

**UNIVERSITY OF MASSACHUSETTS DARTMOUTH
DEPARTMENT OF COMPUTER & INFORMATION SCIENCE**

PILL REMINDER APPLICATION

**CIS 600: MASTER'S PROJECT
FINAL REPORT**

**PRESENTED BY
SAI ALEKHYA RAVI (02101882)**

**PROJECT ADVISOR
DR. AMIR AKHAVAN MASOUMI**

Table of Contents

ABSTRACT	V
CHAPTER 1: INTRODUCTION	1
1.1 OBJECTIVE	1
1.2 EXISTING SYSTEM:	1
1.2.1 EXISTING SYSTEM DISADVANTAGES:	1
1.2.2 LITERATURE REVIEW	2
1.3 PROPOSED SYSTEM:	4
1.3.1 PROPOSED SYSTEM ADVANTAGES	4
1.3.2 PROPOSED SYSTEM DISADVANTAGES	4
CHAPTER 2: PROJECT DESCRIPTION	5
2.1 GENERAL:	5
2.2 METHODOLOGIES:	5
2.2.1 MODULES NAME:	5
2.2.2 MODULE EXPLANATION:	5
2.2.3 MODULE DIAGRAM:	6
CHAPTER 3: REQUIREMENTS ENGINEERING	9
3.1 GENERAL	9
3.2 HARDWARE REQUIREMENTS	9
3.3 SOFTWARE REQUIREMENTS	9
3.4 FUNCTIONAL REQUIREMENTS	10
3.5 NON-FUNCTIONAL REQUIREMENTS	10
CHAPTER 4: DESIGN ENGINEERING	12
4.1 GENERAL	12
4.2 FUNCTIONAL INTERACTION OVERVIEW	12
4.3 STRUCTURAL BLUEPRINT	14
4.4 INSTANCE MAP	15
4.5 LIFECYCLE DIAGRAM	16
4.6 INTERACTION FLOW DIAGRAM	17

4.7 COMMUNICATION MAP	18
4.8 WORKFLOW REPRESENTATION	19
4.9 COMPONENT OVERVIEW	20
4.10 DATA FLOW DIAGRAM:	21
4.11 E-R DIAGRAM:.....	24
4.12 DEPLOYMENT LAYOUT	25
4.13 SYSTEM OVERVIEW DIAGRAM	26
4.14 CONCLUSION	26

CHAPTER 5: DEVELOPMENT TOOLS 27

5.1 ABOUT ANDROID	27
5.2 WHAT IS ANDROID?	27
5.3 ABOUT OPEN HANDSET ALLIANCE (OHA).....	27
5.4 FEATURES OF ANDROID.....	27
5.5 CATEGORIES OF ANDROID APPLICATIONS	27
5.6 HISTORY OF ANDROID	28
5.7 ANDROID ARCHITECTURE	28
5.8 ANDROID CORE BUILDING BLOCKS	29
5.9 ANDROID ACTIVITY LIFECYCLE	31

CHAPTER 6: SNAPSHOTS 33

CHAPTER 7: SOFTWARE TESTING..... 35

7.1 GENERAL	35
7.2 DEVELOPING METHODOLOGIES.....	35
7.3 TYPES OF TESTS	35
7.3.1 UNIT TESTING	35
7.3.2 FUNCTIONAL TEST	36
7.3.3 SYSTEM TEST.....	36
7.3.4 PERFORMANCE TEST.....	36
7.3.5 INTEGRATION TESTING	36
7.3.6 ACCEPTANCE TESTING.....	36
7.2.7 BUILD THE TEST PLAN.....	37

CHAPTER 8: APPLICATION 38

8.1 APPLICATIONS.....	38
8.2 FUTURE ENHANCEMENT:.....	38

CHAPTER 9: CONCLUSION & REFERENCE	<u>39</u>
9.1 CONCLUSION	39
9.2 REFERENCE:.....	40
APPENDIX	<u>41</u>

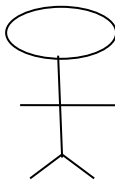
List of Figures

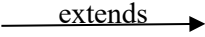

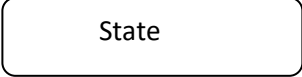
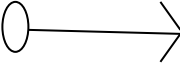
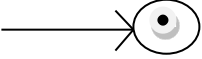
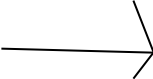
Figure 1: Module Diagram: Registration	6
Figure 2: Module Diagram: Login	6
Figure 3: Module Diagram: Add Tablets.....	7
Figure 4: Module Diagram: Add Reminder.....	7
Figure 5: Module Diagrams: View/Update Tablets	8
Figure 6: Module Diagram: View/Update Reminders	8
Figure 7: Use-Case Diagram	13
Figure 8: Class Diagram	14
Figure 9: Object Diagram.....	15
Figure 10: State Diagram.....	16
Figure 11: Sequence Diagram.....	17
Figure 12: Collaboration Diagram	18
Figure 13: Activity Diagram	19
Figure 14: Component Diagram	20
Figure 15: Data Flow Diagram: Level 0	21
Figure 16: Data Flow Diagram: Level 1	22
Figure 17: Data Flow Diagram	23
Figure 18: E-R Diagram	24
Figure 19: Deployment Diagram	25
Figure 20: System Architecture	26
Figure 21: Activity Lifecycle	32
Figure 22: Home Page	33
Figure 23: Set Reminder Page	34

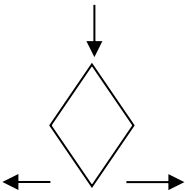
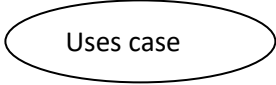
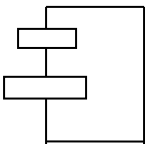
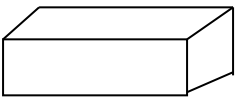
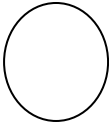
ABSTRACT



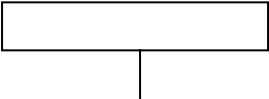
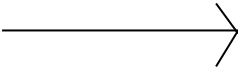
This is an innovative System for any user mostly targeting the aged population as a medical helper. The user or anyone behalf of the user can enter the medicine reminders such as tablet color, name, quantity and when it should be taken with a reminder. The reminder helps the user to investigate the details of tablets that he or she must take and will also speak out the details. This is advance application where a doctor himself can enter the details for the patients helping him or her to remind the user. The System also helps the user to click picture of medicines or make use of OCR where the system will make use of the camera and print the text on the screen and the user must tap the text to save them. The System uses Mobile Vision API for Implementing OCR. This can also be a medium of not typing the name of a medicine while using OCR instead saving lot of time. This system can be useful for a person where he has many tablets to intake. Thus, the system named medical helper.

LIST OF SYMBOLS

S.NO	NAME	NOTATION	DESCRIPTION
1.	Class	<div style="display: flex; align-items: center; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>+ public</i> <i>-private</i> </div> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>Class Name</i> <hr/> <i>-attribute</i> <hr/> <i>-attribute</i> </div> </div>	Represents a collection of similar entities grouped together.
2.	Association	<div style="display: flex; align-items: center; justify-content: space-around; margin-bottom: 10px;"> <div style="border: 1px solid black; padding: 5px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px; text-align: center;">NAME</div> <div style="border: 1px solid black; padding: 5px;">Class B</div> </div> <div style="display: flex; align-items: center; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px;"></div> <div style="border: 1px solid black; padding: 5px;">Class B</div> </div>	Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single class.
4.	Aggregation	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="display: flex; justify-content: space-around; width: 100%;"> <div style="border: 1px solid black; padding: 5px;">Class A</div> <div style="border: 1px solid black; padding: 5px;">Class A</div> </div> <div style="display: flex; justify-content: space-around; width: 100%; margin-top: 10px;"> <div style="text-align: center;">↑</div> <div style="text-align: center;">↑</div> </div> <div style="display: flex; justify-content: space-around; width: 100%;"> <div style="border: 1px solid black; padding: 5px;">Class B</div> <div style="border: 1px solid black; padding: 5px;">Class B</div> </div> </div>	Interaction between the system and external environment

5.	Relation (uses)	Uses	Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.

12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interact ion between the system and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.

17.	External entity		Represents external entities such as keyboard,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message 	Represents the message exchanged.

LIST OF ABBREVIATION

S.NO	ABBREVIATION	EXPANSION
1.	DB	Database
2.	JVM	Java Virtual Machine
3.	JSP	Java Server Page
4.	CB	Collective Behavior
5.	RSSS	Ramp secret sharing scheme
6.	JRE	Java Runtime Environment

CHAPTER 1: INTRODUCTION

The reality is that many of us will need assistance in our later years. In some cases, people need nothing more than occasional visits from a home nurse, some light housekeeping, meals on wheels, and visitors willing to talk and notify about pills. While there is a movement to make aging at home possible for more people, it is not always an option. Dementia and other illnesses can require around the clock medical care and monitoring, things often more easily given in a professional facility than at home. On the other end, increasing number of smart systems opens area where medical treatment can be utilized to completely new level. In this paper we show a working solution how a smart home can be utilized to help people with medication related reminders. Proposed flow starts when a new medication prescription is taken from the doctor.

1.1 OBJECTIVE

The System uses Mobile Vision API for Implementing OCR (Optical Character Recognition). This can also be a medium of not typing the name of a medicine while using OCR instead saving lot of time. This system can be useful for a person where he has many tablets to intake. Thus, the system named medical helper.

1.2 Existing System:

1. Users must enter the name and quantity/dose of the tablet/capsule manually every time. It cannot be added automatically.
2. Users must enter the reminder about the times of dosage manually i.e. 2 or 3 times in a day.
3. Users must manually select the duration of the reminder.
4. They are not facilitating anything regarding the original prescription. Everything needs to be done manually. We need an app which can reduce a lot of the manual work and automate stuff. Also, the existing systems have some major drawbacks.

1.2.1 Existing System Disadvantages:

1. Reminders cannot be set automatically. There is a need for manual work in setting the reminder.
2. A lot of time is consumed in manually setting the reminders.
3. They don't facilitate storing the original prescription.

4. The possibility exists for the existing systems to hang down due to the manual work involved.

1.2.2 LITERATURE REVIEW

TITLE: ArduMed-Smart Medicine Reminder for Old People

AUTHOR: Mayuresh Waykole, Vatsalya Prakash, Himanshu Singh, Nalini N (VIT University, Vellore)

YEAR: 2016

REVIEW

The paper ArduMed – Smart Medicine Reminder for Old People authored by Mayuresh Waykole and Vatsalya Prakash, Himanshu Singh and Nalini (VIT University, Vellore 2016) addresses the pertinent problem of forgetfulness in the case of old people and their scheduled medications. It is remarkable that this work has been aimed at solving a real-life problem with severe public health consequences, more so for the aged care.

Strengths:

Addressed Issue: The focus on the elderly population also depicts the understanding of the target group, with all its problems that include not only forgetfulness but also the burden of polypharmacy.

Application of Smart Technology: The application of Arduino based technology indicates a more practical, quicker, and efficient means of developing a dependable medication reminder system. This selection must help the general design of the system being uncomplicated and cheap.

Health Improvement: This system has the potential of improving the health of older users while reducing the economic burden on health care provision by minimizing medication non-adherence, which in turn prevents complications.

ArduMed is a positive advancement towards the enhancement of medication adherence in the elderly population through smart technology. Once additional development and assessment is done, it can be an important resource in the elderly healthcare space augmenting the health status and quality of life.

TITLE: Portable Medicine Reminder and Automatic Monitoring System

AUTHOR: Namrata Kataki, Assistant Professor

YEAR: 2016

REVIEW

The article, "Portable Medicine Reminder and Automatic Monitoring System" deals with a very common problem encountered among patients, especially the elderly, and those suffering from chronic illnesses which is non-compliance with medications. The author comes up with a solution that seeks to combine the two aspects of convenience and automation to ease the medication schedules and health observation. This review examines the merits, demerits, the research aspect and such other factors that are likely to be crucial to the outcome of the work.

Strengths:

Considerable Impact on Practice: The topic addresses a major gap in the practice of healthcare, which is the issue of a patient taking the medications as scheduled. The potential solution may ease complications arising from skipped doses, hence promoting a better quality of care for patients.

Creativity: Inventions that could smartly merge reminders for taking medicine and a system for monitoring body conditions demonstrate an innovative outlook. Due to the use of automation, there is a lesser need for human help which makes the system suitable for people with little or no tech knowledge.

Attention to Mobility: With the priority of mobility, it implies that the system can be suitable for active people who may need to take medication on-the-go. This characteristic is especially noteworthy nowadays when almost everything centers around mobility.

Technical Integration: Considering the abstract, this seems to rely on some new technology e.g. IoT, embedded systems etc. This is also in sync with the current direction in health care technology advancement.

Potential Impact:

The suggested device would probably assist many elderly who are on chronic medications and have problems in maintaining compliance. If the device is implemented and rolled out efficiently, it is likely to go a long way in cutting down on the cost of health care, readmissions and support patient autonomy.

Conclusion:

In my opinion, Namrata Kataki's work in the field of health care technologies is quite commendable, as it fulfills one of the endless gaps which exist in medication management. It is clear improvement that the device could be enhanced as a starting point for more stored evidence and function validation studies in a future progression of smart healthcare. This study lays a solid groundwork for related works in this area to be built on.

1.3 Proposed System:

The medicine reminder system will have one duty and that would be to remind the user that he is due for taking the medicine. We are trying to make sure that the user never forgets to take the medicine and hence we do the reminder. In the case that patient is outside, we have a mobile reminder app which will remind using mobile notifications for that time. The mobile application can be installed in the android devices. It will add recurring events to the mobile's calendar and will alert the user when he must take the medicine with the list of medicines and its prescribed dosage.

1.3.1 Proposed System Advantages

- The System can be operated only if the user is a registered user thus securing data.
- The System speaks out the reminders as well gives a detail about the reminder.
- The System also implements OCR by using Mobile Vision API by Google to capture text.
- The System doesn't require an internet connection.

1.3.2 Proposed System Disadvantages

- Since the system doesn't use internet, the data is saved offline and is phone dependent.
- If the phone is formatted or lost the data is lost.

CHAPTER 2: PROJECT DESCRIPTION

2.1 GENERAL:

This is an innovative System for any user mostly targeting the aged population as a medical helper. The user or anyone behalf of the user can enter the medicine reminders such as tablet color, name, quantity and when it should be taken with a reminder. The reminder helps the user to look into the details of tablets that he or she has to take and will also speak out the details. This is advance application where a doctor himself can enter the details for the patients helping him or her to remind the user. The System also helps the user to click picture of medicines or make use of OCR where the system will make use of the camera and print the text on the screen and the user has to tap the text to save them. The System uses Mobile Vision API for Implementing OCR. This can also be a medium of not typing the name of a medicine while using OCR instead saving lot of time. This system can be useful for a person where he has many tablets to intake. Thus the system named medical helper.

2.2 METHODOLOGIES:

2.2.1 MODULES NAME:

This project having the following eight modules:

- Registration
- Login
- Add Tablets
- Add Reminder
- View/Update Tablets
- View/Update Reminders

2.2.2 MODULE EXPLANATION:

- **Registration:** The user needs to register into the system.
- **Login:** The user has to login to make use the of the system and is remembered until he logs out.
- **Add Tablets:** The user must add the tablets and its details.
- **Add Reminder:** The user can mark reminders and add tablets and add other attributes along with them.
- **View/Update Tablets:** The user can view and update the tablet details.
- **View/Update Reminders:** The user can view and update the reminder details.

- **OCR:** The user is allowed to use OCR to scan medicines or object with text.
- **Picture:** Click and save picture for future reference.

2.2.3 MODULE DIAGRAM:

Registration



Figure 1: Module Diagram: Registration

In this registration page the user will start to give their information. this information is then stored in the system.

Login:

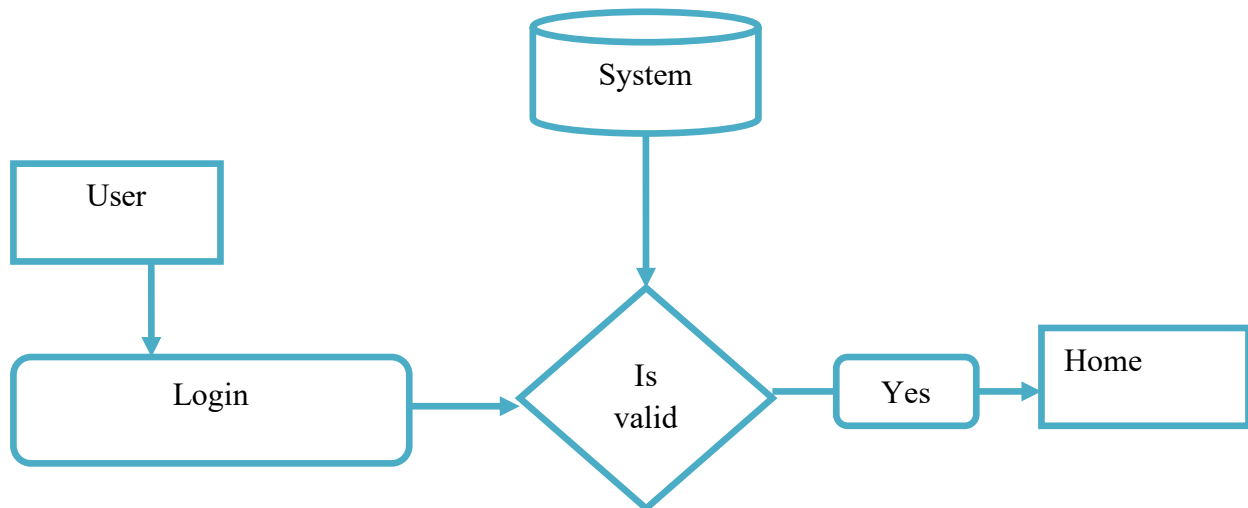


Figure 2: Module Diagram: Login

In the login page, the user gives their credentials. once the system confirms the credentials, the user has access to the home page.

Add Tablets:

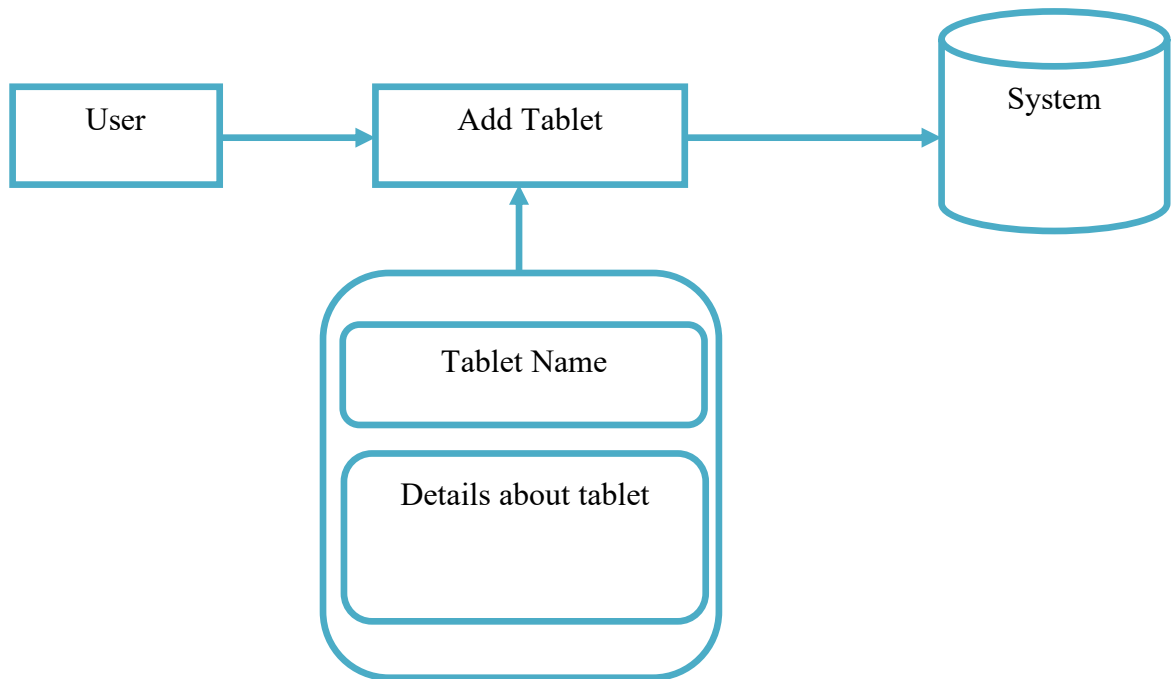


Figure 3: Module Diagram: Add Tablets

In the add tablet section, the user gets to add the name of the tablet. this information is stored in the system.

Add Reminder:

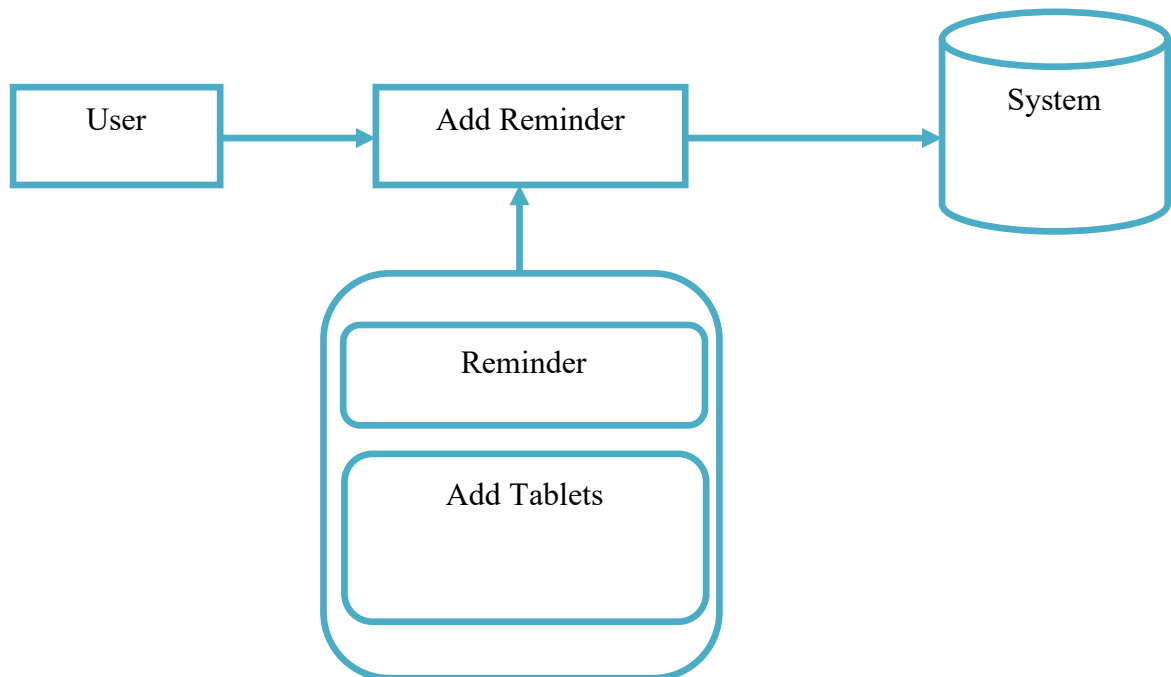


Figure 4: Module Diagram: Add Reminder

In the add reminder, the user gets to add the reminder to the tablet that was stored in the system.

View/Update Tablets:

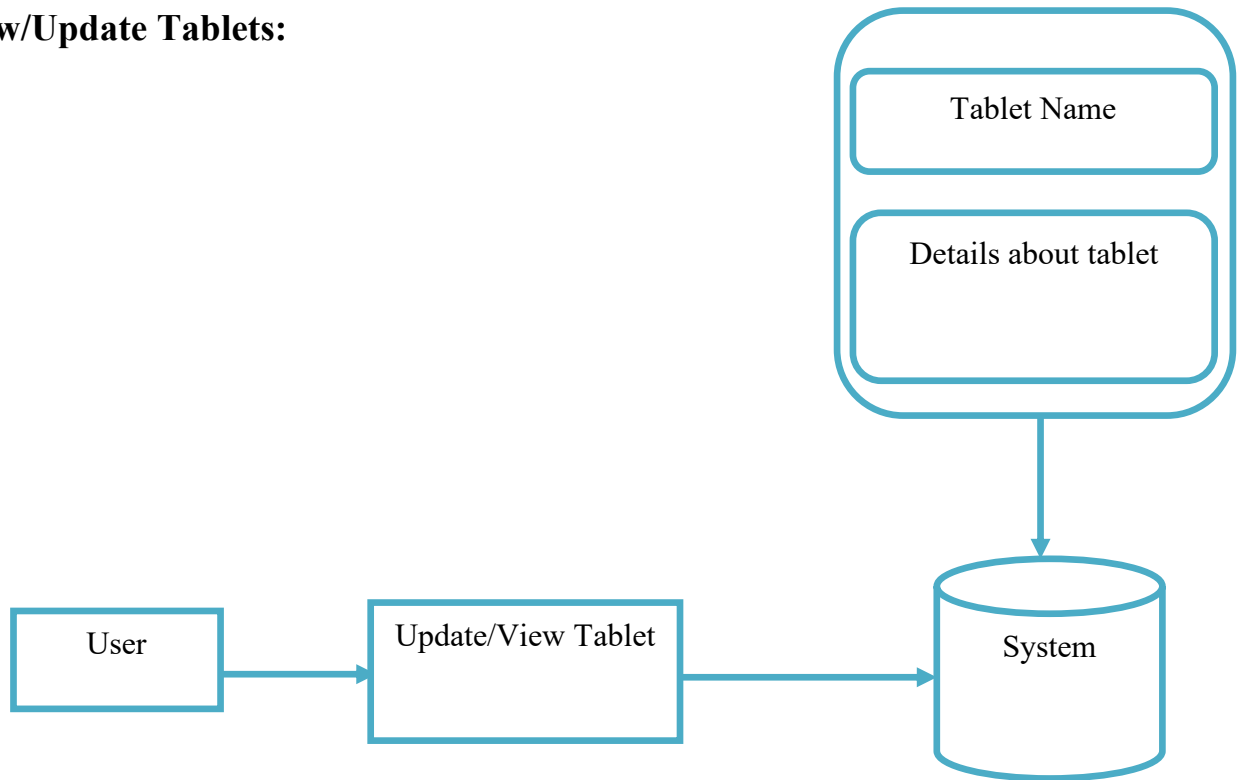


Figure 5: Module Diagrams: View/Update Tablets

To edit the information, the user can go to the view/update tablet. the user can get the onformation or make edits.

View/Update Reminders:

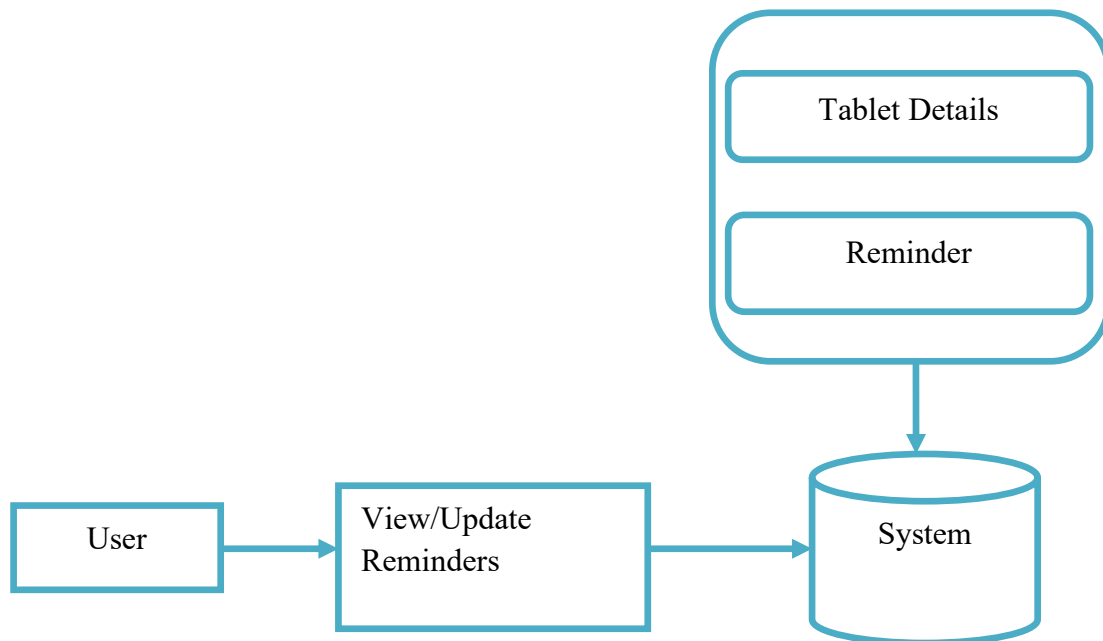


Figure 6: Module Diagram: View/Update Reminders

To edit the reminders, the user can go to the view/update reminders. the user can get the reminders information or make edits.

CHAPTER 3: REQUIREMENTS ENGINEERING

3.1 GENERAL

The project provides an effective solution in resolving all the arguments that occurs in organizations by considering all employees opinions. Android voting system using some main Hardware & Software requirements.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

HARDWARES

- Processor – i3
- Hard Disk – 5 GB
- Memory – 1GB RAM
- Android Phone with kitkat and higher.

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

SOFTWARES

- Windows XP, Windows 7(ultimate, enterprise)
- Android Studio

3.4 FUNCTIONAL REQUIREMENTS

- A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs.
- The system has an admin login that has overall control over it. Admin feed the issues or arguments in the system along with desired options.
- It provides authentication to individuals who wish to get connected. The registration details are stored in the database and whenever the Employees logs in, the employees credentials are retrieved to check whether the employee is an authorized employee or not.
- If the details entered do not match with any of the existing data then the system displays a warning.
- These questions can then be visible to all the employees through android devices. Employees have to first create an account into the system for casting their votes.
- At the end of the voting process the system counts all the votes casted and generates a brief report of the total votes accounted for yes, no and neutral. Eventually, the report is made available to admin, and he may view the maximum votes casted for.
- Hence the system helps admin to receive appropriate response from employees for the matters in question.

3.5 NON-FUNCTIONAL REQUIREMENTS

- **Portability:** It should run on specified platforms successfully. To achieve this we should test the product on all platforms before launching the product. If our project runs successfully on different platforms then our system is portable in nature.
- **Reliability:** The system should perform its intended functions under specified conditions. If our system satisfies all the specified conditions, then it is Reliable in nature.

- **Reusability:** The system should be extremely reusable as a whole or part. Make the system modularize and make sure that modules are loosely coupled. This project is having reusability nature because we can reuse whole or part of this project on other systems.
- **Robustness:** The system on the whole should be robust enough to perform well under different circumstances without any inconsistencies.
- **Testability:** The product of a given development phase should satisfy the conditions imposed at the start of that phase.
- **Usability:** It should be perfect and comfortable for users to work.
- **Security:** The system is completely based on the security. This system will provide security base on the password.

CHAPTER 4: DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

4.2 Functional Interaction Overview

The Use Case Diagram depicted in Figure 7 describes the system along with its users. Such diagrams identify system capabilities and all the users of the system, including the admins. It includes all system features such as user registration, log in, reminder creation, and managing the details of the tablets. The use of this diagram also clarifies in what the system achieves and who the target users are.

The user is the main primary user of this system who does the following actions:

- Create an account in the system
- Sign in to use the services
- Attach tablets and reminders
- Edit or delete already existing reminders

These interactions allow to fully consider how the user is getting help.

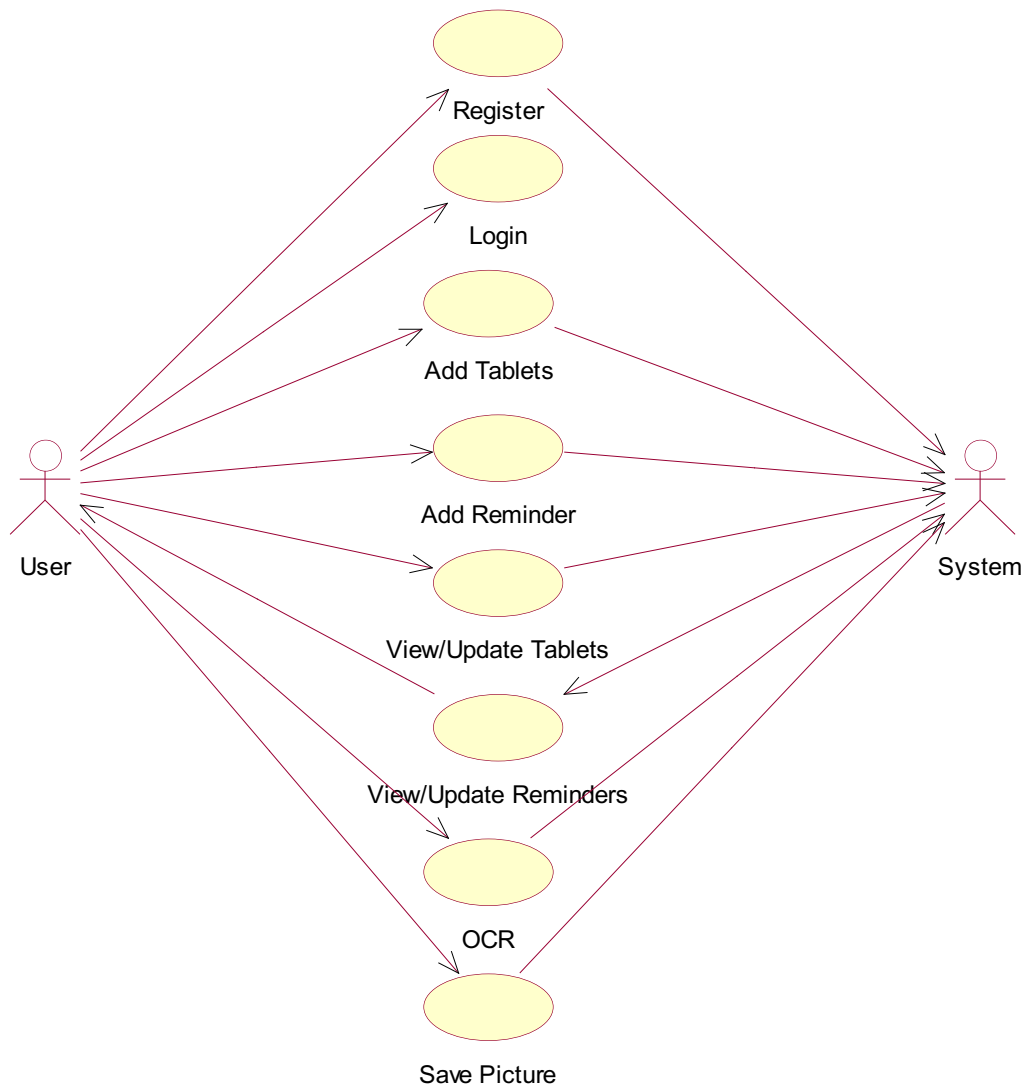


Figure 7: Use-Case Diagram

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

4.3 Structural Blueprint

The Class Diagram in Figure 8 represents diagrams that define the system as well as classes, attributes, and relationships of classes. It shows how parts work together to provide a particular service. For instance, the class User is linked with classes Reminder and Tablet which shows relation between a user, reminders of when to take medications, and the tablets.

Examples of such items are:

- User Class: Has data types like username and email.
- Tablet Class: Contains information such as the name of a tablet, dosage and the time to.
- Reminder Class: Controls the time of notifications and the corresponding tablet details.

This system makes it easier for maintain the system.

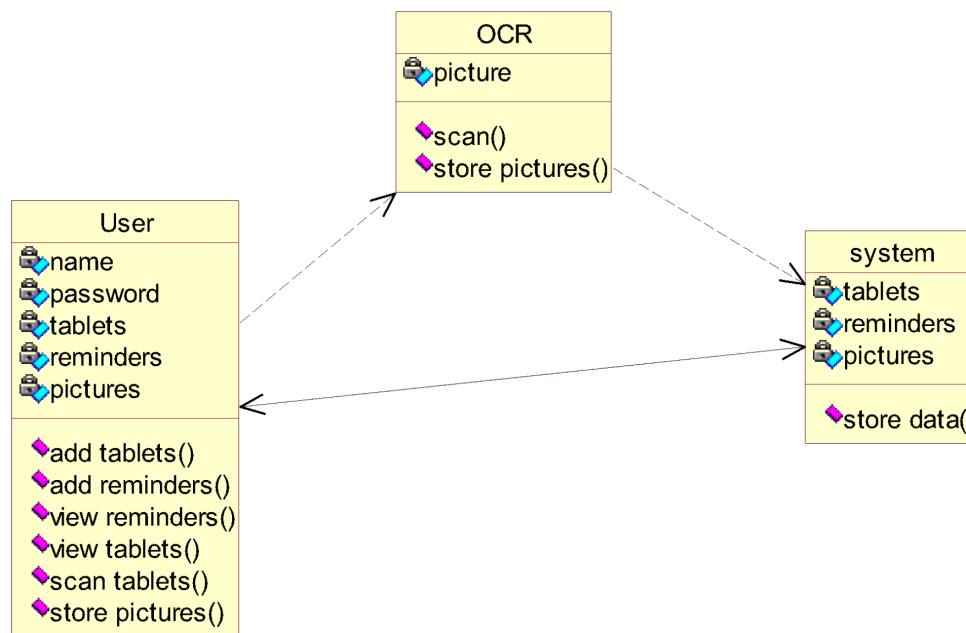


Figure 8: Class Diagram

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

4.4 Instance Map

The Object Diagram specified in Figure 9 traces a moment within the system depicting the objects of the classes with their links. It indicates the possible scenarios of using the system in practice, say when a user wants to set a reminder for taking some specific type of tablet.

This makes it possible to better appreciate the interconnections of systems in motion, such as:

- A user setting more than one reminder for several tablets.
- Certain characteristics of tablets inscribing their distinctiveness.

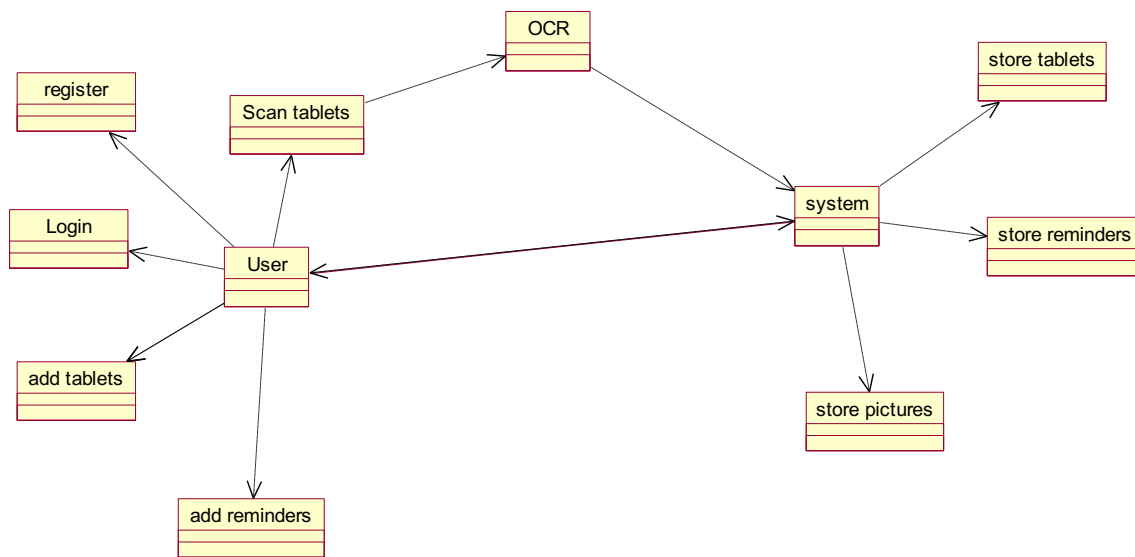


Figure 9: Object Diagram

In the above diagram talks about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

4.5 Lifecycle Diagram

Figure 10 is a State Chart for reminding a user which shows the various status of the reminder and how they change. This picture illustrates the lifespan of the reminders, that is to say, from when the reminder was created to the time it is been sent.

Notable key states are:

- Created: It corresponds to the time the user attempts to define a reminder.
- Active: It corresponds to the time the reminder had been set and is waiting for a notification.
- Completed: It corresponds to the time the reminder was sent.

This assurance offers a good insight into every possible step of any state change.

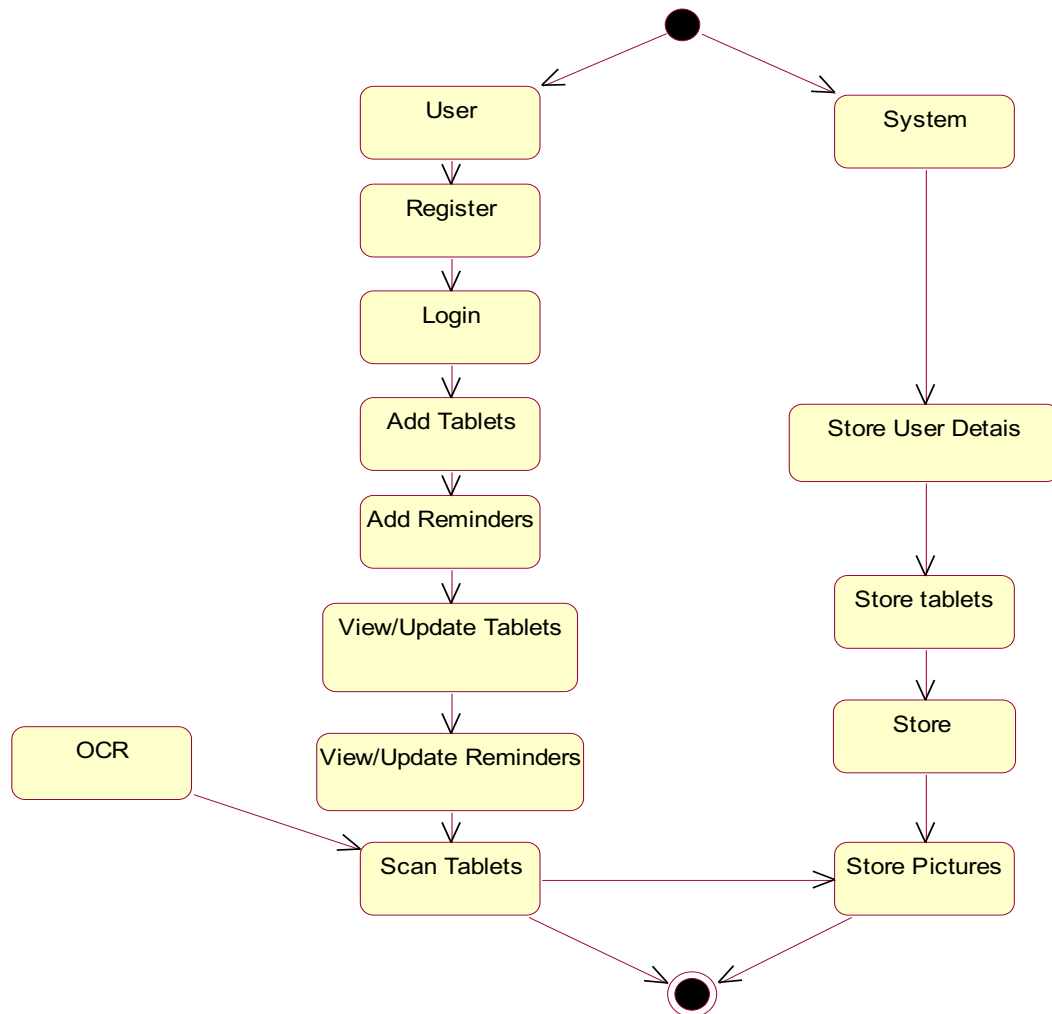


Figure 10: State Diagram

State diagram is a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states.

4.6 Interaction Flow Diagram

The black box model of the system is expressed in terms of sequence of activities and time in the case of creation of a reminder. The user tasks, systems responses are shown simultaneously.

In this phase, the following activities are carried out:

1. The user logs into the system.
2. The user adds a tablet and associated reminder.
3. The pertinent information is saved by the system and the reminder is set.
4. The messages are activated at the appropriate time.

The sequential representation makes the interaction between the user and the system quite clear.

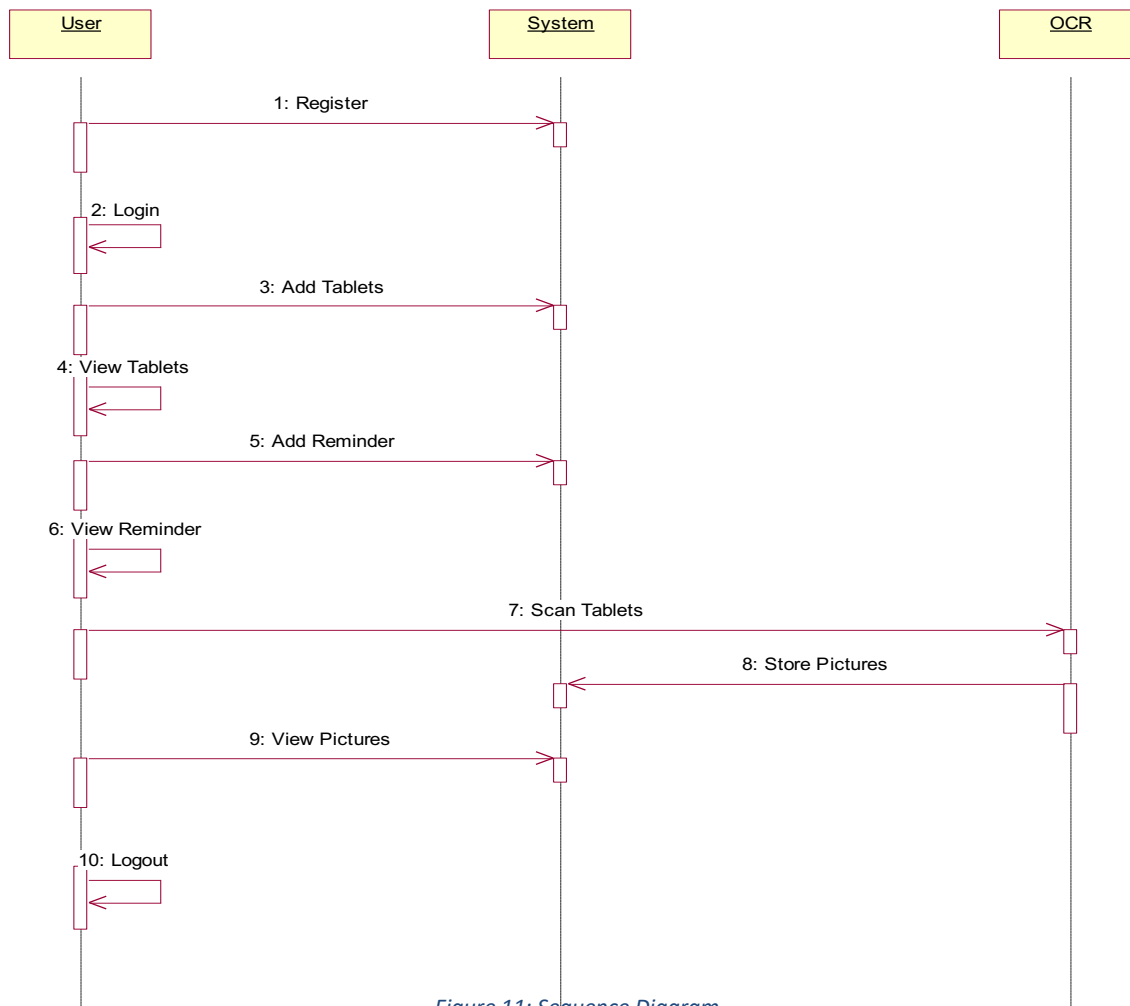


Figure 11: Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

4.7 Communication Map

The Collaboration Diagram presented in Figure 12 concentrates on the relationships and message flow between several objects. It focuses on the structural relations of the message sequence chart.

This is brought out especially in the telecommunication means where actions thereat on the user for instance change the state of Recall Responder and Tablet classes.

Here is how the interaction between objects is illustrated. For example, reminders and tablets classes are updated when a user performs some actions.

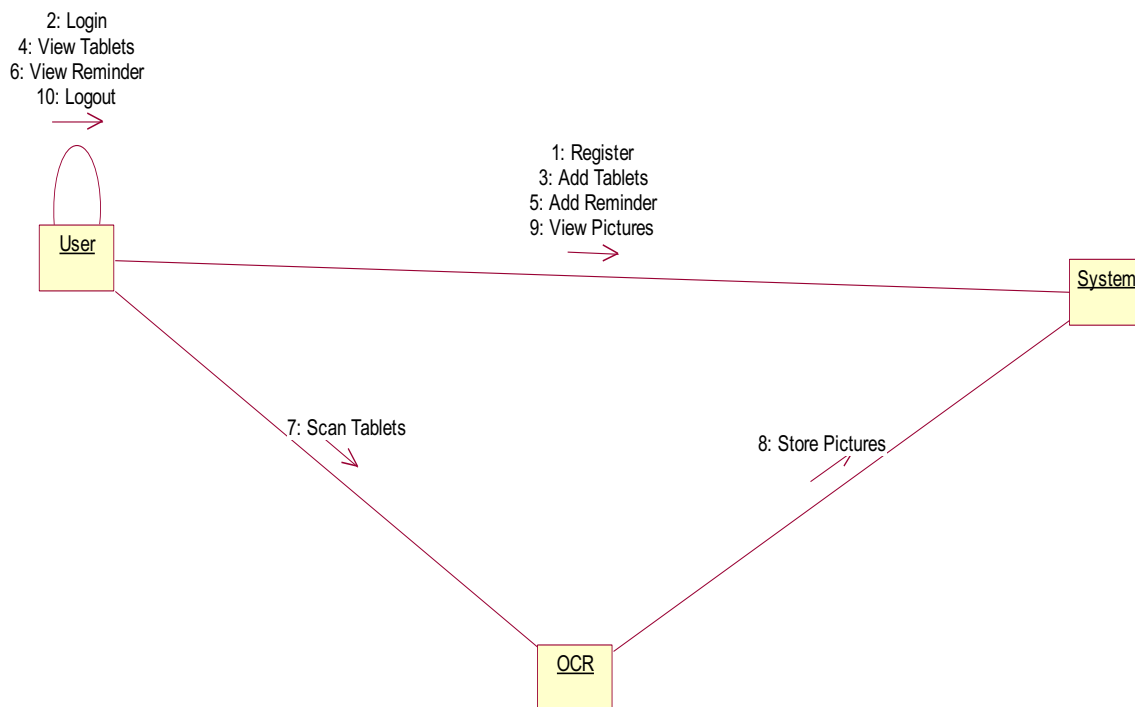


Figure 12: Collaboration Diagram

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

4.8 Workflow Representation

The Activity Diagram presented in Figure 13 illustrates the process within a reminder setup. This shows the order of the performance of tasks, together with the points in which tasks are taken at the same time.

These actions entail the following:

1. Details of the tablet are entered by the user.
2. Input provided is checked by the system.
3. The reminder on an event is formed and set.

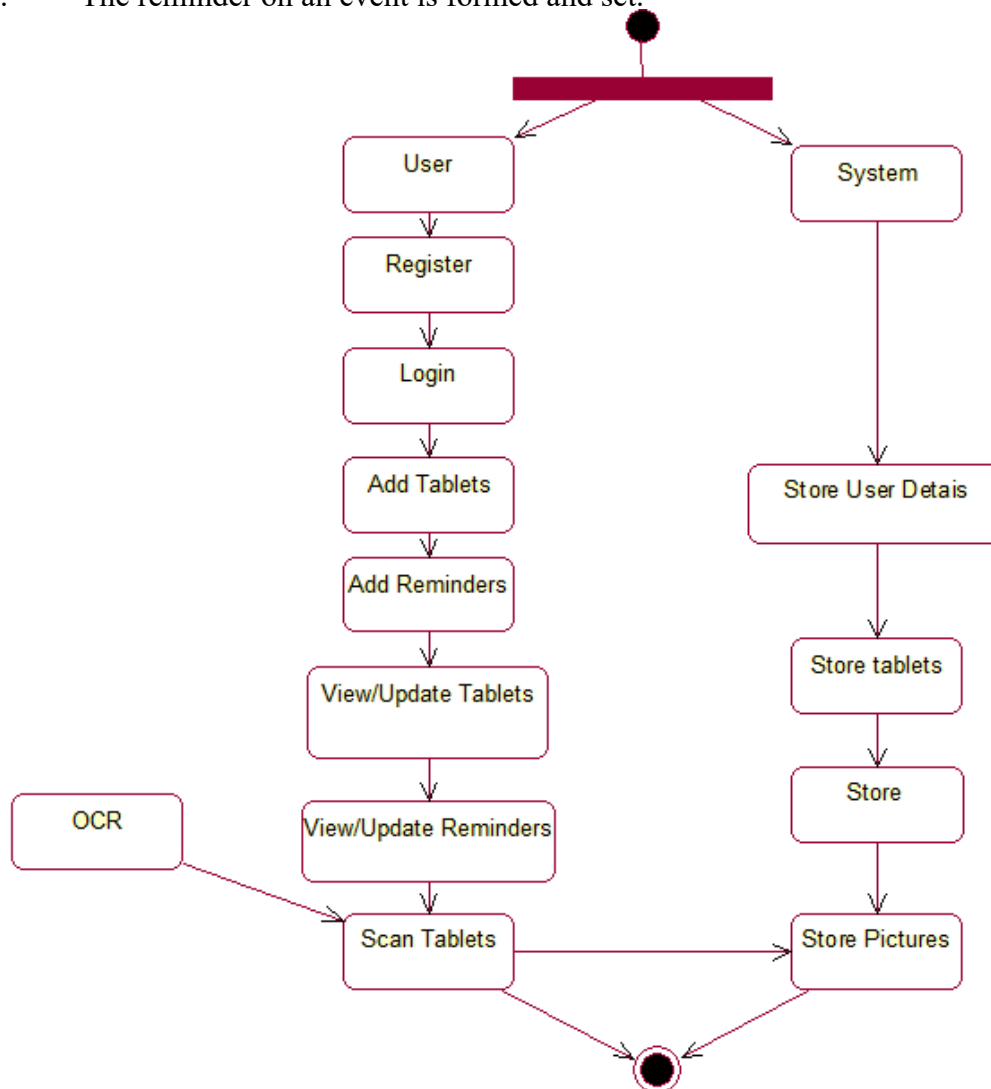


Figure 13: Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

4.9 Component Overview

The physical structure of a system is shown in the component diagram where main components are interconnected.

With respect to this, the components include the following:

- Interfaces modules.
- Storage devices for particulars of the tablets and reminder.
- Notification interfaces for sending notifications.

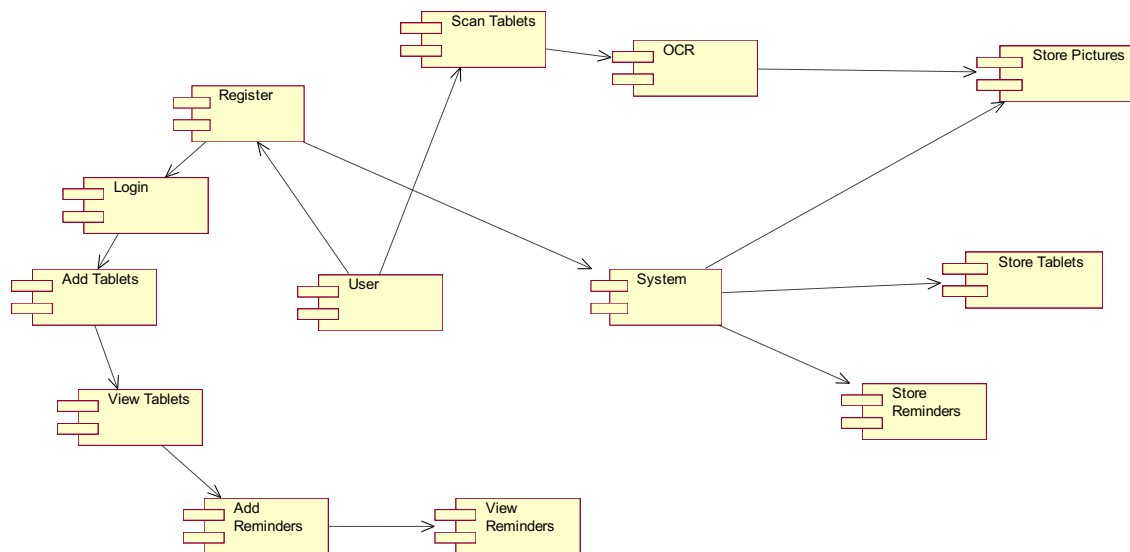


Figure 14: Component Diagram

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query, and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

4.10 Data Flow Diagram:

Level 0:

In Figure 15, the data flow in the system is illustrated. It indicates the way user information is first entered and then saved into their account, for instance, log in details along with the type of tablet.

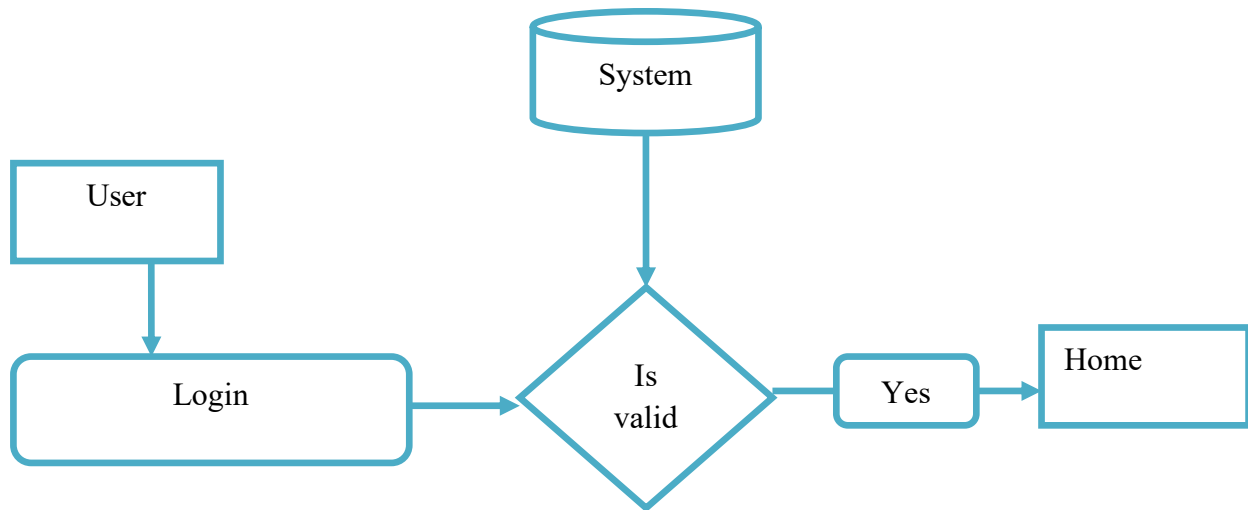


Figure 15: Data Flow Diagram: Level 0

The user submits his login information in a login screen that is shown as a single view, and this overview needs to be validated by the System in the logon area. If the credentials are successfully passing the check of the system, denoted as the Is valid condition, the user is taken to the Home page. This is implemented to prevent unauthorized entry into the system.

Level 1:

Emphasis was placed on the use of tablet and reminder data while the process was performed in Figure 16 in relation to further data flow. The diagram shows the logic processes which are interfacing with the user input through the data exchanges with the system and therefore assures accuracy of data input and output processes.

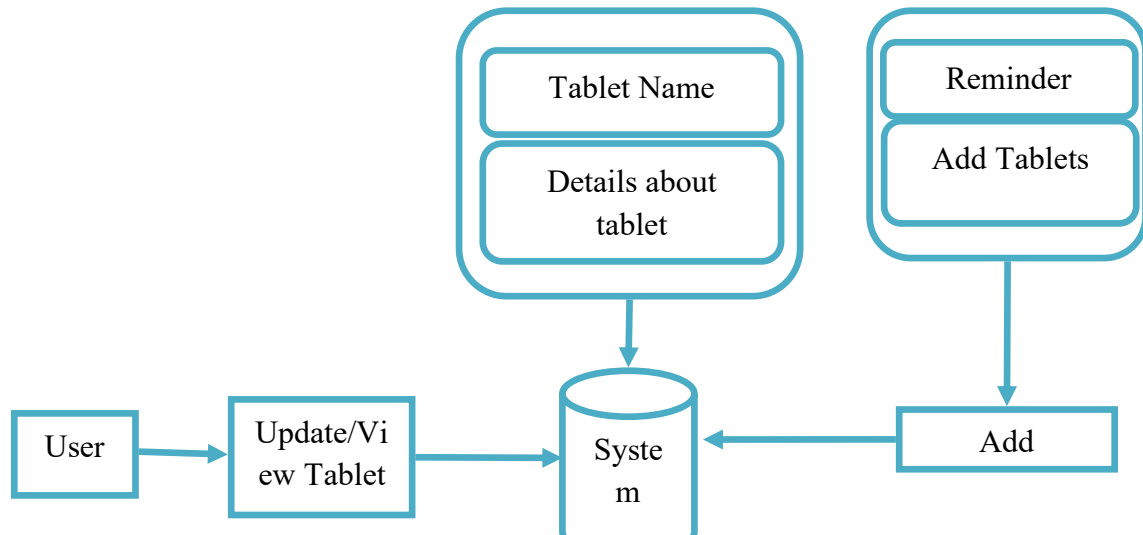


Figure 16: Data Flow Diagram: Level 1

The diagram above displays in detail how reminders and tablets are being handled. For instance, the User can change or check some information about the tablet name and the information about it by updating or viewing it through the Update or View Tablet Module, which concerns the System. The same is the case with the Add module, which includes reminders related to tablets as tablets are created and all the relevant information is stored and dealt with in the Systems.

Level 2:

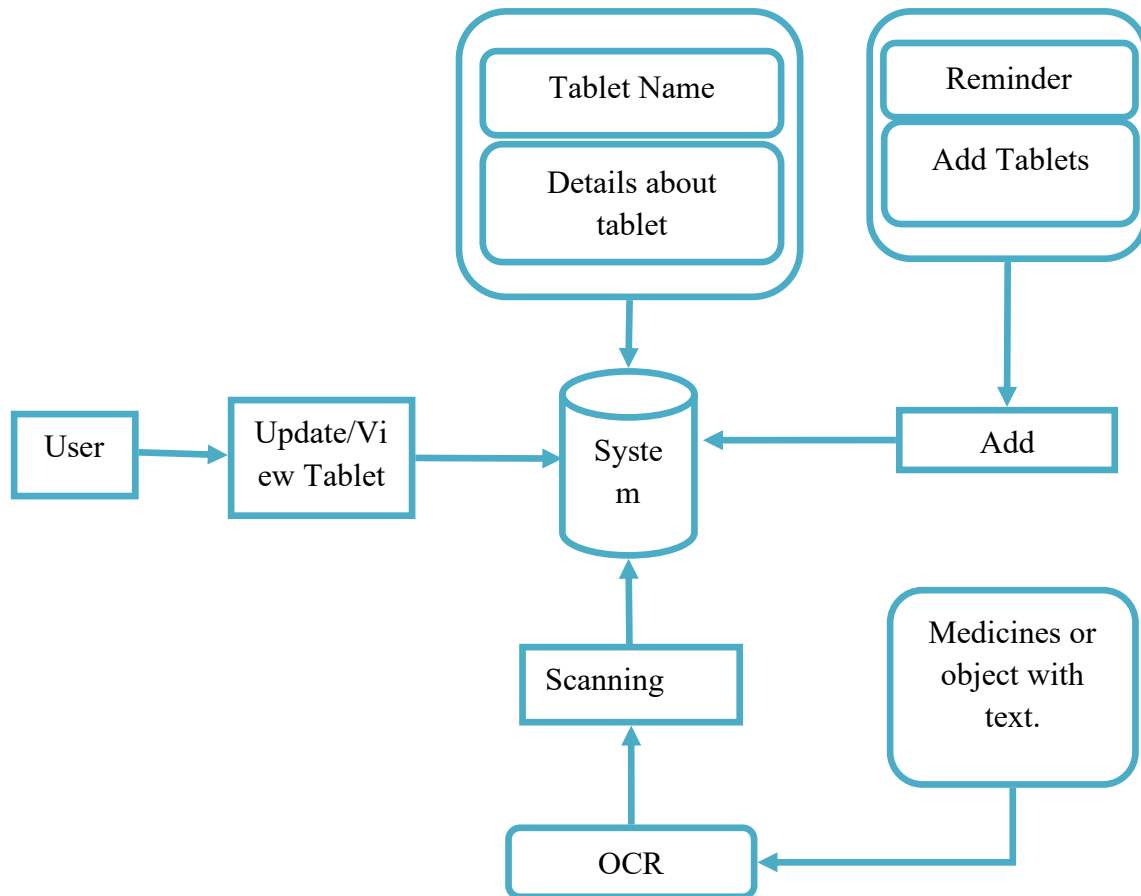


Figure 17: Data Flow Diagram

A Data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

4.11 E-R Diagram:

The overall structure of the database can be observed on the Entity-Relationship Diagram (Figure 18) with an emphasis on the relation of the User, Tablet, and Reminder entities.

This diagram addresses data organization and retrieval, which are necessary for the operation of the system.

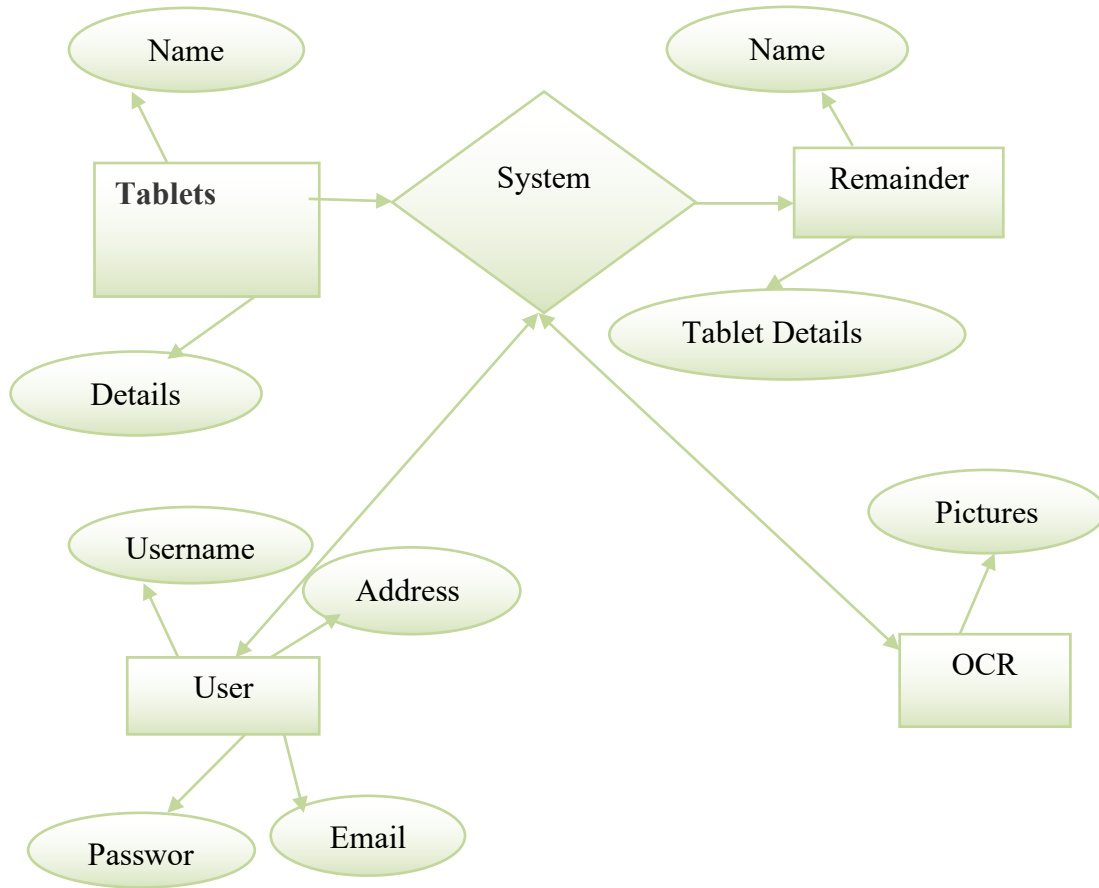


Figure 18: E-R Diagram

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

4.12 Deployment Layout

The Deployment Diagram in Figure 19 describes how the system is deployed. It reveals how the uniform software components are beacon distributed among the physical hardware nodes.

These devices include the Identified factors such as:

- mobile applications for tablets or smart phones.
- desktop applications connected to the server site.

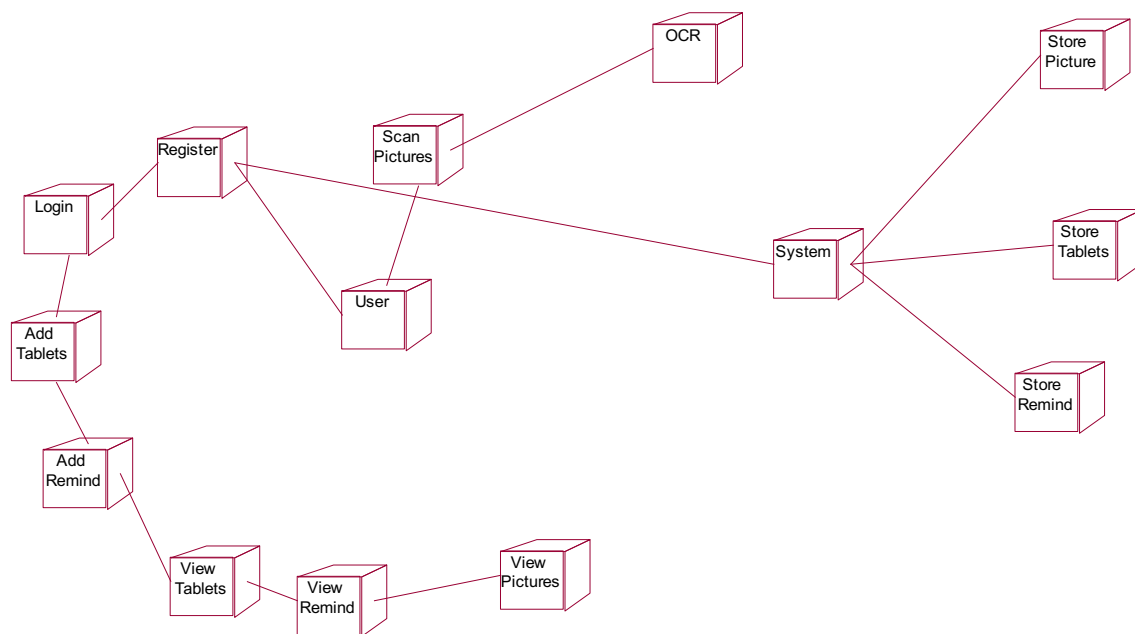


Figure 19: Deployment Diagram

In the Unified Modeling Language, a deployment diagram depicts how deploys are wired together to form larger deployment and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query, and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

4.13 System Overview Diagram

The System Architecture in Figure 20 incorporates all parts and serves as the comprehensive overview of the system. It makes sure that all the functionalities are integrated in a manner that best serves the requirements of the user.

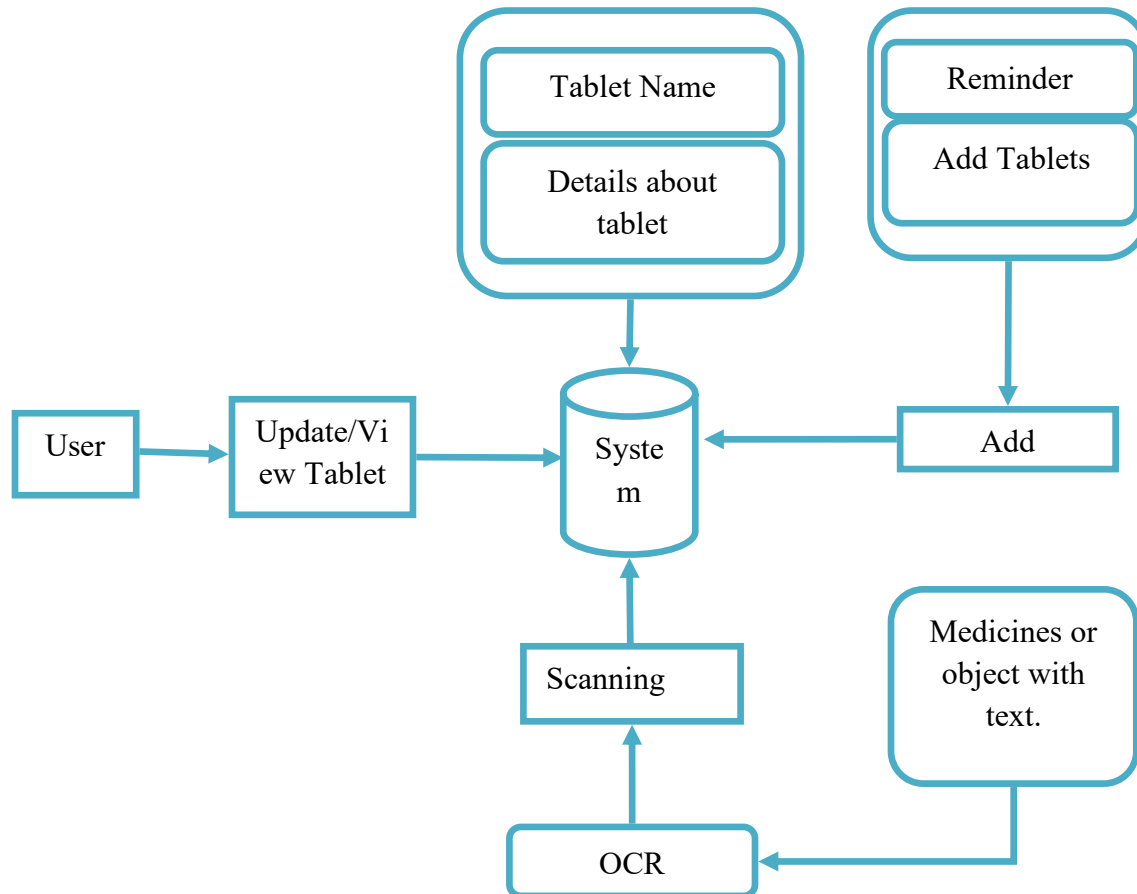


Figure 20: System Architecture

The system has an admin login that has overall control over it. Admin feed the issues or arguments in the system along with desired options. These questions can then be visible to all the employees through android devices. Employees must first create an account into the system for casting their votes. At the end of the voting process the system counts all the votes casted and generates a brief report of the total votes accounted for yes, no and neutral. Eventually, the report is made available to admin, and he may view the maximum votes casted for. Hence the system helps admin to receive appropriate response from employees for the matters in question.

4.14 Conclusion

This chapter also guarantees that all of the system design is recorded and elaborated upon. Each diagram clarifies particular parts and processes, thus making it possible to completely comprehend the functional and structural design of the system.

CHAPTER 5: DEVELOPMENT TOOLS

5.1 About android

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smart phones, electronic book readers, set-top boxes. It contains a linux-based Operating System, middleware and key mobile applications. It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

5.2 What is android?

Android is a software package and linux based operating system for mobile devices such as tablet computers and smart phones. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. The goal of android project is to create a successful real-world product that improves the mobile experience for end users. There are many code names of android such as Lollipop, Kitkat, Jellybean, Ice cream Sandwich, Froyo, Ecliar, Donut etc which is covered in next page.

5.3 About Open Handset Alliance (OHA)

It's a consortium of 84 companies such as Google, Samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc. It was established on 5th November 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Platform.

5.4 Features of Android

- 1) It is open source. Anyone can customize the Android Platform.
- 3) There are a lot of mobile applications that can be chosen by the consumer.
- 4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services (SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

5.5 Categories of Android applications

- Entertainment
- Tools
- Communication

- Productivity
- Personalization
- Music and Audio
- Social
- Media and Video
- Travel and Local etc.

5.6 History of Android

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October 2003.

2) On 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin, Rich Miner, Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS. In 2008, HTC launched the first android mobile.

5.7 Android Architecture

1. Linux kernel: It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

2. Native libraries (middleware): On the top of linux kernel, there are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc. The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3. Android Runtime: In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4. Application Framework: On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5. Applications: On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernel.

5.8 Android Core Building Blocks

An android component is simply a piece of code that has a well-defined life cycle e.g. Activity, Receiver, Service etc.

The core building blocks, or fundamental components of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity:

An activity is a class that represents a single screen. It is like a Frame in AWT.

View:

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent:

Intent is used to invoke components. It is mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.
- For example, you may write the following code to view the webpage.

Example:

```
Intent intent=new Intent(Intent.ACTION_VIEW);  
  
intent.setData(Uri.parse("http://www.gurunanak.com"));  
  
startActivity(intent);
```

Service:

Service is a background process that can run for a long time. There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider:

Content Providers are used to share data between the applications.

Fragment:

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml:

It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD):

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

APK File:

An apk file is created by the framework automatically. If you want to run the android application on the mobile, transfer and install it.

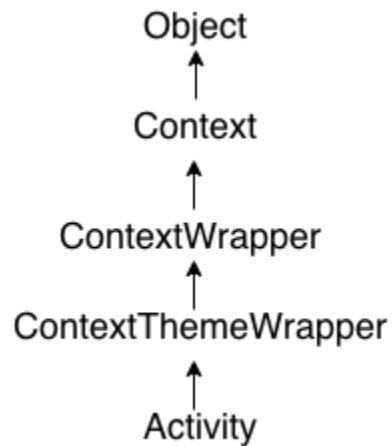
Resources:

It contains resource files including activity_main, strings, styles etc.

Manifest file:

It contains information about package including components such as activities, services, content providers etc.

5.9 Android Activity Lifecycle



Android Activity Lifecycle is controlled by 7 methods of `android.app.Activity` class. The android Activity is the subclass of `ContextThemeWrapper` class. An activity is the single screen in android. It is like window or frame of Java. By the help of activity, you can place all your UI components or widgets in a single screen. The 7-lifecycle method of Activity describes how activity will behave at different states.

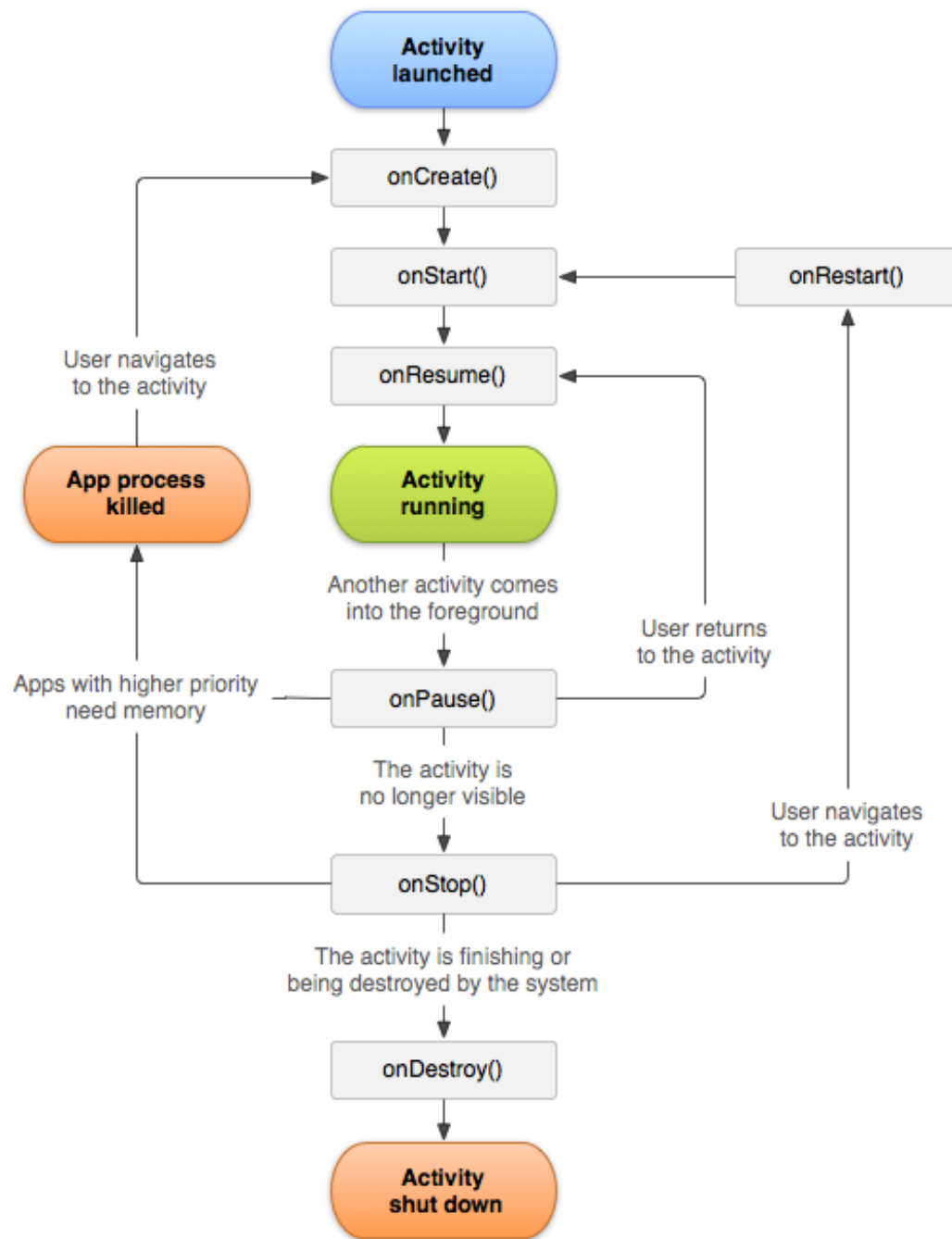


Figure 21: Activity Lifecycle

This picture depicts the different stages that an activity might go through during its life cycle. To be viewed and interacted with, an activity has to enter into and run through the use of `onCreate()`, `onStart()` and `onResume()` callbacks (Activity running). In case the user navigates away, the activity goes to `onPause()` and if it is not visible anymore, it gets to the `onStop()` state. The action can be resumed with `onRestart()`, `onStart()` and `onResume()` functions. In case there is a need for memory reclamation or there is a deliberate cessation of the activity, the system moves into the `onDestroy()` state in which the activity is obliterated. With such endurance, promptness, optimal resource allocation, correct performance of system limitations and user movements are guaranteed.

CHAPTER 6: SNAPSHOTS

6.1 SNAPSHOTS

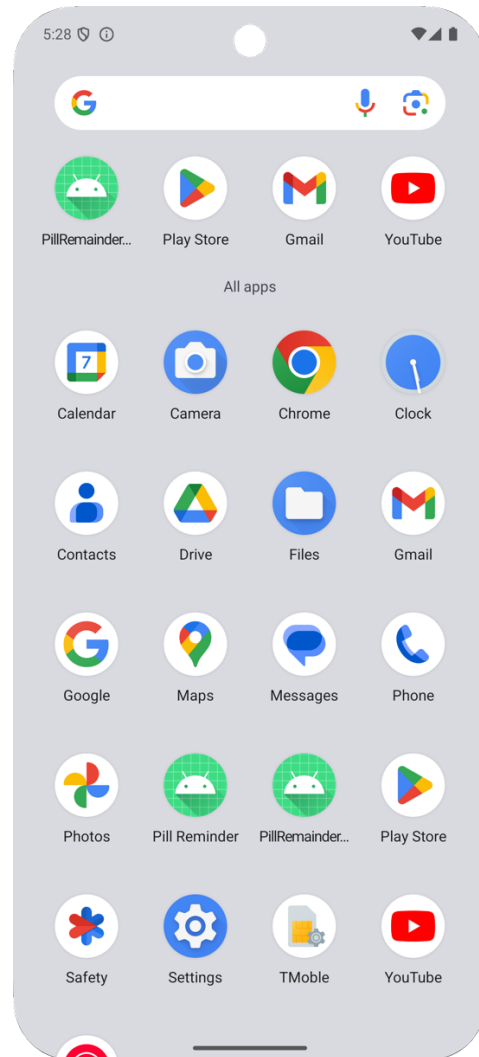


Figure 22: Home Page

User gets to View the app in their home page and with the downloaded apps

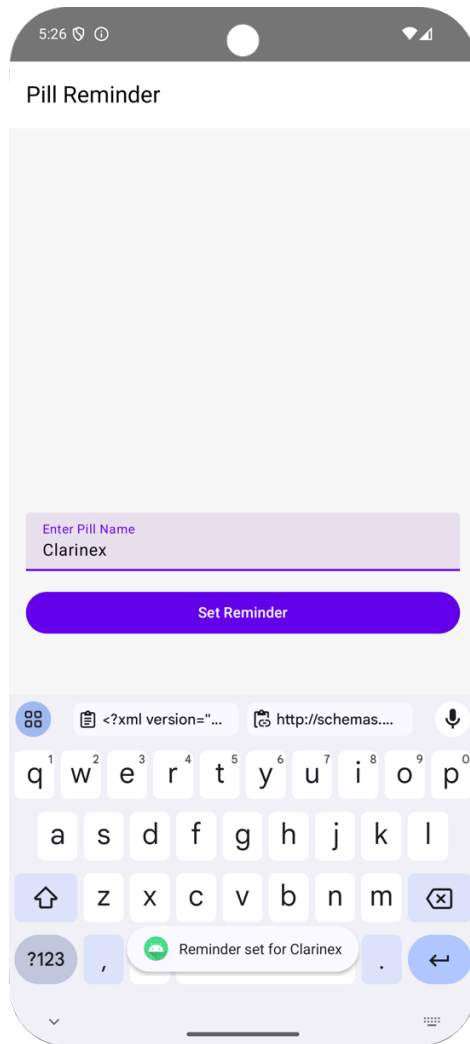


Figure 23: Set Reminder Page

User can see the remainder's set by them.

CHAPTER 7: SOFTWARE TESTING

7.1 GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

7.3 Types of Tests

7.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.3.2 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

7.3.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

7.3.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

7.3.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updating process

7.2.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identify the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 8: APPLICATION

8.1 APPLICATIONS

- This system is used to help the user to remind the tablets and its details while OCR for make the work easy.
- There are many similar applications in the market having a huge amount of success.

8.2 FUTURE ENHANCEMENT:

In future a low-cost, useful model for an automatic pill reminder and disperser box has been designed using simple electronics applications. For easy detection and alert, a buzzer and a LCD display has been attached so that the person in concern takes his pill in time in the right quantity without personalized supervision. The device also records the time and date of taking pills as a useful database for future medical consultation. Family members are alerted, if pills are not taken on time. This easy-to-use device can be a convenient option for households where family members have work-hour compulsions or are compelled to keep a mistress for the member with medical complications.

CHAPTER 9: CONCLUSION & REFERENCE

9.1 CONCLUSION

Elderly people play an important role in the society. They are part of the priority group of healthcare. Therefore, it is necessary to create new devices using the emerging technology to improve their lives quality. Based on open-source solutions, a new alternative to remind medicine dosages is proposed. Arduino Wemos as main controller works totally right and give many other opportunities to develop. The objective of creating a device that allows the organization of several medication schedules, automatic opening system and an effective notification system, is reached. This system reduces family member's responsibility towards ensuring the correct and timely consumption of medicines by alerting the user to get the pill at the time.

9.2 REFERENCE:

<http://ieeexplore.ieee.org/document/7560923/>

- [1] Juan Marcelo Parra 1, Wilson Valdez, Andrea Guevara, Priscila Cedillo, Jose Ortiz Segarra, "INTELLIGENT PILLBOX: AUTOMATIC AND PROGRAMMABLE ASSISTIVE TECHNOLOGY DEVICE", Proceedings of the IASTED International Conference Biomedical Engineering (BioMed 2017).
- [2] "An Electronic Pillbox for Continuous Monitoring of Medication Adherence", Tamara. L. Hayes, Member, IEEE, John M. Hunt, Member IEEE, 28th IEEE EMBS Annual International Conference New York City, USA, Aug 30-Sept 3, 2006..
- [3] Huai-Kuei Wu1, Member, IEEE, Chi-Ming Wong, PangHsing Liu, Sheng-Po Peng, Xun-Cong Wang, Chih-Hi Lin and Kuan-Hui Tu "A Smart Pill Box with Remind and Consumption Confirmation Functions", 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE).
- [4] MacLaughlin, Eric J., et al. "Assessing medication adherence in the elderly." *Drugs & aging* 22.3 (2005): 231-255.
- [5] G. Eason, B. Noble, and I.N. Sneddon Lewis, Kermit E., and Arthur S. Roberts Jr. "Automatic pill dispenser and method of administering medical pills." U.S. Patent No. 4,573,606. 4 Mar. 1986.
- [6] Shaw, Thomas J. "Automatic pill dispensing apparatus." U.S. Patent No. 5,609,268. 11 Mar. 1997.
- [7] MacLaughlin, Eric J., et al. "Assessing medication adherence in the elderly." *Drugs & aging* 22.3 (2005): 231-255. 2005): 231-255.
- [8] Fang, Kerry Y., Anthony J. Maeder, and Heidi Bjering. "Current trends in electronic medication reminders for self care." *The Promise of New Technologies in an Age of New Health Challenges: Selected Papers from 5th Global Telehealth Conference 2016*, Auckland, New Zealand, 1-2 November 2016. 2016.
- [9] Billingsley, Luanne, and Ann Carruth. "Use of technology to promote effective medication adherence." *The Journal of Continuing Education in Nursing* 46.8 (2015): 340-342.
- [10] Patel, Samir, et al. "Mobilizing your medications: an automated medication reminder application for mobile phones and hypertension medication adherence in a high-risk urban population." (2013): 630-639

APPENDIX

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.saialekhya.pillreminder">

    <application

        android:allowBackup="true"

        android:label="@string/app_name"

        android:roundIcon="@mipmap/ic_launcher_round"

        android:supportsRtl="true"

        android:theme="@style/Theme.PillReminder">

        <activity android:name=".MainActivity"

            android:exported="true">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

    </application>

</manifest>
```

```
</intent-filter>
```

```
</activity>
```

```
<!-- Add android:exported to other components with intent filters -->
```

```
<receiver android:name=".ReminderReceiver"
```

```
    android:exported="true" />
```

```
</application>
```

```
</manifest>
```

MainActivity.kt

```
package com.saialekhya.pillreminder
```

```
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
```

```
import android.widget.Button
```

```
import android.widget.EditText
```

```
import android.widget.TimePicker
```

```
import android.widget.Toast
```

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)


        // Link UI components

        val pillNameEditText: EditText = findViewById(R.id.pillNameEditText)

        val timePicker: TimePicker = findViewById(R.id.timePicker)

        val setReminderButton: Button = findViewById(R.id.setReminderButton)


        // Set listener on the "Set Reminder" button

        setReminderButton.setOnClickListener {

            val pillName = pillNameEditText.text.toString()

            val hour = timePicker.hour

            val minute = timePicker.minute
```

```
        // Display a message indicating the reminder time

        Toast.makeText(this, "Reminder set for $pillName at $hour:$minute",
Toast.LENGTH_SHORT).show()

    }

}

}
```

ExampleInstrumentedTest

```
package com.saialekhya.pillremainder

import androidx.test.platform.app.InstrumentationRegistry

import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test

import org.junit.runner.RunWith

import org.junit.Assert.*
```

```
/**
```

```
 * Instrumented test, which will execute on an Android device.
```

```
 *
```

```
 * See [testing documentation](http://d.android.com/tools/testing).
```

```
 */
```

```
@RunWith(AndroidJUnit4::class)
```

```
class ExampleInstrumentedTest {
```

```
    @Test
```

```
    fun useAppContext() {
```

```
        // Context of the app under test.
```

```
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
```

```
        assertEquals("com.saialekhya.pillremainder", appContext.packageName)
```

```
    }
```

```
}
```

ExampleUnitTest

```
package com.saialekhya.pillremainder
```



```
import org.junit.Test
```

```
import org.junit.Assert.*
```

```
/**
```

```
 * Example local unit test, which will execute on the development machine (host).
```

```
 *
```

```
 * See [testing documentation](http://d.android.com/tools/testing).
```

```
 */
```

```
class ExampleUnitTest {
```

```
    @Test
```

```
    fun addition_isCorrect() {
```

```
        assertEquals(4, 2 + 2)
```

```
    }
```

```
}
```

```
activity_main.xml
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:orientation="vertical"

    android:padding="16dp">
```

```
<EditText

    android:id="@+id/pillNameEditText"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:hint="Enter Pill Name" />
```

```
<TimePicker

    android:id="@+id/timePicker"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:timePickerMode="spinner" />
```

```
<Button  
  
    android:id="@+id/setReminderButton"  
  
    android:layout_width="wrap_content"  
  
    android:layout_height="wrap_content"  
  
    android:text="Set Reminder" />
```

```
</LinearLayout>
```

colors.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
  
    android:layout_width="match_parent"  
  
    android:layout_height="match_parent"  
  
    android:orientation="vertical"  
  
    android:padding="16dp">
```

```
<EditText  
  
    android:id="@+id/pillNameEditText"
```

```
android:layout_width="match_parent"

android:layout_height="wrap_content"

android:hint="Enter Pill Name" />
```

```
<TimePicker
```

```
    android:id="@+id/timePicker"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:timePickerMode="spinner" />
```

```
<Button
```

```
    android:id="@+id/setReminderButton"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Set Reminder" />
```

```
</LinearLayout>
```

strings.xml

```
<resources>

    <string name="app_name">Pill Reminder</string>

</resources>
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <!-- Base application theme -->

    <style                                name="Theme.PillReminder"
parent="Theme.MaterialComponents.DayNight">

        <item name="colorPrimary">@color/purple_500</item>

        <item name="colorOnPrimary">@color/white</item>

        <item name="colorSecondary">@color/teal_200</item>

        <item name="colorOnSecondary">@color/black</item>

        <item name="android:colorBackground">@color/white</item>

        <item name="colorSurface">@color/white</item>

        <item name="colorOnBackground">@color/black</item>

        <item name="colorOnSurface">@color/black</item>

        <item                                name="android:statusBarColor"
tools:targetApi="l">@color/purple_700</item>
```

</style>

</resources>

build.gradle.kts (project: PillReminder)

```
buildscript {
```

```
    repositories {
```

```
        google()
```

```
        mavenCentral()
```

```
    }
```

```
    dependencies {
```

```
        classpath("com.android.tools.build:gradle:8.7.3")
```

```
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.10")
```

```
    }
```

```
}
```

build.gradle.kts (module: app)

```
plugins {
```

```
    id("com.android.application")
```

```
id("org.jetbrains.kotlin.android")

}

android {

    namespace = "com.saialekhya.pillreminder"

    compileSdk = 34

    defaultConfig {

        applicationId = "com.saialekhya.pillreminder"

        minSdk = 21

        targetSdk = 34

        versionCode = 1

        versionName = "1.0"

    }

    buildFeatures {

        compose = false

    }

}
```

```
}

kotlinOptions {

    jvmTarget = "1.8"

}

}

dependencies {

    implementation("androidx.core:core-ktx:1.12.0")

    implementation("androidx.appcompat:appcompat:1.7.0")

    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.1")

    implementation("com.google.android.material:material:1.9.0")


    // Test dependencies

    testImplementation("junit:junit:4.13.2")

    androidTestImplementation("androidx.test.ext:junit:1.1.5")
```



```
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

gradle.properties (project properties)

Project-wide Gradle settings.

IDE (e.g. Android Studio) users:

Gradle settings configured through the IDE **will override**

any settings specified in this file.

For more details on how to configure your build environment visit

http://www.gradle.org/docs/current/userguide/build_environment.html

Specifies the JVM arguments used for the daemon process.

The setting is particularly useful for tweaking memory settings.

```
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
```

When configured, Gradle will run in incubating parallel mode.

This option should only be used with decoupled projects. For more details, visit

<https://developer.android.com/r/tools/gradle-multi-project-decoupled-projects>

```
# org.gradle.parallel=true
```

AndroidX package structure to make it clearer which packages are bundled with the

Android operating system, and which are packaged with your app's APK

<https://developer.android.com/topic/libraries/support-library/androidx-rn>

android.useAndroidX=true

Kotlin code style for this project: "official" or "obsolete":

kotlin.code.style=official

Enables namespacing of each library's R class so that its R class includes only the

resources declared in the library itself and none from the library's dependencies,

thereby reducing the size of the R class for that library

android.nonTransitiveRClass=true