HEALTH CHECKS
IN
SPRING BOOT

TEAM DETAILS

Team Head: Dr. M. Sureshkumar
Associate Professor/I.T.,
Sri Sairam Engineering College

Team Mentor: Ms. V. Narmadha
Assistant Professor/I.T.,
Sri Sairam Engineering College

Team Members:

1. Theerej C
2. Arathi P
3. Sakthivel M
4. Shiva Sankar
5. Vishwa Raviraj S
6. Anurega T R
7. Kishore M A
8. Rajesh N
9. Prasanth K
10. Guru Sanjay R K
11. Sri Ganesh
12. Sunil Raj B

College: Sri Sairam Engineering College
Chennai

TECHNICAL SOLUTION APPROACH

CONTENTS

1 INTRODUCTION

The pace of technical advancement and innovation in the field of creating applications for the web is accelerating all the time. Monitoring the Workflow and the continuous health of all online services, endpoints, and databases is essential. In this Part, we will look at how the popular Java framework Spring Boot may be altered to simplify the development of web services, endpoints, databases, and other components connected with online and enterprise-level applications. Spring Boot reduces the complexity of preparing Spring apps for production by requiring less configuration. If a program offers a service that can be used to gauge a participant's sense of well-being, then Coherence might benefit from it. With the help of the provided API, an app's developer may add more health checks to their product. By keeping an eye on things, you can make sure that a part is doing what it's supposed to be doing as efficiently as possible. If you're designing microservices, it's especially crucial that you keep an eye on the API environment and fix any issues as soon as possible. The health check API is available for developers to utilize if their apps provide a service that might be used to determine a user's overall health inside the Coherence ecosystem.

1.1 ABOUT THIS DOCUMENT

An overview of the Health Check API and the approach used to ensure that all involved services, endpoints, and other servers are operating at peak health are provided in this paper. It explains not just the flowchart, but also the sequence diagram and the component diagram. There was extensive discussion of several Test Cases, from which a comprehensive solution was built.

1.1.1. PURPOSE AND SCOPE

The major goal of this API development is to track the efficiency of web services and detect issues before they spiral out of control. Restoring the service's operational condition after it has fallen into disrepair will show that it is unable to interact with its dependent services. This module guarantees that the healthcheck API's provided services are completely autonomous and far more versatile than competing frameworks. Data from APIs that examine the health of a service instance may include the timestamps of component execution or of connections to downstream services. Check the health of your services and their dependencies with the help of APIs designed for this purpose. You can keep the parts of your app that rely on failed services up and running with the help of health checks.

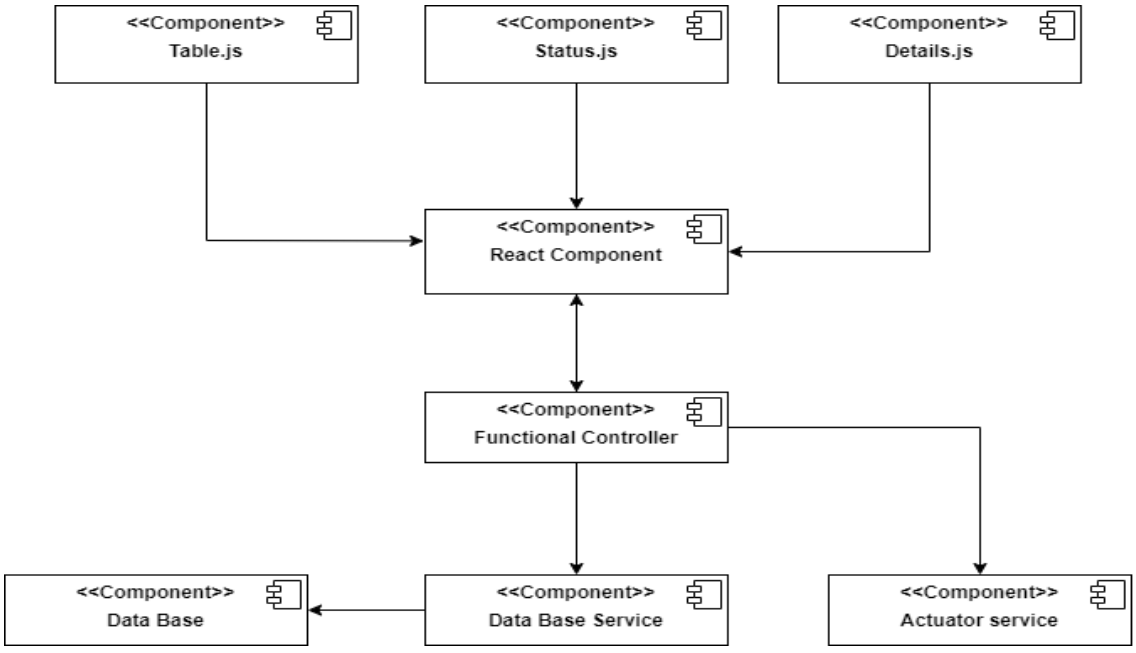
The Scope of the Component:

- This module's primary responsibility is to track and report on the operational state of the application's many endpoints, services, databases, and other modules.
- The actuator will then be the primary factor in determining whether or not the endpoints and other components are healthy or functioning.

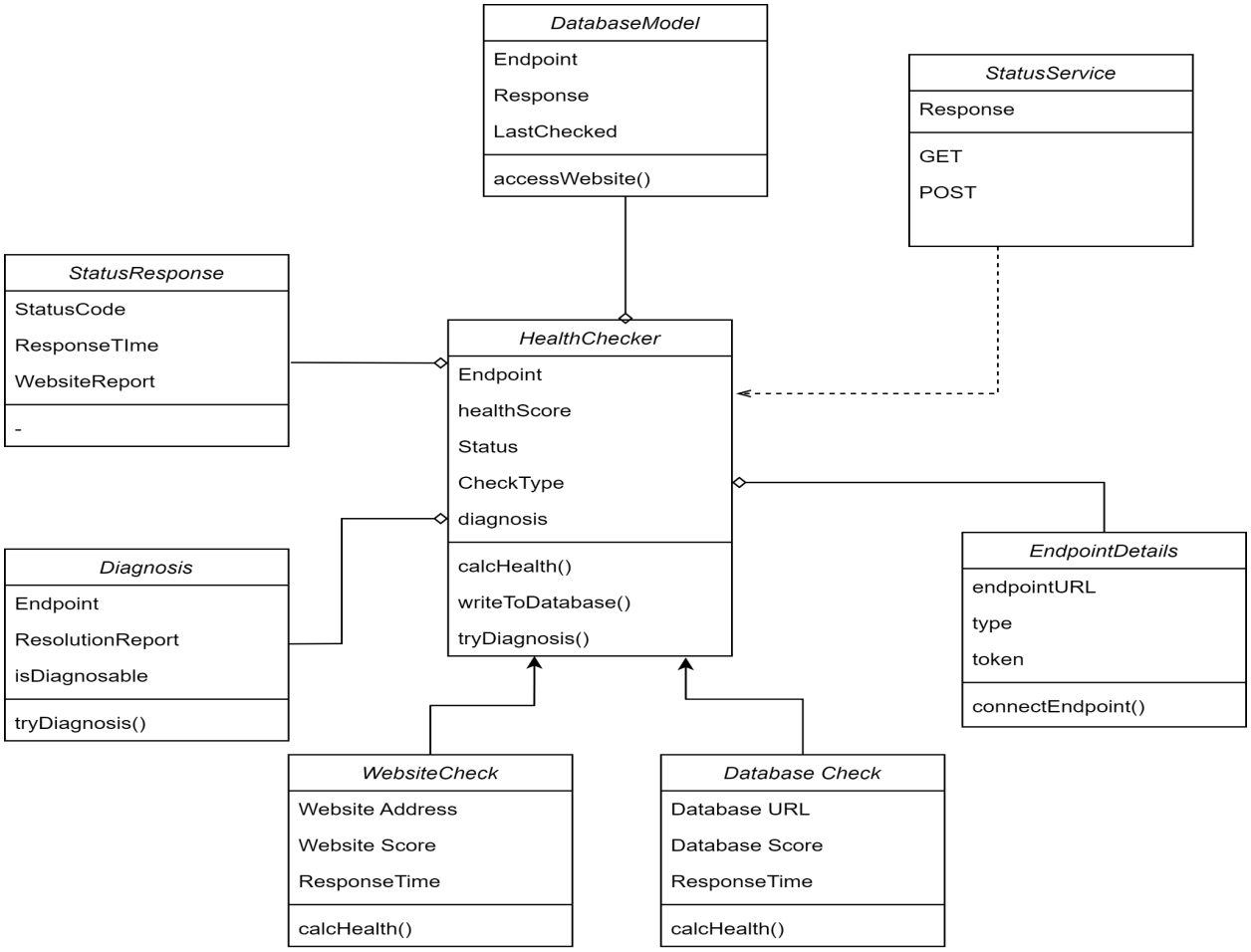
2. COMPONENT DESIGN

2.1 COMPONENT DESIGN DIAGRAM

The Finish points are the primary focus of this section. The user initiates the request, which is then passed to the authorisation module for review. After that, the request is sent to the application's login endpoint. This is the start of the app's backend. After that, the request is routed to the agency responsible for doing the actual health check. The user's request will be received by the UI layer, which will then forward it to the service's designated endpoint. The service layer will then look for the requested app or component. The service is responsible for sending the request and receiving the answer. After checking the request against the logs, the answer is sent out to the client. The outcomes of the tested components are recorded in the logs for further review. The user is then informed of the error's diagnosis.



CLASS DIAGRAM

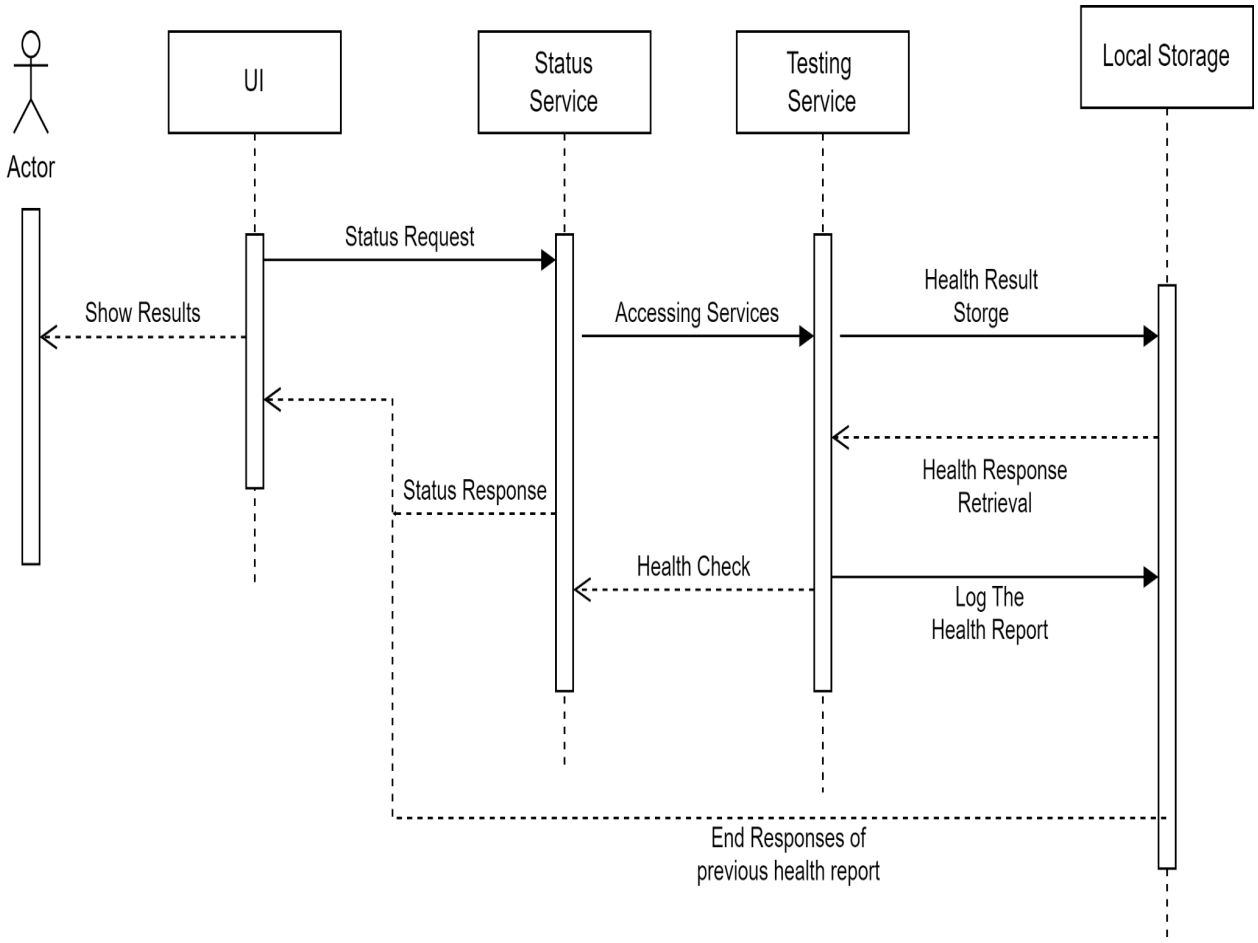


2.1.1 OVERALL WORKFLOW

SEQUENCE DIAGRAM

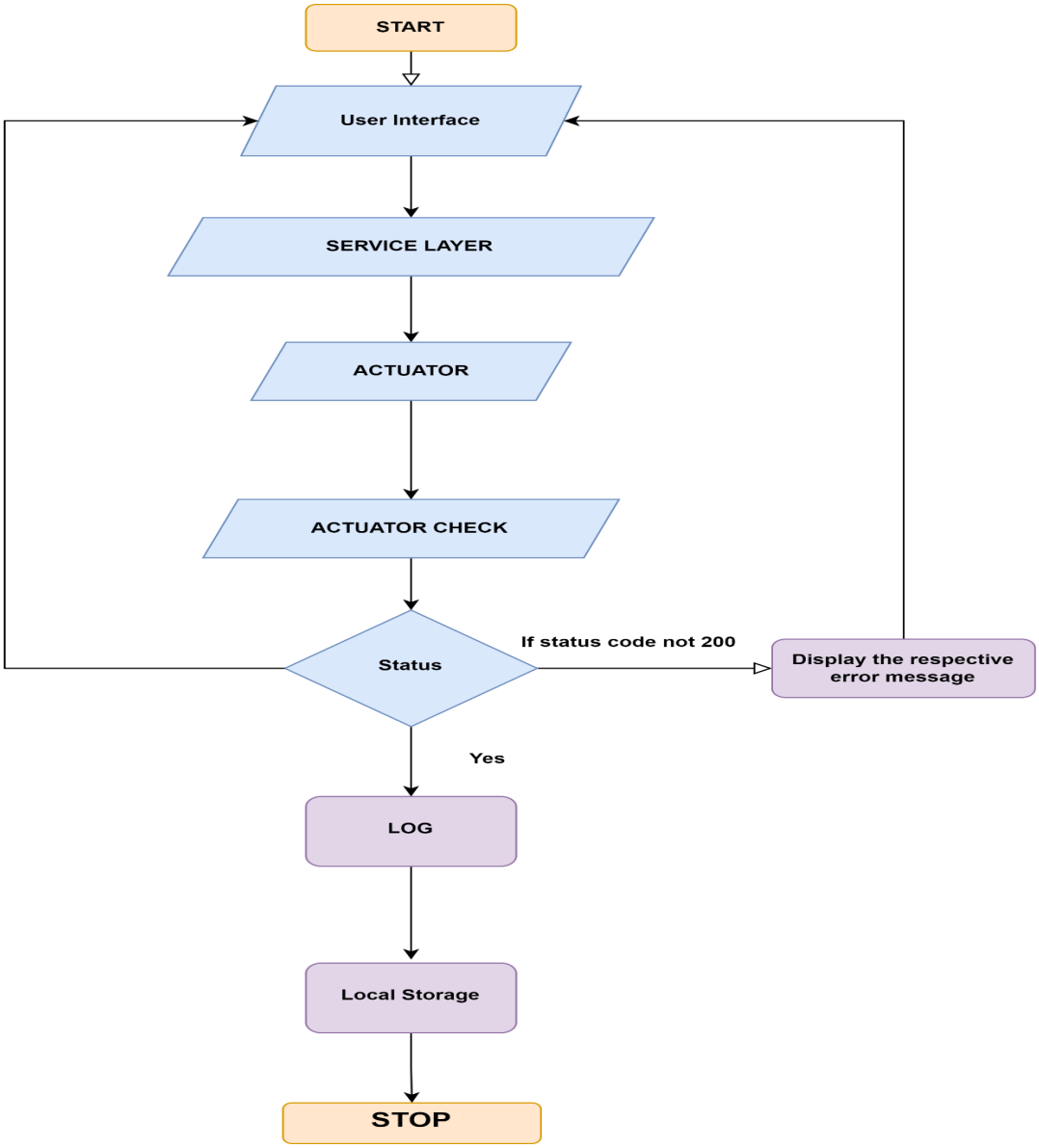
The status service receives the request from the UI and forwards it to the requesting services. This service of access is delivered to the service of testing. Health data is sent to local storage from testing services. The retrieval of health responses is the stage between local storage and the testing service. The testing service also records medical history information. The results of the health check have been obtained from the laboratory. It is sent to the status service, which in turn relays the status answer to the UI. The actor is presented with the output results. The local storage can be of type of log or any file format.

The flow is like: The request comes from the user to the health check endpoint. The actuator checks the health condition. Then the response is sent to the user using the front end.



2.1.2 LOW LEVEL DESIGN

The high-level layout depicts the whole process, from beginning to end. Actuator receives the flow after it has been processed by the UI and the service layer. The signal is sent to the actuator check module through this actuator. If the status code is not 200 or the status is yes, the status has been verified. If so, the signal is sent to the log, then to local storage, finally to the end-of-process state, if applicable. The actuator in the spring boot is an main module which is responsible for health checks. We can access all the end points, databases, components in the application using the actuator.



3 TECHNOLOGY & FRAMEWORKS TO BE USED

1. SpringBoot Java

- To create web applications using the Spring framework, developers often turn to Spring Boot.
- It is meant to streamline the development of standalone, enterprise-ready Spring applications.
- With the many tools available in Spring Boot, application deployment can be performed rapidly and with little effort.
- They include a command-line interface that expedites the creation, execution, and testing of applications; auto-configuration, which automatically configures many of the components required for an application's operation; starters, which give pre-configured dependencies for typical use cases; and so on.
- Several other types of applications, including web apps, microservices, and APIs, are constructed with Spring Boot.
- Because of its user-friendliness and feature-richness, it is often used by programmers who need to create high-quality, scalable programs rapidly.
- Spring Boot is a useful tool for developing current websites since it is compatible with many different frameworks and framework technologies.

2. React

- To put it simply, React is a JavaScript library for developing front-ends.
- ReactJS is a JavaScript toolkit for creating reusable graphical user interface components. It is fast, adaptable, and declarative.
- It's free and solely concerns itself with the presentation layer of applications.
- When building complicated user interfaces, ReactJS's encapsulated components handle state management and other tasks independently.
- As the component functionality is implemented in JavaScript rather than templates, rich data can be sent across the app without exposing the state to the DOM.
- React makes it possible to add new functionality without having to completely rewrite the application.

- Developing faster apps via more responsive user interfaces is ReactJS's major focus.
- Virtual DOM is a JavaScript object that ReactJS utilizes to speed up applications.
- Client-side and server-side usage, as well as integration with other frameworks, are all possible with ReactJS.
- It employs data patterns and components to make huge applications easier to understand and update.

4 SOLUTION APPROACH

A health checker is a diagnostics tool used to evaluate the state of a program or computer network. If everything checks out, you'll get a result saying "system is healthy." When the health check endpoint reports an unfavorable result, the system does a diagnostic to determine what went wrong. Patterns that make huge programs easier to comprehend and manage provide the basis of the solution for developing a health checker using Spring boot.

- Using React.js, design a user interface for health trackers.
- To build a Spring application and include all necessary libraries.
- Create a spring-boot actuator-dependent checkpoint for system health. The current state of an application may be retrieved through a health check endpoint in the form of a JSON object.
- Each program may make use of a health indicator interface, which can be implemented at any point in time, to perform a health check and provide the current health state.
- Add a personalized health check by adding an actuator to the component.
- By submitting a GET request to the predefined endpoints and comparing the replies to the anticipated data, we can test the functionality of the health check endpoint.
- The system or application is in Good Condition if the response is normal. A warning will appear if the answer indicates that the system or application is unhealthy.

When a healthy endpoint reverts to an unhealthy one, the health checker will identify the cause of the change. The Spring Boot problem-solving methodology known as "Solution" is as follows:

- Set up a logging system to keep track of errors and issues.
- If a check for health status at an endpoint yields an unfavorable result, make a note of the issue and any relevant context.
- Problems are found through analyzing log files. The error message may be searched for this information.
- If a serious problem has occurred, it is best to resolve the issue before restarting the program or the computer.
- Repeated checks will be made to ensure a successful health check at each endpoint.
- The API's health may be checked and fixed with the aid of the health check feature.
- This helps in determining whether or not the other program is functioning normally.

TESTING

1. Test case 1: Validate given url
   Input: Check if given url exist and is valid
   Output: Success message

2. Test case 2: URL invalid / does not exist.
   Input: Check if given url exist and is invalid
   Output: Error message saying url does not exist.

3. Test case 3: Malicious url
   Input: Check if the input url is safe
   Output: Block if url is malicious, proceed if safe.

4. Test case 4: Invalid DB
   Input: Giving a invalid DB details
   Output: Output must show unhealthy/does not exist