A Project/Dissertation Review-ETE Report

On

SUNSHINE WEATHER ANDROID APPLICATION

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted to: Mr. A. Arul Prakash

Submitted by:

SHIVA NAND MISHRA
19021011400 / 19SCSE1010211

VIVEKANAND
19021011539 / 19SCSE1010355

SCHOOL OF COMPUTING SCIENCE & ENGINEERING

Galgotias University, Greater Noida

October – 2021

ACKNOWLEDGEMENT

In performing our assignment, we had to take the help and guideline of some respected person, who deserve our greatest gratitude. The completion of this assignment gives us much Pleasure. We would like to show our gratitude Mr. A. Arul Prakash., Project Instructor, Galgotias University who introduced us to the Methodology of work, and whose passion for the "underlying structures" had lasting effect and for giving us a good guideline for assignment throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this project. Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

ABSTRACT

Nowadays, it is typical that people need detailed and easily available information about current weather and weather forecast. A solution to this requirement can be a mobile phone application which communicates with a data server and provides all useful information in an easily useable user interface. As a consequence, an Android based mobile application is provided.

Sunshine Weather App – A Fully featured Weather forecast Android application implemented with all the best practices like content providers, loaders, notifications, services, Shared Prefrences , map Intents, networking , databases, View Recycling and many more. It makes use of all the 4 android fundamental components: Activity, Content providers, Services and BroadCast Receivers.

INTRODUCTION

When it comes to the weather, there is no lack of applications to choose from to get your information. But there is very little diversity when it comes to those apps.

Sunshine weather app, is looking to change all that. Using information delivered from the barometer sensor in the phone, as well as user-generated reports of local weather, the app is trying to change the way we consume information about the day's weather.

"Right now, all these apps are showing you a bunch of numbers, without actually taking into account how the weather feels to you, based on how you experience different types of weather.

This might sound simple, but in reality most of the hundreds of weather apps available for your phone are simply repackaging the same government-issued reports, according to Stroponiati. Those reports, in turn, come from satellites (which only see the bigger picture) or weather stations, which Stropionati describes as "few and far between."

"The more users we have, with phones offering up sensor data and users submitting weather reports, the more accurate we will get.

The idea is that weather feels different to everyone, so the more hyperlocal the reports can get, the better prepared users will be to plan for their day.

When signing up, users are asked to provide the address of their home and their office, or any other oft-frequented place. They're also asked to enter a "comfort zone," which is a generally preferred temperature range. From there, Sunshine offers a personalized push notification each morning helping that user decide what to wear for the day, etc.

Users can also check on predicted weather for the next 24 hours at any time, as well as input their own weather report based on more generalized options (like sunny, cloudy, windy, etc.) instead of exact temperatures.

Purpose:

The purpose of developing weather app is to fetch the data in the need of taking information about weather worldwide. Another purpose for developing this software is to generate the report automatically at the end of the session or in the between of the session.

System Analysis:

Analysis can be defined as breaking up of any whole so as to find out their nature, function etc. It defines design as to make preliminary sketches of; to sketch a pattern or outline for plan. To plan and carry out especially by artistic arrangement or in a skilful wall. System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and an intellectual orientation.

Identification Of Need:

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studies to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The System is viewed as a whole and the input to the system are identified. The outputs from the organization are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and Decisional variables, analysis and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing

system. Now the existing system is subjected to close study and problem area are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Feasibility Study:

Feasibility analysis begins once the goals are defined. It starts by generating broad possible solutions, which are possible to give an indication of what the new system should look lime. This is where creativity and imagination are used. Analysts must think up new ways of doing things- generate new ideas. There is no need to go into the detailed system operation yet. The solution should provide enough information to make reasonable estimates about project cost and give users an indication of how the new system will fit into the organization. It is important not to exert considerable effort at this stage only to find out that the project is not worthwhile or that there is a need significantly change the original goal. Feasibility of a new system means ensuring that the new system, which we are going to implement, is efficient and affordable. There are various types of feasibility to be determined. They are, Technical feasibility: The technical requirement for the system is economic and it does not use any other additional Hardware and software. Technical evaluation must also assess whether the existing systems can be upgraded to use the new technology and whether the organization has the expertise to use it. Install all upgrades framework into the .Net package supported widows based application. this application depends on Microsoft office and intranet service ,database. Enter their attendance and generate report to excel sheet. Economically Feasibility: Development of this application is highly economically feasible. The only thing to be done is making an environment with an effective supervision. It is cost effective in the sense that has eliminated the paper work completely. The system is also time effective because the calculations are automated which are made at the end of the month or as per the user requirement. Operational Feasibility: The system working is quite easy to use and learn due to its simple but attractive interface. User requires no special training for operating the system. Technical performance include issues such as determining whether the system can provide the right information for the Department personnel student details, and whether the system can be organized so that it always delivers this information at the right place and on time using intranet services. Acceptance revolves around the current system and its personnel.

Hardware and Software Requirements:

- Hardware Required

- Standard computer with at least i3 processor Standard computer with 4GB of RAM
- Standard computer with 100GB of free space
- Active Internet Connectivity with good bandwidth

Software Required:

- Android Studio
- Ms Office
- Draw.io / Star UML
- SYSTEM DESIGN
- GANTT CHART

1	Feasibility Study	01 Oct 2021
2	System Specification	04 Oct 2021
3	System Architecture	07 Oct 2021
4	Planning	09 Oct 2021
5	Design	11 Oct 2021
6	Coding	-
7	Test Plan	-
8	Testing	-
9	Documentation	-

Sequence Diagram:

Sequence diagrams can be useful reference diagrams for businesses and other organizations. Try drawing a sequence diagram to:

Represent the details of a UML use case.

Model the logic of a sophisticated procedure, function, or operation. See how tasks are moved between objects or components of a process.

Plan and understand the detailed functionality of an existing or future scenario.

Popular Sequence Diagram Uses:

Usage Scenario – A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system**.**

Method Logic - Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to Usage Scenario - A usage scenario is a diagram of how your system could potentially be used. It's a great explore the logic of any function, procedure, or complex process.

Service Logic - If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.

Activity diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Purpose of Activity Diagrams:

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Where to Use Activity Diagrams?

The basic usage of activity diagram is similar to other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages. Activity diagram is suitable for modelling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system.

Class diagram:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Purpose of Class Diagrams:

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

Where to Use Class Diagrams?

Class diagram is a static diagram and it is used to model the static view of a. The static view describes the vocabulary of the system. Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the system static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system. Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.

Class diagram clearly shows the mapping with object-oriented languages such as Java, C++, etc. From practical experience, class diagram is generally used for construction purpose

User Interface Design:

The design of user interfaces for machines and software, such as computers, home appliances, mobile devices and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centred design). Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs. Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in

certain types of projects and have skills centred on their expertise, whether that be software design user research, web design or industrial design.

Implementation and Testing:

A software system test plan is a document that describes the objectives, scope, approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be through enough to be useful but not so through that no one outside the test group will read it.

Introduction:

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements. The main purpose of testing is to detect errors and error prone areas in a system. Testing must be through well planned. A partially tested system is to detect errors and error prone areas in a system. Testing must be through well planned. A partially tested system is as bad as an untested system. And the price of an untested and under tested system is high.

Objectives of Testing:

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview:

The following represents the overall flow of the testing process: Identify the requirements to be tested. All test cases shall be derived using the current Program Specification. Identify which particular test(s) will be used to test each module. Review the test data and test cases to ensure

that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit Identify the expected results for each test. Document the test case configuration, test data, and expected results. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR). Successful unit testing is required before the unit is eligible for component integration/system testing. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

Test Cases:

Test case is an object for execution for other modules in the architecture does not represent any interaction by itself. A test case is a set of sequential steps to execute a test operating on a set of predefined inputs to produce certain expected outputs. There are two types of test cases: manual and automated. A manual test case is executed manually while an automated test case is executed using automation. In system testing, test data should cover the possible values of each parameter based on the requirements. Since testing every value is impractical, a few values should be chosen from each equivalence class. An equivalence class is a set of values that should all be treated the same. Ideally, test cases that check error conditions are written separately from the functional test cases and should have steps to verify the error messages and logs. Realistically, if functional test cases are not yet written, it is ok for testers to check for error conditions when performing normal functional test cases. It should be clear which test data, if any is expected to trigger errors.

Testing Steps:

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. A strategy must provide guidance for the practitioner and a set of milestones for the manager. Because the steps of the test strategy occur at a time when deadline pressure begins to rise, progress must be measurable and problems must surface as early as possible. Following testing techniques are well known and the same strategy is adopted during this project testing.

Unit testing:

Unit testing focuses verification effort on the smallest unit of software design- the software component or module. The unit test is white-box oriented. The unit testing implemented in every module of Weather Application. By giving correct manual input to the system, the data’s are stored in database and retrieved. If you want required module to access input or gets the output from the End user. any error will accrued the time will provide handler to show what type of error will accrued.

Integration Testing:

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface.

Performance Testing:

Performance testing is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white-box tests are conducted. This project reduce attendance table, codes. It will generate report fast. No have extra time or waiting of results entered correct data will show result few millisecond. Just used only low memory of our system. Automatically do not getting access at software. Get user permission and access to other applications.

Validation:

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists. a) The function or performance characteristics confirm to specification and are accepted. b) A deviation from specification is uncovered and a deficiency lists is created. c) Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

White Box Testing:

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Black box testing:

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing:

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity.

Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hardcopy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

User Acceptance Testing:

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point: a) Input Screen Design. b) Output Screen Design. c) Format of reports and other outputs.

Integration Testing:

Software testing is always used in association with verification and validation. In the testing phase of this project our aim is to find the answer to following two questions. Whether the software matches with the specification (i.e. process base) to verify the product. Whether this software in one client what wants (i.e. product base) to validate the product. Unit testing and integration testing has been carried out to find the answer to above questions. In unit testing

each individual module was test to find any unexpected behaviour if exists. Later all the module was integrated and flat file was generated.

Functional Testing:

These are the points concerned during the stress test: Nominal input: character is in putted in the place of digits and the system has to flash the message "Data error" Boundary value analysis: exhaustive test cases have designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

Conclusion:

By this system weather forecasting report generation becomes easy. Less chances of malfunctioning are there. The system has reached a steady state but still improvements are to be made. The system is operated at a high level of efficiency and all the work and user associated with the system understand its advantage. It was intended to solve as requirement specification. In future this system can be implemented to all over the world and will be designed for cross platform.

Bibliography

www.android.com

www.developer.android.com