

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Belagavi-590010**



**MINI PROJECT REPORT**

**ON**

**“ SOIL CONSULTANCY SYSTEM ”**

**Submitted in partial fulfillment for the requirements for the **seven-semester** assignment**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**For the Academic year 2020-2021**

**Submitted by:**

**Sanmati RM**

**1MV17CS097**

**Utkarsh Srivastava**

**1MV17CS116**

**Project carried out at:**

**Sir M. Visvesvaraya Institute Of Technology**

**Bangalore - 562157**

**Under the guidance of:**

**Mr. Suraj Kumar BP**

**Associate Professor, Department Of CSE**

**Sir M. Visvesvaraya Institute Of Technology, Bangalore**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER**

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**

**HUNASAMARANAHALI, BANGALORE- 562157**

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**

**BENGALURU -562157**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING CERTIFICATE**



**CERTIFICATE**

It is certified that the project work entitled “ **SOIL CONSULTANCY SYSTEM** ” is a bonafide work carried out by **Sanmati Rm(1MV17CS097)**, **Utkarsh Srivastava (1MV17CS116)** in partial fulfillment for the requirements of the mini-project for the VII semester curriculum, Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the academic year **2020-2021**. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

**Name & Signature of Guide**

Mr. Suraj Kumar BP  
Associate Prof. & Internal Guide  
Dept. Of CSE, Sir MVIT  
Bengaluru -562157

**Name & Signature of HOD**

Dr. Bhanu Prakash G C  
HOD, Dept of CSE  
Sir MVIT Bengaluru - 562157

**External Examination:**

**Name of the Examiners**

- 1)
- 2)

**Signature with date**

# DECLARATION

We hereby declare that the entire mini-project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: **Bengaluru**

Date: **17/01/2021**

Signature of Students:

**SANMATI RM (1MV17CS097 )**

**UTKARSH SRIVASTAVA (1MV17CS116)**

## **ABSTRACT**

Agricultural productivity highly influences the economy of any country, especially in India where agriculture makes up about 20% of the country's GDP. In such situations, the yield from agricultural land becomes very important. Certain measures must be taken to ensure that best yield is obtained from a piece of land. This can be achieved by ensuring that the right kind of fertilizers are applied to the soil. Using the soil nutrition information, the condition of the soil can be checked and the right amount of fertilizer can be applied. Our project aims to make this soil consultancy system accessible to all the farmers so as to ensure better yield.

# TABLE OF CONTENTS

<b>Certificate</b>	<b>ii</b>
<b>Declaration</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>Table Of Contents</b>	<b>v</b>
<b>List Of Figures</b>	<b>vi</b>
<b>Chapter 1: INTRODUCTION</b>	
1.1 Introduction of the project	7-10
1.2 Objective of the project	10
<b>Chapter 2: REQUIREMENT ANALYSIS</b>	
2.1 About Front End	11-15
2.2 About Back End	16-17
2.3 About Connection	17-18
<b>Chapter 3: SOFTWARE REQUIREMENT &amp; SPECIFICATIONS</b>	
3.1 Hardware Requirements	19
3.2 Software Requirements	19
3.3 Functional Requirements	19-20
<b>Chapter 4: ANALYSIS AND DESIGN</b>	
4.1 Introductions	21
4.2 Design Strategy	21-23
<b>Chapter 5: IMPLEMENTATIONS</b>	
5.1 Table design	24
5.2 Database Structure	25
5.3 Front-End Implementation	25-27
5.4 Back-End Implementation	28-30
5.5 Result & Snapshots	31-32
<b>Chapter 6: TESTING &amp; SNAPSHOTS</b>	<b>33-36</b>

## **LIST OF FIGURES**

<b>Sl.No</b>	<b>Title of Figure</b>	<b>Page No.</b>
1.	A profile of the soil reveals a sequence of horizons	7
2.	Product Architecture and tech stack	21
3.	Database Structure	25
4.	Front-End Implementation	25-27
5.	Back-End Implementation	28-30
6.	Login Page	31
7.	Report Page	31
8.	Final Consultancy Report	32
9.	Testing Snapshots	36

# **CHAPTER 01**

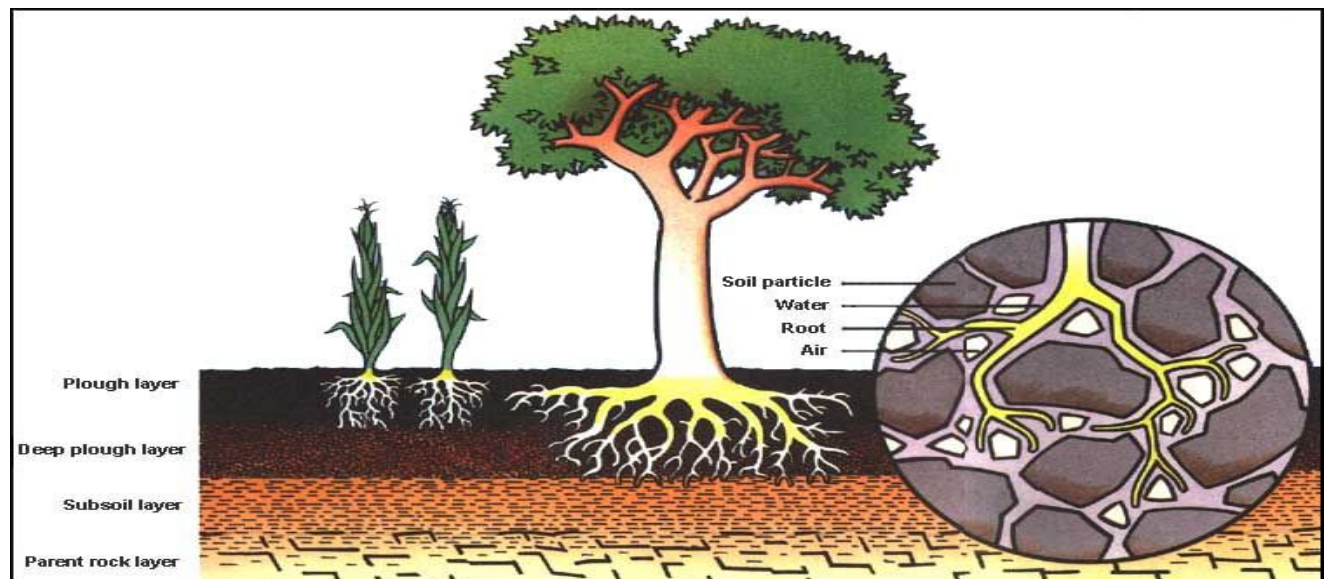
## **INTRODUCTION**

### **1.1 INTRODUCTION TO THE PROJECT**

#### **Soil**

Soil covers most of the land surface of the earth in a thin layer, ranging in thickness from a few centimeters to several meters. It is composed of inorganic matter (rock and mineral particles), organic matter (decaying plants and animals), living plants and animals (many of them microscopic), water, and air.

Soil forms as rocks slowly crumble away. Air and water collect between the particles, and chemical changes occur. Plants take root, binding the particles together, shielding the surface from the elements, drawing up minerals from lower layers, and attracting animal life. Bacteria and fungi break down plants and animals remain into fertile humus. The speed of this process varies. Once destroyed, the soil is for all practical purposes lost forever. Fertile soils teem with life.



**Fig:** A profile of the soil reveals a sequence of horizons, varying in color and texture according to their composition. Plant roots work their way between the soil particles, binding and aerating the soil.

## **What is Agriculture ?**

Agriculture is the science and art of cultivating plants and livestock. Agriculture was the key development in the rise of sedentary human civilization, whereby farming of domesticated species created food surpluses that enabled people to live in cities. The history of agriculture began thousands of years ago. After gathering wild grains beginning at least 105,000 years ago, nascent farmers began to plant them around 11,500 years ago. Pigs, sheep, and cattle were domesticated over 10,000 years ago. Plants were independently cultivated in at least 11 regions of the world. Industrial agriculture based on large-scale monoculture in the twentieth century came to dominate agricultural output, though about 2 billion people still depended on subsistence agriculture into the twenty-first.

## **Present Challenges Facing by existing Agriculture System Used By Farmers**

- Organic matter maintains the soil structure. It also acts as a buffer for chemical fertilizers, adding to their beneficial effects and reducing possible harm. In fact, the organic content and structure of the soil have to be managed as carefully as the nutrient content.
- As agriculture has become more intensive and extensive, mineral fertilizer use has increased. Between 1981 and 1991, the world's annual use of fertilizers rose from 81 to 96 kilograms per hectare of cropland. This average, however, conceals huge differences in usage Zimbabwe, one of Africa's higher users, used only 56 kilograms per hectare a year in 1989-91.
- When fertilizer levels correspond to the needs of specific soils and crops and the structure of the soil is conserved, yields can be sustained indefinitely. Overuse or under-use of fertilizer can lead to crop failures. Over-application can also cause pollution: excess nutrients leach out of the soil into groundwater, streams, rivers, and lakes, making their water unfit for consumption or boosting the growth of algae, which can suffocate entire aquatic ecosystems.
- To grow, plants need nitrogen, phosphorus, potassium, and a range of other elements. However fertile the soil, growing crops will use up its nutrients. Farmers once compensated for this by spreading animal manure and plant waste on their fields. Increasingly, these have been replaced by man-made fertilizers.



## **Importance of Right kind of Fertilizers in Agriculture**

- Just as humans need essential minerals and nutrients for strong, healthy growth, so do the world's crops. Our core business of fertilizer production is intimately connected with agricultural productivity and food production. The role of fertilizers in food production is usually underestimated. Fertilizers are food for plants
- Fertilizers replace the nutrients that crops remove from the soil. Without the addition of fertilizers, crop yields and agricultural productivity would be significantly reduced. That's why mineral fertilizers are used to supplement the soil's nutrient stocks with minerals that can be quickly absorbed and used by crops.
- Thus, to meet human nutritional needs in the crops and meat we eat, we need to replace what we take out. The key is to get this balance right and to maintain a level of nutrients in soils that will support our crops without applying excess.
- Each crop draws down from these reserves and we need to replace them with fertilizers, every year and after every crop.
- Fertilizers increase plants' tolerance towards pests. This has reduced their reliance on insecticides and herbicides, thereby, producing healthier crops. Consequently, diseases have reduced, providing aesthetic value to the crops.
- Fertilizers improve the water holding capacity of the plants and increase root depth. The potassium content present in the fertilizers strengthens the straws and stalks of the plants.
- The phosphorus present in the fertilizers helps in the faster development of roots and the formation of seeds in the plants. Nitrogen in the fertilizers enhances the growth of the plants which can be characterized by the green color of the plants.
- Since the chemical fertilizers adversely affect soil fertility, biofertilizers were brought into use. These are substances that contain living or latent cells, and even micro-organisms. They provide the soil with the necessary nutrients and microbes for the growth of the plants. They help the soil to retain its fertility. They are environment-friendly and also destroy pathogenic components responsible for causing disease in plants. Acetobacter and Rhizobium are two such widely used biofertilizers.
- Put simply – we use fertilizer to:
  - ➔ Provide nutrients not available in the soil.
  - ➔ Replace nutrients removed at harvest.
  - ➔ Balance nutrients for better product quality and higher yield.

## **Why Soil must be checked and diagnosed properly before putting fertilizer ?**

- Soil testing is important before applying phosphorus and other nutrients to determine the critical level and if it is below the critical level you need to calculate the amount of P fertilizer to make up for sustainable yield of crops but if it is above the critical level, there is no need to apply extra P fertilizer because it will lead to nutrient antagonism between P/Mg, N/P Ca/P and may affect the uptake of Mg, Ca in soils.
- It must be noted that the application of fertilizers to soils without a soil test is like a medical doctor prescribing drugs to a sick patient without a thorough clinical examination. Besides, a soil test will reduce the cost of fertilizers to apply, decreases pollution of the environment, and enhances the stability of soil equilibrium.

## **Proposed System**

The proposed system is having many added advantages which is having a higher number of Economic advantages for Farmers. This system modular will allow the farmer to get a detailed analysis of the soil in a simple report that can be easily understood by the farmer, thus eliminating the need to be dependent on other individuals and authorities for help. When the farmer obtains the analysis for the composition of the soil structure, then the farmer can directly use the system for evaluating the results to get the proper quantities of the fertilizers and manure needed to be added in the soil for greater and improved yield.

## **1.2 OBJECTIVES TO THE PROJECT**

The spread of internet facilities and technology around the world has led to availability of smart phones with internet connections becomes available to almost every person around the world. This helps the soil consultancy system to have the following advantages

- Consultancy is done free of cost, which makes it more accessible and affordable.
- No need for physical meetings which results in reduction of travelling time and expenditure.
- Accurate Consultancy which can be trusted.
- More reach to farmers, there are no theoretical geo-graphic limitations.
- Safety of customer privacy is ensured, no need to log-in and provide personal details.

## **CHAPTER 02**

### **FRONT END AND BACK END**

#### **2.1 About Front End**

##### **HTML**

- Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delimited by tags, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.
- HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), the former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

## CSS

- Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file and reduce complexity and repetition in the structural content.
- Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices.
- The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.
- The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

## JavaScripts

- JavaScript often abbreviated as JS, is a high-level, just-in-time compiled, object-oriented programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.
- Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it.
- As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.
- Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.
- Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design. JavaScript was influenced by programming languages such as Self and Scheme. The JSON serialization format, used to store data structures in files or transmit them across networks, is based on JavaScript.

## ReactJS

- It is also known as React.js or ReactJS is an open-source, front-end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.
- However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing. React Router is an example of such a library.
- Some of the React Features includes :
  - ➔ **JSX** – JSX is a JavaScript syntax extension. It isn't necessary to use JSX to React to development, but it is recommended.
  - ➔ **Components** – React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects. Unidirectional data flow.
  - ➔ **Flux** – React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps to keep your data unidirectional.
  - ➔ **License** – React is licensed under Facebook Inc. Documentation is licensed under CC BY 4.0
- React Advantages:
  - ➔ Uses virtual DOM which is a JavaScript object. This will improve apps performance since JavaScript virtual DOM is faster than the regular DOM.
  - ➔ Can be used on the client and server-side as well as with other frameworks.
  - ➔ Components and data patterns improve readability, which helps to maintain larger apps.
- React Limitations:
  - ➔ Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
  - ➔ Uses inline templating and JSX, which might seem awkward to some developers.

## **Material-UI**

- Material design is a design language developed by Google to help designers and end-users replicate Google's work, and explain why things in Google look and respond the way they do.
- What is material design?
  - ➔ Material design is a design language developed by Google which, at its core, is an extremely sophisticated and well-defined set of guidelines to help both designers and end-users replicate Google's work as well as explain why things in Google look and respond the way they do.
- The goal of material design is to better unite fundamental design principles with technology.
- Material design has three core principles:
  1. Material is the metaphor Our on-screen design should be a metaphor for off-screen things, especially pen and paper. This means that on-screen buttons should look like real-life buttons, elements should have shadows, and "the fundamentals of light, surface, and movement" need to be respected. Material design uses the idea that user experience is enhanced when elements on a screen mirror (to an extent) how those things look in the off-screen world. For example, a pop-up ad – should have a subtle shadow, because all 3D objects have shadows.
  2. Bold, graphic, and intentional Everything that we design should be deliberate and big. No more wishy-washy colors like beige. Be bold! Use magenta! The takeaway here is that material design aims to guide designs that make sense straight away, is easy to follow (e.g. bold colors, headlines), and creates a clean and uncluttered experience (for example, with negative space).
  3. Motion provides meaning Motion can be used to help move the user and create meaning, in particular, movement is especially useful in providing feedback to the user. For example, on Android, when you hold down an icon, it pops out a little to tell you that it's ready to be moved. That small movement is a great way to communicate with the user. That's what material design aims for.
- Importance:

First, material design is important in the context of flat design. For years, the material design represents the first major shift away from that trend, although parts of it like the principle of material being the metaphor even pushes back towards early-2000s skeuomorphic design. Second, material design approaches design on screens from a much more tactile direction than other design languages or approaches have before. The material design guidelines place all objects on the screen within a 3D matrix, rather than 2D. The last reason that material design is a big deal is its ability to span screen sizes.

## 2.2 About Back End

### Cassandra

- Avinash Lakshman, one of the authors of Amazon's Dynamo, and Prashant Malik initially developed Cassandra at Facebook to power the Facebook inbox search feature. Facebook released Cassandra as an open-source project on Google code in July 2008. In March 2009 it became an Apache Incubator project. On February 17, 2010, it graduated to a top-level project.
- Apache Cassandra is a free and open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients. Cassandra offers the distribution design of Amazon DynamoDB with the data model of Google's Bigtable.
- The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

- Main features:

#### → **Distributed**

Every node in the cluster has the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.

#### → **Supports replication and multi-datacenter replication**

Replication strategies are configurable. Cassandra is designed as a distributed system, for the deployment of large numbers of nodes across multiple data centers. Key features of Cassandra's distributed architecture are specifically tailored for multiple-data center deployment, redundancy, failover, and disaster recovery.

#### → **Scalability**

Designed to have read and write throughput both increase linearly as new machines are added, with the aim of no downtime or interruption to applications.



→ **Fault-tolerant**

Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

→ **Tunable consistency**

Cassandra is typically classified as an AP system, meaning that availability and partition tolerance are generally considered to be more important than consistency in Cassandra. Writes and reads offer a tunable level of consistency, all the way from "writes never fail" to "block for all replicas to be readable", with the quorum level in the middle.

→ **MapReduce support**

Cassandra has Hadoop integration, with MapReduce support. There is support also for Apache Pig and Apache Hive.

→ **Query language**

Cassandra introduced the Cassandra Query Language (CQL). CQL is a simple interface for accessing Cassandra, as an alternative to the traditional Structured Query Language (SQL).

→ **Eventual Consistency**

Cassandra manages the eventual consistency of reads, upserts, and deletes through Tombstones.

## **2.3 About Connection**

### **REST APIs using Spring Boot**

- Spring Boot is an open-source Java-based framework used to create a micro Service. It is developed by the Pivotal Team and is used to build stand-alone and production-ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.
- Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.

- Advantages

Spring Boot offers the following advantages to its developers –

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

- Goals

Spring Boot is designed with the following goals –

- To avoid complex XML configuration in Spring
- To develop production-ready Spring applications in an easier way
- To reduce the development time and run the application independently
- Offer an easier way of getting started with the application.

- We choose Spring Boot because of the features and benefits it offers as given here –

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto-configured; no manual configurations are needed.
- It offers an annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

## **CHAPTER 03**

### **SPECIFICATIONS**

#### **3.1 HARDWARE REQUIREMENTS**

- PROCESSOR : INTEL i3 2.0 OR HIGHER VERSION
- RAM : MINIMUM 2GB
- HARD DISK : 60GB OR ABOVE

#### **3.2 SOFTWARE REQUIREMENTS**

- SOFTWARE (SERVER) : XAMPP
- SUPPORTED BROWSERS : Google Chrome/ Mozilla Firefox/ Internet Explorer
- Atom
- Windows 7/ 8.1/10/ Vista

#### **3.3 Functional Requirements**

The Functional Requirements refer to anything that might be a customer-facing or admin-facing feature or piece of functionality. Some examples include wishlist management, product zooming, social sharing buttons, currency conversion, etc. These features are addressed either natively through our client's selected e-commerce platform, via a third-party app/plugin, or custom-built directly into their website. The cost/time associated with Functional Requirements is simply a reflection of how complicated it might be to implement those features within the website.

### **Following Factors are used to measure software development quality:**

Each attribute may be the accustomed measure of the product performance. These attributes may be used for Quality assurance similarly to quality control. Quality assurance activities are directed towards the prevention of the introduction of defects and internal control activities are aimed toward detecting defects in products and services.

#### **1. Reliability**

Measure if the product is reliable enough to sustain in any condition. Give systematically correct results. Product dependability is measured in terms of the operation of the project underneath different operating atmospheres and different conditions.

#### **2. Maintainability**

Different versions of the product ought to be easy to maintain. For development, it ought to be easy to feature code to the existing system, and ought to be easy to upgrade for brand new options and new technologies from time to time. Maintenance ought to be value effective and simple. The system is easy to take care of and correct defects or make a change within the software system.

#### **3. Usability**

This can be measured in terms of ease of use. The application should be user-friendly. Easy to use for input preparation, operation, and also for interpreting of output.

#### **4. Portability**

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioural issues related to porting.

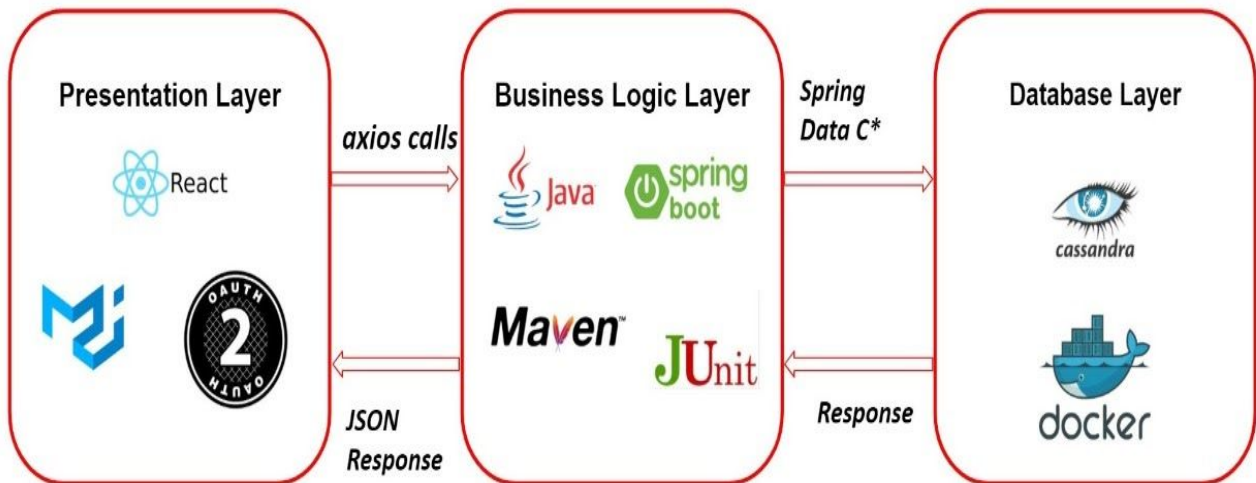
## CHAPTER 4

### ANALYSIS & SYSTEM DESIGN

#### 4.1 INTRODUCTION

System design is the first design stage for devising the basic approach to solving the problem. During system design, developers decide the overall structures and styles. The system architecture determines the organization of the system into subsystems. In addition, the architecture provides the context for the detailed decisions that are made in later stages. During design, developers make decisions about how the problem will be solved, first at the high level and then with more detail.

#### 4.2 DESIGN STRATEGY



**Fig: Product architecture and tech stack**

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This added flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system.

For example, the user interface of a web application could be redeveloped or modernized without affecting the underlying functional business and data access logic underneath. This architectural system is often ideal for embedding and integrating 3rd party software into an existing application. This integration flexibility also makes it ideal for embedding analytics software into pre-existing applications and is often used by embedded analytics vendors for this reason. 3-tier architectures are often used in cloud or on-premises based applications as well as in software-as-a-service (SaaS) applications.

- **Presentation Tier-** The presentation tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.
- **Application Tier-** The application tier contains the functional business logic which drives an application's core capabilities. It's often written in Java, .NET, C#, Python, C++, etc.
- **Data Tier-** The data tier comprises the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.

### **Benefits of Using a 3-Layer Architecture**

There are many benefits to using a 3-layer architecture including speed of development, scalability, performance, and availability. As mentioned, modularizing different tiers of an application gives development teams the ability to develop and enhance a product with greater speed than developing a singular code base because a specific layer can be upgraded with minimal impact on the other layers. It can also help improve development efficiency by allowing teams to focus on their core competencies. Many development teams have separate developers who specialize in front- end, server back-end, and data back-end development, by modularizing these parts of an application you no longer have to rely on full stack developers and can better utilize the specialties of each team.

Scalability is another great advantage of a 3-layer architecture. By separating out the different layers you can scale each independently depending on the need at any given time. For example, if you are receiving many web requests but not many requests which affect your application layer, you can scale

your web servers without touching your application servers. Similarly, if you are receiving many large application requests from only a handful of web users, you can scale out your application and data layers to meet those requests without touching your web servers. This allows you to load balance each layer independently, improving overall performance with minimal resources. Additionally, the independence created from modularizing the different tiers gives you many deployment options. For example, you may choose to have your web servers hosted in a public or private cloud while your application and data layers may be hosted onsite. Or you may have your application and data layers hosted in the cloud while your web servers may be locally hosted, or any combination thereof.

By having disparate layers you can also increase reliability and availability by hosting different parts of your application on different servers and utilizing cached results. With a full stack system you have to worry about a server going down and greatly affecting performance throughout your entire system, but with a 3-layer application, the increased independence created when physically separating different parts of an application minimizes performance issues when a server goes down.

## **CHAPTER 5**

### **IMPLEMENTATION**

#### **5.1 Table Design**

```
CREATE TABLE soilks.report (  
  reportid uuid PRIMARY KEY,  
  boron float,  
  calcium float,  
  chlorine float,  
  copper float,  
  crop text,  
  iron float,  
  magnesium float,  
  manganese float,  
  molybdenum float,  
  nickel float,  
  nitrogen float,  
  phosphor float,  
  potassium float,  
  sulphur float,  
  zinc float  
) WITH bloom_filter_fp_chance = 0.01  
  AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}  
  AND comment = '' AND compaction = {'class':  
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32',  
'min_threshold': '4'} AND compression = {'chunk_length_in_kb': '64', 'class':  
'org.apache.cassandra.io.compress.LZ4Compressor'}  
  AND crc_check_chance = 1.0  
  AND dclocal_read_repair_chance = 0.1  
  AND default_time_to_live = 0  
  AND gc_grace_seconds = 864000  
  AND max_index_interval = 2048  
  AND memtable_flush_period_in_ms = 0  
  AND min_index_interval = 128  
  AND read_repair_chance = 0.0  
  AND speculative_retry = '99PERCENTILE';
```



## 5.2 Database Structure

```
docker exec -it a17aba2b77c6d1714886244013b00d8d89457bdc0b27cbeec7910fe23afe /bin/sh
cqlsh:soilks> desc soilks;

CREATE KEYSPACE soilks WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'} AND durable_writes = true;

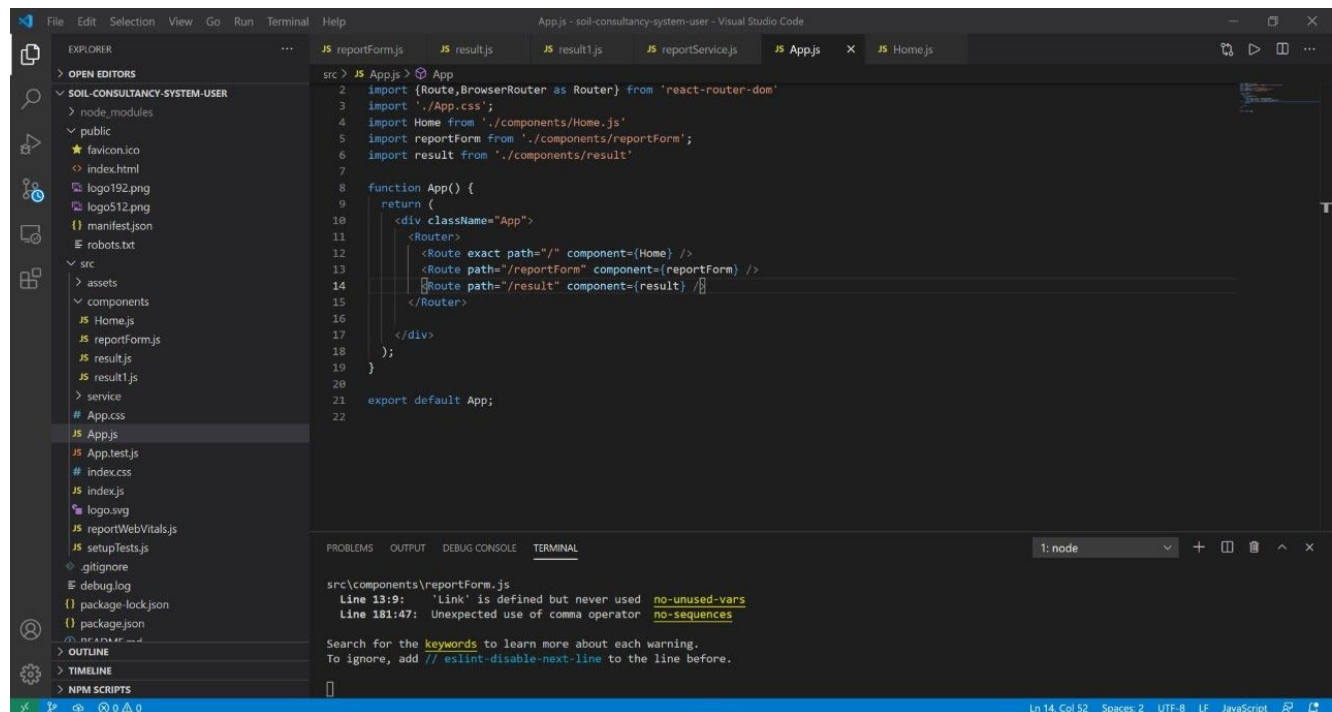
CREATE TABLE soilks.report (
  reportid uuid PRIMARY KEY,
  boron float,
  calcium float,
  chlorine float,
  copper float,
  crop text,
  iron float,
  magnesium float,
  manganese float,
  molybdenum float,
  nickel float,
  nitrogen float,
  phosphor float,
  potassium float,
  sulphur float,
  zinc float
) WITH bloom_filter_fp_chance = 0.01
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
AND comment = ''
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
AND crc_check_chance = 1.0
AND dclocal_read_repair_chance = 0.1
AND default_time_to_live = 0
AND gc_grace_seconds = 864000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair_chance = 0.0
AND speculative_retry = '99PERCENTILE';

cqlsh:soilks> select * from report;

reportid | boron | calcium | chlorine | copper | crop | iron | magnesium | manganese | molybdenum | nickel | nitrogen | phosphor | potassium | sulphur | zinc
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
561809ac-26d9-419b-82a3-c8c7f51b46b4 | 150 | 100 | 125 | 89 | Ragl | 20 | 80 | 56 | 20 | 102 | 40 | 20 | 10 | 60 | 101
427312ba-59c5-4e33-80f1-6835b1d5516d | 150 | 100 | 125 | 89 | Sugarcane | 20 | 80 | 56 | 20 | 102 | 100 | 40 | 50 | 60 | 101
4b982658-19ad-42af-9628-5497d130f377 | 150 | 100 | 125 | 89 | Tomato | 20 | 80 | 56 | 20 | 102 | 100 | 100 | 100 | 60 | 101
06833639-bc97-46a2-9c1e-a72989cca992 | 150 | 100 | 125 | 89 | Paddy | 20 | 80 | 56 | 20 | 102 | 40 | 20 | 20 | 60 | 101
d742731e-e1f8-4c27-904a-fc7c9c1859cb | 150 | 100 | 125 | 89 | Maize | 20 | 80 | 56 | 20 | 102 | 60 | 30 | 40 | 60 | 101

(5 rows)
cqlsh:soilks>
```

## 5.3 Front-End Implementation Code



```
File Edit Selection View Go Run Terminal Help
App.js - soil-consultancy-system-user - Visual Studio Code

EXPLORER
> OPEN EDITORS
> SOIL-CONSULTANCY-SYSTEM-USER
  > node_modules
  > public
  > favicon.ico
  > index.html
  > logo192.png
  > logo512.png
  > {} manifest.json
  > {} robots.txt
  > src
    > assets
    > components
      > JS Home.js
      > JS reportForm.js
      > JS result.js
      > JS result1.js
      > service
    > App.css
    > App.js
    > App.test.js
    > index.css
    > index.js
    > logo.svg
    > reportWebVitals.js
    > setupTests.js
  > .gitignore
  > debug.log
  > {} package-lock.json
  > {} package.json
  > .prettierrc
  > OUTLINE
  > TIMELINE
  > NPM SCRIPTS

src > JS App.js > App
2 import {Route,BrowserRouter as Router} from 'react-router-dom'
3 import './App.css';
4 import Home from './components/Home.js'
5 import reportForm from './components/reportForm';
6 import result from './components/result'
7
8 function App() {
9   return (
10     <div className="App">
11       <Router>
12         <Route exact path="/" component={Home} />
13         <Route path="/reportForm" component={reportForm} />
14         <Route path="/result" component={result} />
15       </Router>
16     </div>
17   );
18 }
19
20 export default App;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

src\components\reportForm.js

Line 13:9: 'Link' is defined but never used [no-unused-vars](#)

Line 181:47: Unexpected use of comma operator [no-sequences](#)

Search for the [keywords](#) to learn more about each warning.

To ignore, add `// eslint-disable-next-line` to the line before.

Ln 14, Col 52 Spaces: 2 UTF-8 LF JavaScript

```
src > service > JS reportService.js > reportService > retrieveAllReport
1 import axios from 'axios';
2
3 const REPORT = 'http://localhost:8082/soil-consultancy-system'
4
5 class reportService {
6
7   createReport(report)
8   {
9     return axios.post(`${REPORT}/report`, report);
10  }
11
12  retrieveAllReport()
13  {
14    return axios.get(`${REPORT}/reports`);
15  }
16
17  retrieveReportById(id)
18  {
19    return axios.get(`${REPORT}/report/${id}`);
20  }
21 }
22 export default new reportService();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

src\components\reportForm.js  
Line 13:9: 'Link' is defined but never used [no-unused-vars](#)  
Line 181:47: Unexpected use of comma operator [no-sequences](#)

Search for the **keywords** to learn more about each warning.  
To ignore, add `// eslint-disable-next-line` to the line before.

Ln 13, Col 47 Spaces: 4 UTF-8 CRLF JavaScript

```
src > components > JS reportForm.js > reportForm > render > width
283 value=(this.state.chlorine)
284 />
285 <TextValidator
286 style={{ width: 500 }}
287 label='Nickel(ppm)'
288 onChange=(this.handleChange('nickel'))
289 value=(this.state.chlorine)
290 />
291 </Card>
292 </Grid></Grid>
293 <Grid container spacing={10} style={{marginLeft:'32px'}}>
294 <Grid item>
295 <Card className={classes.card3}>
296 <Typography variant='h6' style={{align:'center',color:'black'}}>CROP TYPE</Typography>
297 <TextValidator
298 style={{width:400}}
299 select
300 label='Crop Type'
301 onChange=(this.handleChange('crop'))
302 value=(this.state.crop)
303 >
304 {Crop.map(option => {
305   <MenuItem key={option.value} value={option.value}>{option.label}</MenuItem>
306 })}
307 </TextValidator>
308 </Card>
309 </Grid>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

src\components\reportForm.js  
Line 13:9: 'Link' is defined but never used [no-unused-vars](#)  
Line 181:47: Unexpected use of comma operator [no-sequences](#)

Search for the **keywords** to learn more about each warning.  
To ignore, add `// eslint-disable-next-line` to the line before.

Ln 298, Col 36 Spaces: 2 UTF-8 CRLF JavaScript

```
src > components > JS Homejs > Home > render
7 import Button from '@material-ui/core/Button';
8 import Typography from '@material-ui/core/Typography';
9
10 class Home extends React.Component{
11
12   render(){
13     return(
14       <div className="home">
15         <Card className="Card" style={{backgroundColor: '#c8e6c9'}}>
16           <CardActionArea>
17             <CardContent>
18               <Typography variant="subtitle1" color="textSecondary">
19                 Don't know what fertilizer to use to increase your crop's yield? Click here to find out!
20               </Typography>
21             </CardContent>
22           </CardActionArea>
23           <CardActions>
24             <Button variant="contained" color="primary" style={{marginLeft: '35%'}} href="/reportForm">Let's go</Button>
25           </CardActions>
26         </Card></div>
27       )
28     }
29   }
30 }
31
32 export default Home;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

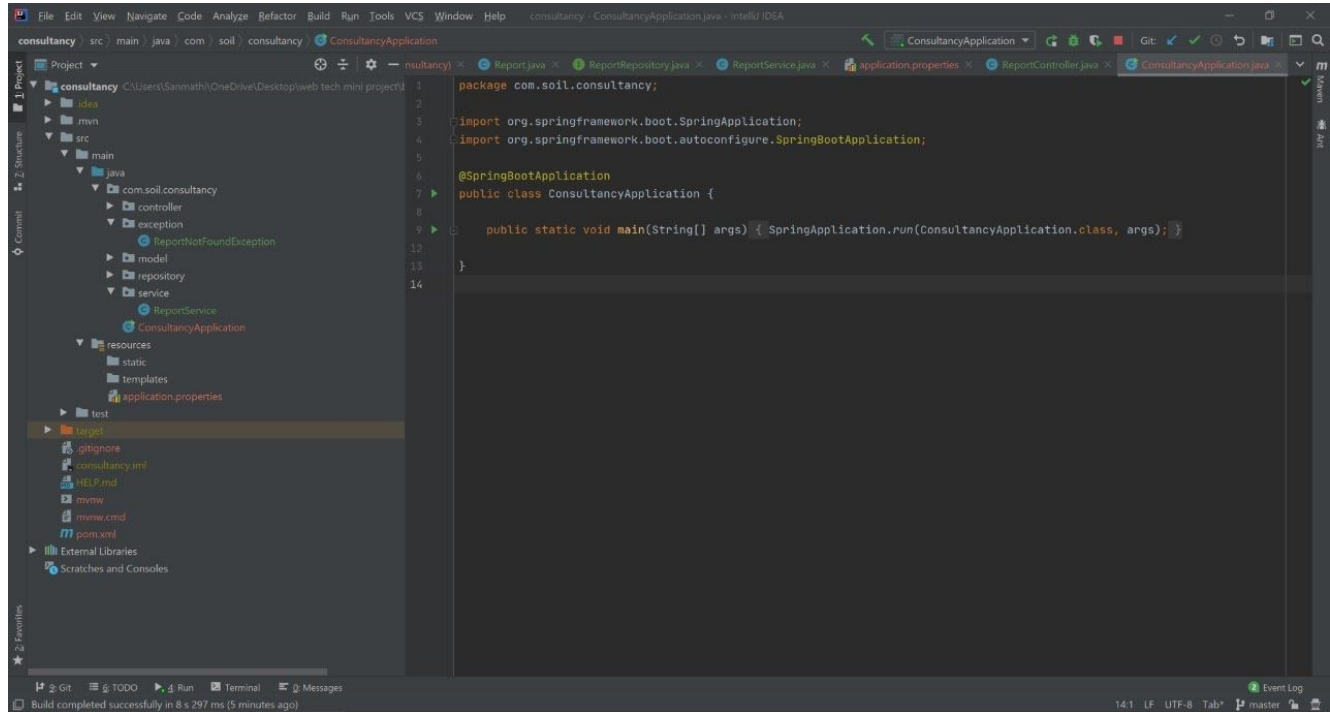
src\components\reportForm.js  
Line 13:9: 'Link' is defined but never used [no-unused-vars](#)  
Line 181:47: Unexpected use of comma operator [no-sequences](#)

Search for the **keywords** to learn more about each warning.  
To ignore, add `// eslint-disable-next-line` to the line before.

```
src > components > JS resultjs > default
27 });
28
29
30 class result extends React.Component{
31
32   constructor(props){
33     super(props);
34     this.state={
35       reports:[]
36     }
37   }
38
39   render(){
40     const { classes } = this.props;
41     return(
42       <div>
43         <AppBar position="static">
44           <Toolbar>
45             <Typography variant="h5" className={classes.title}> Soil Consultancy System</Typography>
46           </Toolbar>
47         </AppBar>
48         <Typography variant="h5" style={{marginTop:100}}><b>CONSULTANCY REPORT</b></Typography>
49         <Grid container spacing={6} style={{marginLeft:30, marginTop:20}}>
50           <Grid item>
51             <Card className={classes.card}>
52               <Typography variant="h6" className={classes.typo} style={{marginTop:10}}><b>Nitrogen</b></Typography>
53               <Typography variant="body1" className={classes.typo}>Crop Type: Sugarcane</Typography>
54               <Typography variant="body1" className={classes.typo}>Amount of Nitrogen Present : 100 kg/acre</Typography>
55               <Typography variant="body1" className={classes.typo}>Ideal dose of Nitrogen required : 112-228 kg/acre</Typography>
56               <Typography variant="body1" className={classes.typo} style={{color: 'red'}}><b>Deficiency</b></Typography>
57               <Typography variant="body1" className={classes.typo}>Fertilizer to be applied : Urea (46% N)</Typography>
58               <Typography variant="body1" className={classes.typo}>Amount of Fertilizer to be applied = ((100+10)*10
59             </Card>
60           </Grid>
61           <Grid item>
62             <Card className={classes.card}>
63               <Typography variant="h6" className={classes.typo} style={{marginTop:10}}><b>Phosphorus</b></Typography>
64               <Typography variant="body1" className={classes.typo}>Crop Type: Sugarcane</Typography>
65               <Typography variant="body1" className={classes.typo}>Amount of Phosphorus Present : 100 kg/acre</Typography>
66               <Typography variant="body1" className={classes.typo}>Ideal dose of Phosphorus required : 112-228 kg/acre</Typography>
67               <Typography variant="body1" className={classes.typo} style={{color: 'red'}}><b>Deficiency</b></Typography>
68               <Typography variant="body1" className={classes.typo}>Fertilizer to be applied : Urea (46% N)</Typography>
69               <Typography variant="body1" className={classes.typo}>Amount of Fertilizer to be applied = ((100+10)*10
70             </Card>
71           </Grid>
72         </Grid>
73       </div>
74     )
75   }
76 }
```

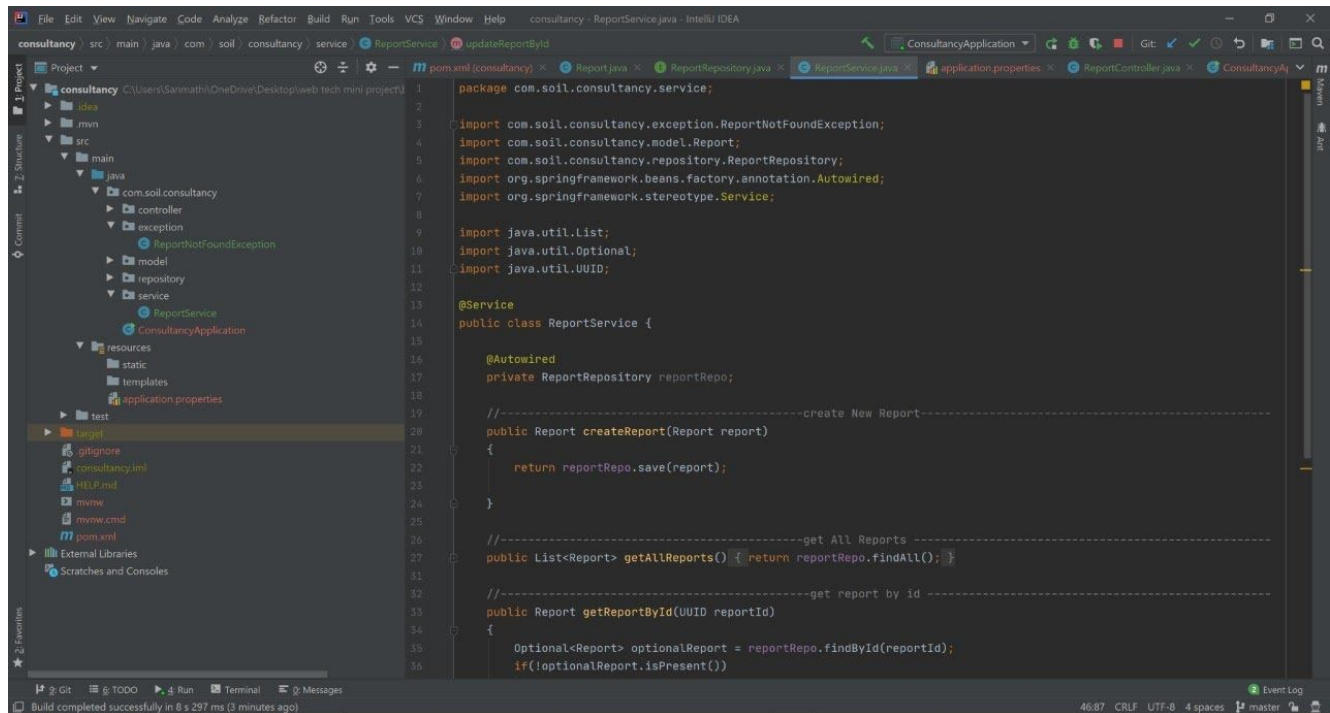
Ln 92, Col 44 Spaces: 4 UTF-8 CRLF JavaScript

## 5.4 Back-End Implementation Code



```
1 package com.soil.consultancy;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class ConsultancyApplication {
8
9     public static void main(String[] args) { SpringApplication.run(ConsultancyApplication.class, args); }
10
11 }
12
13
14
```

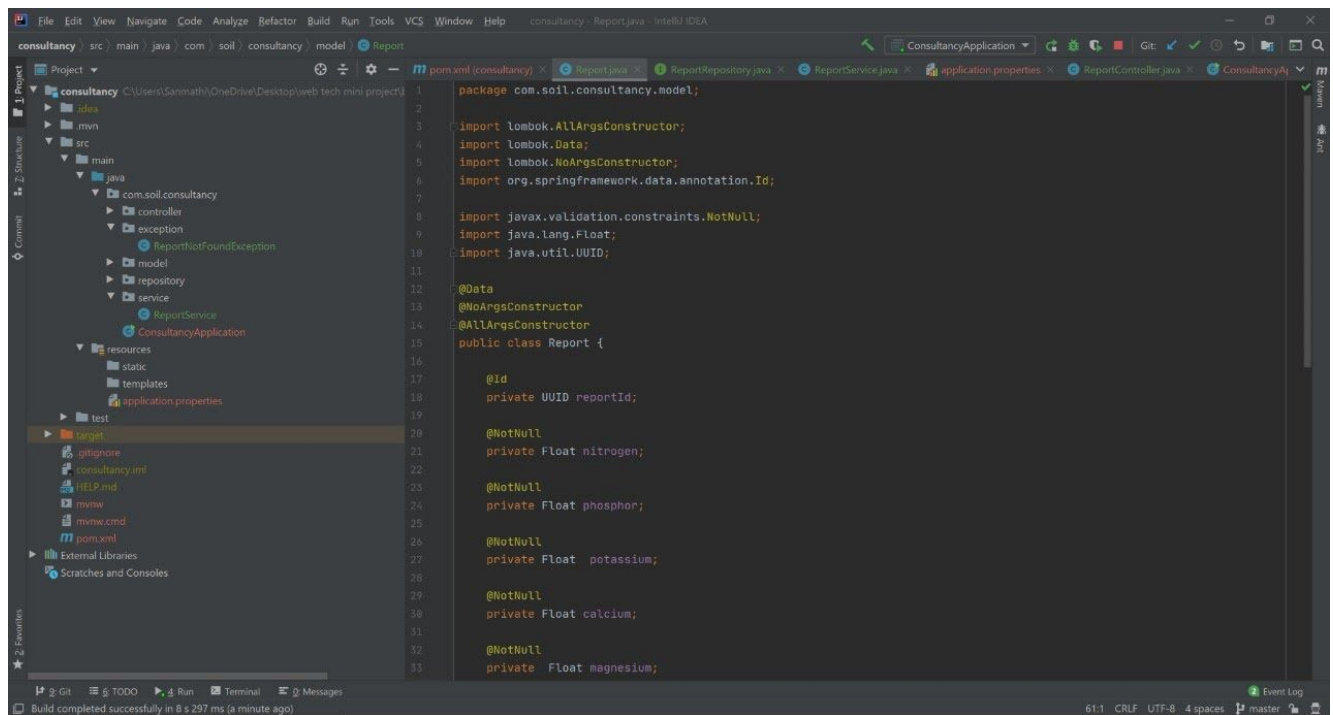
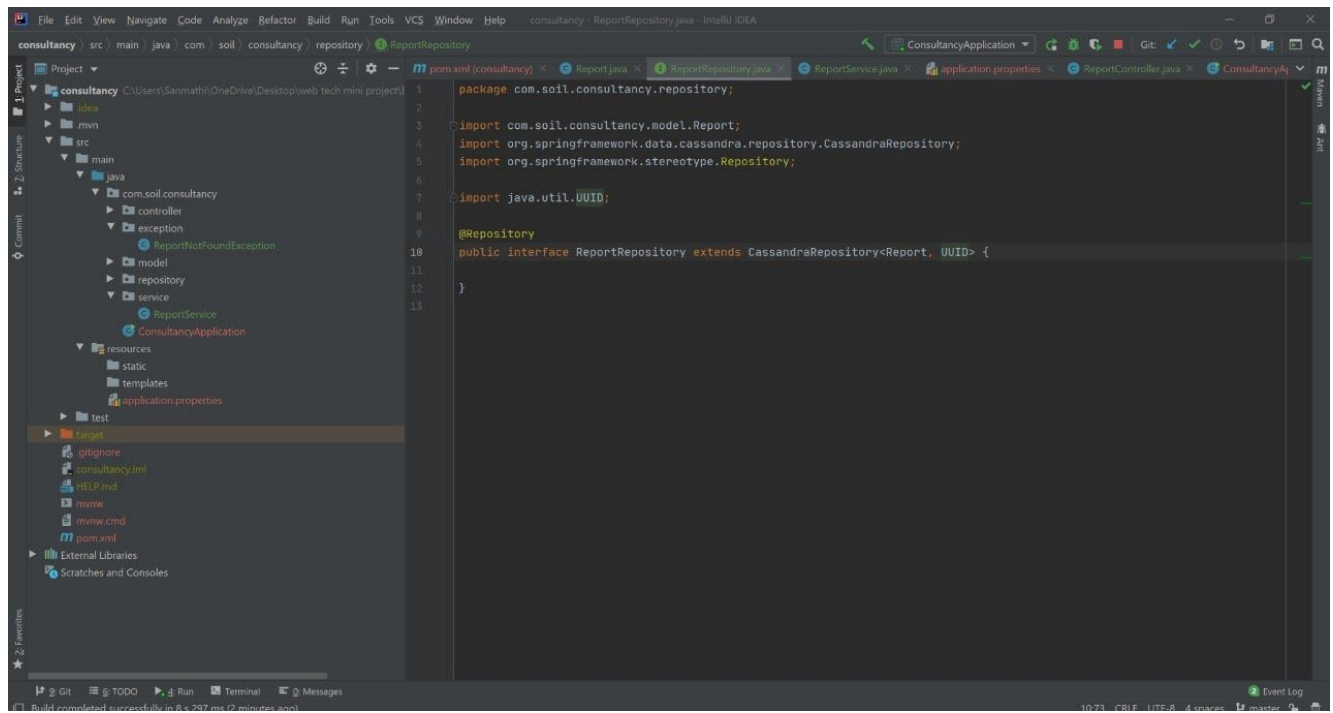
Build completed successfully in 8 s 297 ms (5 minutes ago)

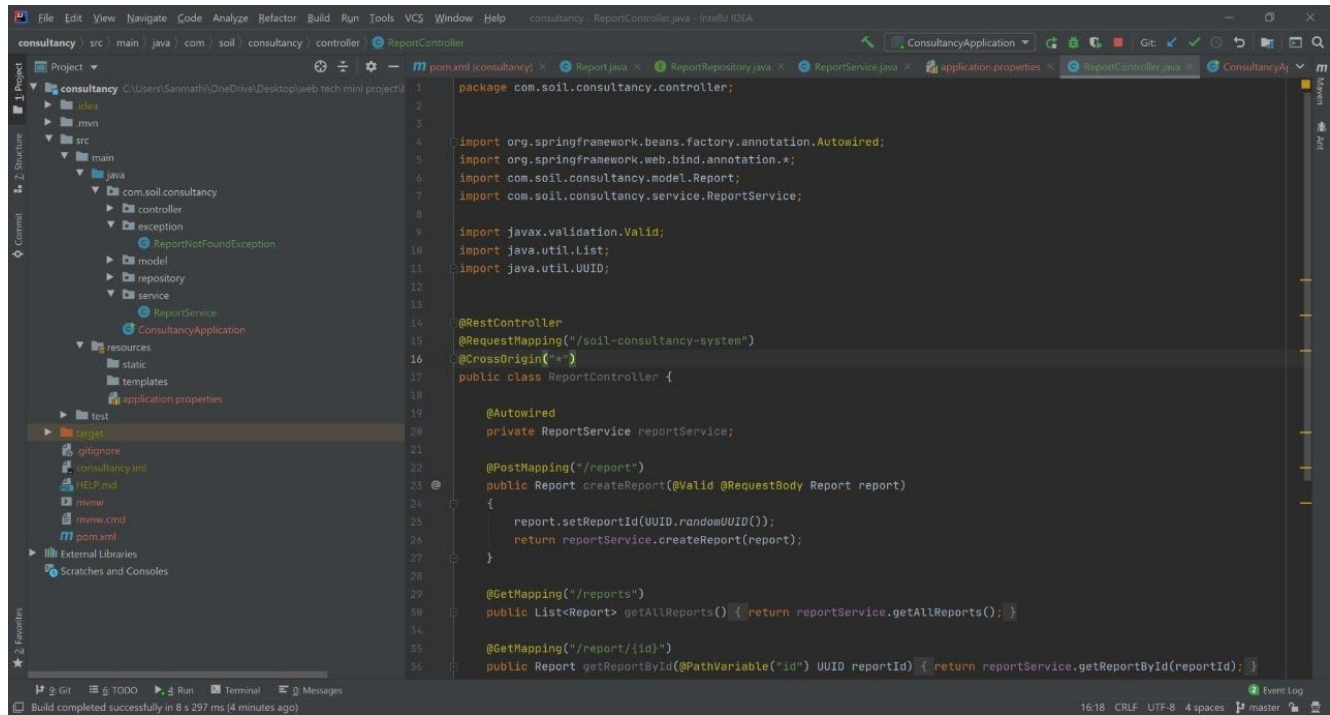
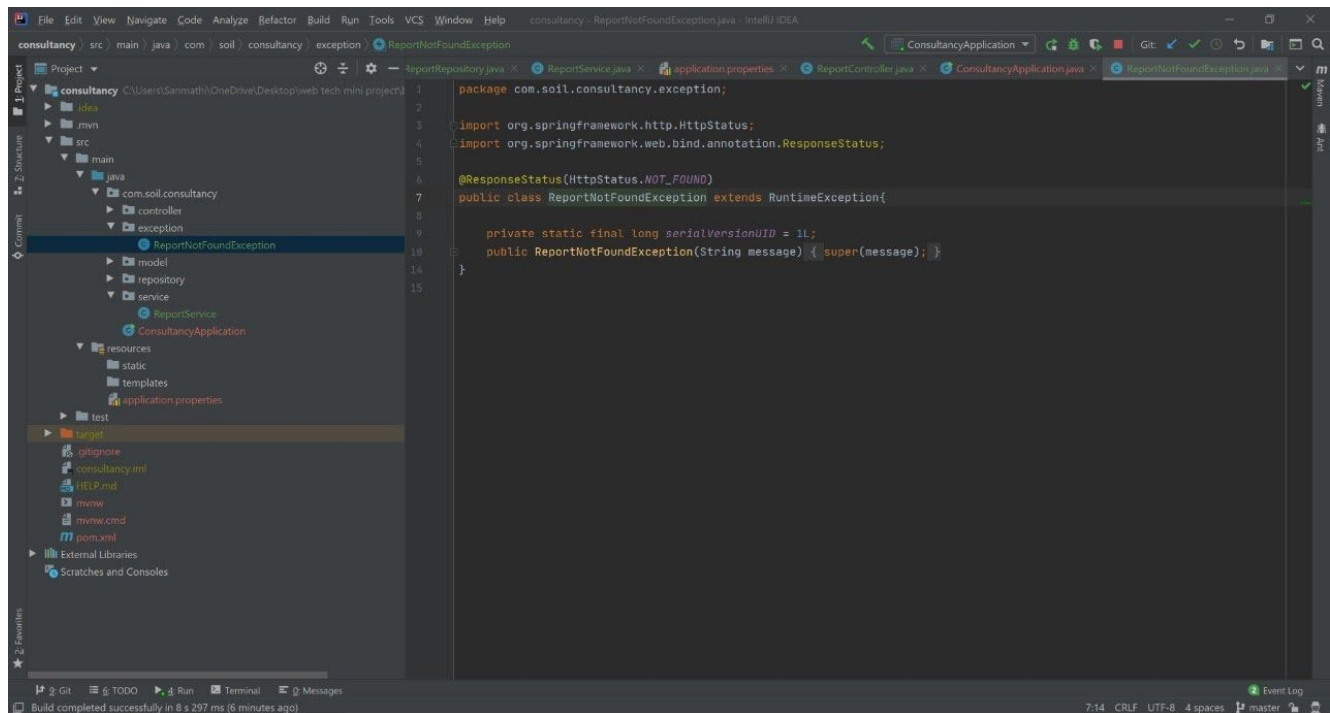


```
1 package com.soil.consultancy.service;
2
3 import com.soil.consultancy.exception.ReportNotFoundException;
4 import com.soil.consultancy.model.Report;
5 import com.soil.consultancy.repository.ReportRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import java.util.List;
10 import java.util.Optional;
11 import java.util.UUID;
12
13 @Service
14 public class ReportService {
15
16     @Autowired
17     private ReportRepository reportRepo;
18
19     //-----create New Report-----
20     public Report createReport(Report report)
21     {
22         return reportRepo.save(report);
23     }
24
25     //-----get All Reports -----
26     public List<Report> getAllReports() { return reportRepo.findAll(); }
27
28     //-----get report by id -----
29     public Report getReportById(UUID reportId)
30     {
31         Optional<Report> optionalReport = reportRepo.findById(reportId);
32         if(!optionalReport.isPresent())
33         {
34             throw new ReportNotFoundException("Report not found");
35         }
36         return optionalReport.get();
37     }
38
39 }
```

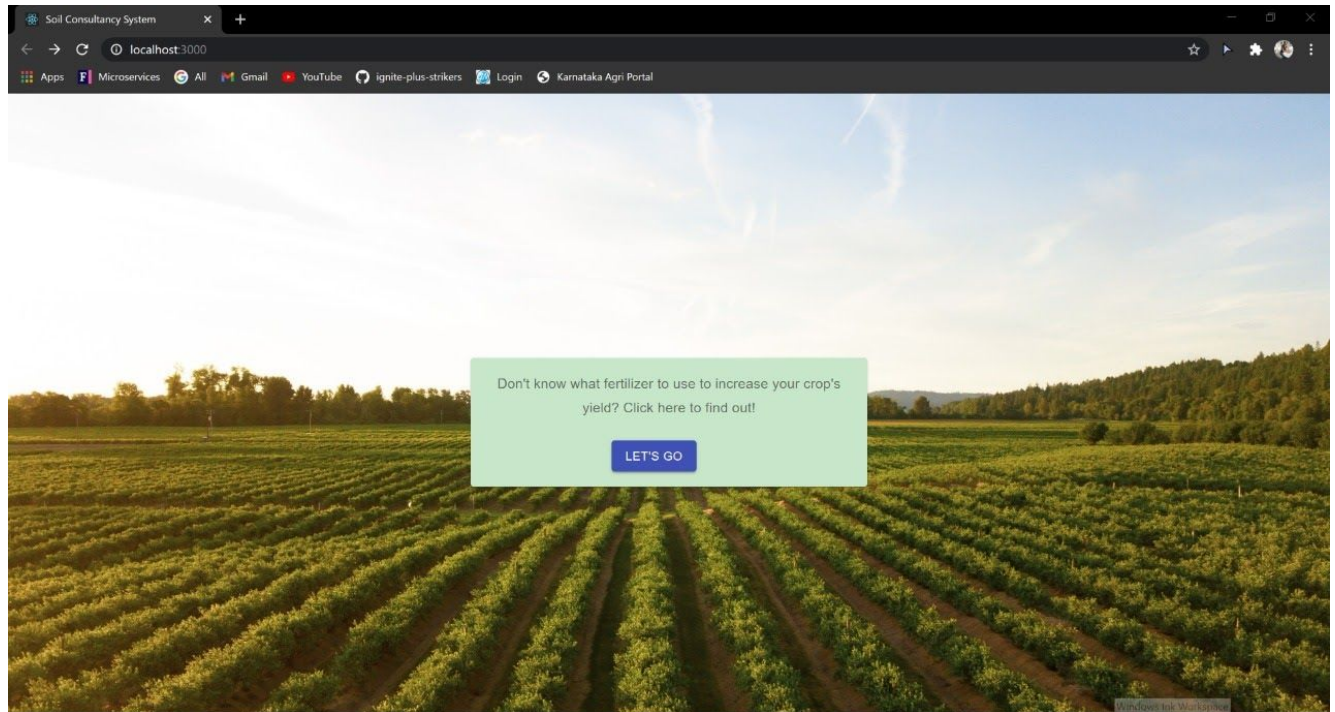
Build completed successfully in 8 s 297 ms (5 minutes ago)







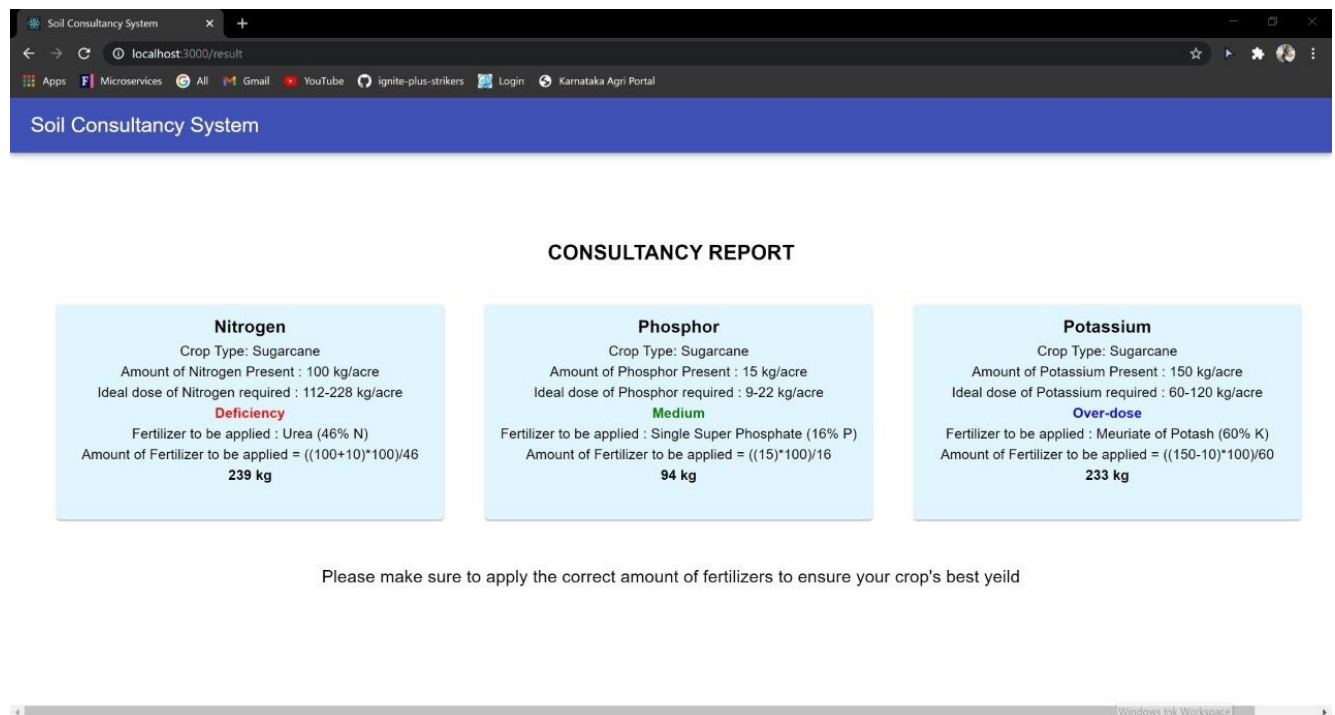
## 5.5 RESULTANT SNAPSHOTS



**Fig:** LogIn Page

**Fig:** ReportDetails

A screenshot of the 'Soil Consultancy System' report details page. The browser's address bar shows 'localhost:3000/reportForm'. The page has a blue header with the text 'Soil Consultancy'. Below the header, the title 'REPORT DETAILS' is centered, followed by the instruction 'Please enter the report details below'. The form is organized into four light blue boxes. The first box, titled 'PRIMARY NUTRIENTS', contains three input fields for 'Nitrogen(kg/acre)', 'Phosphor(kg/acre)', and 'Potassium(kg/acre)'. The second box, titled 'SECONDARY NUTRIENTS', contains three input fields for 'Calcium(ppm)', 'Magnesium(ppm)', and 'Sulphur(ppm)'. The third box, titled 'MICRO-NUTRIENTS', contains four input fields for 'Iron(ppm)', 'Zinc(ppm)', 'Manganese(ppm)', and 'Copper(ppm)'. The fourth box, titled 'CROP TYPE', is currently empty. The browser's top bar includes various icons and links such as 'Apps', 'Microservices', 'All', 'Gmail', 'YouTube', 'ignite-plus-strikers', 'Login', and 'Karnataka Agri Portal'.



**Fig:** Final Consultancy Report



## **CHAPTER 6**

### **TESTING AND SNAPSHOTS**

#### **JUnit Testing**

JUnit is an open source Unit Testing Framework for JAVA. It is useful for Java Developers to write and run repeatable tests. Erich Gamma and Kent Beck initially develop it. It is an instance of xUnit architecture. As the name implies, it is used for Unit Testing of a small chunk of code. Developers who are following test-driven methodology must write and execute unit tests first before any code. Once you are done with code, you should execute all tests, and it should pass. Every time any code is added, you need to re-execute all test cases and make sure nothing is broken.

#### **What is JUnit Testing ?**

Before discussing Junit testing in detail, it is imperative to understand what is Unit Testing?

Unit Testing is used to verify a small chunk of code by creating a path, function or a method. The term "unit" exists earlier than the object-oriented era. It is basically a natural abstraction of an object oriented system i.e. a Java class or object (its instantiated form).

Unit Testing and its importance can be understood by below-mentioned points:

- Unit Testing is used to identify defects early in the software development cycle.
- Unit Testing will compel to read our own code. i.e. a developer starts spending more time reading than writing.
- Defects in the design of code affect the development system. A successful code breeds the confidence of the developer.

#### **Why do you need JUnit Testing ?**

- It finds bugs early in the code, which makes our code more reliable.
- JUnit is useful for developers, who work in a test-driven environment.
- Unit testing forces a developer to read code more than writing.
- You develop more readable, reliable and bug-free code which builds confidence during development.

## **Features and advantages JUnit5**

- JUnit has added many new features in JUnit4. You can understand it easily by comparing JUnit 3.x and JUnit 4.x.
- Below is quick comparison between JUnit4.x and JUnit 3.x -
- All the old assert statements are the same as before.
- Most of the things are easier in JUnit4 as:
  - With JUnit 4 you are more capable of identifying exceptions. You can define expected exceptions as a parameter while using `@test` annotation.
  - Parameterized test is introduced, which enables us to use parameters.
  - JUnit4 still can execute JUnit3 tests.
- JUnit 4 can be used with java5 or higher version.
- While using JUnit4, you are not required to extend `JUnit.framework.TestCase`. You can just create a simple java class.
- You need to use annotations in spite of the special method name as before. Instead of using the setup method, you need to use `@before` annotation. Instead of using the teardown method, put `@after` annotation. Instead of using `testxxxx` before the method name, use `@test` annotation.

## **MockMVC framework & test pyramid**

Succeeding with Agile: Software Development Using Scrum, Mike Cohn describes the test automation pyramid as a test automation strategy in terms of quantity and effort that must be spent in the different types of tests. The test pyramid is also used by other authors to describe a test portfolio in terms of cost and speed. The base of the pyramid are unit tests, meaning that unit tests are the foundation of the testing strategy and that there should be many more unit tests than high level end-to-end tests.

As you know, unit testing focuses on testing the smallest component of a software. In object oriented programming this is usually a single class or interface, or a method within this class. The test is performed in isolation of other units, in order to achieve this dependencies are usually mocked. One of the main traits of unit tests is that their execution is really fast.

Similarly, `integration_tests` focus on testing a combination or group of the aforementioned components. Technically speaking, tests using `MockMvc` are in the boundaries between unit and integration tests. They aren't unit tests because endpoints are tested in integration with a Mocked MVC container with mocked inputs and dependencies. But we might also don't consider these tests as integration tests, if done right, with `MockMvc` we are only testing a single component with a mocked platform, but not a combination of components.

Technicalities aside, when preparing a test portfolio, tests using Spring MVC test framework should be treated as unit tests and be part of the foundation of the testing strategy.

This test will validate that the method returns the expected list whenever a null argument is passed. It will add 100% coverage for the exposed controller method, even though many aspects of the implementation won't be tested, more specifically:

- **Content negotiation headers:** This specific controller is configured to produce only `application/json` content.
- **Response codes:** With `MockMvc` we're able to check if the response code matches the expected one, even if it's produced outside the controller method or post processed by an annotation/aspect.
- **JSON serialization/deserialization:** `MockMvc` will help us validate that JSON is deserialized as expected when performing a request with content, and correctly converted to the response body.  
Availability of response headers
- Many other scenarios and configurations that happen outside the method specific implementation but are part of the endpoint implementation (Validation, Security, Exception Handling...)

### **WebMvcTest annotation**

The easiest way to run a test with `MockMvc` is to use the `@WebMvcTest` Spring boot annotation. This annotation will configure the `SpringRunner` JUnit test runner to deploy a partial web application by auto configuring only the required components for an MVC application. In my opinion, this is not the best approach to test your application as for each test suite run a mocked application will be deployed. This may be an overkill if you only want to test units of a controller and not the integration of the controller with the rest of components (test pyramid).

## SNAPSHOTS

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
consulancy : ReportServiceControllerTest.java - IntelliJ IDEA
consulancy | src | test | java | com | soil | consulancy | ReportServiceControllerTest | createReport
ReportServiceControllerTest | Git | Run | Test | Find | Search
ReportServiceControllerTest | ReportRepository.java | ReportService.java | application.properties | ReportController.java | ConsulancyApplication.java | ReportNotFoundException.java | ReportServiceControllerTest.java
@RunWith(SpringRunner.class)
@WebMvcTest(ReportController.class)
public class ReportServiceControllerTest {

    @Autowired
    private MockMvc mvc;

    @MockBean
    private ReportController reportController;

    @Test
    public void getReports() throws Exception {
        Report report = new Report();
        report.setNitrogen(120f);
        report.setPhosphor(20f);
        report.setPotassium(100f);
        report.setBoron(100f);
        report.setCalcium(100f);
        report.setChlorine(100f);
        report.setCopper(100f);
        report.setIron(100f);
        report.setMagnesium(100f);
        report.setManganese(100f);
        report.setMolybdenum(100f);
        report.setNickel(100f);
        report.setSulphur(100f);
        report.setZinc(100f);
        report.setCrop("Maize");

        List<Report> allReports = singletonList(report);

        given(reportController.getAllReports()).willReturn(allReports);
    }
}

```

Tests passed: 2 (moments ago)

83.51 CRLF UTF-8 4 spaces master Event Log

The screenshot displays the IntelliJ IDEA IDE with a Java test file named `ReportServiceControllerTest.java` open. The file is part of a project named `consultancy`, located in the `src \ test \ java \ com \ soil \ consultancy` directory. The test class `ReportServiceControllerTest` contains two test methods: `createReport` and `getReport`.

The `createReport` method is annotated with `@Test` and `throws Exception`. It uses `given()` to set up a `Report` object with various nutrient values (Nitrogen, Phosphor, Potassium, Boron, Calcium, Chlorine, Copper, Iron, Magnesium, Manganese, Molybdenum, Nickel, Sulphur, Zinc) and a crop type of "Maize". It then uses `when()` to call `reportController.createReport(report)` and `willReturn()` to specify the expected `Report` object. Finally, it uses `mvc.perform()` to send a `POST` request to `"/soil-consultancy-system/report"` with the report data as JSON.

The `getReport` method is also annotated with `@Test`. It uses `given()` to set up a list of `Report` objects and then uses `when()` to call `reportController.getAllReports()`. It expects a `200` status and uses `willReturn()` to specify the expected list of reports.

The IDE interface shows the Project tool window on the left with the project structure, the Run tool window at the bottom, and the Event Log on the right. The status bar at the bottom indicates that 2 tests passed in 2 minutes.

```
12 List<Report> allReports = singletonList(report);
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63 @Test
64 public void createReport() throws Exception {
65     Report report = new Report();
66     report.setNitrogen(120f);
67     report.setPhosphor(20f);
68     report.setPotassium(100f);
69     report.setBoron(100f);
70     report.setCalcium(100f);
71     report.setChlorine(100f);
72     report.setCopper(100f);
73     report.setIron(100f);
74     report.setMagnesium(100f);
75     report.setManganese(100f);
76     report.setMolybdenum(100f);
77     report.setNickel(100f);
78     report.setSulphur(100f);
79     report.setZinc(100f);
80     report.setCrop("Maize");
81
82
83     given(reportController.createReport(report)).willReturn(report);
84
85     mvc.perform(post( uriTemplate: "/soil-consultancy-system/report" )
86         .content(asJsonString(report))
87     )
88     .andExpect(status().isOk());
89
90     List<Report> allReports = singletonList(report);
91
92     when(reportController.getAllReports()).willReturn(allReports);
93
94     mvc.perform(get( uriTemplate: "/soil-consultancy-system/reports" )
95         .contentType(APPLICATION_JSON)
96         .andExpect(status().isOk());
97
98     }
99
100 }
```

