

MAJOR PROJECT REPORT

Continuous Integration Pipeline Implementation for Tech11Software

Submitted By

ASWIN G SUGUNAN (Reg.No . 121436–)

JEFFIN JACOB (Reg.No . 12143611)

NITIN SURESH (Reg.No . 121436–)

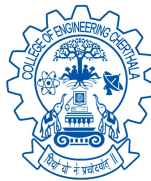
VISHNU BOSE (Reg.No . 121436–)

under the guidance of

Mrs. GREESHMA N GOPAL

(Assistant Professor)

(Computer Science & Engineering)



MARCH 2017

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING, CHERTHALA
PALLIPPURAM P O, ALAPPUZHA-688541,
PHONE: 0478 2553416, FAX: 0478 2552714
<http://www.cectl.ac.in>**

MAJOR PROJECT REPORT

Continuous Integration Pipeline Implementation

for Tech11Software

Submitted By

ASWIN G SUGUNAN (Reg.No . 121436–)

JEFFIN JACOB (Reg.No . 12143611)

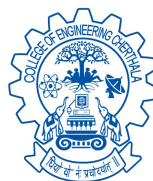
NITIN SURESH (Reg.No . 121436–)

VISHNU BOSE (Reg.No . 121436–)

under the guidance of

Mrs. GREESHMA N GOPAL

In partial fulfilment of the requirements for the award of the degree
of
Bachelor of Technology
in
Computer Science and Engineering
of
Cochin University Of Science And Technology



MARCH 2017

Department of Computer Science and Engineering
College of Engineering, Cherthala
Pallippuram P O, Alappuzha-688541
Phone: 0478 2553416, Fax: 0478 2552714

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING CHERTHALA
ALAPPUZHA-688541



C E R T I F I C A T E

This is to certify that, the major project report submitted by

Name:

Date:04/03/2017

Reg.No:

On *Continuous Integration Pipeline Implementation for Tech11Software* in partial fulfillment of the requirements for the award of the degree, **B. Tech. Computer Science & Engineering** of **Cochin University of Science & Technology**.

Guide

Co-ordinator

HoD

Mrs. Greeshma N Gopal K

Mrs. Sreelakshmi Murali

Dr. Preetha Theresa Joy

Assistant Professor

Assistant Professor

Professor

Computer Science&Engg

Computer Science&Engg

Computer Science&Engg

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude to the people who have been instrumental in the successful completion of our major project. For every endeavour in our life, the help of the Lord has always followed. We have yet again experienced his loving kindness while preparing for this project. We would like to express our sincere thanks to the Principal, **Dr. Mini M. G**, for her valuable support and advice. We would also like to thank our Head Of Computer Science Department, **Dr.Preetha Teresa Joy** (Professor, Computer Science and Engineering Department) for her support. We express our heartfelt gratitude to our project co-ordinator, **Mrs. Sreelakshmi Murali** (Assistant Professor, Computer Science and Engineering Department) for her valuable help and support. We would also like to thank our project Guide, **Mrs. Greeshma N Gopal** (Assistant Professor, Department of Computer Science) for her support and guidance. We are indebted to all teaching and non-teaching staff of the Department of Computer Science and Engineering, friends and family for their co-operation and support, without which we could never have completed the project this well.

ABSTRACT

Software development, as we know it today, is a demanding area of business with its fast-changing customer requirements, pressures of an ever shorter time-to-market, and unpredictability of market. With the shift towards modern continuous deployment pipelines, releasing new software versions early and often has become a concrete option also for an ever growing number of practitioners.

Continuous delivery is a software development practice where new features are made available to end users as soon as they have been implemented and tested. In such a setting, a key technical piece of infrastructure is the development pipeline that consists of various tools and databases, where features flow from development to deployment and then further to use.

Contents

1	INTRODUCTION	1
1.1	PURPOSE	1
1.2	PRODUCT SCOPE	2
2	OVERALL DESCRIPTION	3
2.1	PRODUCT PERSPECTIVE	3
2.1.1	Proposed System	3
2.2	SOLUTION FUNCTION	4
2.3	OPERATING ENVIRONMENT	6
2.4	GANTT CHART	7
2.5	COST ESTIMATION	9
2.6	EXTERNAL INTERFACE REQUIREMENTS	9
2.6.1	User Interface	10
2.6.2	Network Requirements	10
2.6.3	Software Requirements	10
3	SYSTEM DESIGN	11
3.1	MODULES	11
3.1.1	SENSING MODULE	11
3.1.2	ANDROID APPLICATION	11
3.1.3	SERVER SIDE	11
3.2	DATA FLOW DIAGRAMS	12
3.2.1	Purpose	12

3.2.2	Description	12
3.3	USE CASE DIAGRAMS	15
3.3.1	Purpose	15
3.3.2	Description	15
3.4	SEQUENCE DIAGRAM	16
3.4.1	Purpose	16
3.4.2	Description	16
3.5	ACTIVITY DIAGRAM	17
3.5.1	Purpose	17
3.5.2	Description	18
3.6	HARDWARE DIAGRAM	19
3.6.1	Purpose	19
3.6.2	Description	19
3.7	ER DIAGRAM	20
3.7.1	Purpose	20
3.7.2	Description	20
4	IMPLEMENTATION	24
4.1	LANGUAGES AND PLATFORM USED	24
4.1.1	Python	25
4.1.2	Android	25
4.1.3	Jsp	25
4.2	SCREEN SHOTS	26
4.2.1	Android Application	26
4.2.2	Web page	28
5	TESTING	30
5.1	CODE TESTING	31
5.2	UNIT TESTING	31
5.3	INTEGRATION TESTING	31

5.4	VALIDATION TESTING	32
5.5	SYSTEM TESTING	32
5.6	OUTPUT TESTING	33
5.7	GOAL OF TESTING	33
5.8	PASS/FAIL CRITERIA	34
5.9	PASS CRITERIA	34
5.10	FAIL CRITERIA	34
5.11	TEST REPORTS	34
5.12	BLACK BOX TESTING	35
6	FUTURE SCOPE	37
7	CONCLUSION	38
7	REFERENCES	39
	INDEX	41

List of Figures

2.1	Gantt chart	8
2.2	COCOMO Model Coefficients	9
3.1	Notations used in DFD	13
3.2	Level 0 DFD	13
3.3	Level 1 DFD	14
3.4	Use case Diagram	15
3.5	Sequence Diagram	21
3.6	Activity Diagram	22
3.7	Hardware Diagram	23
3.8	ER Diagram	23
4.1	user registration in android application	26
4.2	Connecting to Server	27
4.3	Accident notification in web page	28
4.4	Map in web page	29

Chapter 1

INTRODUCTION

Software development, as we know it today, is a demanding area of business with its fast-changing customer requirements, pressures of an ever shorter time-to-market, and unpredictability of market. With the shift towards modern continuous deployment pipelines, releasing new software versions early and often has become a concrete option also for an ever growing number of practitioners.

Continuous delivery is a software development practice where new features are made available to end users as soon as they have been implemented and tested. In such a setting, a key technical piece of infrastructure is the development pipeline that consists of various tools and databases, where features flow from development to deployment and then further to use.

1.1 PURPOSE

The objective of the project is to put in place a Continuous Integration framework for product development activities of Tech11 Software. This would enable the Tech11 team to rapidly bring a product change or feature to production gaining market advantage.

This activities of this project will involve accessing different CI integration approaches and solutions available, identify the feasibility of those solution by doing POCs and demos, fine tune the final solution and set up the CI infrastructures, educate the developers on CI culture.

1.2 PRODUCT SCOPE

The scope of the system is to help software developers to ensure new features are made available as soon as the program has been implemented and tested. This product also helps in reducing the time needed to develop a software and also acts a guideline for future software developments

Chapter 2

OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

Continuous Integration is based on continuous performance of acts of integration of source code, testing, building and deployment in response to each change to the source code of the project submitted by the developer and for the use of tools for support of the development and testing by compliance with the established procedure automatically. The proposed system directs the user (developers) for time and cost effective production and solves majority of the Integration hell problem.

Integration Hell refers to the point in production when members on a delivery team integrate their individual code. In traditional software development environments, this integration process is rarely smooth and seamless, instead resulting in hours or perhaps days of fixing the code so that it can finally integrate. Continuous Integration (CI) aims to avoid this completely by enabling and encouraging team members to integrate frequently (e.g., hourly, or at least daily).

2.1.1 Proposed System

The Proposed system is to help software developers to ensure new features are made available as soon as the program has been implemented and tested. This product also helps in reducing the time needed to develop a software and also acts a guideline for future software developments.

2.2 SOLUTION FUNCTION

Continuous integration the practice of frequently integrating one's new or changed code with the existing code repository should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.[9] Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build. The practicalities of doing this in a multi-developer environment of rapid commits are such that it is usual to trigger a short time after each commit, then to start a build when either this timer expires, or after a rather longer interval since the last build. Many automated tools offer this scheduling automatically.

- **Version Control**

This practice advocates the use of a revision control system for the project's source code. All artefacts required to build the project should be placed in the repository. In this practice and in the revision control community, the convention is that the system should be buildable from a fresh checkout and not require additional dependencies. Extreme Programming mentions that where branching is supported by tools, its use should be minimised.[9] Instead, it is preferred for changes to be integrated rather than for multiple versions of the software to be maintained simultaneously.

- **Artifact Manager**

While many developers have adopted Maven as a build tool, most have yet to understand the importance of maintaining a repository manager both to proxy remote repositories and to manage and distribute software artifacts. A repository stores two types of artifacts: releases and snapshots. Release repositories are for stable, static release artifacts and snapshot repositories are frequently updated repositories that store binary software artifacts from projects under constant development. While it is possible to create a repository which serves both release and snapshot artifacts, repositories are usually segmented into release or snapshot repositories serving different consumers and maintaining differ-

ent standards and procedures for deploying artifacts. Much like the difference between a production network and a staging network, a release repository is considered a production network and a snapshot repository is more like a development or a testing network. While there is a higher level of procedure and ceremony associated with deploying to a release repository, snapshot artifacts can be deployed and changed frequently without regard for stability and repeatability concerns.

- **Continuous Integration Handler**

While there are many tools, I will focus on one of the most popular, Jenkins CI. This is one of the more popular (open source) tools available. Jenkins CI (the continuation of a product formerly called Hudson) allows continuous integration builds in the following ways:

- It integrates with popular build tools (ant, maven, make) so that it can run the appropriate build scripts to compile, test and package within an environment that closely matches what will be the production environment
- It integrates with version control tools, including Subversion, so that different projects can be set up depending on projection location within the trunk.
- It can be configured to trigger builds automatically by time and/or changeset. (i.e., if a new changeset is detected in the Subversion repository for the project, a new build is triggered.)
- It reports on build status. If the build is broken, it can be configured to alert individuals by email.

Jenkins is an automation engine with an unparalleled plugin ecosystem to support all of your favorite tools in your delivery pipelines, whether your goal is continuous integration, automated testing, or continuous delivery.

- **Test Automator**

In software testing, test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual

outcomes with predicted outcomes.[1] Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or perform additional testing that would be difficult to do manually. Test automation is critical for continuous delivery and continuous testing. There are many approaches to test automation, however below are the general approaches used widely:

- Graphical user interface testing. A testing framework that generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.
- API driven testing. A testing framework that uses a programming interface to the application to validate the behaviour under test. Typically API driven testing bypasses application user interface altogether. It can also be testing public (usually) interfaces to classes, modules or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.

- **Continuous Deployment**

Continuous deployment is the next step of continuous delivery: Every change that passes the automated tests is deployed to production automatically. Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

2.3 OPERATING ENVIRONMENT

The system is expected to be operated in Linux as well as in windows with the support of respective JRE (Java Runtime Environment). This system based project is completely platform independent. The most important requirement is the Internet connection. This tool is coded using JDK 1.6.

2.4 GANTT CHART

Gantt chart is a graphical representation of allocation of resources to the activities. Here our resource is time. A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Some Gantt charts also show the dependency (i.e., precedence network) relationships between activities.

Gantt charts have become a common technique for representing the phases and activities of a project work breakdown structure (WBS), so they can be understood by a wide audience. Although a Gantt chart is useful and valuable for small projects that fit on a single sheet or screen, they can become quite unwieldy for projects with more than about 30 activities. Larger Gantt charts may not be suitable for most computer displays. A related criticism is that Gantt charts communicate relatively little information per unit area of display. That is, projects are often considerably more complex than can be communicated effectively with a Gantt chart. Although project management software can show schedule dependencies as lines between activities, displaying a large number of dependencies may result in a cluttered or unreadable chart.

Because the horizontal bars of a Gantt chart have a fixed height, they can misrepresent the time-phased workload (resource requirements) of a project, which may cause confusion especially in large projects. A related criticism is that all activities of a Gantt chart show planned workload as constant. In practice, many activities (especially summary elements) have front loaded or back-loaded work plans, so a Gantt chart with percent-complete shading actually may lead to mis-communications on the true schedule performance status.

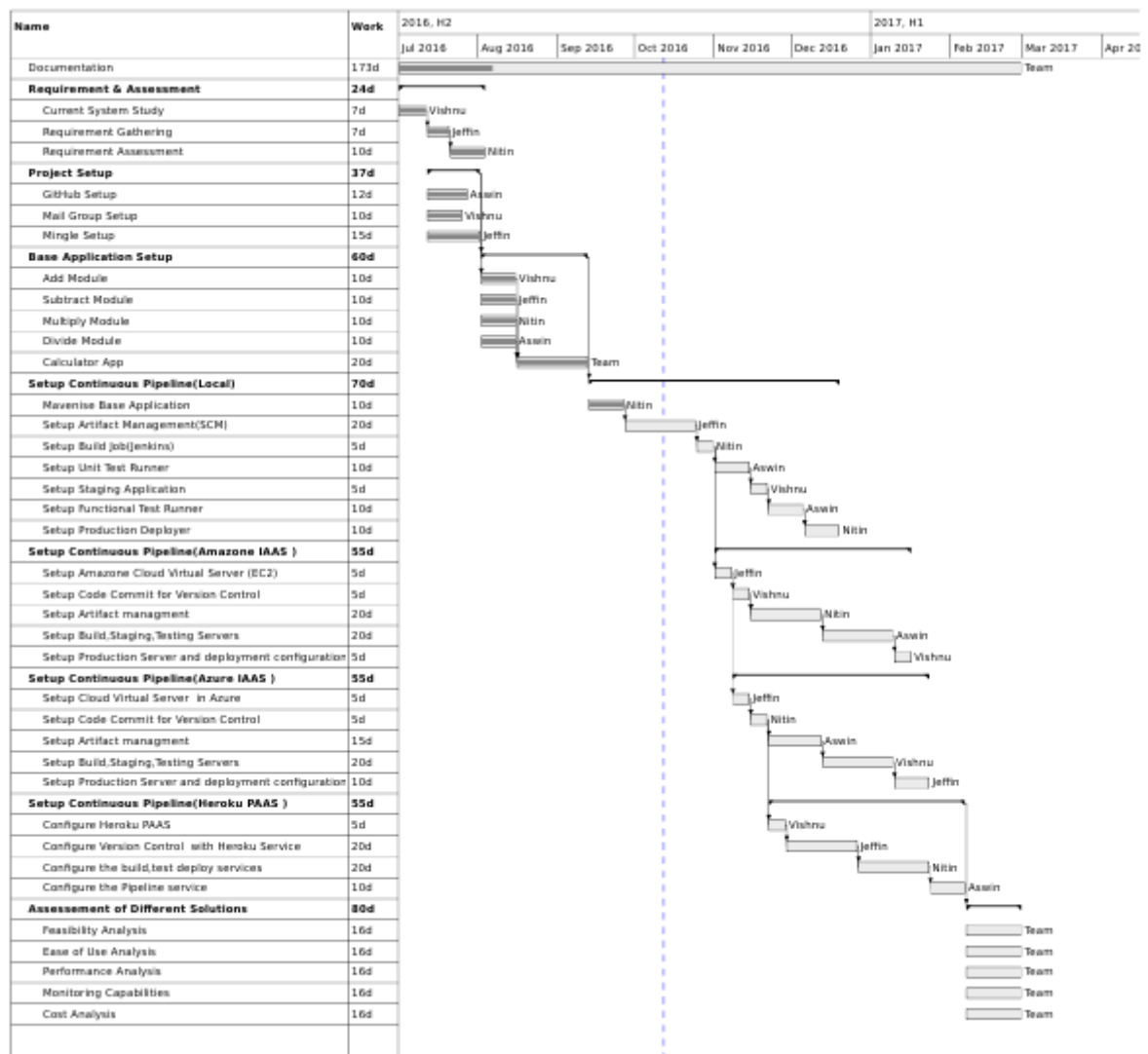


Figure 2.1: Gantt chart

2.5 COST ESTIMATION

Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of lines of code (KLOC).

COCOMO applies to three classes of software projects:

- Organic projects - small teams with good experience working with less than rigid requirements.
- Semi-detached - medium teams with mixed experience working with a mix of rigid and less than rigid requirements
- Embedded projects - developed within a set of tight constraints (hardware, software, operational ...)

The basic COCOMO equations take the form:

Effort Applied = $a(\text{KLOC})^b$ [person-months]

Development Time = $c(\text{Effort Applied})^d$ [months]

People required = Effort Applied / Development Time [count]

The coefficients a, b, c and d are given in the following table.

Software project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Figure 2.2: COCOMO Model Coefficients

2.6 EXTERNAL INTERFACE REQUIREMENTS

This system based software can be used in any operating system such as Microsoft Windows, Linux or any kind of user application interface, since it is platform independent.

2.6.1 User Interface

Interface hardware shall be encapsulated in a set of classes that isolate hardware specifications from the rest of the software. In particular, interfaces for specific hardware boards shall be implemented as derived classes of an abstract class.

2.6.2 Network Requirements

- Systems need minimum speed internet connection.

2.6.3 Software Requirements

- Git
- Jenkins
- Checkstyle
- Findbugs
- Mingle

Chapter 3

SYSTEM DESIGN

3.1 MODULES

Based upon the level of the product, project had been divided into 3 modules.
Sensing, Android application and Server module.

3.1.1 SENSING MODULE

The external hardware used is the Raspberry pi and it is connected to the server. In the sensing module, sensor is activated when the accident occurs. Arduino is used for converting analog sensor values to digital values.

3.1.2 ANDROID APPLICATION

In this module it fetches the information from the collision sensor and pass it the alert to the server. It also fetches the accurate GPS location of the accident site. A map is also displayed showing the exact accident location. The shortest path for reaching the accident location is also shown along with the map.

3.1.3 SERVER SIDE

In this module it passes the alert message to the nearest control station and to the emergency number provided by the user.

3.2 DATA FLOW DIAGRAMS

3.2.1 Purpose

The data flow diagram (DFD) is used for classifying system requirements to major transformation that will become programs in system design. This is the starting point of the design phase that functionally decomposes the required specifications down to the lower level of details. It consists of a series of bubbles joined together by lines. Bubbles: Represent the data transformations. Lines: Represent the logic flow of data. Data can trigger events and can be processed to useful information. Systems analysis recognizes the central goal of data in organizations. This data flow analysis tells a great deal about how organization objectives are accomplished.

3.2.2 Description

- Process : Describes how each input data is converted to output data.
- Data Store : Describes the repositories of data in a system.
- Data Flow : Describes the data flow between Processes, Data stores, Entities.
- Entity : An external entity causing the origin of data.

<u>Elements Reference</u>	<u>Symbols</u>
Data Flow	
Process	
Database	
Entity	

Figure 3.1: Notations used in DFD

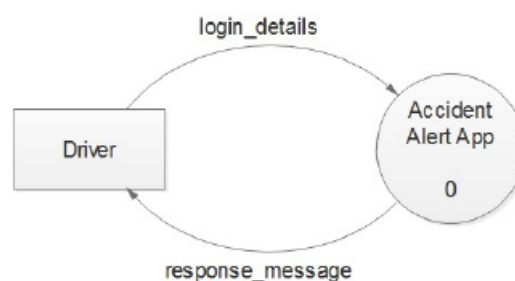
LEVEL 0

Figure 3.2: Level 0 DFD

Description

The driver install the application by providing his name and emergency number to be contacted in case of accident. The driver receives a confirmation message.

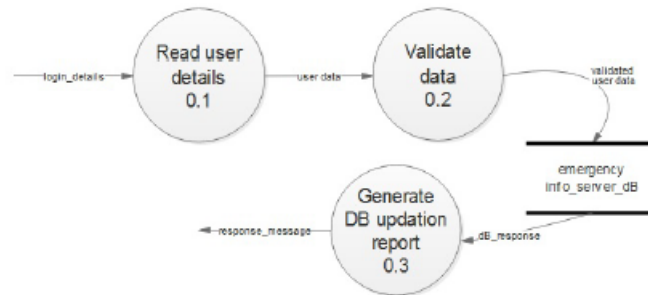
LEVEL 1

Figure 3.3: Level 1 DFD

Description

During app installation the user register his name and phone number to be contacted in case of emergency. The name and the emergency number is validated. Emergency number is registered to the database. The database is updated. Response message is passed to user.

3.3 USE CASE DIAGRAMS

3.3.1 Purpose

Use case diagram in the Unified Modeling language (UML) is a type of behavioral diagram. Its purpose is to represent the graphical overview of the functionality provided by a system in terms of actors, their goals (represented use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Role of the actors in the system can be depicted.

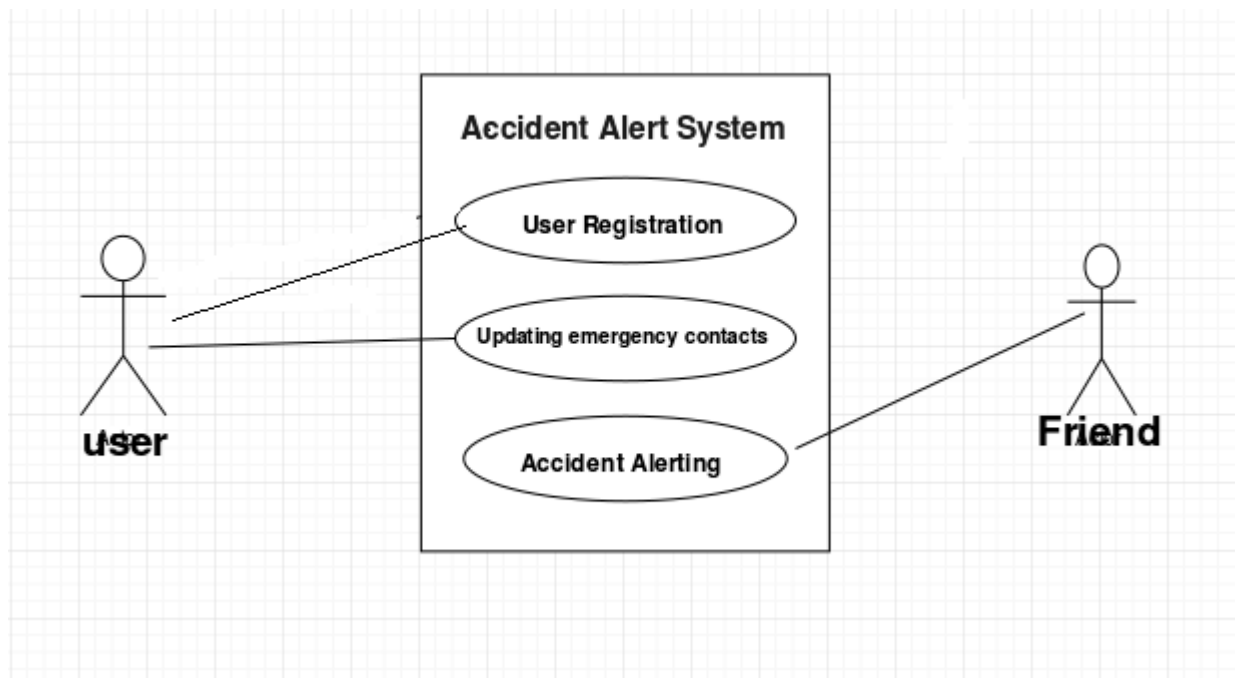


Figure 3.4: Use case Diagram

3.3.2 Description

The user can register by providing his number and emergency number to be contacted. Accident alerting system alerts the emergency contact and nearest control station.

3.4 SEQUENCE DIAGRAM

3.4.1 Purpose

A Sequence diagram depicts the sequence of actions that occur in a system. It portrays the different perspectives of behavior of the system and different types of inferences can be drawn from them. The invocation of method since each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system.

3.4.2 Description

When the accident occurs the sensing module senses the pressure variations and if it is above the predefined threshold value then it sends an alert to the android application module. Android application fetches the GPS location of the accident site and sends it to the server. Server sends the accident alert message to the nearest control station and the emergency number provided by the user.

3.5 ACTIVITY DIAGRAM

3.5.1 Purpose

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

3.5.2 Description

After the user registration, the details are updated in the database. While driving the user need to initialise the raspberry pi module and thus monitoring the sensor. If an accident occurs, then the GPS location of the site is fetched and is send to the server. Server sends the message to alert receivers. Otherwise it continues to monitor. While user is not driving, he can initialise sleep mode.

3.6 HARDWARE DIAGRAM

3.6.1 Purpose

The purpose of this diagram is to show the "as deployed" logical view of logical application components in a distributed network computing environment. The diagram is useful for the following reasons:

- Enable understanding of which application is deployed where.
- Establishing authorization, security, and access to these technology components.
- Understand the Technology Architecture that support the applications during problem resolution and trouble shooting.

3.6.2 Description

The collision sensor sensor senses the pressure variations and sends it to the raspberry pi through the arduino. Collision information is passed to the drivers mobile phone.

3.7 ER DIAGRAM

3.7.1 Purpose

Database is recognized as a standard and is available virtually for every computer system. The general theme behind every database is to integrate all the information. The database is an integrated collection of data and provides centralized access to the data. The user authentication in the project and accident details is implemented using the database.

3.7.2 Description

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. Here vehicle number is the primary key. Name, phone number, emergency number are other attributes of the driver entity. Helpline number have an IS A relationship with control station and emergency number.

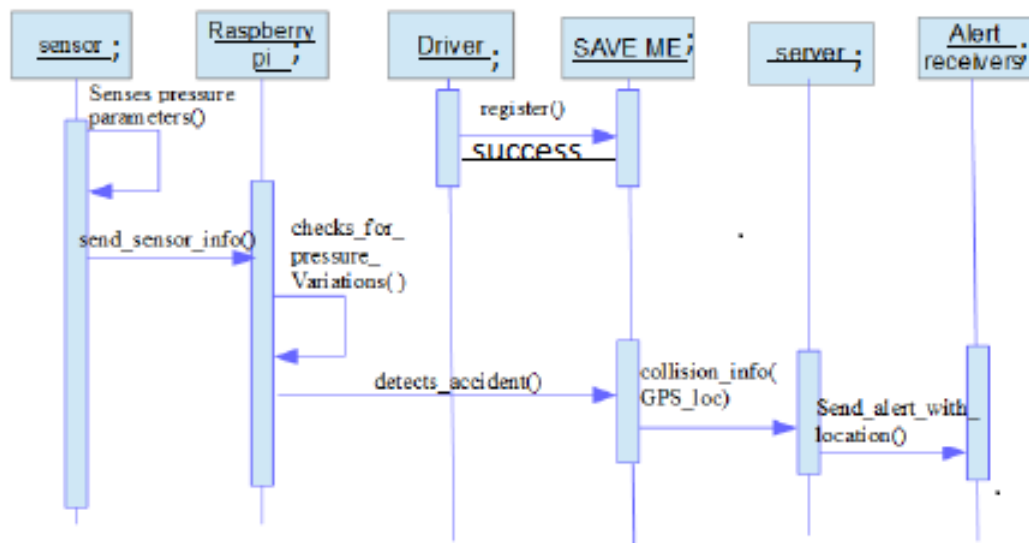


Figure 3.5: Sequence Diagram

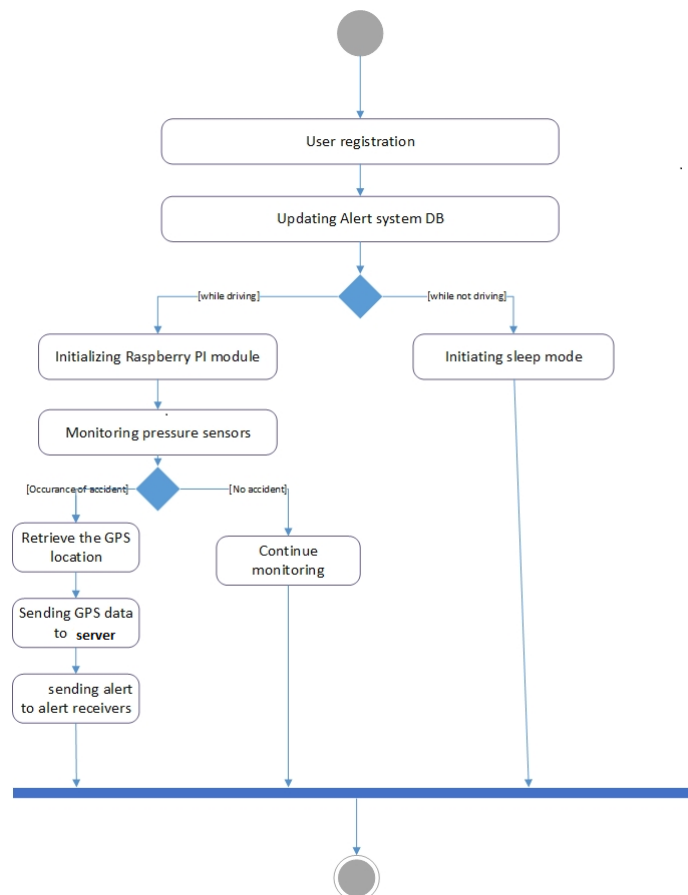


Figure 3.6: Activity Diagram

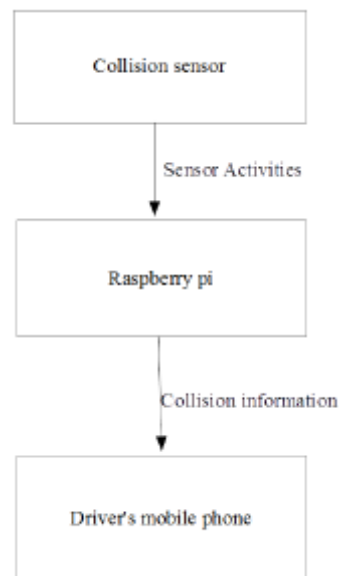


Figure 3.7: Hardware Diagram

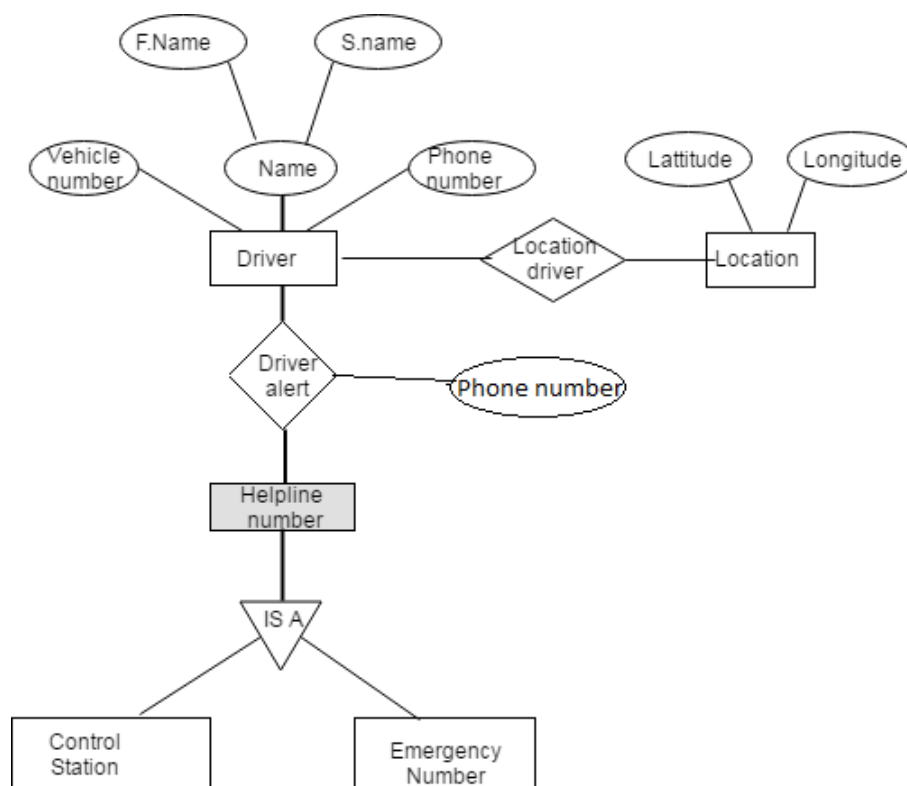


Figure 3.8: ER Diagram

Chapter 4

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence about the new system to users that it will work effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of achieve the changeover, an evolution of change over method. The product is developed in Java environment in android platform. The software used for the development are Android Studio and SQLyog for the database. The hardware platform is Raspberry pi, collision sensor and arduino. This chapter may explain our implementation details.

4.1 LANGUAGES AND PLATFORM USED

The product is developed in Java environment and we are using SQLyog for database. SQLyog Server automatically tunes many of the server configuration options, therefore requiring little, if any, tuning by a system administrator. Although these configuration options can be modified by the system administrator, it is generally recommended that these options be left at their default values, allowing SQLyog Server to automatically tune itself based on run-time conditions.

4.1.1 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. Python is managed by the non-profit Python Software Foundation.

4.1.2 Android

Android Studio is the official integrated development environment (IDE) for Android platform development. Android is a mobile operating system based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation. Android is designed primarily for touch screen mobile devices.

4.1.3 Jsp

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. A Java Server Pages component is a type of Java servlet that is designed to fulfill the role of a user interface. JSP is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun-Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing Java Beans components, passing control between pages and sharing information between requests.

4.2 SCREEN SHOTS

4.2.1 Android Application

- When a user installs SAVE ME in their mobile phone, a screen for app registration may appear of the following form.

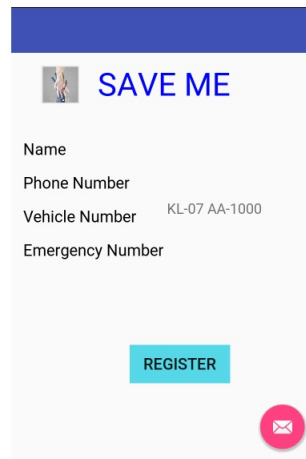


Figure 4.1: user registration in android application

- Any user using the app has to press the connect button in order to establish connection with the server. A screen of the following format may appear.

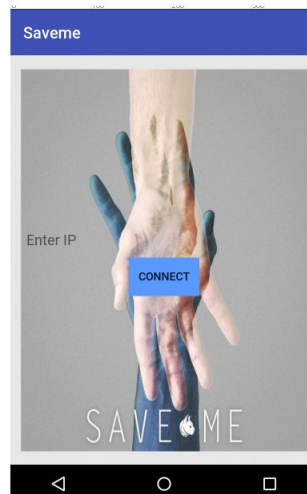


Figure 4.2: Connecting to Server

4.2.2 Web page

- Whenever an accident occurs the server sends an accident notification to the near-by control station of the accident location.

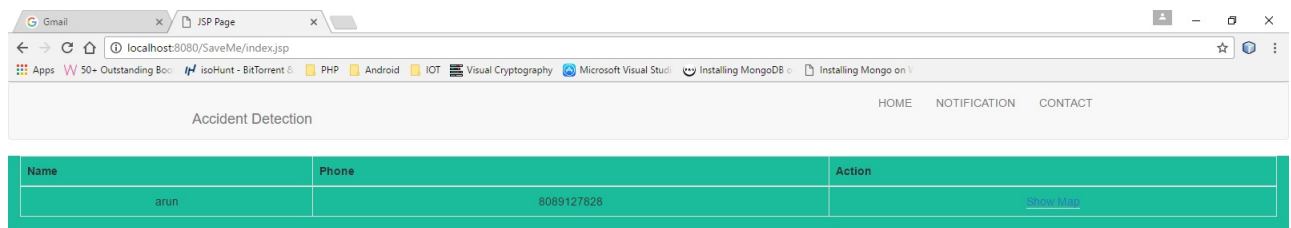


Figure 4.3: Accident notification in web page

- A map will appear with the indication of the accident area in the web page as in the following way for locating the exact location.

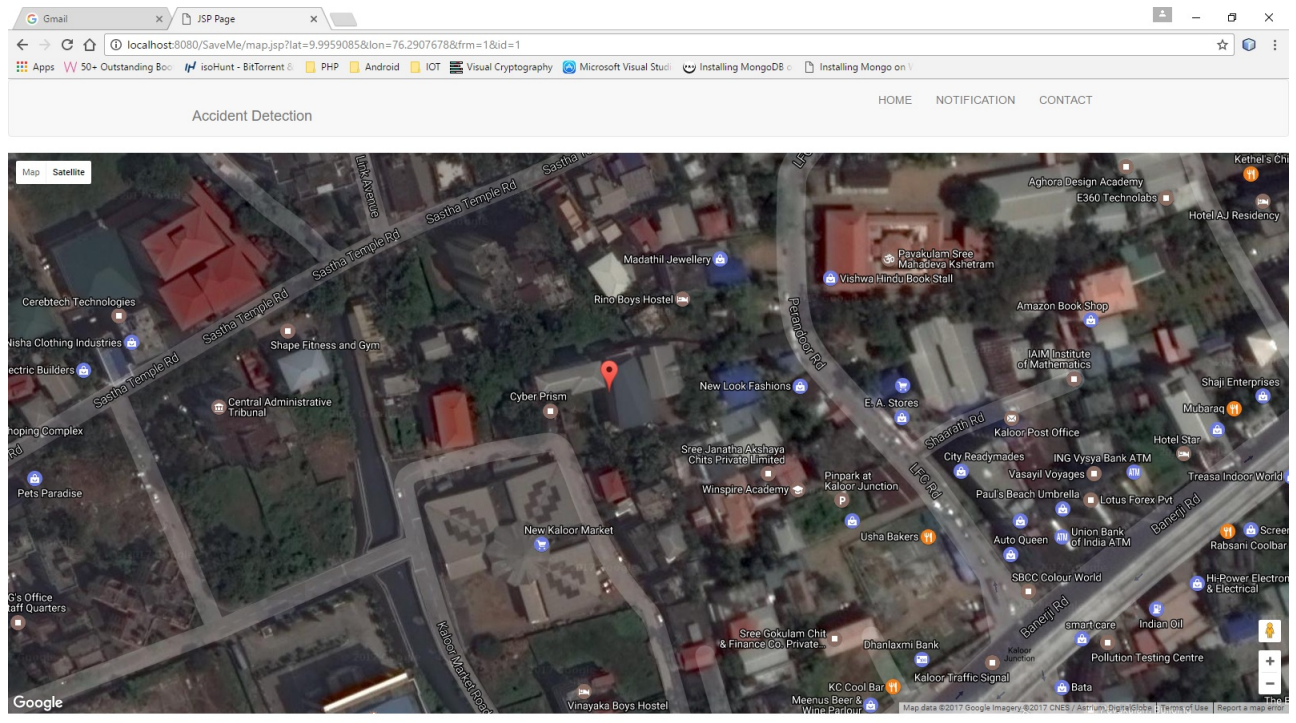


Figure 4.4: Map in web page

Chapter 5

TESTING

When a system is developed, it is hoped that it performs properly. In practice, however some errors always occur. The main purpose of testing an information system is to find the error and correct them. A successful test is one that finds an error. System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is the process of executing a program with the intent of finding an as yet undiscovered error. Nothing is complete without testing. Testing is vital to the success of the system.

The main objectives of the system testing are:

- To ensure during operation the system will perform as per specification.
- To make sure that the system meets user requirements during operation.
- To verify that the controls incorporated in the system function as intended.

If the testing conducted successfully, it will uncover errors in the software. As a secondary benefit, testing demonstrates that the software functions appear to be working according to specification and that performance requirements appear to have been satisfied.

The system Save Me is tested in such a way that almost all errors that may occur are found and corrected. The test process carried out in this system includes the following:

5.1 CODE TESTING

In code testing the logic of the developed system is tested. For this every module of the program is executed to find any error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions. This testing was done side by side with coding. This examines the logic of the program. In Java special test cases are used for testing the code. Every part of the program was tested in this phase.

5.2 UNIT TESTING

Unit testing is undertaken after a module has been coded and reviewed. Before carrying this testing, the unit test cases have to be designed and the test environment for the unit under test has to be developed. The various test cases are driver and stub modules. The main objective is to determine the correct working of the individual modules. During the testing each module is isolated from other modules and individually unit tested. It involves a precise definition of the test cases, testing criteria, and management of test cases. The modules that are tested include Android module, Server module and Sensing module.

5.3 INTEGRATION TESTING

System testing does not test the software as a whole, but rather than integration of each module in the system. The primary concern is the compatibility of individual modules. One has to find areas where modules have been designed with different specifications of data lengths, type and data element name. Testing and validation are the most important steps after the implementation of the developed system. The system testing is performed to ensure that there are no errors in the implemented system. The software must be executed several times in order to find out the errors in the different modules of the system. Each of the modules were integrated together and subjected to testing.

5.4 VALIDATION TESTING

Validation refers to the process of using the new software for the developed system in a live environment i.e., new software inside the organization, in order to find out the errors. The validation phase reveals the failures and the bugs in the developed system. It will come to know about the practical difficulties the system faces when operated in the true environment. Validation test was performed in the Login section. By testing the code of the implemented software, the logic of the program can be examined. A specification test is performed to check whether the specifications stating the program are performing under various conditions. Apart from these tests, there are some special tests conducted which are given below:

- **Peak load test** This determines whether the new system will handle the volume of activities when the system is at the peak of its processing demand. The test has revealed that the new system is capable of handling the demands at the peak time.
- **Storage testing** This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available.
- **Performance time testing** This test determines the length of the time used by the system to process transaction data.

5.5 SYSTEM TESTING

After all units of a program have been integrated together and tested, system testing is taken up. It is same for both procedural and object oriented programming. System tests are designed to validate a fully developed system to assure that it meets its requirements. The system test cases can be classified into performance and functionality test cases. The functionality test cases are designed to check whether the software satisfies the functional requirements as documented in the SRS document. The performance tests on the other hand test the conformance of the system with non-functional requirements of the system.

5.6 OUTPUT TESTING

After the performance of unit testing, the next step is output testing. No system would be useful if it does not produce the required output in the specific format, thus output format on the screen is found to be correct when the format was designed in the system phase according to the user need.

The maintenance of software is the time period in which software product performs useful works. Maintenance activities involve making enhancement to software product, adapting product to new environment and correcting problems. It includes both the improvement of the system function.

It may involve the continuing involvement of a large proportion of computer department resources. The main task may be to adapt existing system in a changing environment. System should not be changed casually following informal requests. To avoid unauthorized amendments, all requests for change should be channeled to a person nominated by management. The nominated person has sufficient knowledge of the organizations computer based systems to be able to judge the relevance of each proposed change.

No annual costs for support or maintenance are required. Of course, the individual system components come with limited warranty from the manufacturers, eg:, the PC, mobile phones, etc.

There is no obligation to purchase or pay for any extended maintenance or support.

5.7 GOAL OF TESTING

Many users may use our project. So the project designer must test all the modules of the project. The main goal of our project is, whenever user uses our project, it should run without any error.

5.8 PASS/FAIL CRITERIA

The pass/fail criteria specifies a set of constraints whose satisfaction leads to approval or disapproval of the proper functioning of the system .

5.9 PASS CRITERIA

The system must meet all the functional and non-functional requirements. Pass all the test cases, get the expected response, and get acceptable performance to be tested pass.

- User can login and submit request.
- Officials can verify and process the request
- Periodic update of database

5.10 FAIL CRITERIA

If one of the following situations happens, the system is considered to fail:

- User cannot login
- Database updation failure
- OTP sending failed
- Human error

5.11 TEST REPORTS

The detailed test reports prepared for each function. A sample test report is given below:

Test case preparation helps the user and the developer to find and fix the errors easily and in advance. Save Me is well tested with the proper test cases and thus passed a better unit test.

Table 5.1: Test Report

Name	SAVE ME
Version	1.0
Author	Anamika G V, Mekha S, Priyanka S, Safeena Ashraf.
Approved By	Self
Date	Monday, 6th March 2017
Role	User can operate the system.
Prerequisite	The user is logged into the system
User/Actor	System Response
Registration	User can register and use the system.
Run Applications	Any accident may be detected and notification may be reported.
Handle User Data	The server can handle various user data
The test project was successfully saved	

Elaborated test cases also prepared subjected the system for thorough testing. The test cases prepared are in the above format.

5.12 BLACK BOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. Test case preparation helps the user and the developer to find and fix the errors easily and in advance.

The test cases prepared are in the following format:

Table 5.2: Test Cases

Test	Usecase step	Action performed /User input	Expected Result /system response	Actual result	Do expected and actual result correspond
1	Admin	Create user	Successful creation of user	As Expected	Yes
2	User	Enter name phone no. emergency no. vehicle no.	Display success. Next page contain a connect button	As Expected	Yes
3	User	Presses the connect button	Connection establish between user and raspberry pi	As Expected	Yes
4	Sensor	Crack in sensor when accident occurs	arduino converts analog signal to digital and send to pi	As Expected	Yes
5	Sensor	Pi sense threshold and send to phone	Get notification in phone page	As Expected	Yes
6	User	Send accident information to control station and emergency no.	Get location of accident and take rescue action	As Expected	Yes

Chapter 6

FUTURE SCOPE

In the proposed system all details about the system is designed into an application. SAVE ME: An Automatic Accident Detection And Alert System For Automobiles is used for providing help to the accident victims. There are many cases where valuable lives are lost due to the lack of help at the correct time. We are developing this application for providing help in case of accidents for the victims. It is designed as a web based application, which will work smoothly on any computer system with internet connection. The application sends the accident message to the nearest control station as well as to the emergency number provided by the application user. The application provides the exact GPS location of the accident site which makes it very helpful for the police to provide immediate help for the victim. The document automation software is also supported by a powerful database where the documents are arranged, making updates and collaboration easy and fast. The system proposed here is meant for use in any automobile with the user having android mobile phone. The basic steps in application processing is same everywhere, and hence the extended version of this system can be used in advanced areas too.

Chapter 7

CONCLUSION

SAVE ME: An Automatic Accident Detection And Alert System For Automobiles is used for providing help to the accident victims. There are many cases where valuable lives are lost due to the lack of help at the correct time. We are developing this app for providing help in case of accidents for the victims. The app sends the accident message to the the nearest control station as well as to the emergency number provided by the app user. The app provides the exact GPS location of the accident site which makes it very helpful for the police to provide immediate help for the victim. With the help of this app we may be able to save many lives as there is sureity of help.

REFERENCES

- [1] Bhandari Prachi, Dalvi Kasturi, Chopade Priyanka,“ Intelligent Accident-Detection And Ambulance- Rescue System” *INTERNATIONAL JOURNAL OF SCIENTIFIC and TECHNOLOGY RESEARCH*, VOLUME 3, ISSUE 6, JUNE 2014
- [2] Collision avoidance system.Dec.2013.[Online].Available:[https://en.wikipedia.org/ wiki- i/Collision/avoidance/system](https://en.wikipedia.org/wiki/Collision_avoidance_system).
- [3] Raspberry pi,Sept.2013.[Online]. Available:[https://en.wikipedia.org/wiki/Raspberry/Pi](https://en.wikipedia.org/wiki/Raspberry_Pi).

GLOSSARY

ER : Entity Relationship
GPS : Global Positioning System
HTTP : Hyper Text Transfer Protocol
IPV4 : Internet Protocol Version 4
JSP : Java Server Page
SQL : Structured Query Language
UML : Unified Modeling Language
ERD : Entity Relationshi Diagram
JVM : Java Virtual Machine
DFD : Data Flow Diagram

Index

A

Android

Android Application, 1

Arduino , 7

C

Conclusion , 14

D

Data Flow Diagram , 8

Database, 11

Design, 7

ER Diagram , 16

Existing system,

F

Functional Requirements , 16

G

Gantt chart , 16

Goal of testing,

H

Hardware Requirements , 8

I

Introduction , 1

Integration Testing

J

Java

JSP

Modules , 4

N

Non Functional Requirements , 5

O

Operational Feasibility,8

Output Testing

P

Python

R

References

S

Screen shots

Sequence Diagram

Software Requirements,12

System testing,67

T

Technical Feasibility,8

Test reports

U

UML diagrams, 23 ,16

Use case diagram

Unit Testing

V

Validation Testing