual. For example, the app might use a recommender system to suggest healthy meal options based on an individual's dietary preferences.

These are just a few examples of the algorithms that might be used in a mobile app for fitness and health promotion.

REQUIREMENTS

Technical expertise: A team of skilled developers and designers with experience in building mobile applications is essential.

Health and fitness data: Access to reliable data on health and fitness is necessary to provide personalized recommendations and track progress.

Machine learning algorithms: Advanced machine learning algorithms are required to provide personalized recommendations and track progress over time.

User interface design: A well-designed user interface is essential to make the app accessible and user-friendly.

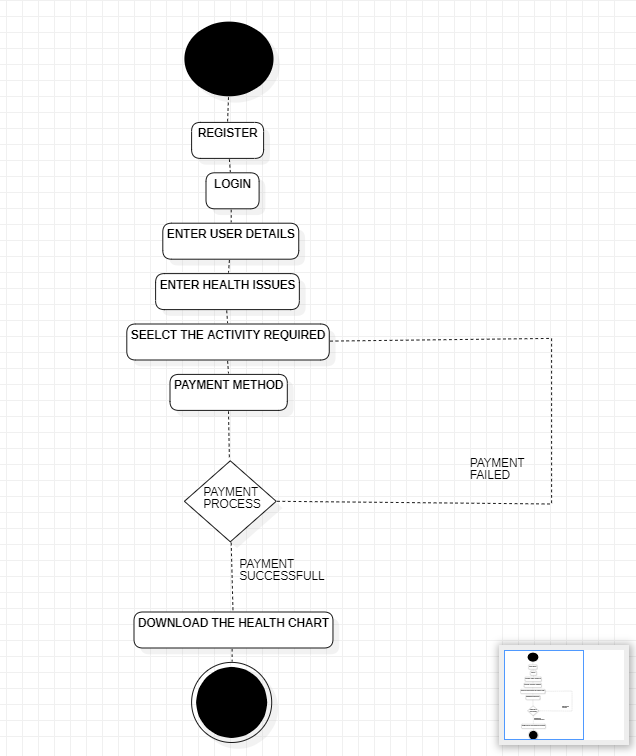
Database management: A database management system is needed to store and organize the vast amount of health and fitness data generated by the app.

Mobile platform support: The app should be developed for both iOS and Android platforms to reach the largest possible audience.

Testing and quality assurance: The app should be thoroughly tested and evaluated for quality and reliability to ensure a positive user experience.

Marketing and promotion: A marketing and promotion strategy is necessary to raise awareness of the app and attract users.

**ACTIVITY DIAGRAM**



**ANDROID**

Android is an open-source operating system for mobile devices, such as smartphones and tablets, developed by Google. It was first released in 2008 and has since become one of the most popular mobile operating systems in the world.

Some key features and characteristics of Android include:

Customization: Android allows for extensive customization, including the ability to change the look and feel of the operating system with custom launchers, icon packs, and themes.

Google services: Android integrates with Google's suite of services, such as Gmail, Google Maps, and Google Drive, providing users with a seamless and integrated experience.

Large app ecosystem: Android has a large and diverse app ecosystem, with millions of apps available on the Google Play Store. This allows users to access a wide range of apps, from games and entertainment to productivity and education.

Open-source: Android is open-source, meaning that the source code is publicly available for anyone to use and modify. This has led to a vibrant community of developers and users who contribute to the development and improvement of the operating system.

Cross-device compatibility: Android is compatible with a wide range of devices, from high-end flagship smartphones to budget-friendly devices, making it accessible to users of all budgets and preferences.

**Java (programming language)**

### History

James Gosling developed the JAVA programming language in June 1991 for a set-top box project. The language's original name, Oak, was taken from an oak tree outside Gosling's office. It was also known as Green. Later, Java was chosen from a list of random phrases. Gosling wanted to create a virtual machine language with notation similar to that of C/C++. Java 1.0 was the initial release to the general public in 1995. It made the "Write Once, Run Anywhere" (WORA) promise and offered free runtimes for well-known platforms.

It was moderately secure and customizable security allowed for limiting network and file access. Major web browsers quickly added support for running secure Java applets inside of web pages. Java gained popularity quickly. With the introduction of Java 2, new versions included numerous configurations created for various kinds of systems. For instance, J2EE was for corporate applications, whereas J2ME was for mobile apps in its substantially condensed form. The Standard Edition was referred to as J2SE. New J2 versions were dubbed Java EE, Java ME, and Java SE in 2006 for marketing purposes.

To codify Java, Sun Microsystems approached Ecma International and the ISO/IEC JTC1 standards organisation in 1997, but it quickly withdrew from the process. The Java Community Process continues to govern Java as a standard. Sun used to offer the majority of their proprietary Java implementations for free, despite the fact that they were proprietary software.

Selling licencing for specialist products like the Java Enterprise System was how Sun made money off of Java. The main difference between Sun's Software Development Kit (SDK) and Runtime Environment (JRE), a subset of the SDK, is that the compiler, utility applications, and many essential header files are absent from the JRE.

Sun released a large portion of Java as free software under the provisions of the GNU General Public License on November 13, 2006. (GPL). Sun completed the process on May 8 by declaring all of Java's core code open source, with the exception of a minor amount of code for which Sun lacked copyright..

### Prime objectives

### The development of the Java language had five main objectives:

### • It ought to employ object-oriented programming techniques.

### • It should enable the execution of the same software across many operating systems.

### • It must to include built-in assistance for utilising computer networks.

### • It must be built to safely run code from remote sources.

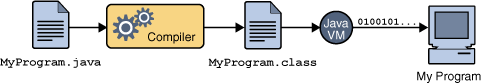
### • It should be simple to use by incorporating the features that were deemed to be beneficial in existing object-oriented languages.The Java Programming Language:

All of the following catchphrases fit the high-level language that is the Java programming language:

Simple, neutral in terms of architecture, object-oriented, portable, distributed, and high performance The Java Language Environment, a white paper by James Gosling and Henry McGilton, defines each of the aforementioned keywords.

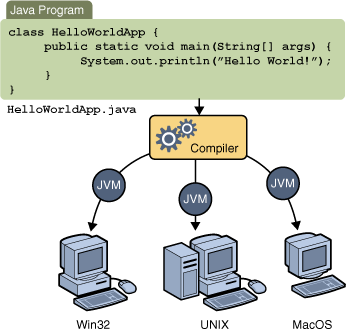
All source code in the Java programming language is initially written in plain text files with the.java extension. The javac compiler subsequently converts the source files into class files.

A class file includes byte codes, the Java Virtual Machine's machine language, rather than native code for your CPU (Java VM). The Java Launcher tool then executes your programme using a Java Virtual Machine instance.



A description of the procedure for developing software.

The same class files can be run on Microsoft Windows, the Solaris TM Operating System (Solaris OS), Linux, or Mac OS due to the Java VM's availability on a variety of operating systems. Some virtual machines, like the Java Hot Spot virtual machine, carry out extra tasks at runtime to speed up your programme. Finding performance bottlenecks and recompiling commonly used code parts (to native code) are just a couple of the jobs that fall under this category.



The same application can run on various platforms thanks to the Java VM.

Platform Java

The hardware or software environment that a program runs in is referred to as a platform. Some of the most well-known operating systems, including Microsoft Windows, Linux, Solaris OS, and Mac OS, have already been mentioned. Most platforms can be characterized as a fusion of the underlying hardware and operating system. The Java platform is unique from most other platforms in that it runs on top of other hardware-based systems and is entirely software-based.

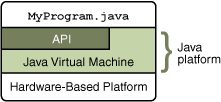
**There are two parts to the Java platform:**

Virtual Machine for Java

Application Programming Interface for Java (API)

The Java Virtual Machine, the foundation of the Java platform and a port to many hardware-based platforms, has already been introduced to you.

The API is a sizable collection of pre-made software elements that offer a wide range of beneficial features. It is organized into libraries called packages that contain similar classes and interfaces. What Can Java Technology Do? which follows, highlights some of the features made available by the API.



The programme is shielded from the underlying hardware by the Java Virtual Machine and API.

The Java platform, which is a platform-independent environment, can sometimes a little slower than native code. However, improvements in virtual machine and compiler technology are bringing performance near to native code levels without endangering portability.

Runtime Environment for Java

Any application deployed on the Java Platform must execute within the Java Runtime Environment, or JRE. A JRE is frequently used by end users as a plug-in for Web browsers and other packages. The Java 2 SDK, usually referred to as the JDK by Sun, is a superset of the JRE that includes development tools such the Java compiler, Javadoc, Jar, and debugger.

The idea of a runtime engine has the special advantage that faults (exceptions) shouldn't 'crash' the system. Additionally, there are tools that attach to the runtime engine in settings where runtime engines are used, such as Java, and they capture debugging information whenever an exception of interest occurs (stack and heap values). The 'root-cause' information for exceptions in Java programmes running in production, testing, or development environments is provided by these automated exception handling solutions.

Uses for JAVA

Blue is a smart card that uses Java Card technology and the safe, universal Java Card API. Blue has a real on-card processing chip that enables enhanced capability and numerous functions on a single card. Any third-party vendor card that offers the requisite Java Card Application Environment can execute applets that adhere to the Java Card API specification (JCAE). A single card can run several applet programmes, and when the card is given to the consumer, more applets and functionality can be added.

Java is used at NASA and can be utilised in chemistry.

• Java is utilised in 2D and 3D applications.

• Java is utilised in graphics programming as well.

• Java is utilised in animations.

• Java is utilised in online and web applications.

**JSP :**

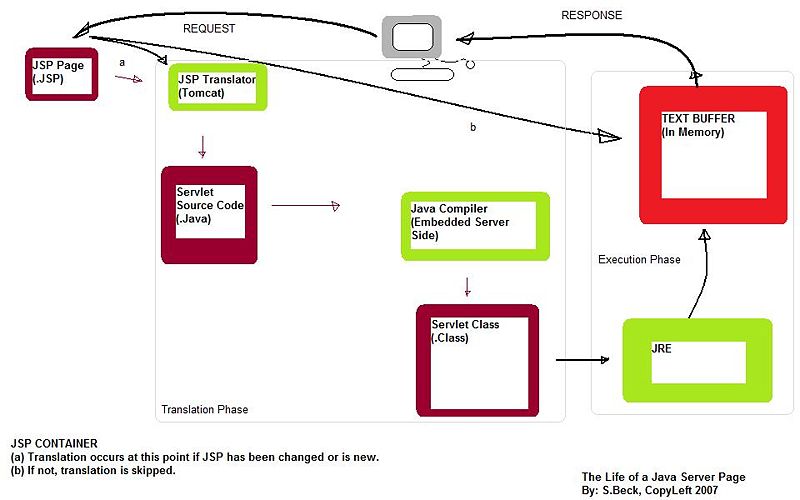
In response to a Web client request, Java Server Pages (JSP), a Java technology, enables software developers to dynamically generate HTML, XML, or other types of documents. The technique enables the embedding of Java code and specific pre-defined actions into static content.

Additional XML-like tags, referred to as JSP actions, are added to the JSP syntax and can be used to call built-in functionality. The technology also enables the development of JSP tag libraries, which function as extensions to the basic HTML or XML tags. A platform-independent method of expanding a Web server's functionality is through tag libraries.

A JSP compiler converts JSPs into Java Servlets. A JSP compiler may provide byte code for the servlet directly or it may produce a servlet in Java code that is subsequently built by the Java compiler. JSPs can also be interpreted instantly, which speeds up the reloading of updates.

Using Java Server Pages (JSP) technology, dynamic web content may be produced easily and quickly. Rapid creation of server- and platform-independent web applications is made possible by JSP technology.

**Architecture OF JSP**

**[](http://upload.wikimedia.org/wikipedia/en/4/46/JSPLife.jpg)**

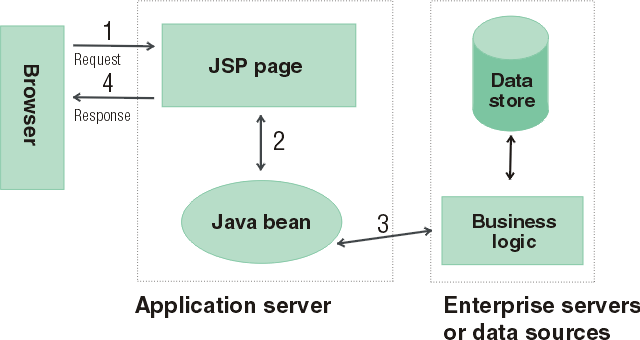
## The Advantages of JSP

Dynamic Server Pages (ASP). A comparable Microsoft technology is ASP. There are two benefits to JSP. First off, because Java was used to write the dynamic portion rather than Visual Basic or another MS-specific language, it is more capable and user-friendly. Second, it can run on non-Microsoft web servers and other operating systems. just Servlet. Nothing that a Servlet could do in theory, you could do with JSP. However, writing (and editing!) standard HTML is more practical than having a tonne of println statements generate the HTML. Additionally, by separating the appearance from the content, you can assign certain duties to other people: The HTML can be created by your Web page designers, leaving spaces for your Servlet programmers to insert the dynamic content.

Server-Side Contains (SSI). A static Web page can have externally-defined components using the widely-accepted mechanism known as SSI. JSP is superior since it enables you to construct the dynamic portion using Servlet rather than a separate programme. Additionally, SSI isn't really designed for "real" programmes that use form data, link to databases, and other things like that; rather, it's simply meant for simple inclusions. JavaScript. On the client, HTML can be dynamically generated using JavaScript. This is a helpful functionality, but it can only be used when the dynamic information is dependent on the environment of the client.

Cookies are the only HTTP and form submission data that are accessible to JavaScript. JavaScript can't access server-side resources like databases, catalogues, price data, and the like because it runs on the client. dynamic HTML Of course, dynamic information cannot be contained in standard HTML. Because JSP is so simple and convenient, it is quite possible to improve HTML sites that only slightly benefit from the addition of small amounts of dynamic data. Dynamic data previously couldn't be used in all but the most advantageous situations due to the cost of employing it.

**ARCHITECTURE OF JSP**



• A JSP page receives a request from the browser.

The JSP page and a Java bean exchange information.

• A database is linked to the Java bean.

• The JSP page reacts to browser requests.

FRONT END SERVLETS

A software developer can add dynamic content to a Web server running on the Java platform by using the Java Servlet API. The generated material is often HTML, but it could also be other types of data, including XML. The Java equivalent of non-Java dynamic Web content technologies like PHP, CGI, and ASP.NET is a servlet. Servlet can use HTTP cookies, session variables, or URL rewriting to retain state over numerous server transactions.

The javax package hierarchy contains the servlet API. The term "Servlet" describes how a Web container and a Servlet are anticipated to communicate. In essence, a Web container is the part of a Web server that communicates with the Servlet. The URL requester must have the appropriate access privileges, and the Web container is in charge of managing the lifecycle of Servlets, mapping a URL to a specific Servlet, and managing Servlet mappings.

An object called a servlet is one that processes requests and responds to them. Java objects that represent Servlet requests and responses, as well as objects that reflect the Servlet setup parameters and execution environment, are defined by the basic Servlet package.

the javax package. Servlet. Http defines subclasses of the generic Servlet elements that are particular to HTTP, such as session management objects that keep track of numerous requests and responses sent back and forth between a client and a Web server. As a Web application, Servlet may be packaged in a WAR file.

Java Server Pages (JSP) or alternatively template engines like Web Macro can automatically produce Servlets. A variation of the model-view-controller design called "Model 2" frequently employs Servlets together with JSPs.

Servlet technology is CGI programming's response to Java technology. They are applications that create Web pages and run on a web server. For a variety of reasons, creating Web pages on the fly is advantageous (and frequently done).

The user-submitted data form the basis of the Web page. Programs that process orders for e-commerce sites and search engine results pages, for instance, are both developed in this manner. The data is often updated. For instance, a website with weather information or news headlines might be built dynamically, returning an older version of the page if it is still current. The website makes use of data from business databases and other similar sources.

You might use this to create a Web page for an online store that provides the current prices and quantity of inventory, for instance.

The Environment for Servlet Run-time

Because a Servlet is a Java class, a service that we refer to as a Servlet engine must execute it on a Java virtual machine. When a servlet is first requested, or alternatively as soon as the servlet engine is started, the servlet engine loads the servlet class. Up until it is explicitly emptied or the Servlet engine is shut down, the Servlet remains loaded to process additional requests.

A built-in Servlet engine is present in certain Web servers, including Sun's Java Web Server (JWS), W3C's Jigsaw, and Gefion Software's Lite Web Server (LWS). A Servlet engine add-on module is necessary for other Web servers, including Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS), and Apache from the Apache Group. All Servlet requests are intercepted by the add-on, which then executes them and sends the results back to the client via the Web server. The WAI Cool Runner by Gefion Software, Web Sphere by IBM, JRun by Live Software, and Servlet Exec by New Atlanta are a few examples of Servlet engine add-ons.

The Java Servlet Development Kit (JSDK), which may be downloaded from Sun's official Servlet site, combines all Servlet API classes and a basic Servlet-enabled Web server. I advise you to download the JSDK and experiment with the sample Servlet to get you started with Servlets.

SERLET LIFE CIRCLE

• The steps in the Servlet lifecycle are as follows:

• During startup, the container loads the Servlet class.

The init() method is called by the container. Before the Servlet can handle any requests, this method must be invoked to initialise it. The init() method is only ever invoked once over the lifetime of a Servlet. The Servlet is ready to fulfil client requests after initialization.

### Each request is handled in a different thread. For each request, the container invokes the Servlet's service() function.

### The request is routed to the appropriate method by the service() method after identifying the type of request being made. An implementation of these methods must be supplied by the Servlet developer. When a request is made for a method that the Servlet does not implement, the method of the parent class is called, and usually an error is returned to the requester. Finally, the container invokes the destroy() method to deactivate the Servlet. Like init(), the destruct() method is only used once during a Servlet's lifetime.

Objects of Request and Response

HttpServletRequest and HttpServletResponse are two relevant parameters for the do Get method. These two classes allow you complete access to all request-related data and enable you to manage the output that is sent to the client as a request-response. The names of the environment variables can change between implementations, and some may not be available by all Web servers, but you can read environment variables and stdin with CGI to learn more about the request.

In a standardised manner, the HttpServletRequest object offers the same details as the CGI environment variables, in addition to more. Additionally, depending on the type of request, it offers techniques for retrieving HTTP parameters from the query string or the request body (GET or POST).

For both sorts of requests, Servlet developers access parameters in the same way. Other techniques let you access all request headers and aid with cookie and date header parsing.

The HttpServletResponse returns either an OutputStream or a PrintWriter instead of publishing the response to stdout as you would with CGI. The PrintWriter is designed for text output, whereas the OuputStream is for binary data, like a GIF or JPEG image. Additionally, you can modify the status code and all response headers without utilising certain Web server CGI parameters like Non Parsed Headers (NPH). This makes installing your Servlet simpler.

ServletConfig and Servlet Context:

There is only one Servlet Context in each application. All Servlets can use this object to access facts about containers or application-level metadata. On the other hand, every Servlet receives a unique ServletConfig object. For a servlet, this object's initialization arguments are provided. Either the ServletConfig object or the Servlet Request object can be used by a developer to get the reference to the Servlet Context.

One servlet context contains all servlets. All servlets on a single host belong to the same context in Servlet API implementations of versions 1.0 and 2.0, however with version 2.1 of the API, the context gains strength and can be considered as the embryonic stage of the Application idea.

You can combine a number of servlets into one context and support many contexts on the same host with the help of many servlet engines that implement the Servlet 2.1 API. In the 2.1 API, the Servlet Context is in charge of the state of its servlets and is aware of the resources and characteristics that are available to them. Here, we will solely examine the use of Servlet Context properties for information sharing amongst a set of Servlets.

Get Attribute, Set Attribute, and Remove Attribute are the three Servlet Context methods that deal with context attributes. Additionally, the servlet engine might offer options for configuring a servlet context with initial value for attribute.

This is a wonderful addition to the servlet startup arguments for configuration data required by a collection of servlets, such as the database identifier we discussed earlier, a URL for a style sheet for an application, the name of a mail server, etc.

Java programmers can use the JDBC Java Database Connectivity (JDBC) framework to create applications that can access data held in databases, spreadsheets, and flat files. No matter what database management system is employed to maintain the database, JDBC is frequently used to connect a user programme to a "behind the scenes" database. JDBC is cross-platform in this sense.

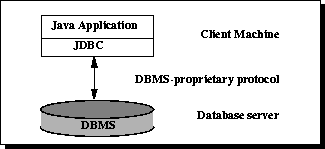
### This article will give an overview and sample code that shows database access from Java programmes that utilise the JDBC API classes, which can be downloaded for free from Sun's website.

### A data source is a database that another software relates to. Numerous data sources, including Microsoft and Oracle products, already adhere to the Open Database Connectivity standard (ODBC). Many old C and Perl programmes link to data sources using ODBC. Many of the similarities between database management systems were unified through ODBC. JDBC advances the level of abstraction by building on this capability. Java programmes can now connect to ODBC-capable database programmes thanks to JDBC-ODBC bridges.

### JDBC Architecture

### **Two-tier and Three-tier Processing Models**

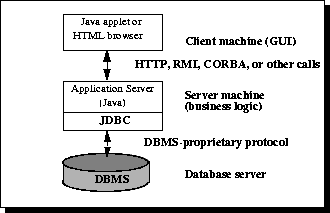
The JDBC API supports both two-tier and three-tier processing models for database access.



A Java applet or application communicates directly with the data source in the two-tier paradigm. A JDBC driver that can interface with the specific data source being accessed is necessary for this. The database or other data source receives the user's commands, processes them, and returns the results to the user. The user may be connected to the data source through a network on a different computer. In a client/server configuration, the user's computer serves as the client while the computer hosting the data source serves as the server. The network could be the Internet or an intranet, which, for instance, links workers within a company.

Commands are sent to a "middle tier" of services in the three-tier paradigm, which subsequently transmits the commands to the data source. The middle tier receives the results from the middle tier, which then provides them to the user when the data source has processed the commands.

Because the middle tier allows for the maintenance of control over access and the kinds of adjustments that can be made to company data, MIS directors find the three-tier model to be particularly alluring. Another benefit is that it makes application deployment simpler. Finally, the three-tier architecture frequently offers performance benefits.



Up until recently, the intermediate tier was frequently programmed in performance-oriented languages like C or C++. However, the Java platform is quickly replacing other platforms as the preferred one for middle-tier development thanks to the advent of optimising compilers that convert Java byte code into effective machine-specific code and technologies like Enterprise JavaBeansTM. This is a huge bonus because it enables the use of Java's multithreading, security, and robustness capabilities.

The JDBC API is widely employed in the middle tier of a three-tier architecture as businesses use the Java programming language for building server code. JDBC supports distributed transactions, detached rowsets, connection pooling, and other capabilities that make it a server technology. A Java middle tier can also access a data source thanks to the JDBC API.

**Testing**

**The various levels of testing are**

1. White Box Testing
2. Black Box Testing
3. Unit Testing
4. Functional Testing
5. Performance Testing
6. Integration Testing
7. Objective
8. Integration Testing
9. Validation Testing
10. System Testing
11. Structure Testing
12. Output Testing
13. User Acceptance Testing

**White Box Testing**

A type of software testing called "white-box testing" (also called "clear box testing," "glass box testing," "transparent box testing," and "structural testing") examines an application's internal mechanisms rather than its functionality (i.e. black-box testing). In white-box testing, test cases are created using programming knowledge and an internal viewpoint of the system. The tester selects inputs to test various code routes and identify the necessary outputs. Similar to evaluating the nodes in a circuit, such as by in-circuit testing (ICT).

In the software testing process, white-box testing can be used at the unit, integration, and system levels, but it is typically done at the unit level.

It may test the connections between subsystems during a system-level test, as well as the connections between units during integration. Although this approach to test design can find a lot of faults or issues, it might miss needs that aren't present or sections of the specification that haven't been implemented.

Techniques for designing white-box tests include:

* Control flow testing
* Data flow testing
* Branch testing
* Path testing
* Statement coverage
* Decision coverage

A technique for testing an application at the source code level is called "white-box testing." The above-mentioned design methodologies—control flow testing, data flow testing, branch testing, path testing, statement coverage, decision coverage, and modified condition/decision coverage—are used to produce the test cases. These methods are used in "white-box testing," which examines any vulnerable code while establishing an environment free of errors.

These White-box testing methods serve as the foundation for white-box testing, which is fundamentally concerned with carefully testing an application at the source code level in order to catch any buried issues in the future. These many methods test each visible path in the source code to reduce errors and establish an environment free of errors.

Knowing which line of the code is being executed and being able to determine what the intended output should be are the two main goals of white-box testing.

**Levels**

1. Unit testing. Before any integration with previously tested code takes place, white-box testing is carried out during unit testing to make sure the code is functioning as expected. Unit testing using white-box testing finds any errors early on and helps with any errors that occur later after the code is merged with the rest of the application, preventing any errors from occurring later.

2. Testing for integration. At this phase, white-box testing is created to examine how each interface interacts with the others. The integration testing looks at the accuracy of the behaviour in an open environment by using white-box testing for any interactions of interfaces that are familiar to the programmer. The unit level testing made sure that each code was tested and working appropriately in an isolated environment.

3. Regression analysis. Recycled white-box test cases are used for white-box testing during regression testing at the unit and integration testing levels.

In order to test something using white-box testing, you must have a thorough comprehension of the source code that is being tested. To design test cases that exercise every observable path for testing, the programmer needs to have a thorough grasp of the application. Once the source code has been comprehended, it can be examined in order to develop test cases. The three fundamental phases that white-box testing follows in order to develop test cases are as follows:

Input consists of various requirements, functional specifications, meticulous document design, correct source code, and security specifications. White-box testing is currently in the preparation stage, when all of the fundamental data is laid out.

Processing Unit entails carrying out risk analysis in order to direct the entire testing process, create an appropriate test plan, carry out test cases, and communicate results. Building test cases during this stage will ensure that the application is thoroughly tested and that the findings are captured appropriately.

Prepare a final report that includes the preparations and outcomes from the previous steps.

Black Box Testing

A software testing technique called "black-box testing" looks at an application's functionality, or what the software does, without looking at its internal components or workings (see white-box testing). Practically every level of software testing, including unit, integration, system, and acceptance testing, may be conducted using this test methodology. It often makes up the majority, if not higher-level testing, but it can also rule unit testing.

**Test Protocols**

It is not necessary to have in-depth knowledge of the code or internal structure of the application or of programming in general. The tester is aware of what the software is meant to do, but not how it actually accomplishes that. For example, the tester may be aware that a specific input leads to a specific, deterministic outcome but may not be aware of how the software generates the output in the first place.

**Case Studies**

Specifications and requirements, or what the application is supposed to do, are the foundation upon which test cases are created. Typically, test cases are produced from external descriptions of the software, such as requirements, specifications, and design criteria.

### Non-functional tests may also be utilised, despite the fact that functional tests represent the majority of the tests used. Without being aware of the internal structure of the test object, the test designer chooses both legitimate and incorrect inputs and figures out the correct output.

### **Test design techniques**

Typical black-box test design techniques include:

* Decision table testing
* All-pairs testing
* State transition tables
* Equivalence partitioning
* Boundary value analysis

**Unit testing**

Unit testing is a technique used in computer programming to check the suitability of individual pieces of source code, sets of one or more computer programme modules with related control data, usage processes, and operating procedures. A unit can be thought of as the smallest tested component of an application, intuitively. A unit in procedural programming might be a whole module, but they are typically just one function or process. A unit in object-oriented programming is frequently a full interface, like a class, although it also could be a single method. Programmers or occasionally white box testers write unit tests as part of the development process.

Each test scenario should ideally be independent of the others. To help with module isolation testing, substitutes such method stubs, dummy objects, fakes, and test harnesses can be utilised. Software engineers generally create and execute unit tests to make sure that code adheres to design specifications and operates as expected. Its application might be structured as a part of build automation or very manual (pencil and paper).

Testing cannot analyse every execution path in any but the simplest programmes, hence it cannot detect every error in the programme. It holds true for unit testing as well. Additionally, by definition, unit testing only examines the functioning of the individual units.Consequently, it won't detect integration issues or more general system errors (such as functions performed across multiple units, or non-functional test areas such as performance).

Unit testing should be conducted in conjunction with other software testing procedures because it can only demonstrate the existence or absence of specific faults, not the absence of all defects. Other methods, such as the use of formal methods to demonstrate that a software component has no unexpected behaviour, are necessary in order to guarantee accurate behaviour for every execution path and every conceivable input and to ensure the absence of mistakes.

Combinatorial problems are common in software testing. For instance, every Boolean decision statement necessitates the execution of two tests, one with a "true" and one with a "false" conclusion. Programmers consequently frequently require 3 to 5 lines of test code for each line of code they write.

Clearly, this takes time, and the effort may not be worthwhile. There are also numerous issues that are difficult to test at all, such as nondeterministic issues or issues involving several threads. Additionally, the code used in a unit test is probably at least as flawed as the code it is testing. In The Mythical Man-Month, Fred Brooks says, "Never carry two chronometers to sea." Take one or three at all times.

The difficulty of creating up practical and realistic tests is another issue that arises while writing unit tests. To ensure that the application component under test behaves as a component of the entire system, it is required to establish pertinent initial conditions. The test will not be exercising the code in a realistic environment if these initial conditions are incorrect, which reduces the usefulness and precision of unit test results.

Throughout the software development process, strict discipline is required in order to reap the intended benefits from unit testing. It is crucial to keep meticulous records of all changes made to the source code of this or any other unit in the system, as well as the tests that have been run.

Version control software must be used. The version-control software can offer a list of the source code modifications (if any) that have been made to the unit since a previous version of the unit passed a particular test.

If a process to ensure that test case failures are reviewed daily and immediately addressed is not put in place and ingrained into the team's workflow, the application will develop independently of the unit test suite, leading to an increase in false positives and lowering the effectiveness of the test suite.

The issue of unit testing embedded system software is distinct: You cannot easily run a test programme in the real deployment environment, as you do with desktop programmes, because the software is being developed on a separate platform than the one it will finally run on.

Functional testing

Functional testing is a sort of black box testing and a quality assurance (QA) procedure that bases its test cases on the requirements of the software component being tested. Internal programme structure is rarely taken into account while testing functions; instead, input is provided and the results are examined (not like in white-box testing). Functional testing often outlines the system's functionality.

System testing "validates a programme by evaluating it against the published user or system requirements," whereas functional testing "verifies a programme by checking it against... design document(s) or specification(s)" (Kane, Falk, Nguyen 1999, p. 52).

Five steps are commonly included in functional testing. identifying the tasks that the software is intended to carry out

1. Creating input data in accordance with the function's requirements

2. Making output decisions based on the function's specifications

3. the test case's execution

4. The evaluation of actual and anticipated results

**MODULES**

* **USER**

**A user module in a software system is typically responsible for managing user accounts, authentication, and authorization**

* **WORKOUT SCHEDULING**

**This module allows users to set specific fitness goals, such as running a 5K race or losing 10 pounds, and track their progress towards those goals. The module may also include tools to create a personalized fitness plan to help users achieve their goals.**

* **FOOD MANAGEMENT**

**A meal plan or a diet plan, is a tool that helps individuals plan their meals and make healthier food choices.**

* **WEEKLY WORKOUT**

**This module allows users to track their progress towards their fitness goals, such as weight loss, muscle gain, or endurance. The module may include tools to measure body fat percentage, muscle mass, and other biometric data.**

* **WORKOUT BENEFITS:**

**This module allows users to track their exercise routines, including the type of exercise, duration, and intensity. The module may also include pre-set workouts or allow users to create their own workouts.**

**Performance testing**

Performance testing in software engineering generally refers to testing done to ascertain how a system performs in terms of responsiveness and stability under a specific workload. Additionally, it can be used to look into, gauge, confirm, or evaluate other system quality characteristics like scalability, dependability, and resource utilisation.

A subset of performance engineering, which aims to include performance into a system's implementation, design, and architecture, is performance testing.

**Testing types**

### **Load testing**

The most basic type of performance testing is load testing. A load test is typically performed to determine how the system will respond to a particular predicted load. This load might be the anticipated number of people using the programme concurrently, each doing a set number of transactions in the allotted time. The results of this test will reveal the reaction times of all significant, time-sensitive business transactions. This straightforward test can by itself indicate application software bottlenecks if the database, application server, etc. are also monitored.

### **Stress testing**

Stress testing is typically done to determine the system's maximum capacity. This type of test is used to assess the system's robustness under extreme load and aids application administrators in determining if the system will function well if the current load is significantly higher than the anticipated maximum.

Soak testing

To ascertain whether the system can withstand the continuous predicted load, soak testing, also known as endurance testing, is frequently carried out. Memory use is tracked throughout soak tests to look for any potential leaks. Performance deterioration is very crucial yet sometimes disregarded. That is, to make sure that the throughput and/or reaction times are just as excellent as or better than they were at the start of the test.

In essence, it is placing a heavy strain on a system for an extended period of time. The objective is to learn how the system operates under prolonged use.

Spike testing

Spike testing involves abruptly boosting the number of users or the load they produce while monitoring the system's response. Finding out whether performance will degrade, the system will malfunction, or it will be able to manage significant changes in load is the aim.

### **Configuration testing**

Tests are developed to ascertain the impact of configuration changes to the system's components on the system's performance and behaviour rather than testing for performance from the perspective of load. A frequent illustration would be testing with various load-balancing techniques.

Isolation Testing

Isolation testing, which entails repeating a test execution that led to a system issue, is not exclusive to performance testing. used frequently to identify and validate the fault domain.

 Integration testing

The stage of software testing where separate software modules are merged and tested as a unit is known as integration testing (sometimes known as integration and testing, abbreviated I&T). It takes place between unit testing and validation testing. The goal of integration testing is to produce an integrated system that is ready for system testing by taking as input modules that have undergone unit testing, grouping them into bigger aggregates, applying the tests outlined in an integration test plan to those aggregates.

**Purpose**

Integration testing is used to confirm that the key design elements meet the functional, performance, and reliability criteria. Black box testing is used to exercise these "design pieces," which are assemblages (or groupings of units), through their interfaces, simulating success and error instances with the right parameter and data inputs. Individual subsystems are exercised through their input interfaces, and inter-process communication and the simulated use of shared data regions are tested.

After testing individual modules, or unit testing, test cases are created to see whether all the components within assemblages communicate correctly, such as between procedure calls or process activations. The general method is one of "building blocks," where confirmed assemblages are added to a basis that has already been verified and is then utilised to enable the integration testing of more assemblages.

The big bang, top-down, and bottom-up methods of integration testing are a few diverse variations. Other integration patterns include client/server integration, distributed services integration, collaboration integration, backbone integration, layer integration, and high-frequency integration.

Big Bang

This method involves coupling all or the majority of the generated modules to create a full software system or a significant portion of the system, which is then utilised for integration testing. The Big Bang technique is excellent for speeding up the integration testing procedure. However, the integration process will be made more difficult and the testing team may not be able to accomplish the purpose of integration testing if the test cases and their results are not properly documented.

Usage Model testing is a subset of Big Bang Integration testing. Both software and hardware integration testing can make use of usage model testing. Running user-like workloads in integrated user-like environments serves as the foundation for this kind of integration testing. This method of testing directly proofs the environment while indirectly verifying the individual components through use.

Usage Model testing has an upbeat testing philosophy because it anticipates seeing few issues with the separate components. The component developers' ability to perform the isolated unit testing for their product is crucial to the plan. The technique aims to prevent repeating developer testing and instead fleshes out issues brought on by the interaction of the components in the environment.

Usage model testing can be more effective and give better test coverage for integration testing than traditional focused functional integration testing. Care must be taken while designing the user-like workloads to produce realistic scenarios when exercising the environment in order to be more effective and accurate. This inspires confidence that the target customers will experience the integrated environment as planned.

Bottom-up and Top-down

A method of integrated testing known as "bottom up testing" involves evaluating the most basic elements first and then using the results to help test the more complex ones. Up till the component at the top of the hierarchy is tested, the process is repeated.

All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top Down Testing is a method of integrated testing where the top integrated modules are tested first, followed by step-by-step testing of the module's branch all the way to its conclusion.

An approach called "sandwich testing" combines top-down and bottom-up testing.

The primary benefit of the Bottom-Up methodology is that flaws are more readily discovered. It is simpler to locate a missing branch link with Top-Down.

**Validation and Verification**

Verification and validation are separate processes that are used in tandem to make sure that a system, service, or product complies with requirements and specifications and serves the intended purpose. These are essential elements of an ISO 9000-compliant quality management system. Sometimes the phrase "Independent" (or IV&V) comes before the words "verification" and "validation," denoting that the verification and validation are to be carried out by an impartial third party.

It has been suggested that the questions "Are you building the right thing?" and "Are you building it right?" can be used to indicate validation and verification, respectively.

The way that these words are actually used varies. They are even sometimes used interchangeably.

The PMBOK guide, an IEEE standard, defines them as follows in its 4th edition

* "Validation. the confirmation that a system, service, or product satisfies the needs of its target audience. Acceptance and suitability with external clients are frequently involved. Compared to verification"
* "Verification. the assessment of a system's, product's, or service's compliance with a rule, specification, or other requirement. Frequently, it is an internal process. Compare this to validation."
* A product, service, or system's compliance with a set of basic design requirements needs to be verified through verification. Verification methods are carried out throughout the development phase by executing particular tests to model or simulate all or a portion of a system, product, or service, followed by a review or analysis of the modelling outcomes.
* During the post-development phase, verification processes entail repeating tests on a regular basis to make sure the system, service, or product stays in compliance with the original design criteria, specifications, and laws throughout time.
* It is a procedure used to determine whether a system, service, or product complies with rules, requirements, or other conditions set at the beginning of a development phase. Verification may take place during manufacturing, scale-up, or development. This frequently happens inside. A product, service, or system's development and verification processes should produce a product, service, or system (or component thereof, or set thereof) that satisfies the initial requirements. This is the goal of validation.
* Validation processes for new development flows or verification flows may entail modelling both flows and utilising simulations to foresee flaws or gaps that could result in inaccurate or insufficient verification or development of a good, service, or system (or portion thereof, or set thereof).

Then, a development flow or verification flow for a good, service, or system may be approved based on a set of validation specifications, rules, and requirements (or portion thereof, or set thereof). Additional validation procedures also include those created specifically to ensure that changes made to an already qualified development flow or verification flow will result in the production of a product, service, or system (or portion of it, or set thereof), and these validations help to keep the flow qualified. It is a procedure for gathering data that offers a high level of assurance that a system, service, or product satisfies the needs for which it is designed.

* This frequently entails end users and other product stakeholders accepting the product's suitability for the purpose. Frequently, this is an outside procedure.

It has been suggested that the questions "Are you building the right thing?" and "Are you building it right?" can be used to indicate validation and verification, respectively. "Creating it right" verifies that the system is accurately carrying out the specifications, whereas "building the right thing" refers back to the user's demands. In some circumstances, it is necessary to establish formal, written procedures or protocols for determining compliance in addition to stated requirements for both.

It's totally conceivable for a product to pass verification but fail validation.

This may occur, for example, if a product is constructed in accordance with the requirements, but those requirements do not sufficiently satisfy the demands of the user.

**Activities**

Design qualification (DQ), installation qualification (IQ), operational qualification (OQ), and performance qualification are the typical components of machinery and equipment verification (PQ). Normally, vendors are in charge of DQ. However, the user can also do DQ by verifying through inspection and testing that the apparatus complies with the published acquisition specification. The users who work in an industrial regulatory environment must thoroughly complete the latter 3Q if the pertinent document or manuals of machinery/equipment are provided by suppliers. Otherwise, the work of validation is the IQ, OQ, and PQ process. Such a situation is typically shown by the absence or loss of vendor documentation for antique machinery or do-it-yourself (DIY) assemblies (e.g., cars, computers etc,) and users should make an effort to obtain DQ documents in advance.

Each DQ, IQ, OQ, and PQ template is typically available online, while the DIY qualification of machinery and equipment can be aided by the vendor's training course materials and tutorials or by published guidance books, such as step-by-step series, if the purchase of machinery and equipment is not bundled with on-site qualification services. This sort of do-it-yourself strategy can be used to determine whether software, computer operating systems, and manufacturing processes are qualified. The most crucial and vital responsibility is to create and archive machinery/equipment qualification reports for auditing reasons, if regulatory compliances are required, as the final step of the operation.

Machines and equipment must be qualified at each location, especially shock-sensitive items that need for balancing or calibration; qualifying must be repeated once the products have been moved. Some equipment certifications' full scales are even time-dependent as consumables wear out (such as filters) or springs need to be recalibrated, necessitating re-certification whenever a predetermined due period has passed. Re-qualification of equipment/machinery should also be done whenever it has been necessary to replace parts, couple it to another device, instal new software, or rearrange the computer's pre-settings, such as the BIOS, registry, disc drive partition table, dynamically linked (shared) libraries, ini file, etc.

Regardless of whether the parts, devices, or software are authentic or not, the qualification document should still include the specifications of the parts, devices, and software as well as restructuring suggestions.

Torres and Hyman have talked about the acceptability of non-original parts for clinical usage and given instructions for equipment users to choose suitable replacements that can prevent negative consequences. Re-qualification of the non-genuine assemblies is not necessary when some regulatory requirements call for the use of genuine parts, equipment, or software. The asset must be recycled instead for non-regulatory uses.

The process is known as certification when machinery/equipment qualification is carried out by a standard-endorsed third party, such as by an ISO standard authorised business for a specific division. The ISO/IEC 15408 certification by an ISO/IEC 27001 recognised organisation currently has a limited coverage; the scheme needs some work to become more well-known.

System Testing

Software and hardware systems are tested as a whole, integrated system to see whether they adhere to the requirements that have been set forth. Black box testing includes system testing, which should not call for any understanding of the logic or code's internal structure.

System testing typically uses the software system itself combined with any suitable hardware system as well as all of the "integrated" software components that have passed integration testing as its input (s). The goal of integration testing is to find any discrepancies between any of the assemblages—groups of integrated software units—or any of the assemblages and the hardware. A more restricted sort of testing called system testing looks for flaws both in the "inter-assemblages" and in the system as a whole.

In the framework of a Functional Requirement Specification (s) (FRS) and/or a System Requirement Specification, system testing is done on the complete system (SRS). System testing examines not only the design but also user behaviour and even a customer's perceived expectations. Additionally, it is meant to test both within and outside of the parameters specified in the software/hardware requirements specification.

**Types of tests to include in system testing**

The following examples are different types of testing that should be considered during System testing:

* Graphical user interface testing
* Usability testing
* Software performance testing
* Compatibility testing
* Exception handling
* Load testing
* Volume testing
* Stress testing
* Security testing
* Scalability testing
* Sanity testing
* Smoke testing
* Exploratory testing
* Ad hoc testing
* Regression testing
* Installation testing
* Maintenance testing Recovery testing and failover testing.
* Accessibility testing, including compliance with:
  + Americans with Disabilities Act of 1990
  + Section 508 Amendment to the Rehabilitation Act of 1973
  + Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

This list provides as a broad framework or basis to start with, even though various testing organisations may recommend various tests as part of System testing.

**Structure testing:**

This process involves going through specific execution routes and testing a program's internal logic.

Output testing is the process of comparing the outputs of test cases with the predictions made during test case design.

• By asking the user what format they prefer, the system under examination evaluates the output it produces or displays.

• In this case, there are two possible output formats: one is a printed format, and the other is an on-screen format.

• The output displayed on the screen is found to be accurate because the format was created during the system design phase taking user demands into account.

• The output is produced as the user's hard copy and meets the requirements.

**User Acceptance Testing:**

• The last stage, before delivery to the customer, is often completed by the customer and involves running the test cases using real data.

• The system under consideration is tested for user acceptance, and changes are made as needed while staying in close contact with the potential system user throughout development.

• To show that the established software system satisfies the requirements provided in the requirement specification, it entails the planning and execution of various types of tests.

MODULES

User management: This module would manage user accounts, authentication, and authorization, allowing users to create profiles and track their progress over time.

Fitness tracking: This module would track users' physical activity, such as exercise and steps taken, and provide personalized recommendations based on their progress.

Nutrition tracking: This module would track users' nutrition habits, such as calorie intake and nutrient balance, and provide recommendations for a healthier diet.

Risk assessment: This module would assess an individual's risk of developing certain conditions based on their fitness and nutrition habits, and provide personalized recommendations to improve their health.

Recommendation engine: This module would use machine learning algorithms to provide personalized recommendations for fitness and nutrition, taking into account the individual's goals, habits, and risk assessment.

Progress tracking: This module would track users' progress over time and provide regular updates on their fitness and health, helping them to stay motivated and on track.

Reporting and analytics: This module would provide detailed reports and analytics on users' fitness and health, allowing them to see their progress and make informed decisions.

ADVANTAGES

Personalized recommendations: The app provides personalized recommendations for fitness and nutrition based on an individual's goals, habits, and risk assessment, making it easier for them to adopt healthier habits.

Convenient and accessible: The app is available on mobile devices, making it easy to use and accessible from anywhere.

Data tracking and analysis: The app tracks and analyzes data on fitness and health, allowing users to see their progress and make informed decisions.

Improved health outcomes: By promoting healthy habits, the app can help to reduce the risk of developing certain conditions and improve overall health outcomes.

Increased motivation: The app provides regular updates on progress and encourages users to stay on track, helping to increase motivation and drive positive behavior change.

LIMITATIONS

Reliance on self-reported data: The accuracy of the app's recommendations and tracking depends on users accurately reporting their fitness and health data.

Limited data sources: The app's recommendations may be limited by the data sources available, and may not take into account other factors that can impact health, such as genetics or environmental factors.

Technical limitations: The app may have technical limitations, such as compatibility issues or security concerns, that can impact its effectiveness and user experience.

Cost: Depending on the features and technology used, building and maintaining the app can be costly, which may limit its accessibility for some users.

FUTURE ENHANCEMENTS

Integration with wearable technology: The integration of wearable devices, such as fitness trackers, could provide a more comprehensive view of an individual's health and allow for more accurate tracking and analysis.

Artificial intelligence and machine learning: The use of AI and machine learning could be enhanced to provide even more personalized recommendations and to continuously improve the app's accuracy and effectiveness over time.

Virtual reality and augmented reality: The use of virtual reality and augmented reality could provide a more engaging and immersive experience for users, making it easier to stick to their fitness and health goals.

Integration with healthcare systems: Integrating the app with healthcare systems could provide a more comprehensive view of an individual's health and allow for easier tracking of progress and risk for certain conditions.

Gamification: Incorporating game-like elements into the app could make it more fun and engaging for users, encouraging continued use and helping to foster positive habits.

CONCLUSION

In conclusion, Pledge Estimation of Healthcare Fitness prevention of condition Using a Mobile App to Improve Fitness and Health has the potential to make a significant impact on the health and well-being of individuals. By providing personalized recommendations and tracking data on fitness and health, the app can help users to adopt healthier habits and reduce their risk of developing certain conditions.

However, it is important to keep in mind that the specific benefits and limitations of the app will depend on the goals and features of the app, as well as the technology and data available. Additionally, the app's success will largely depend on the willingness of individuals to adopt and stick to the healthy habits recommended by the app.

Overall, Pledge Estimation of Healthcare Fitness prevention of condition Using a Mobile App to Improve Fitness and Health offers a promising solution for promoting healthy behavior and improving health outcomes. With the right development and implementation, it has the potential to make a real difference in the lives of individuals and communities.

REFERENCE:

[1] Bender, Chelsea G., Jason C. Hoffstot, Brian T. Combs, Sara Hooshangi, and Justin Cappos. "Measuring the fitness of fitness trackers." In *2017 IEEE Sensors Applications Symposium (SAS)*, pp. 1-6. IEEE, 2017.

[2] Wu, Qinge, Kelli Sum, and Dan Nathan-Roberts. "How fitness trackers facilitate health behavior change." In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 60, no. 1, pp. 1068-1072. Sage CA: Los Angeles, CA: SAGE Publications, 2016.

[3] Henriksen, André, Martin Haugen Mikalsen, Ashenafi Zebene Woldaregay, Miroslav Muzny, Gunnar Hartvigsen, Laila Arnesdatter Hopstock, and Sameline Grimsgaard. "Using fitness trackers and smartwatches to measure physical activity in research: analysis of consumer wrist-worn wearables." *Journal of medical Internet research* 20, no. 3 (2018): e110.

[4] Lynch, Chris, Stephen Bird, Noel Lythgo, and Isaac Selva-Raj. "Changing the physical activity behavior of adults with fitness trackers: a systematic review and meta-analysis." *American Journal of Health Promotion* 34, no. 4 (2020): 418-430.

[5] Fereidooni, Hossein, Tommaso Frassetto, Markus Miettinen, Ahmad-Reza Sadeghi, and Mauro Conti. "Fitness trackers: fit for health but unfit for security and privacy." In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pp. 19-24. IEEE, 2017.

[6] Becker, Moritz, Andreas Kolbeck, Christian Matt, and Thomas Hess. "Understanding the continuous use of fitness trackers: A thematic analysis." (2017).

[7] Coorevits, Lynn, and Tanguy Coenen. "The rise and fall of wearable fitness trackers." In *Academy of Management*. 2016.

[8] Knight, C. H. "Sensor techniques in ruminants: more than fitness trackers." *Animal* 14, no. S1 (2020): s187-s195.

[9] Yang, Rayoung, Eunice Shin, Mark W. Newman, and Mark S. Ackerman. "When fitness trackers don't'fit' end-user difficulties in the assessment of personal tracking device accuracy." In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 623-634. 2015.

[10] Nuss, Kayla, Kristen Moore, Tracy Nelson, and Kaigang Li. "Effects of motivational interviewing and wearable fitness trackers on motivation and physical activity: A systematic review." *American Journal of Health Promotion* 35, no. 2 (2021): 226-235.

[11] Spiel, Katta, Fares Kayali, Louise Horvath, Michael Penkler, Sabine Harrer, Miguel Sicart, and Jessica Hammer. "Fitter, happier, more productive? The normative ontology of fitness trackers." In *Extended abstracts of the 2018 CHI conference on human factors in computing systems*, pp. 1-10. 2018.

[12] Das, Aveek K., Parth H. Pathak, Chen-Nee Chuah, and Prasant Mohapatra. "Uncovering privacy leakage in BLE network traffic of wearable fitness trackers." In *Proceedings of the 17th international workshop on mobile computing systems and applications*, pp. 99-104. 2016.

[13] Miller, Andrew. "Fitness trackers." *XRDS: Crossroads, The ACM Magazine for Students* 20, no. 2 (2013): 24-26.

[14] Zhu, Yaguang, Stephanie L. Dailey, Daniel Kreitzberg, and Jay Bernhardt. "“Social networkout”: Connecting social features of wearable fitness trackers with physical exercise." *Journal of health communication* 22, no. 12 (2017): 974-980.

[15] Cadmus-Bertram, Lisa. "Using fitness trackers in clinical research: what nurse practitioners need to know." *The journal for nurse practitioners* 13, no. 1 (2017): 34-40.