A Thesis/Project/Dissertation Report

on

WEATHER APP

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in
Computer Science Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr. Amit Kumar Goel
Professor

Submitted By

Vaishnavi Rai
18SCSE1140057

Sushant Gangwar
18SCSE1140037

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

DECEMBER-2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"WEATHER APP"** in partial fulfillment of the requirements for the award of the Bachelors of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of **Dr. Amit Kumar Goel,** Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Vaishnavi Rai – 18SCSE1140057

Sushant Gangwar – 18SCSE1140037

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Amit Kumar Goel

Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Vaishnavi Rai-18SCSE1149957 and Sushant Gangwar-18SCSE1140037 has been held on _________________ and his/her work is recommended for the award of Bachelors of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

Problem Stated

Weather apps have become one of the most popular and important application on smartphones. They have capability to provide daily to hourly weather data with dynamic changes as per circumstances of a location. In today's world, for any weather information weather apps are preferred. Many different weather apps are used to show weather details on mobile phone as well as on website. Some apps use fixed day by day data to forecast which tends to have minor to major fault changes as weather is dynamic.

Problem Solution

Our app will use an API capability which will make app robust and database hassle free application. It will help to maintain easy data parsing over internet without any interruption. No user data will be trespassed other than location of user as per his IP address and searched location with end-to-end security. It can also be converted to website as well as local app for phones.

Tools and Technology

Techs used are Visual Studio Code editor and languages used are HTML, CSS, JavaScript and web framework to support cross platform app.

Result

It will be a complete working and cross platform responsive app to show weather data as per user location.

Table of Contents

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

CHAPTER-1

Introduction

Since long time people were forecasting weather on news channel with help of long data analysis and only medium to spread this was through news channel, newspaper and articles. With digitalization and invention of modern era smartphones, to provide weather report on daily basis on each smartphone became a need.

Many Companies started their research and testing of new ideas to make this reality. First weather app was introduced by Apple inc. in 2007, developed only for iOS platform. It was ready to provide weather report as per mobile location with the help of database analysis.

Slowly other companies ported same idea to different OS platforms. With progress, more efficiency was introduced. Earlier apps were using fixed database to analyze daily weather report and were trying to be more accurate and time less in showing data. Many different techniques were used to improve the efficiency such as server database, cloud database, manual database, etc. There were minor to major problems in these techniques related to their efficiency to fetch data or to show accurate data as well. There was a time when many techniques and apps became outdated due to unavailability of data or inaccuracy in data processed.

Since 2017, API was introduced by Google and AWS amazon with their cloud service. This API fetch idea was so unique and easy to handle that many company started focusing on it. With invention of many different API services providing different performance and strength paved new roads to future of cloud database.

Now many companies such as Google Weather app used API service with their server to provide weather report to all users.

In our project, we are trying to build a feasible, responsive and trustable weather app based on API service that can provide weather report as per user’s location.

Formulation of Problem

Many apps still use data based on old analysis which is fixed and out dated as per google report of 2020. In this project, we are building an app that will provide a platform to get weather report on daily basis of user’s location. It will be made cross platform that will work on every OS platform.

Tools and Technology Used

Basic structure will be made using HTML, CSS and Javascript. Web framework like Node.js and Express,js will used to support the backend and database of app. Visual Studio Code will used to create the whole app with coding of each and every module.

CHAPTER – 2

Literature Survey

Technology is by nature progressive, with new and updated device innovations constantly being released. In response, many adaptations and extensions of the seminal Technology Acceptance Model (TAM) by Davis have been developed, including the Technology Acceptance Model 2 (TAM2) by Venkatesh and Davis, and the Unified Theory of Acceptance and Use of Technology model (UTAUT) by Venkatesh, Morris, Davis and Davis. However, of particular interest to this paper is the Technology Acceptance Model for Mobile Services (TAMM) by Kaasinen. As illustrated in Fig. 1, TAMM forms a modification and extension of the original TAM model by Davis, replacing Perceived Usefulness with Perceived Value and two new perceived product characteristics influencing Intention to Use, namely Trust and Perceived Ease of Adoption.

Method

A quantitative, web–based survey instrument was utilized in this study. The Internet provides a particularly appealing means for data collection. The advantages of reduced response time, lowered cost, ease of data entry, flexibility in format, and an ability to capture additional response–set information are common to Web–based collection across a variety of disciplines. Respondents invited to participate in the research encompassed users of weather apps on smartphones from a specific target population, namely students and staff from Southern Cross University, a regionally university based in Australia. Thus, a nonprobability self–selection sampling technique was utilized for the data collection. Exploratory Factor Analysis (EFA) was utilized during the analysis phase of this study to determine the underlying dimensions of user acceptance of smartphone-based weather apps, based on the perspective of a sample of smartphone users. In utilizing EFA, it is important to review the number of cases/sample size when considering the use of factor analysis as an analytical tool. Hair Jr. et al. suggest that the sample size should be 100 or larger. Others postulate the need for higher sample sizes, with Hutcheson and Sofroniou recommending at least 150–300 cases, and Cattell supporting a 250 case/sample rule.

Data Analysis

Data analysis was carried out using SPSS. Data collection resulted in 178 usable responses being received. To establish whether the data appeared to be normal, indicators such as mean, standard deviations and measures for skewness and kurtosis were inspected. Additionally, the 5 per cent trimmed mean (a measure provided by SPSS) for each TAMM item was ascertained. The difference between the original mean and the new trimmed mean was very small across the measures utilized in the analysis. Thus, there is a reasonable assumption that normality has been met.

The current research utilized the results from 178 respondents for data analysis; and thus recognizes a minimum factor loading of .45 when analyzing factors. Correlations within the rotated component matrix, meeting this minimum measure.

CHAPTER – 3
Methodology and System analysis

3.1 ANALYSIS OF EXISTING SYSTEM

Throughout the system analysis, an in-depth, study of end-user information is conducted, for producing functional requirement of the proposed system. Data about the existing ordering system is collected through several fact-finding techniques such as website visit and document review, at the beginning of this stage. The data collected facilities information required during detailed analysis. A study on the current system is performed based on the collected data. As a result, user requirement of the proposed system is determined. At the end of this stage, requirement specification is produced as deliverable.

3.2 THE EXISTING SYSTEM

The existing system happens to be a non-computerized operating system where all operations are done manually by the people carrying paper and to take down the weather data by analysing. Due to manual weather data, forecast lacks with up to dated data and can be wrong most of the time leading casualties.

3.3 PROBLEMS OF EXISTING SYSTEM

Due to manual means being employed by the fast-weather data, it is very difficult to match real time updates. Most of the problems include:

- Mistakes are made in manual update of data.

- The process of collecting data is slow.

- It leads to lack of reliability of weather data on real time.

- The record keeping system is poor.

- Unnecessary time is wasted conveying information through the ladder of authority. Management at times seeks to get a copy of the weather data from and this may take a lot of time to obtain it.

- It causes reduction of production flow. These are the major problems facing the existing system and would be corrected with the help of the proposed system.

3.4. OBJECTIVES OF THE PROPOSED SYSTEM

The proposed system is developed to manage ordering activities in fast weather data access. It helps to get weather data in real time. The system should cover the following functions in order to support the weather data process for achieving the objectives:

- To allow the update of weather data in real time.

- To provide interface that allows check and update.

- To prevent interface that shows wrong data.

- Tools that generate reports that can be used for decision making.
- A tool that allows the management to modify the weather information such as location, add a new location and many others as well as tools for managing user, system menu and promotion records.

3.5 JUSTIFICATION FOR THE NEW SYSTEM

It is the purpose of the new system to address all the problems plaguing the present system. This system will do the analysing and storing of information either automatically or interactively. It will make use of REACT and MONGODB. This will be like this: a report is generated conforming to particular information needed by the management via the monitor. This will require the input of necessary data and record of fast-weather ordering and delivery and then a report is generated. The proposed system will also have some other features such as:

1. Accuracy in handling of data
2. The volume of paper work will be greatly reduced.
3. Fast rate of operation as in making the ordered weather available and delivered on time.
4. Flexibility (i.e. it can be accessed at any time)
5. Easy way to back up or duplicating data in CD’s in case of data loss

6. Better storage and faster retrieval system.

7. Errors in the reports will be greatly minimized.

<u>Feasibility Analysis</u>

The concept of this project is to make it easy to get information about weather. The application will be a mobile application as well as desktop i.e cross platform due to the current atmosphere where it is common to use specified apps for most activities.

Overall App Feasibility

• Considering the major features

Our app provides any weather information on your phone.
It is cross platform.
Safe to use.

• Benefits of the product or service

Information of any location's weather is easily available.

• Redefined Concept:

Old App problems

- There are many apps and website already in market but most of them(approx.. 50%) are outdated because of outdated database and not working anymore.

- Apps which are in market use SMS method and some use fixed database that nearly needs to update every time which is costly and slower to work on.

Our Concept

- We will use a API based database which will automatically update the data.
- App will be cross platform which means can run on any device.

CHAPTER – 4
Functionality

REQUIRED TOOLS

- **Firebase Database**

  Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don’t really want to build, because they’d rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer.

- **Used language- Java, JavaScript**

  It can be combined with native code written in other programming languages, making it very flexible and a good choice for adding new features to existing applications.

- **Visual studio Code**

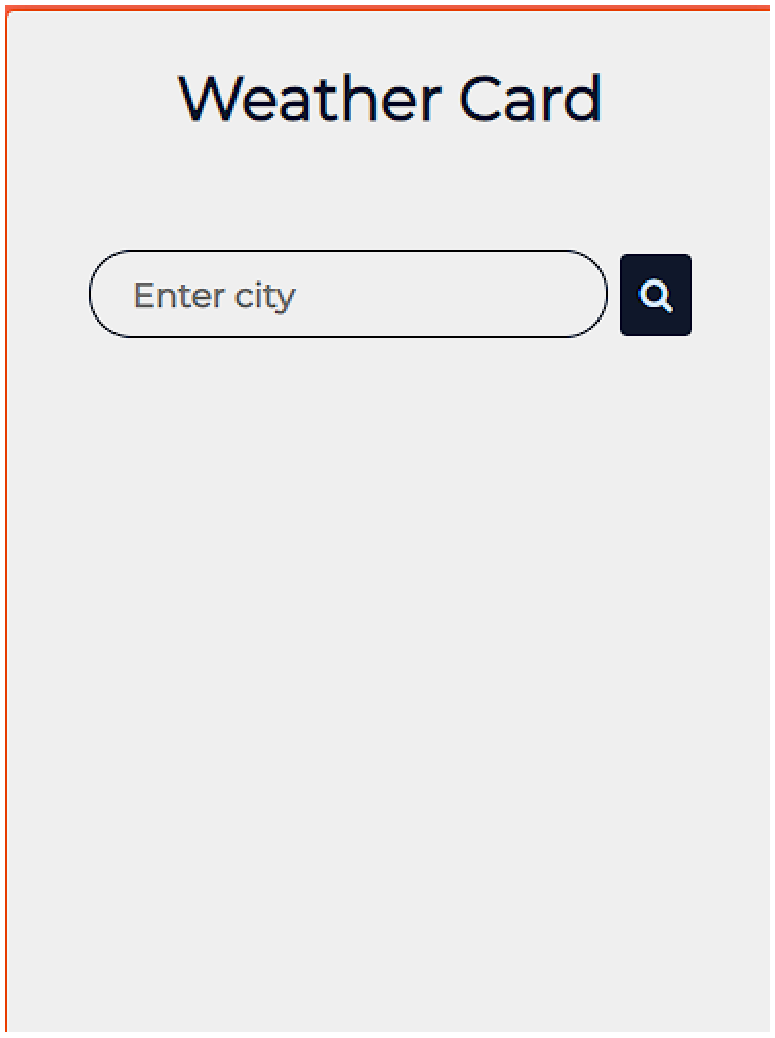
  A code editor for writing the code and making the web app.

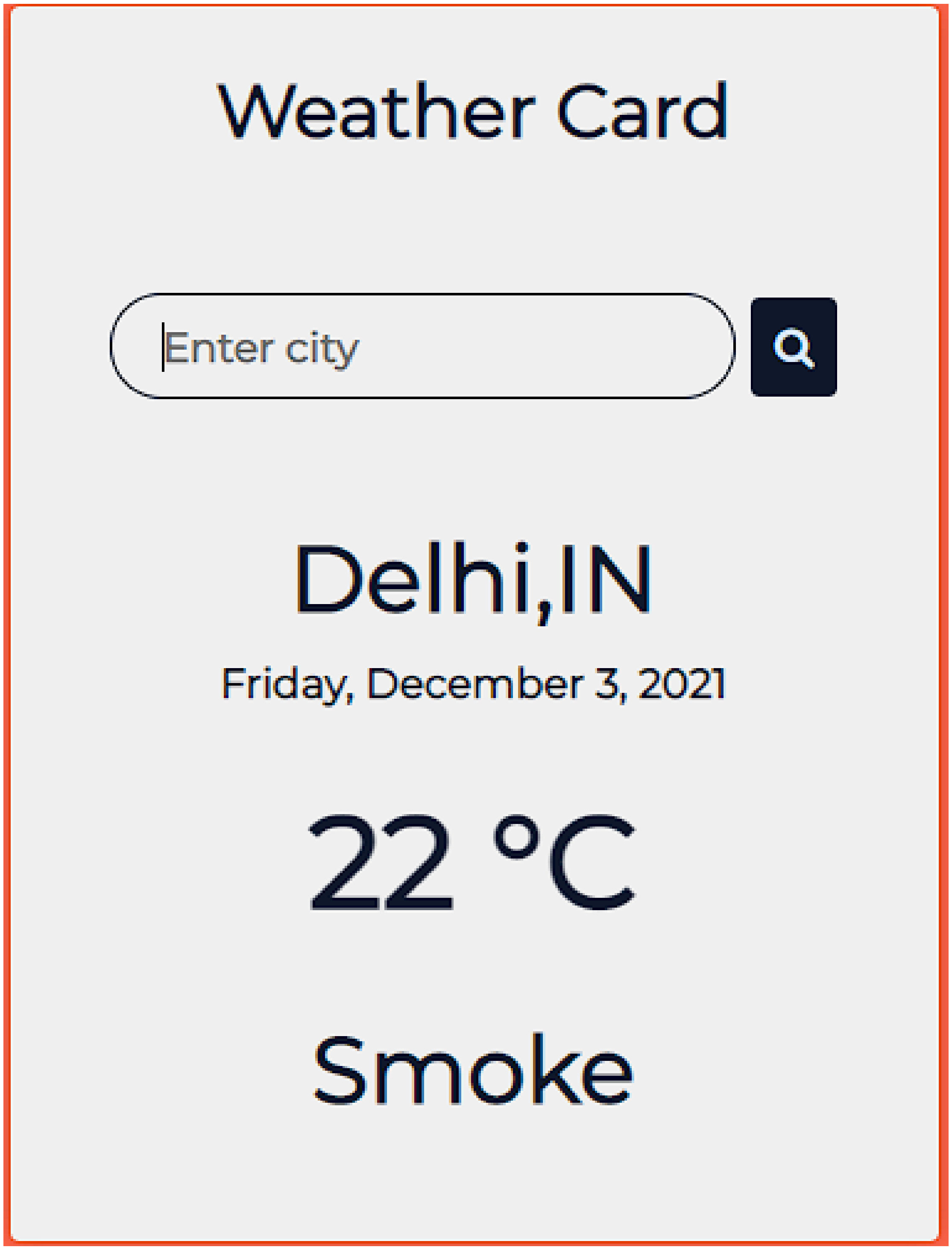
- **Android Studio**

  It is widely used because it has below feature:
  - Gradle-based build support
  - Android-specific refactoring and quick fixes

- Lint tools to catch performance, usability, version compatibility and other problems
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Card Design




Weather Card
Enter city
Delhi,IN
Friday, December 3, 2021
22 °C
Smoke

CHAPTER – 5

System Requirements

SOFTWARE REQUIREMENTS SPECIFICATION

5.1 Hardware Requirements

Number Description

1 PC with 250 GB or more Hard disk.

2 PC with 2 GB RAM.

3 PC with Pentium 1 and above.

5.2 Software Requirements

Number Description Type

1 Operating System Windows XP / Windows

2 Language React, HTML, CSS, JS

3 Database MongoDB

4 IDE Visual Code

5 Browser Google Chrome

In this Section we will do Analysis of Technologies to use for implementing the project.

5.3 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 from 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 delineated 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 behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), 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, colours, 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.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

JavaScript

JavaScript s a high-level, interpreted scripting language that conforms to the ECMA Script 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. The terms Vanilla JavaScript and Vanilla JS refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code. Google's Chrome extensions, Opera's extensions, Apple's Safari 5 extensions, Apple's Dashboard Widgets, Microsoft's Gadgets, Yahoo! Widgets, Google Desktop Gadgets are implemented using JavaScript.

React JS

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It’s ‘V’ in MVC. React JS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and react will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

Features of React.js

There are unique features are available on React because that it is widely popular.

- **Use JSX:** It is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript. It makes it easier for us to create templates.

- **Virtual DOM:** Virtual DOM exists which is like a lightweight copy of the actual DOM. So, for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen.

- **One-way Data Binding:** This feature gives you better control over your application.

- **Component:** A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

- **Performance:** React.js use JSX, which is faster compared to normal JavaScript and HTML. Virtual DOM is a less time taking procedure to update webpages content.

BOOTSTRAP

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Why we use Bootstrap?

- It is Faster and Easier way for Web-Development.
- It creates Platform-independent web-pages.
- It creates Responsive Web-pages.
- It designs the responsive web pages for mobile devices too.
- It is Free and open-source framework available on www.getbootstrap.com

GIT

Git is a free open-source **distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

Git relies on the **basis of distributed development** of a software where more than one developer may have access to the source code of a specific application and can modify changes to it which may be seen by other developers.

Every git working directory is **a full-fledged repository** with complete history and full version-tracking capabilities, independent of network access or a central server.

Git **allows a team of people to work together**, all using the same files. And it helps **the team cope up with the confusion** that tends to happen **when multiple people are editing the same files**.

Characteristics of Git

1. **Strong support for non-linear development**
2. Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.
3. A major assumption in Git is that a change will be merged more often than it is written.
4. **Branches** in Git **are very lightweight**.

Distributed development

1. Git **provides** each developer **a local copy** of the entire development history, and changes are copied from one such repository to another.

2. The changes can be merged in the same way as a locally developed branch very efficiently and effectively.

Compatibility with existing systems/protocol Git has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories.

Github

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

CHAPTER – 6

Modules

App module is divided in two parts:

1) Database

2) Functionality

Database Module is implemented using Firebase and firestore database storage. They provide both online and offline handling of data.

Functionality is implemented using javascript and connectivity with google service API.

Working Explanation

Languages Used

HTML

HTML stands for Hyper Text Markup Language. HTML is the standard markup language for creating Web pages. It describes the structure of a Web page and consists of a series of elements. HTML elements tell the browser how to display the content. Elements also label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

Example:

```
<!DOCTYPE html>
<html>
      <head>
```

```
            <title>Page Title</title>
      </head>
      <body>
            <h1>My First Heading</h1>
            <p>My first paragraph. </p>
      </body>
</html>
```

CSS

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files with extension .css.

JavaScript

JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.

API

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

CONCEPTS

What is PWA?

PWA stands for Progressive Web App. PWA are web applications that have been designed so they are capable, reliable, and installable. These three pillars transform them into an experience that feels like a platform-specific application. They have both capabilities of website and an app.

Database

Firebase’s firestore database with real time updating is used to display information. Also cached database for local assistance when user goes offline is also used to showcase database. It is not required to store any database while using this app.

What are service workers?

A service worker is a type of web worker. It's essentially a JavaScript file that runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and delivering push messages.

CHAPTER – 7

Source Code

Index.html

This page contains base structure of weather input and shows the location data as per API data

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="styles.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
   <title>Weather App</title>
</head>
<body>
   <div class="container">
      <h2 >Weather Card</h2>
      <form class="form">
         <input class="input-city" type="text" placeholder="Enter city"/>
         <button type="submit" class="btn"><span class="fa fa-search"></span></button>
      </form>
      <div class="place info"></div>
      <small class="date info"></small>
      <div class="temp info"></div>
      <div class="condition info"></div>
   </div>

   <Script src="/weather.js"></Script>
</body>
```

</html>

Styles.css

Contains styling of weather page.

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat&display=swap');

body{
    text-align: center;

    margin: 0;
    font-family: 'Montserrat', sans-serif;
    background:tomato;
    max-width:100vw;
    max-height:100vh;
    width:100%;
    height:100vh;
    display:flex;
    justify-content:center;
    align-items:center;
}

.container{
    background-color: #efefef;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    border-radius: 3px;
    color: #0f172a;
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 300px;
    height: 400px;
    margin: auto;
}
.form{
    margin: 1rem 0rem;
}
.input-city{
    background-color: transparent;
    border: 1px solid #0f172a;
    border-radius:40px;

    color: #0f172a;
    display: inline-block;
    font-family: 'Montserrat', sans-serif ;
    font-weight: bold;
    margin: 0.5rem 0rem;
    padding:0.5rem 1rem;
    outline: none;
}
.btn{
    background-color: #0f172a;

    border-radius: 3px;
    border:none;
    color: #DBEAFE;
    display: inline-block;
    font-family: 'Montserrat', sans-serif ;
    font-weight: bold;
    margin: 0.5rem 0rem;
    padding: 0.5rem;
    outline: none;
    border-radius:20px;
}

.info{
     color: #0f172a;
    margin-top: 1rem;
    font-weight: bold;
}
```

```
.place{

    font-size: 30px;

}
.date{

    margin-top: 0.5rem;

}
.temp{


    font-size: 42px;

    margin-top: 1.5rem;

}
.condition{

    font-size: 30px;

    margin: 1.5rem 0rem;
}
```

Weather.js

The complex logic behind showing weather data.

```
window.onload = function(){

  //acessing dom elements
  const input =  document.querySelector(".input-city");

  const form = document.querySelector(".form");
const [place,date,temperature,condition]=
document.querySelectorAll(".place,.date,.temp,.condition");

  //submit handler
  form.addEventListener('submit', async(e) =>{
    e.preventDefault();
    const cityName = input.value;

    //check if there is not any value
    if(!cityName|| /\s/g.test(cityName))return input.value="";
    place.textContent="Loading...";
    date.textContent="";
    temperature.textContent="";
    condition.textContent="";
    try{
      //Api key
      const apiKey = "76c0f72608729b50bd0a1b90110fd036";

      //Url
const url =
`https://api.openweathermap.org/data/2.5/weather?q=${cityName}&appid=${apiKey}&un
its=metric`;

      const  res = await fetch(url);
      const data =await res.json();
      const {name, weather, sys:{country}, main:{temp} } = data;

      place.textContent =`${name},${country}`;

      date.textContent = new Date().toLocaleString("en-US",{
        weekday: 'long',
        year: 'numeric',
        month: 'long',
        day: 'numeric'
      });
      temperature.textContent=`${temp.toFixed(0)} °C`;
      condition.textContent=weather[0].main;
      input.value = "";
    } catch(err){
        console.log(err.message);
        place.textContent="Try Again With proper country name";
    }
  });
}
```

Service Worker

This code helps to register the app as pwa and provide local installation within mobile phone.

Sw.js

```
const staticCacheName ='site-static-v1';
const dynamicCacheName = 'site-dynamic-v1';
const assets=[
    '/',
    '/index.html',
    '/pages/search.html',
    '/pages/new_data.html',
    '/js/app.js',
    '/js/ui.js',
    '/pages/detail.html',
    '/js/materialize.min.js',
    '/css/styles.css',
    '/css/materialize.min.css',
    '/image/search.png',
    'https://fonts.googleapis.com/icon?family=Material+Icons',
    'https://fonts.gstatic.com/s/materialicons/v50/flUhRq6tzZclQEJ-Vdg-
IuiaDsNcIhQ8tQ.woff2',
    '/pages/fallback.html',
];
```

```
//cache size limit function
const limitCacheSize =(name, size)=>{
    caches.open(name).then(cache =>{
        cache.keys().then(keys => {
            if(keys.length > size){
                cache.delete(keys[0]).then(limitCacheSize(name, size));
            }
        })
    })
};

self.addEventListener('install',evt=>{
    evt.waitUntil(
    //console.log('Service worker has been installed');
    caches.open(staticCacheName).then(cache =>{
        console.log('catching shell assets');
        cache.addAll(assets)
    })
    );
});
self.addEventListener('activate',evt=>{
    //console.log('service worker has been activated');
    evt.waitUntil(
        caches.keys().then(keys =>{
            //console.log(keys);
            return Promise.all(keys
                .filter(key => key!== staticCacheName && key!== dynamicCacheName)
                .map(key => caches.delete(key))
                )
        })
    );
});
self.addEventListener('fetch',evt=>{
   if(evt.request.url.indexOf('firestore.googleapis.com') === -1){
    evt.respondWith(
        caches.match(evt.request).then(cacheRes =>{
            return cacheRes || fetch(evt.request).then(fetchRes =>{
            return caches.open(dynamicCacheName).then(cache =>{
                cache.put(evt.request.url, fetchRes.clone());
                limitCacheSize(dynamicCacheName,15);
                return fetchRes;
            })
        });
        }).catch(() => {
            if(evt.request.url.indexOf('.html') > -1){
            return caches.match('/pages/fallback.html');
            }
        })
    );
   }
});
```

CHAPTER 8
APPLICATION DESIGN / IMLEMENTATION

8.1 DESIGN STANDARD

The system is designed with several interaction cues on each web page that makes up the web application. These cues are well-defined such as to make several functionalities that the application exposes to collect, process and output data. Access to these functionalities is made possible by the well-designed user interface which embodies several technologies such as AJAX (Asynchronous JavaScript and XML) to process data. The application is built in a modular form where these functionalities are built into modules.

8.2 OUTPUT SPECIFICATION

The system is designed in such a way that it efficiently provides output to the user promptly and in a well-organized manner. The format for the several outputs is make available on the output web pages. Output can be relayed using the following page modules:

1. **Product_list.jsx**: This display output information for the list of weather delicacies which are currently available.
2. **Search_result.jsx**: This displays output information for the order report.
3. **Home_page.jsx**: This displays output information for the home page items details.

8.3 INPUT SPECIFICATION.

The system is designed to accept several input details efficiently through input forms and user clicks. The data captured through the user keystrokes and clicks are received by specific modules on the system and relayed to the back-end of the system for processing.

8.4 DATABASE SPECIFICATION

The database system used to implement the back-end of the system is MongoDB. Access to the system was made possible by a graphical interface with an ISAM engine. The database name is weather and the structure of the database are as follows:

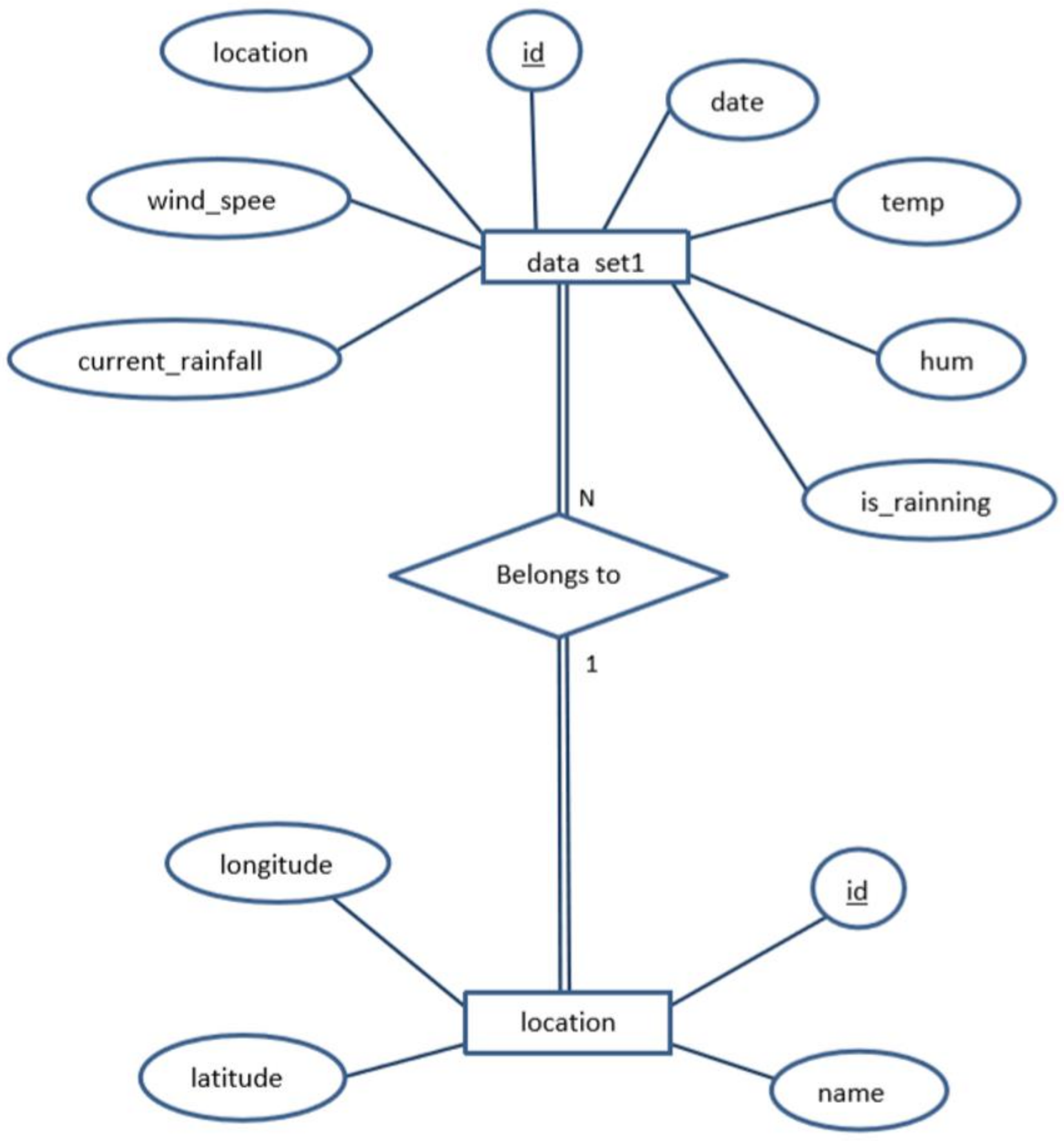
1. User
2. Location

DFD -1

1-level DFD: In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

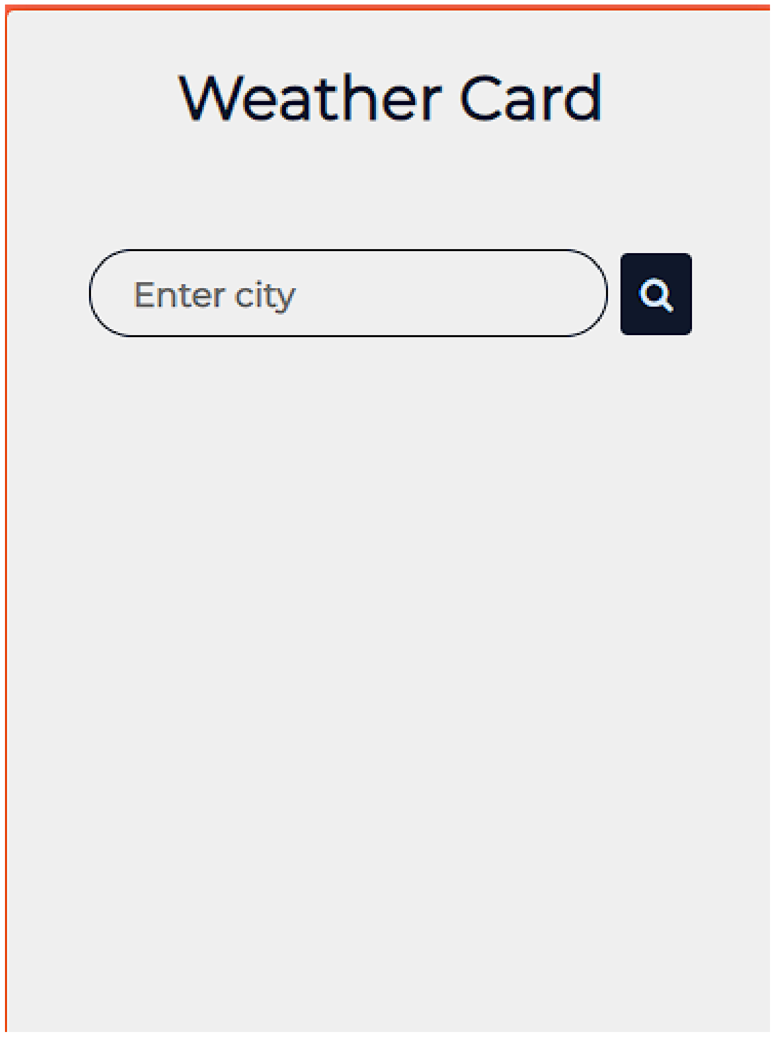
8.5 ER Diagram

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course. An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as: Below shown are the ER diagrams that are used to construct this application Fig-1. This above simulation flow is with respect to customer point of view. And the restaurant manager or staff can keep on track of the orders by viewing the database or by the notification.



8.6 Application

The application starts by displaying the Home Page with a dashboard of delivery, dine out and Nightlife. When the user can navigate or can search for any data. If the user is ordering for first time i.e. then, he/she has to first 'Register' and then they can start viewing the deals. Else, if it's not their first time then they have to 'Login' with all the credentials such as filling his/her first name, last name, phone number, Email Id, address and password. He/she has to choose their favorite dishes from the menu, then place their favorite dishes in the weather cart, this weather cart will help them to customize the orders like increasing the quantity, removing the weather items etc. Once he/she is done customizing their orders, they can checkout and will be redirected to the final order page including their personal details, their orders, total amount to be paid with appropriate payment method. Lastly, they can just pay the amount by selecting the payment method of their choice and simply log-out.

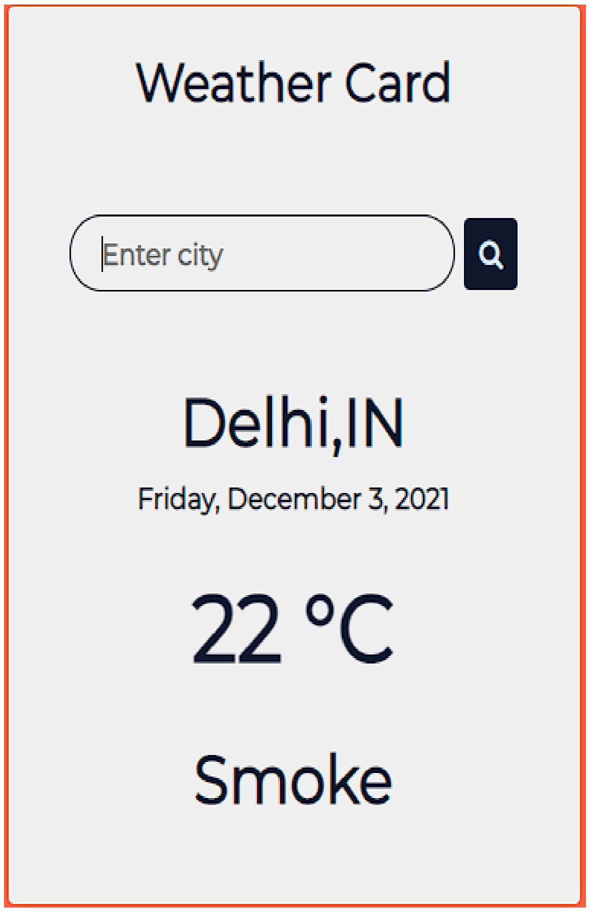


Home Page

It is the home page layout the home page is located in the root directory of a website. Most web server allow the home page to have one of several different filenames.

Examples include index.html, default.html, and home.html. The default filename of a website's home page can be customized on both Apache and IIS servers. Since the home page file is loaded automatically from the root directory, the home page URL does not need need to include the filename.

There is no standard home page layout, but most home pages include a navigation bar that provides links to different sections within the website. Other common elements found on a home page include a search bar, information about the website, and recent news or updates. Some websites include information that changes every day. For example, The Weather home page includes a restaurant and different cuisines for the day.



CHAPTER – 9
TESTING AND IMPLEMENTATION

The term implementation has different meanings ranging from the conversation of a basic application to a complete replacement of a computer system. The procedures however, are virtually the same. Implementation includes all those activities that take place to convert from old system to new. The new system may be totally new replacing an existing manual or automated system or it may be major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Proper implementation is essential to provide a reliable system to meet organization requirement.

9.1: UNIT TESTING

Introduction

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Benefits

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

1) **Find problems early:** Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

2) **Facilitates Change:** Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

3) **Simplifies Integration:** Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

4) **Documentation:** Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

9.2: INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reusability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

9.3: SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

CHAPTER – 10

Conclusion

Now all-weather details can be obtained with use of weather API open to all people under set metrics using our app. This available all over the cross platform without any login or location block.

CHAPTER – 10

References

- USER ACCEPTANCE AND ADOPTION OF WEATHER APPS ON SMARTPHONES: EXPLORATORY FINDINGS

- 2015 WEI International Academic Conference Proceedings