**1.INTRODUCTION**

**1.1 PROJECT OVERVIEW**

The aim of the project is to develop a medicine reminder system that helps senior citizens take the correct medication at the right time and also help doctors or caretakers monitor their patients around the clock. The patient's medical details are stored in IBM cloudant DB and when it is time for patients to take up their medication, the cloud will notify the user’s device.

**1.2 PURPOSE**

Today, most people can expect to live into their seventies and beyond. According to the United Nations, the number of people aged 60 years or older is projected to grow by 56 percent worldwide by 2030. Many of these people will have physical impairments (loss of hearing, eyesight) and yet be self reliant. These senior citizens need assistance while taking their medications. Wealthy people hire personal caretakers for reminding them to take the correct medication at the right time. But most people cannot afford to hire caretakers.

To help these people, a medicine reminder system is developed that reminds self reliant seniors to take the correct medication at the right time. This medicine reminder system will also help doctors monitor patients around the clock. The application is built such that the user can store their medical details in IBM Cloudant DB. The user’s device will receive the medicine name via cloud and notify the user at the right time using voice commands.

**2. LITRATURE SURVEY**

* 1. **EXISTING PROBLEM**

Karantis360 is an automated personal monitoring and alerting system, using intelligent, wirelessly linked to a discreet, mobile device which sends reports and alerts to carers and family members.

It has been developed specifically to promote independent, home living for the elderly, infirm, those living with Alzheimer’s or dementia and to enable early release from hospital for other clients where monitoring is required.

Using a non-intrusive system of sensors, machine learning and automatic data communication, Karantis360 flags exceptions to routines and habits, such as whether your client has got out of bed, is sitting in a chair, has boiled the kettle and so on.

By analyzing activity data and comparing it to expected patterns, the system identifies when your client’s activity is out-of-the-ordinary and sends you an immediate alert, so you can respond quickly and effectively to any potential emergency

* 1. **PROBLEM STATEMENT DEFINITION**

Most people can expect to live into their seventies and beyond. According to the UN, the number of people aged 60 years or older is projected to grow by 56 percent worldwide by 2030. Most of these senior citizens may not have family support or cannot afford to hire personal caretakers. Hence an application is developed for these self-reliant seniors to help them take their proper medications at the right time. The application also helps doctors/caretakers to monitor their patients. Since there is an increase in the number of potential users for the application and there is a necessity for self-reliant seniors, the medicine reminder system is developed to cater to such needs.Figure 2.1 shows customer problem statement.

CUSTOMER PROBLEM STATEMENT:



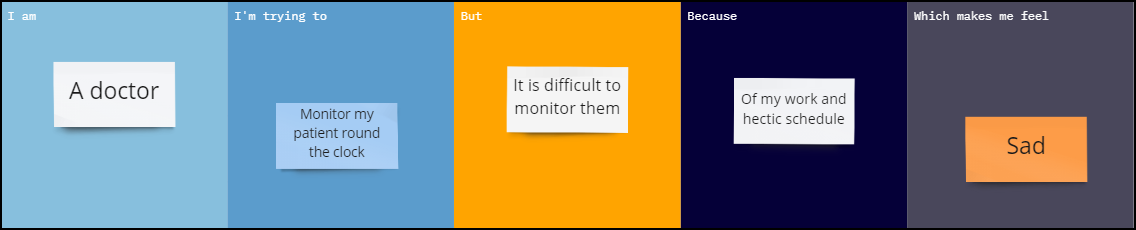


Figure 2.1 CUSTOMER PROBLEM STATEMENT.

Table 2.1 shows problem statement.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | An elderly person | Take my medication on time | I sometimes forget to take my medications at the right time | Of my age and bad memory | Unhappy and frustrated |
| PS-2 | A doctor | Monitor my patient round the clock | It is difficult to monitor them | Of my work and hectic schedule | Sad |

**3. IDEATION AND PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS**

An empathy map is a collaborative visualization used to express clearly what one knows about a particular type of user. It externalizes knowledge about users in order to create a shared understanding of user needs, and aid in decision making. Empathy maps are split into 4 quadrants (Says, Thinks, Does, and Feels), with the user in the middle. Empathy maps provide a glance into who a user is as a whole.The Says quadrant contains what the user says or what he needs. The Thinks quadrant captures what the user is thinking throughout the experience. The Does quadrant encloses the actions the user takes. The Feels quadrant is the user’s emotional state.

Figure 3.1 shows the says, thinks, does and feels about the project

****

Figure 3.1 EMPATHY MAP

**3.2 IDEATION AND BRAINSTROMING**

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. Brainstorming is usually conducted by getting a group of people together to come up with either general new ideas or ideas for solving a specific problem or dealing with a specific situation.

A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity. Both brainstorming and ideation are processes invented to create new valuable ideas, perspectives, concepts and insights, and both are methods for envisioning new frameworks and systemic problem solving.

Figure 3.2, 3.3, 3.4 shows the brainstorm and idea presentation done during planning phase.

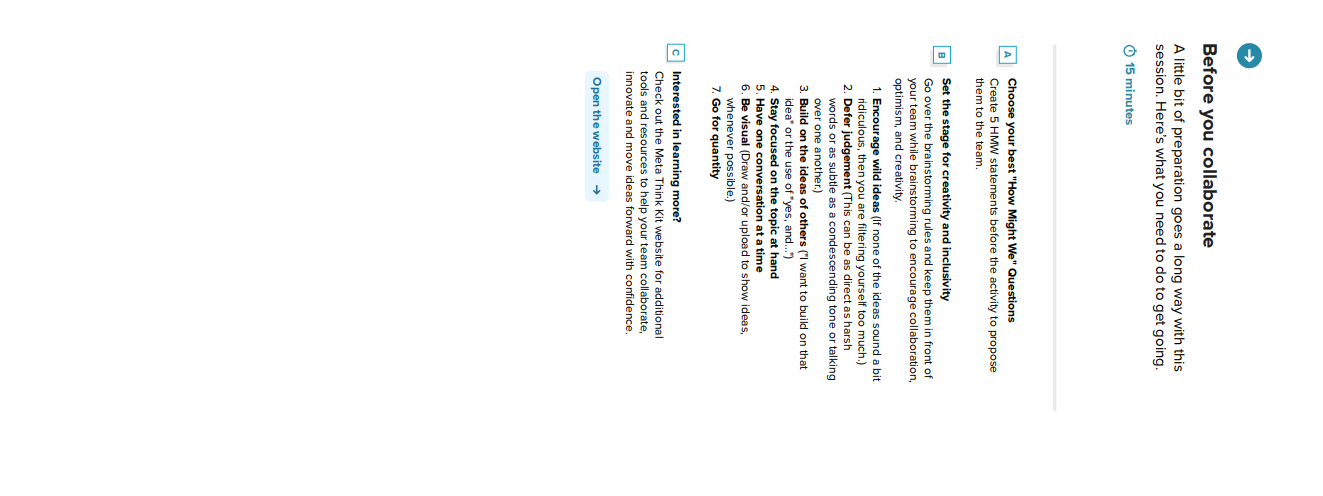
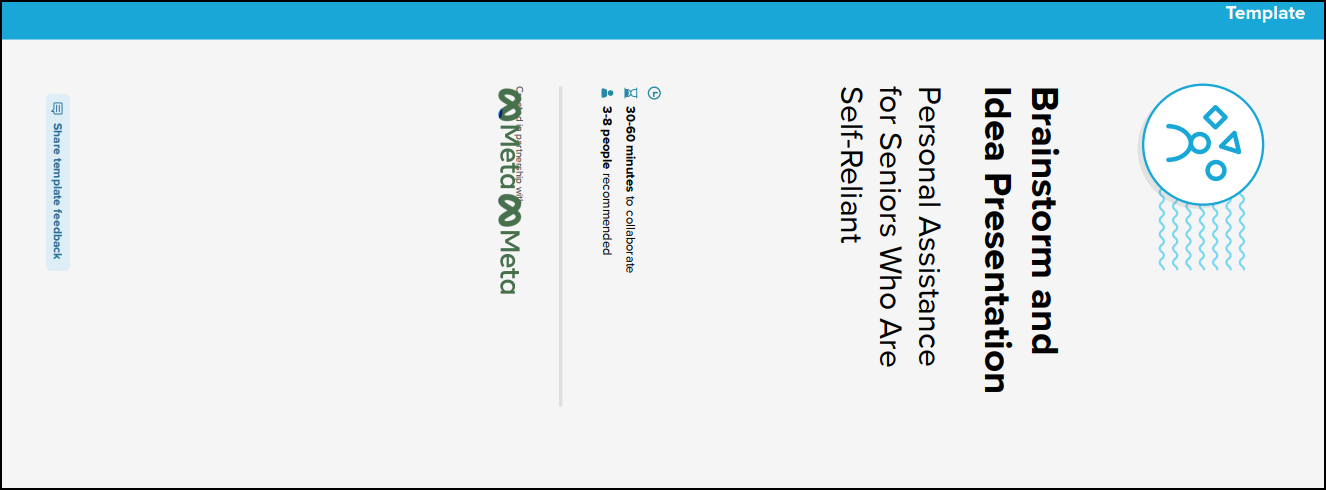
****

Figure 3.2 BRAINSTORMING AND IDEATION PHASE

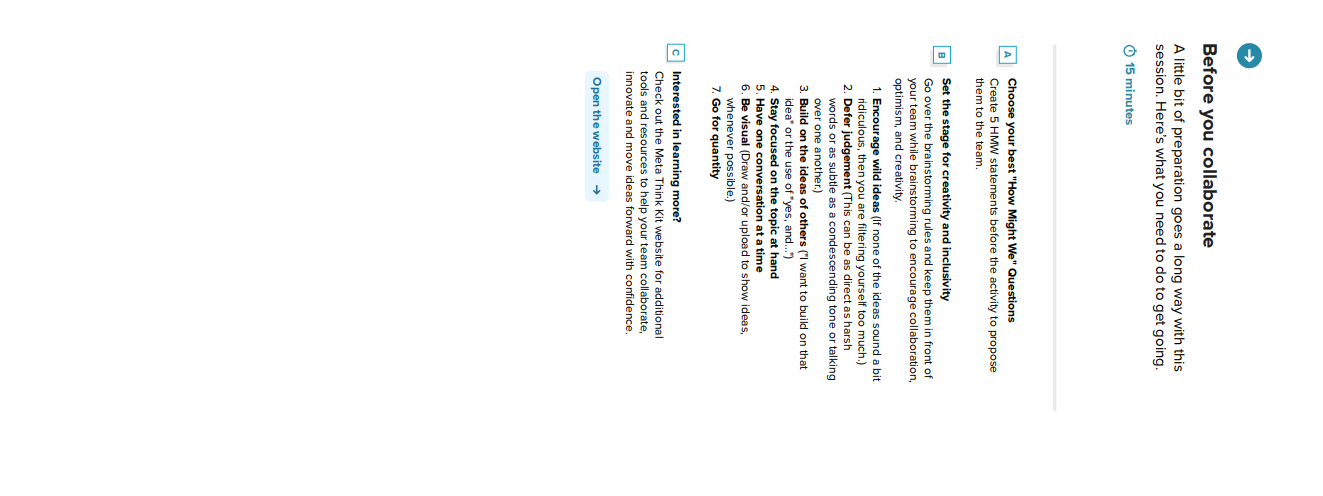
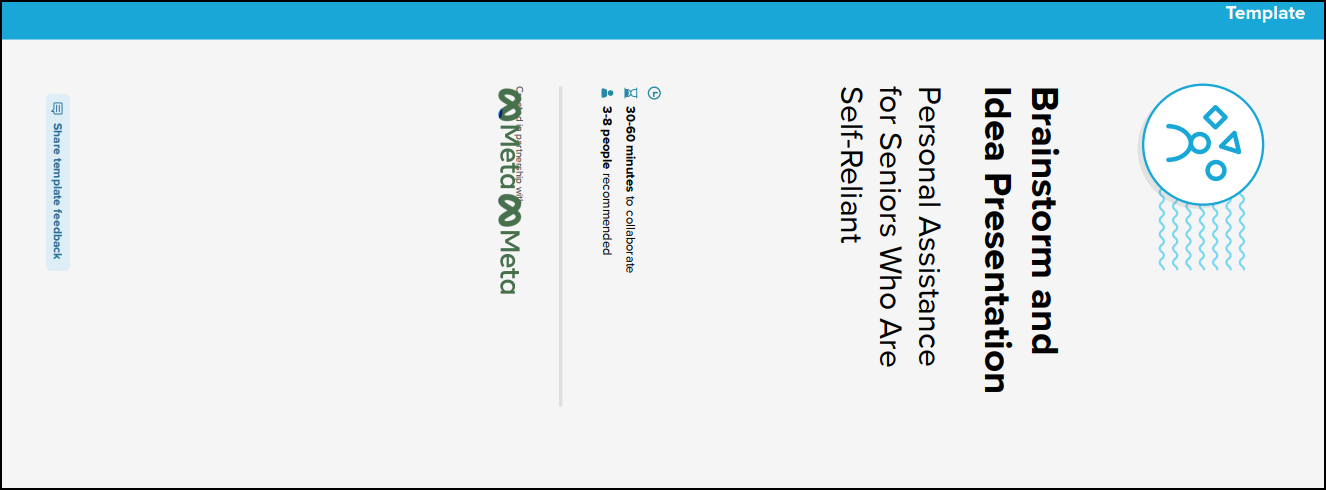
****

Figure 3.2 BRAINSTORMING AND IDEATION PHASE

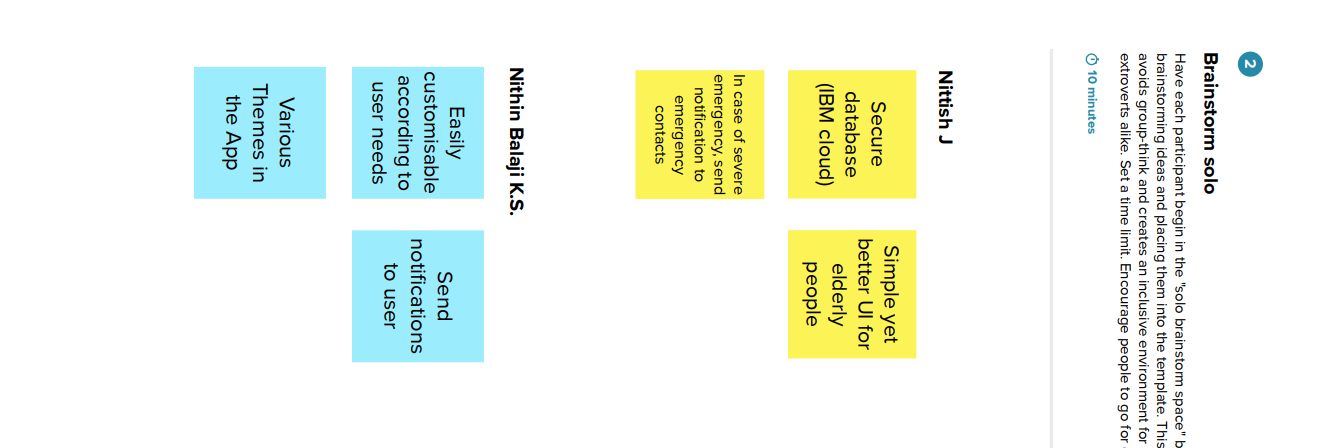
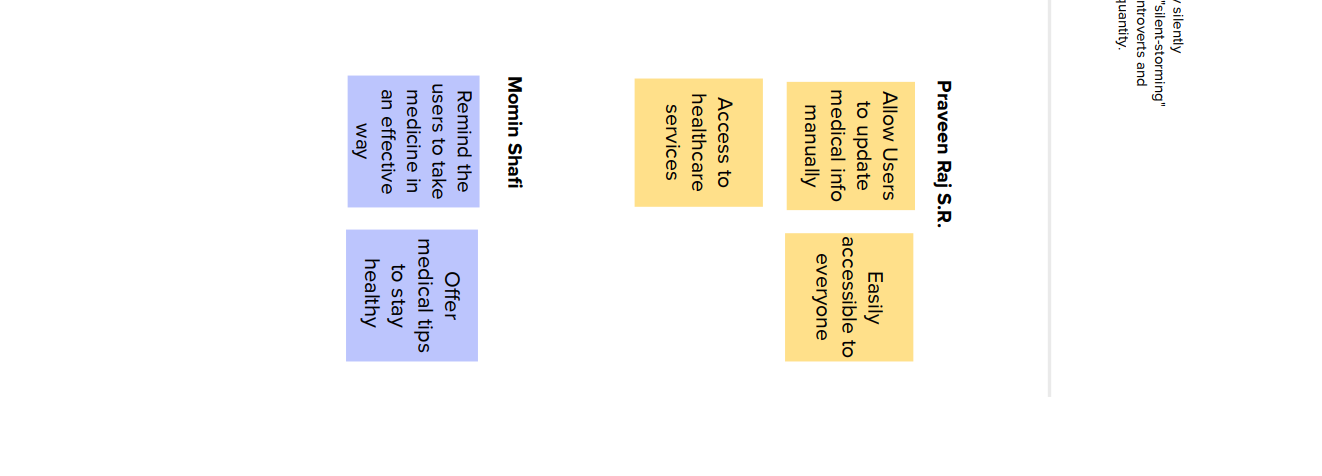
****

Figure 3.3 BRAINSTORMING AND IDEATION PHASE

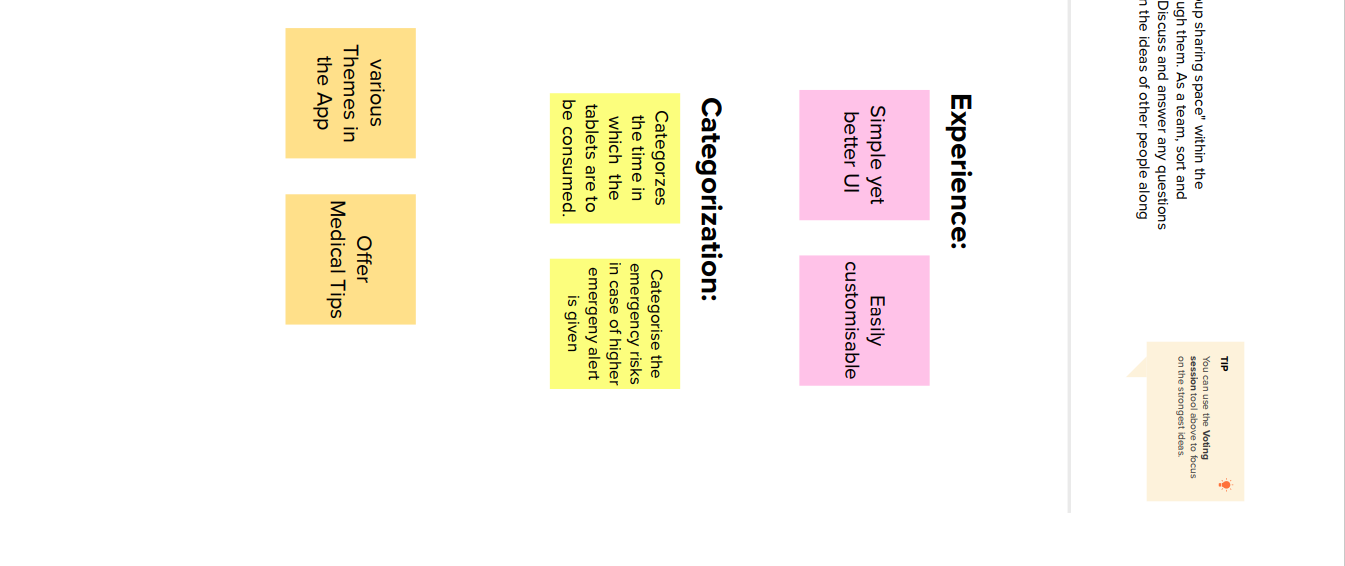
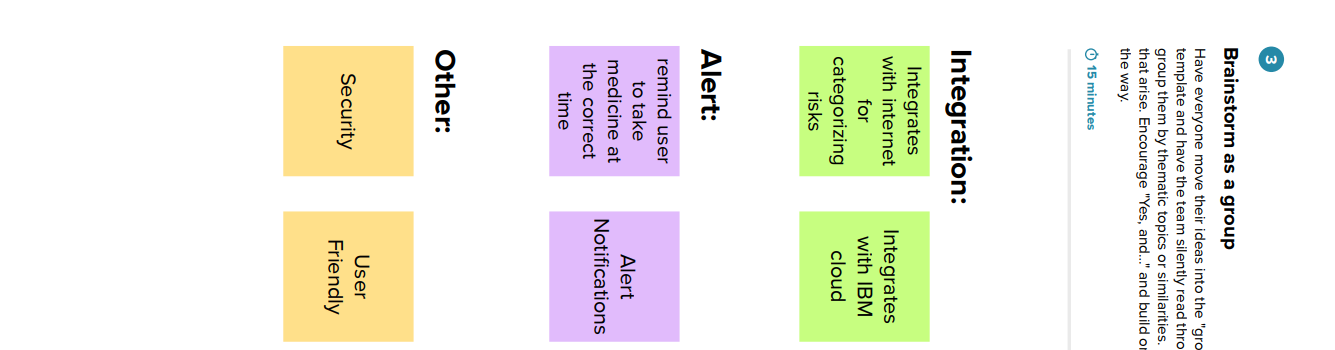
****

Figure 3.4 BRAINSTORMING AND IDEATION PHASE

**3.3 PROPOSED SOLUTION:**

Table 3.1 PROPOSED SOLUTION PARAMETER AND DESCRIPTION

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | Sometimes elderly people forget to take their medicine on time. They also forget what medicine to take at a particular time and the stock gets over. |
| 2. | Idea / Solution description | To remind the users to take medicines on time and alert the user when stocks nearing the end based on an application. |
| 3. | Novelty / Uniqueness | Integration of app with the Internet to get health and medical tips for users on a day to day basis. Emergency SOS message will be sent to all emergency contact listed by the user in case the user presses the SOS icon in case of an emergency. |
| 4. | Business Model (Revenue Model) | The app is designed to help senior citizens. Hence there would be no installation or subscription charges. It is intended to be free for all. Revenue can be generated with advertisements. Since the target audience is huge, advertisement revenue is also expected to be great. |
| 5. | Scalability of the Solution | Most people can expect to live into their seventies and beyond. According to the UN, the number of people aged 60 years or older is projected to grow by 56 percent worldwide by 2030. This means that in the near future, the target audience is going to increase significantly providing a great scope for the future. |

**3.4 PROBLEM SOLUTION FIT:**

The Problem solution fit simply means that one have found a problem with the customer and that the solution one have realised for it actually solves the customers problem. The problem solution fit is an important step towards the Product-Market Fit. The structure of problem solution fit is given below.

**Customer state fit**: To make sure one understand the target group, their limitations and their currently available solutions, against which one is going to compete.

**Problem-Behavior fit**: To help one to identify the most urgent and frequent problems, understand the real reasons behind them and see which behavior supports it. **Communication-Channel fit**: To help one to sharpen the communication with strong triggers, emotional messaging and reaching customers via the right channels.

**Solution guess**: Translate all the validated data one have gathered into a solution that fits the customer state and his/her limitations, solves a real problem and taps into the common behavior of the target group.

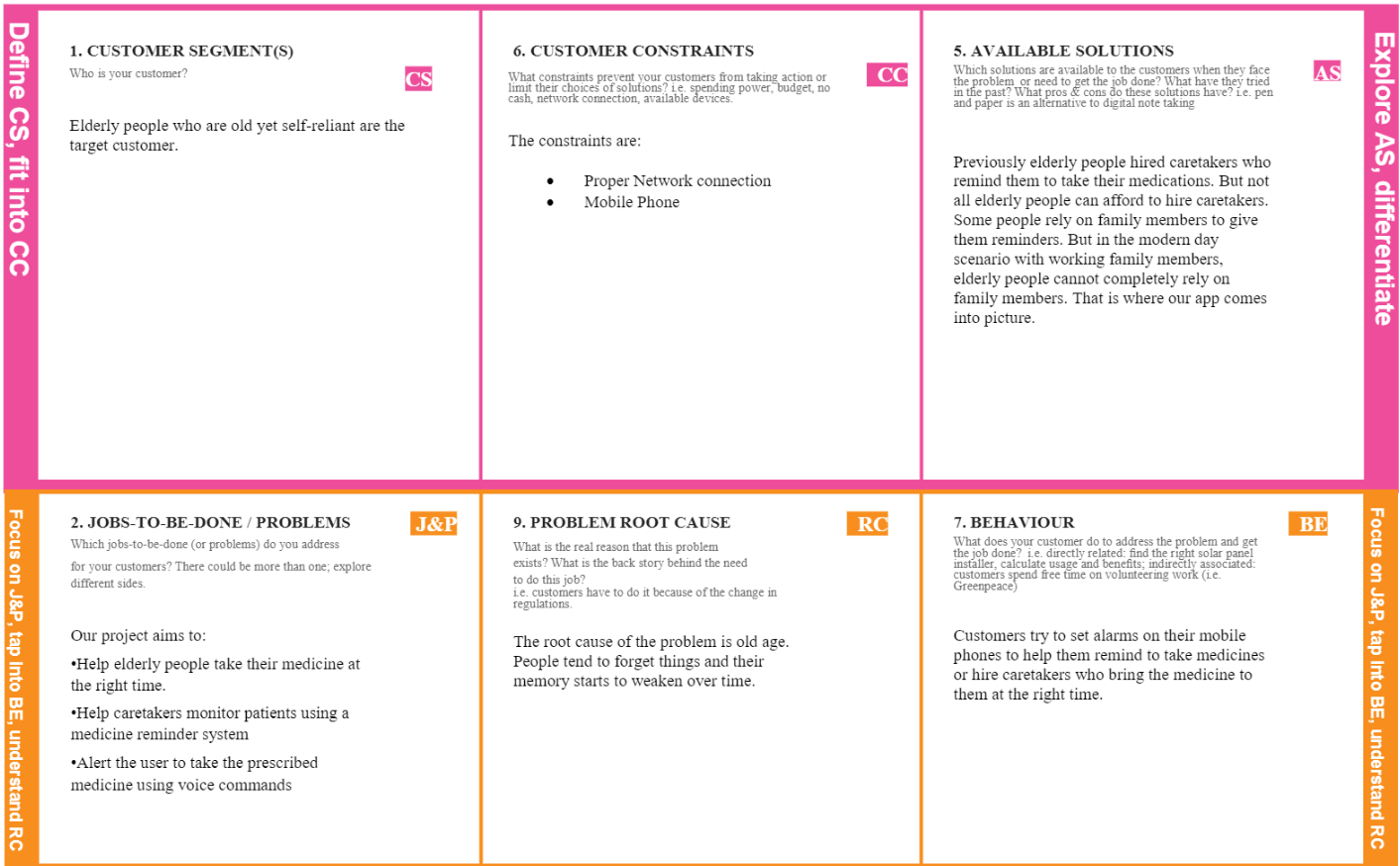
Figure 3.5 and 3.6 shows the problem solution fit****

Figure 3.5 PROBLEM SOLUTION FIT

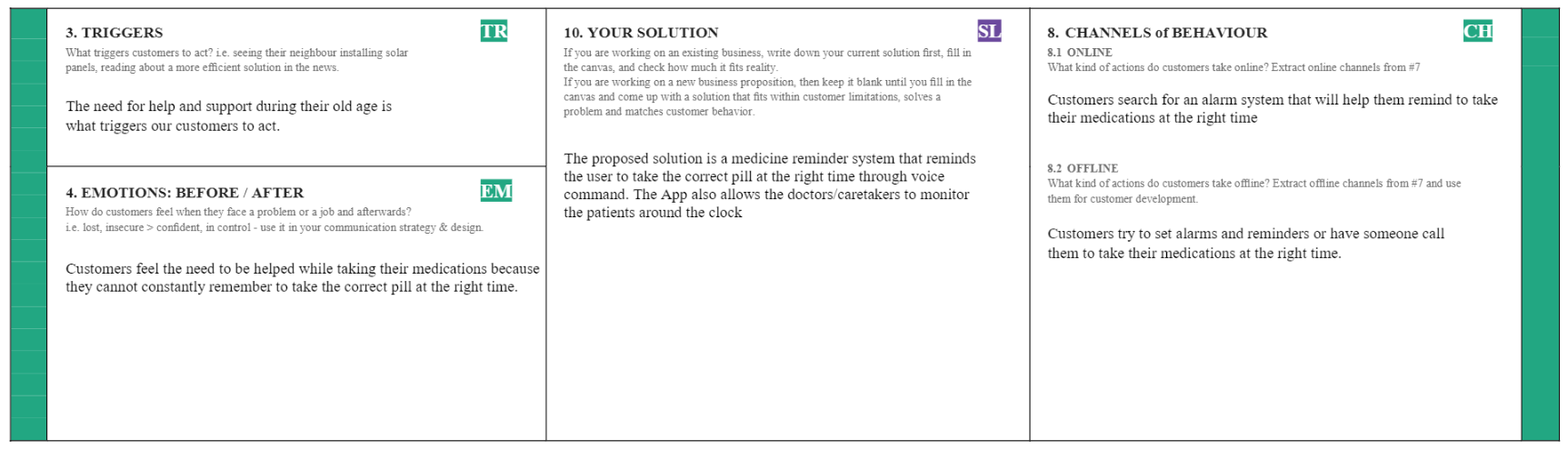
****

Figure 3.6 PROBLEM SOLUTION FIT

**4.REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENTS**

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements

Table 4.1 functional requirements of the proposed solution

|  |  |  |
| --- | --- | --- |
| **FR No**. | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form  Registration through Gmail |
| FR-2 | User Confirmation | Confirmation via Email  Confirmation via OTP |
| FR-3 | Access Cloud Services | Access the cloud services  Store the details in the database  Retrieve the stored information |

**4.2 NON - FUNCTIONAL REQUIREMENTS**

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements

Table 4.2 non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| NFR No. | Non-Functional Requirement | Description |
| NFR-1 | Usability | App must be easy to use by all people. The UI must be simple yet engaging. |
| NFR-2 | Security | The database must be secure since it contains medical information of all the users. It must be strongly encrypted and user login must be secure. |
| NFR-3 | Reliability | The services must be reliable. It will be highly reliable since it uses IBM cloud service. |
| NFR-4 | Performance | The performance standards must be set high when compared to other similar services. |
| NFR-5 | Availability | The service must be available on all platforms like IOS, android etc. |
| NFR-6 | Scalability | Using cloud services would make the service more scalable that using traditional database. |

**5.PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAM:**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

There are four main elements of a DFD — external entity, process, data store, and data flow.

**EXTERNAL ENTITY**

An external entity, which are also known as terminators, sources, sinks, or actors, are an outside system or process that sends or receives data to and from the diagrammed system. They’re either the sources or destinations of information, so they’re usually placed on the diagram’s edges. External entity symbols are similar across models except for Unified, which uses a stick-figure drawing instead of a rectangle, circle, or square.

**PROCESS**

Process is a procedure that manipulates the data and its flow by taking incoming data, changing it, and producing an output with it. A process can do this by performing computations and using logic to sort the data, or change its flow of direction. Processes usually start from the top left of the DFD and finish on the bottom right of the diagram.

**DATA STORE**

Data stores hold information for later use, like a file of documents that’s waiting to be processed. Data inputs flow through a process and then through a data store while data outputs flow out of a data store and then through a process**.**

**DATA FLOW**

Data flow is the path the system’s information takes from external entities through processes and data stores. With arrows and succinct labels, the DFD can show the direction of the data flow. Figure 5.1 shows the data flow diagram for the proposed solution

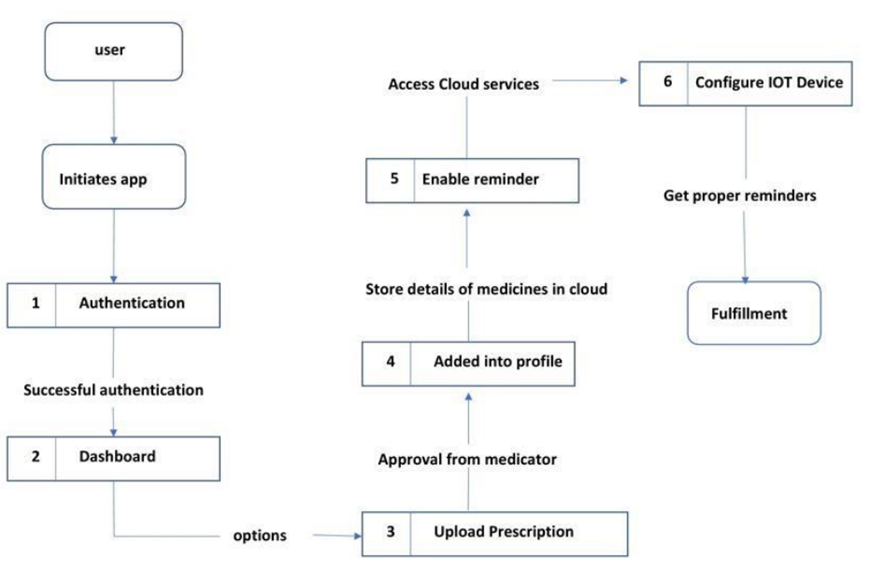
****

Figure 5.1 DATA FLOW DIAGRAM

**5.2 SOLUTION AND TECHNICAL ARCHITECTURE**

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

• Find the best tech solution to solve existing business problems.

• Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.

• Define features, development phases, and solution requirements.

• Provide specifications according to which the solution is defined, managed, and delivered.

Figure 5.2 shows the solution architecture the self reliant and 5.3 shows the technical architecture.

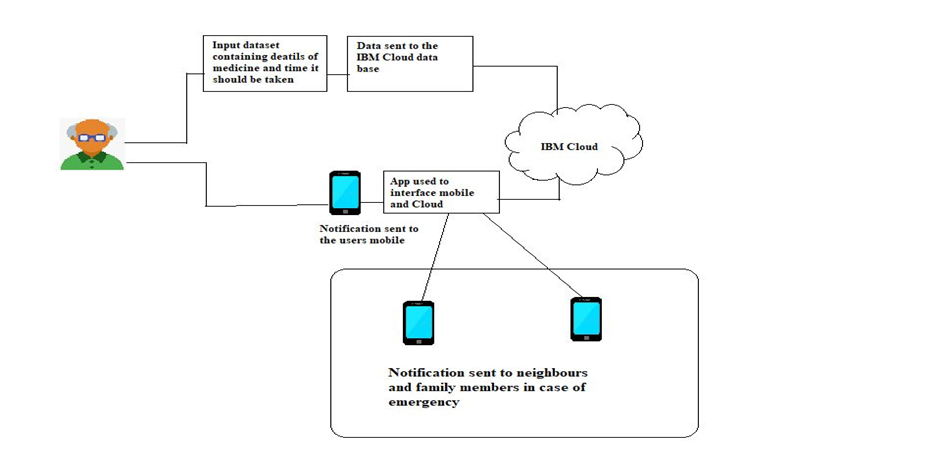
****

Figure 5.2 SOLUTION ARCHITECTURE

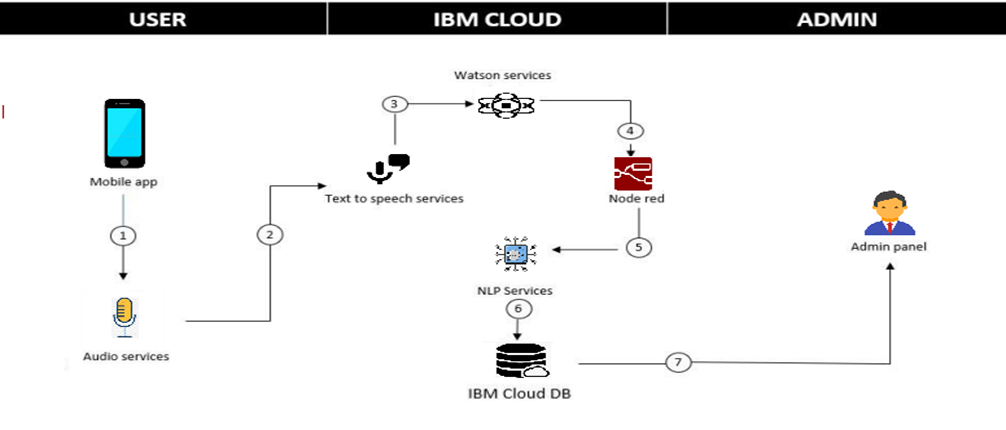
****

Figure 5.3 TECHMOLOGICAL ARCHITECTURE

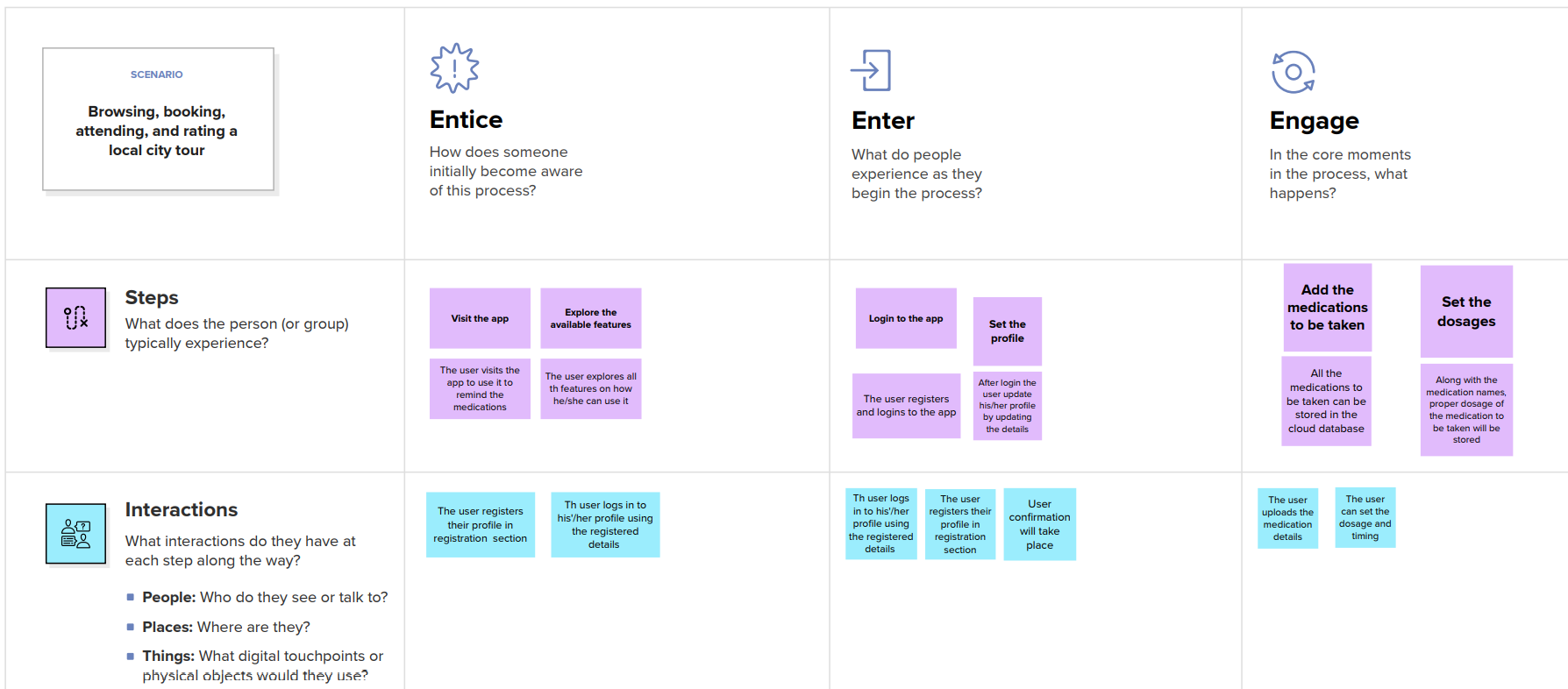
**5.3 USER STORIES**

Table 5.3 shows the user stories

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Type | Functional Requirement | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
| Customer (user app) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can register myself my to use the application | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
|  | Login | USN-3 | As a user, I can register for the application by entering user credentials. | Through successful login, I can access use the application for further needs | High | Sprint-1 |
|  | Dashboard | USN-4 | As a user, I can use consultation feature in the dashboard. | I can get real time consultations through online | Medium | Sprint-2 |
|  |  | USN-5 | As a user, I can use voice configuration feature in the dashboard | I can able to change voice speed and tone | Low | Sprint-2 |
|  |  | USN-6 | As a user, I can store the medication details with proper timing. | I can able to see the history of prescriptions | High | Sprint-1 |
|  |  | USN-7 | As a user, I can use reminder options in the dashboard | I can configure the reminder time to get updates | High | Sprint-1 |
| Administrator | Admin panel | USN-8 | As an app admin , I can use manage users options in the dashboard | I can manage the user reports | High | Sprint-1 |
|  |  | USN-9 | As an app admin , I can use consultation options in the dashboard | I can directly contact the consultant | Medium | Sprint-2 |
|  |  | USN-10 | As an app admin, I can configure new alert option in the dashboard. | can access the device to update to latest version | low | Sprint-2 |

CUSTOMER JOURNEY

Figure 5.4 and 5.5 shows the customer journey

****

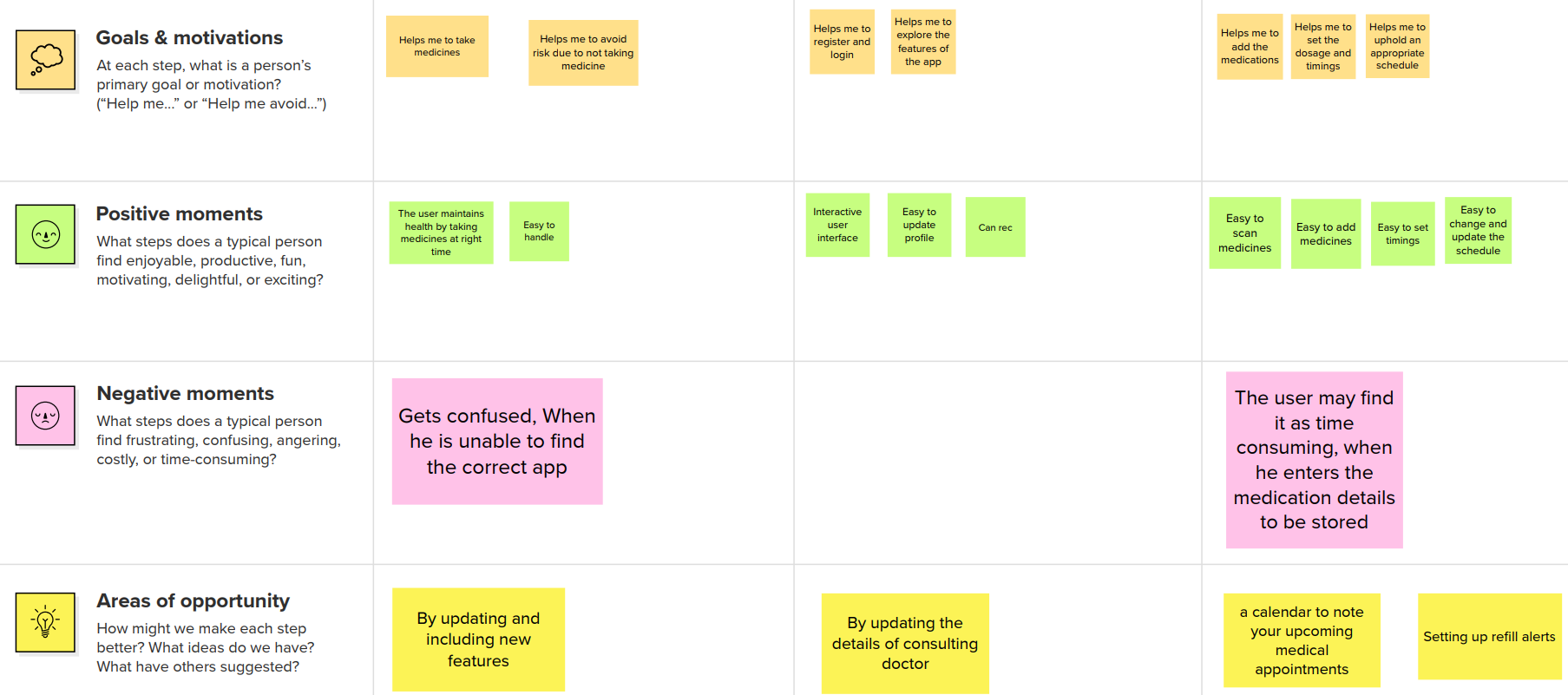
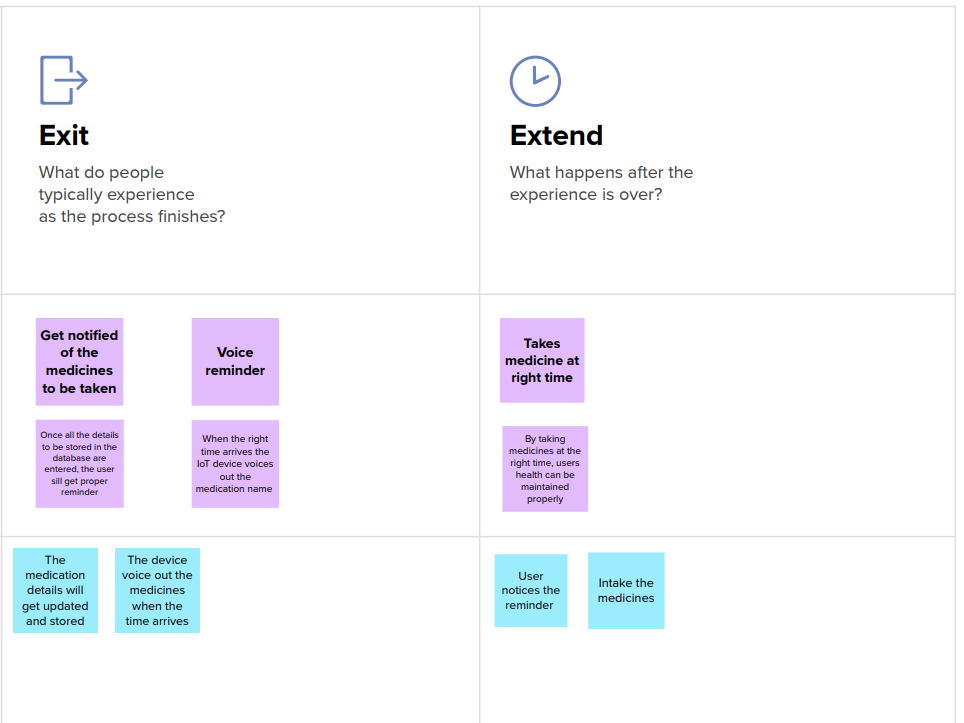
****

Figure 5.4 CUSTOMER JOURNEY

****

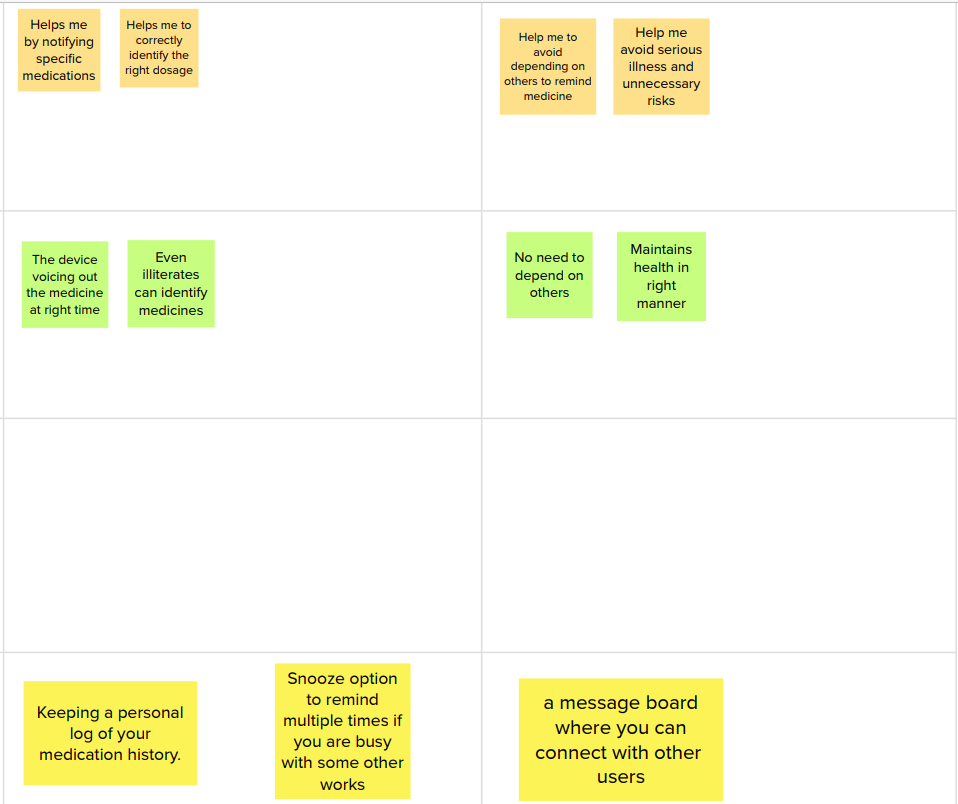
****

Figure 5.5 CUSTOMER JOURNEY

**6.PROJECT PLANNING AND SCHEDULING:**

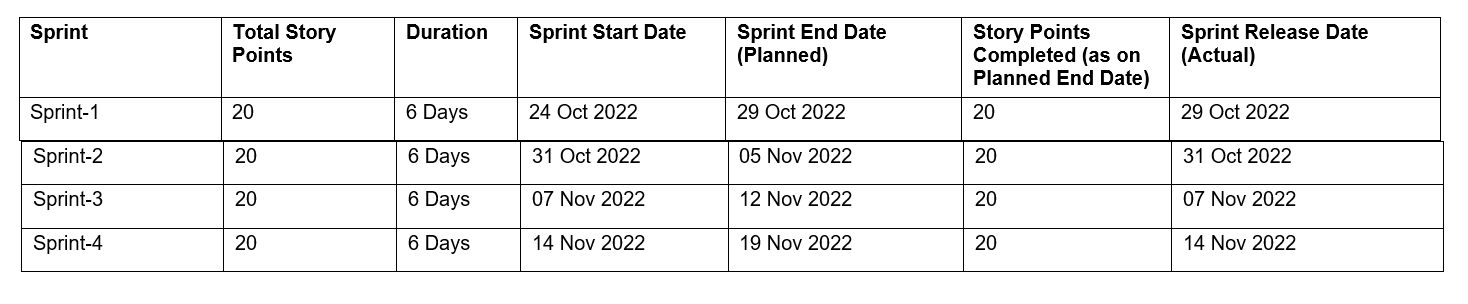
**6.1 SPRINT PLANNING AND ESTIMATION**

Table 6.1 Sprint Schedule and Estimation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** |
| Sprint-1 | IBM Watson IOT platform | USN-1 | Creating devices and board and generating data | 1 | medium |
| Sprint-2 | Storing Data using node-red | USN-2 | Storing the data in IBM Cloudant DB through node-red functions | 2 | High |
| **Sprint** | **Functional**  **Requirement** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** |
| Sprint 3 | Creating frontend app | USN-1 | Creating devices and board and generating data | 1 | medium |
| Sprint-4 | Reminder(TTS) | USN-4 | Getting the speech reminder to users to take their tablet | 1 | High |

**6.2 SPRINT DELIVERY PLAN**

Table 6.2 Sprint delivery plan



**7.CODING AND SOLUTIONING**

**7.1 FEATURE**

**NODE-RED DESIGN WITH FORM NODE**

Figure 7.1 shows the node red schematic

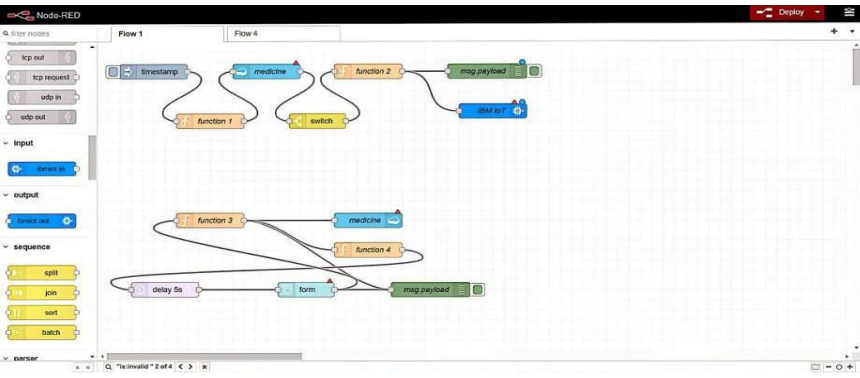
****

Figure 7.1 NODE RED SCHEMATIC

**STEPS TO ADD INPUT FORM TO UI:**

1) Configure the UI input form

2) Add a group with the name user

3) Add a tab with the name Home and press add

4) Press add

5) Give the UI from a name Insert a new User and add the input fields for medicine name, time and date

6) Add a function node and insert code and wire the nodes

7) Open in a new browser tab <https://[your-host]/ui/>

8) Insert required details and press submit

9) Now wire the function as input for the database as shown in about layout

10) Press Deploy.

**RUN AND CLICK URL:**

Figure 7.2 shows URL interface

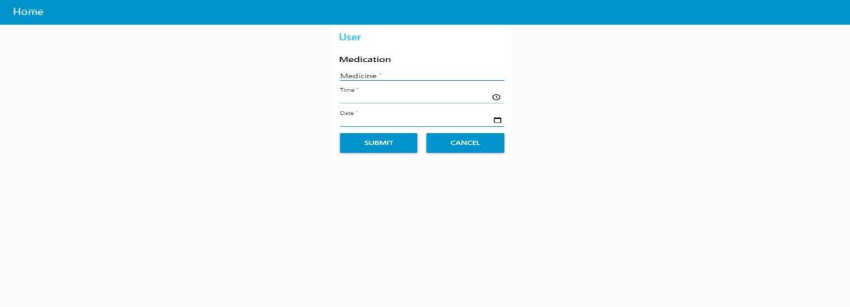
****

Figure 7.2 URL INTERFACE

**WEB UI INTERFACE FOR USERS:**

In the below image, depicts the web UI interface or web application for users to enter their medication details like name, date and time.

On clicking submit the details will be saved in the Cloudant database and will be accessed by function nodes used in the node red design. Upon comparison with time specified in function node the apt medicine name in the text format is passed on to the Text-To-Speech service (Implementation is available in sprint.

**DEBUG IN NODE-RED COMMAND PROMPT:**

Figure 7.3 shows node red command promt

****

Figure 7.3 NODE RED COMMAND PROMPT

**8. TESTING**

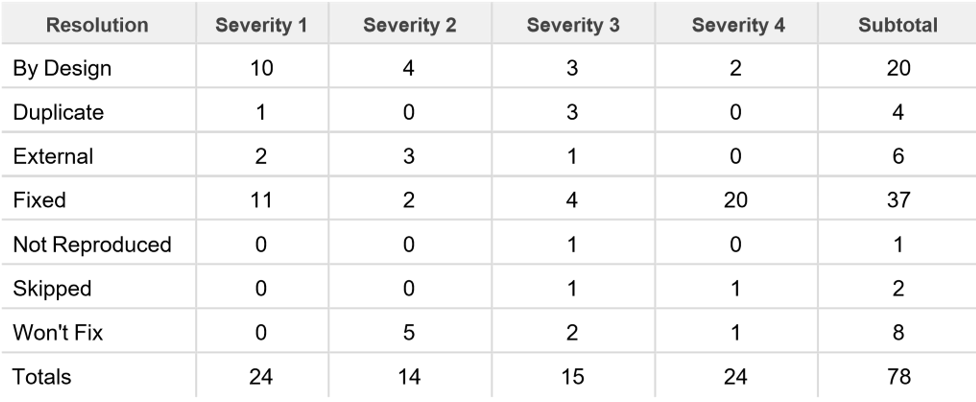
**8.1 TEST CASES AND USER ACCEPTANCE TESTING**

The purpose of this document is to briefly explain the test coverage and open issues of the project-personal assistance for senior citizen who are self-reliant at the time of the lease to User Acceptance Testing (UAT).

**DEFECTIVE ANALYSIS**

Table 8.1 shows the number of resolved or closed bugs at each severity level, and how they were resolved.

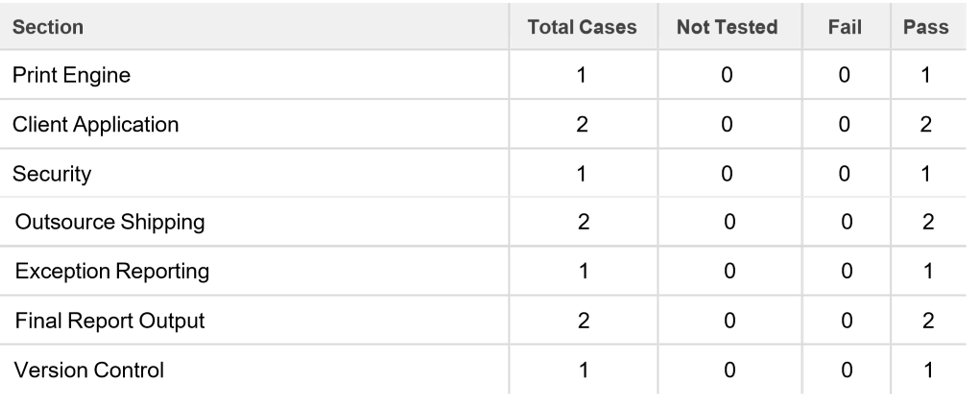
Table 8.1 DEFECTIVE ANALYSIS



**TEST CASE ANALYSIS**

Table 8.2 shows the number test cases passed and failed

Table 8.2 TEST CASE ANALYSIS



**9.RESULTS**

Preparation phase includes creating a database in cloudant,IBM watson IoT platform and devices,node red services and text to speech converter as shown in figures 9.1, 9.2, 9.3 ,9.4.

CREATE A DATABASE IN CLOUDANT DB:

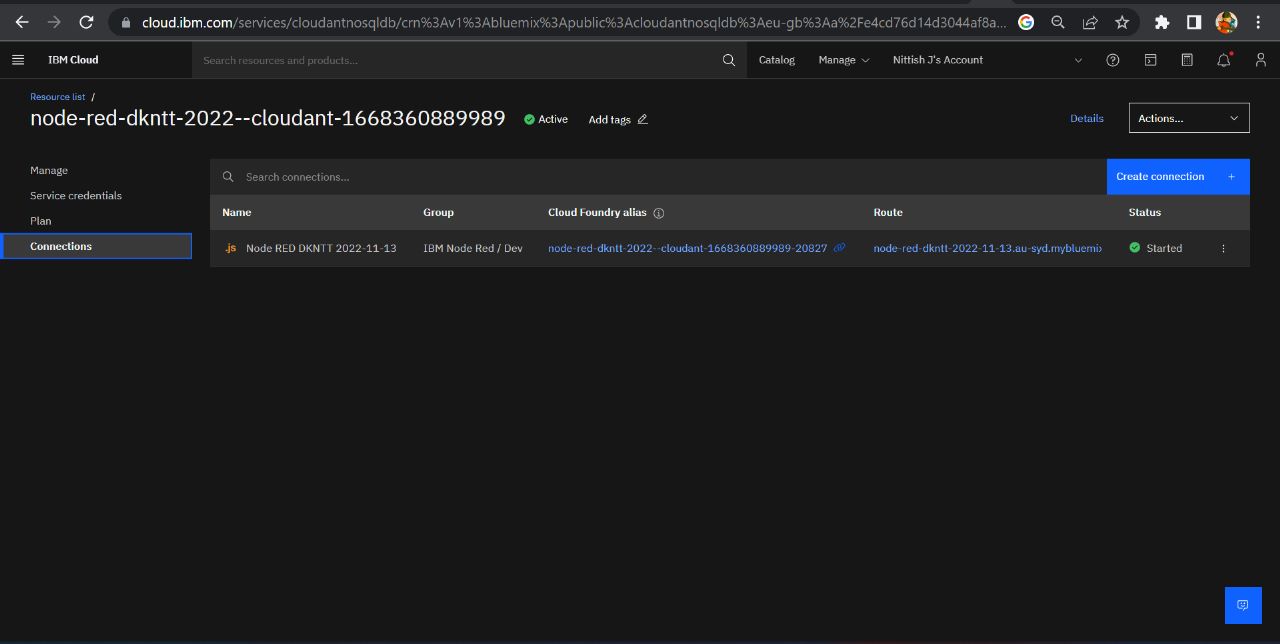


Figure 9.1 IBM CLOUDANT PLATFORM

IBM WATSON IoT PLATFORM & DEVICE :

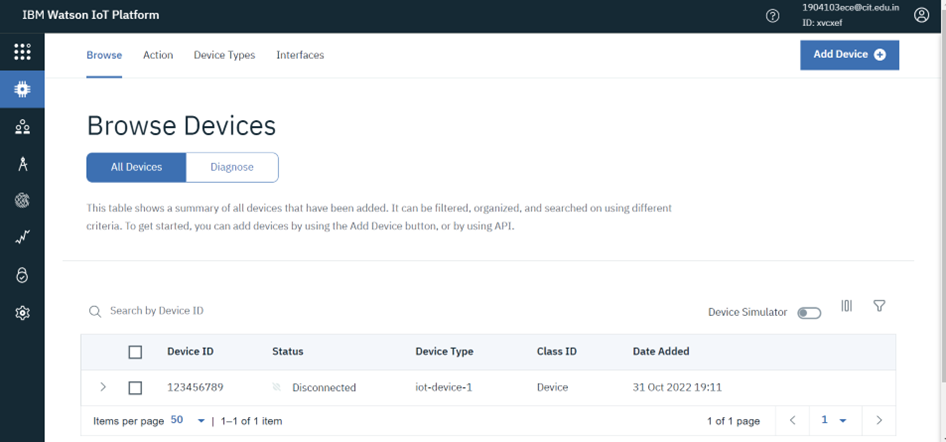
****

Figure9.2 IBM WATSON IOT PLATFORM

NODE-RED SERVICE:

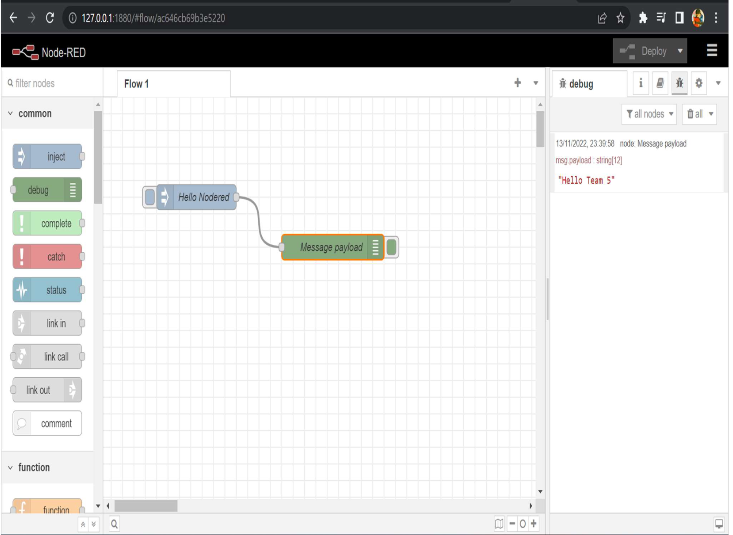
****

Figure 9.3 NODE RED SERVICES

TEXT TO SPEECH SERVICE:

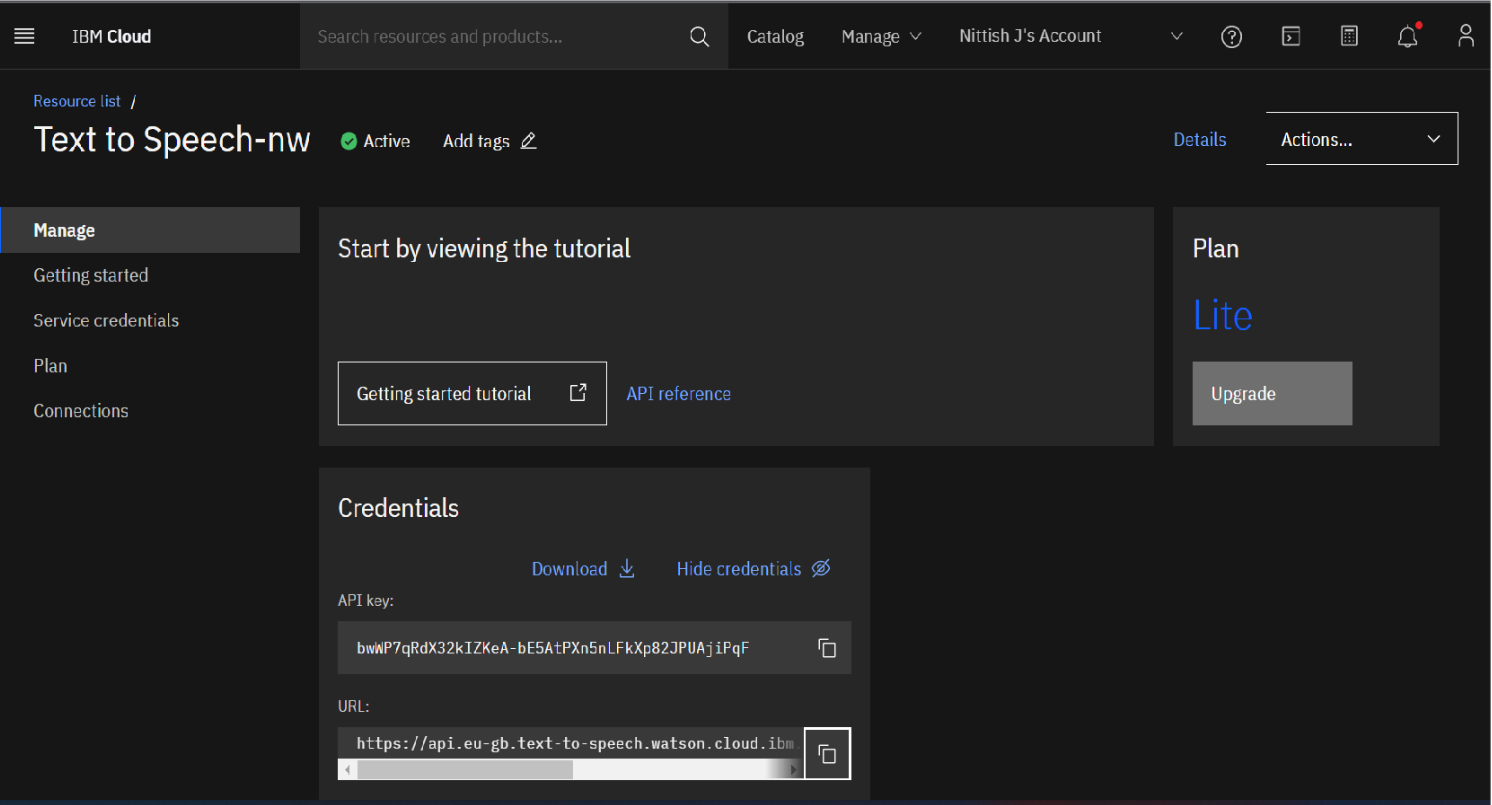
****

Figure 9.4 IBM CLOUD FOR TEXT TO SPEECH SERVICE

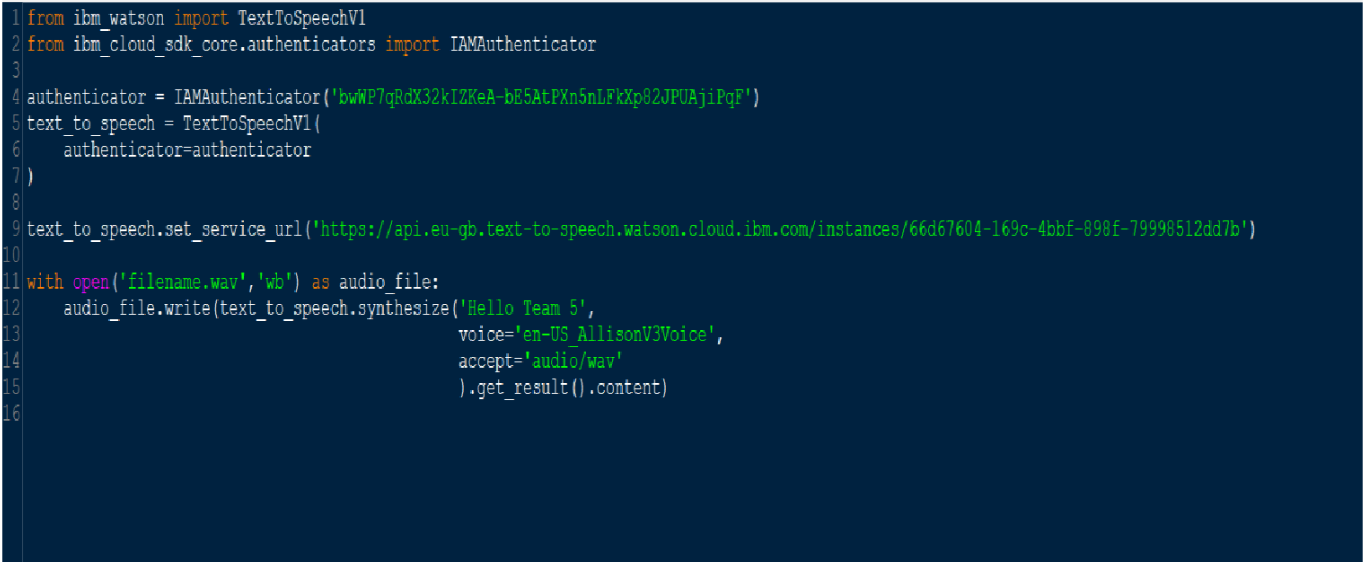
****

Figure 9.5 CODE FOR TEXT TO SPEECH SERVICE

****

Figure 9.6 TEXT TO SPEECH CONVERTED FILE

**LOGIN PAGE:**

Figure 9.7 shows the login page used in app.

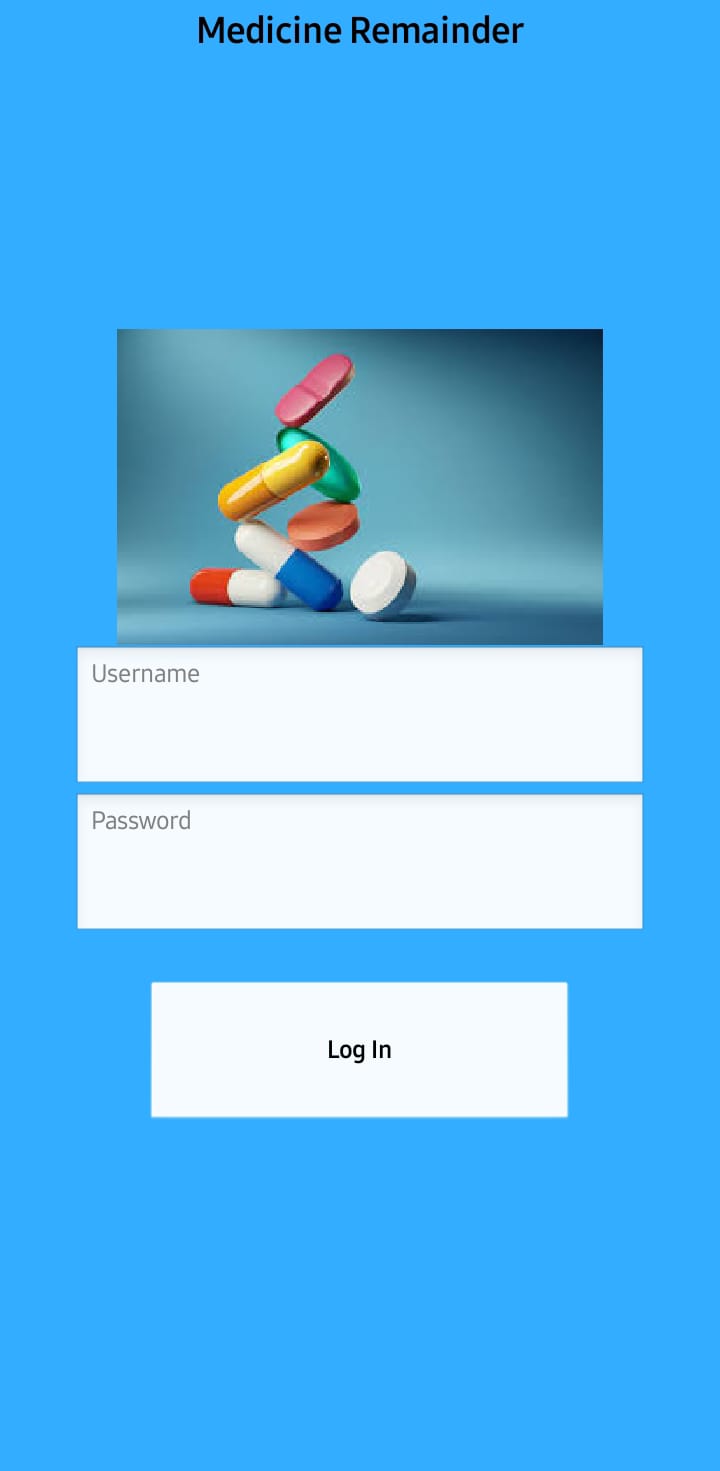
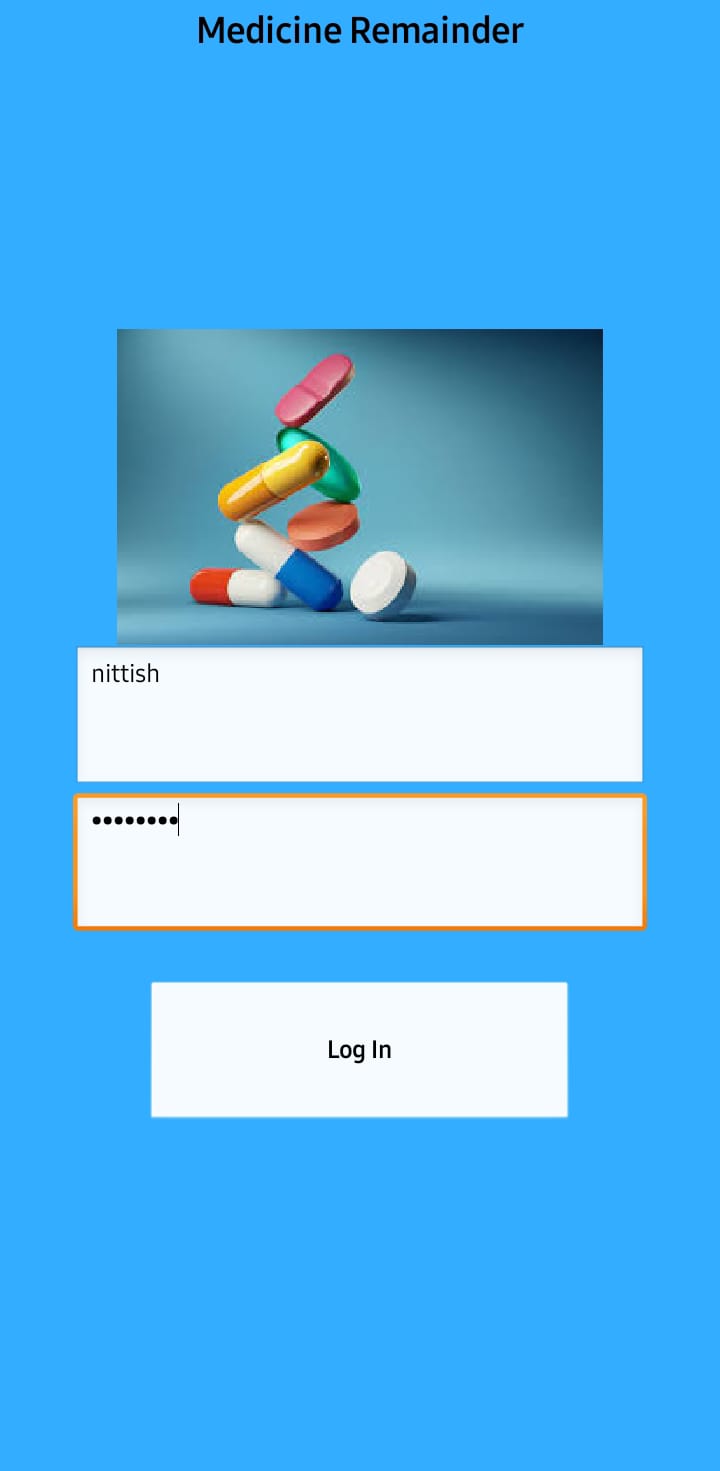
 

Figure 9.7 LOGIN SCREEN

**APP INTERFACE:**

Figure 9.8, 9.9, 9.10 shows the working of app.

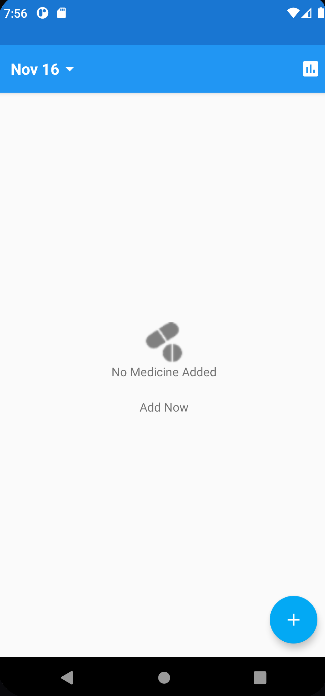
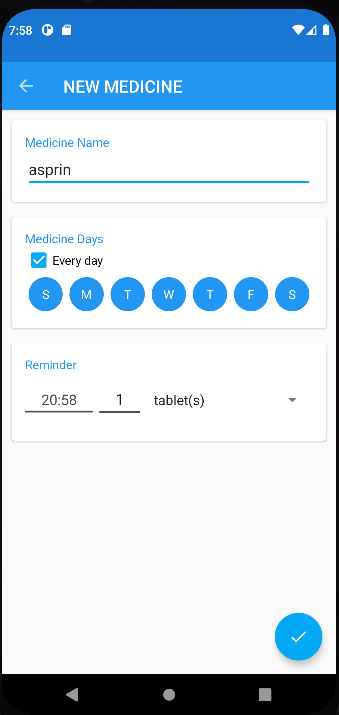
 

Figure 9.8 USER INTERFACE AND MEDICINE ADDING PAGE

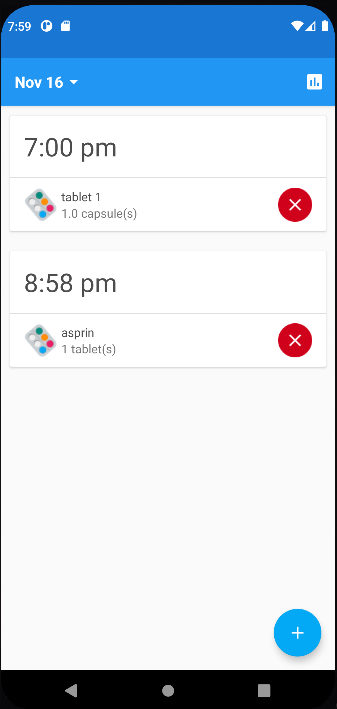
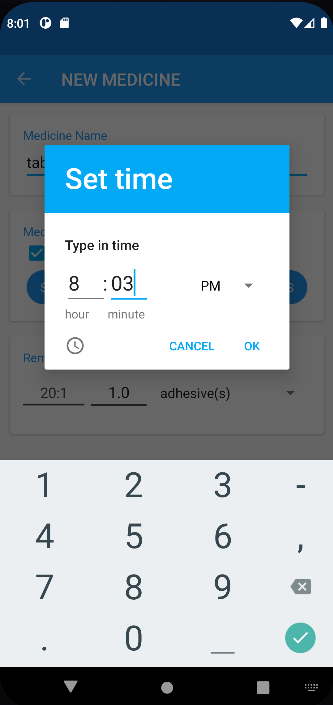
 

Figure 9.9 MEDICINE ADDED TO LIST

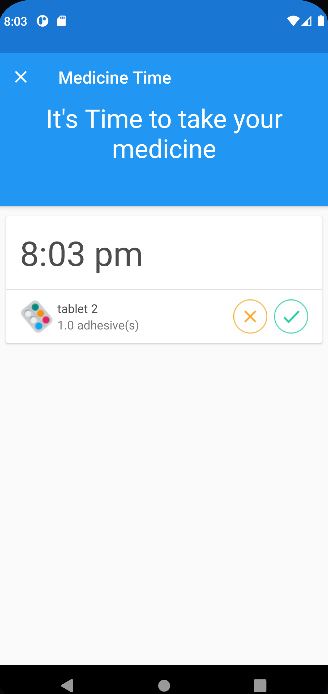
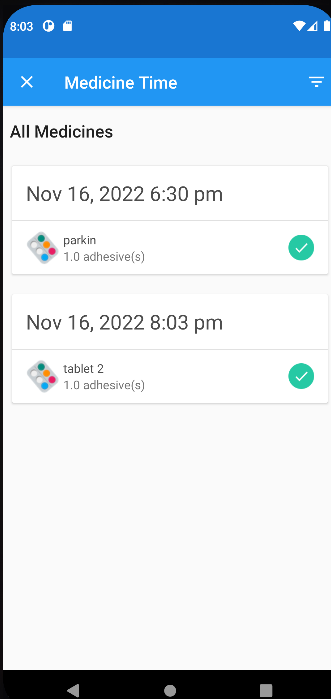
 

Figure 9.10 MEDICINE REMINDER WITH SOU

**10.ADVANTAGES AND DISADVANTAGES**

**ADVANTAGES**

It has been developed specifically to promote independent, home living for the elderly, infirm, and those living with Alzheimer’s or dementia and to enable early release from the hospital for other clients where monitoring is required.

To help these people, a medicine reminder system is developed that reminds self-reliant seniors to take the correct medication at the right time. This medicine reminder system will also help doctors monitor patients around the clock.

**DISADVANTAGES**

Even though it helps older people to do things on time but it won’t compensate the family being with them.

In the failure of application or battery failure in the phone can cause in problems as they only rely on the application and it should be monitored with more caution to prevent such kind of thing. If it is monitored it can be avoided.

**11.CONCLUSION**

We have successfully implemented a Personal Assistance for Seniors Who Are Self-Reliant which is not only very simple but also extensible.A streamlined version of the software that we have deployed enables independent people to manage their medication, medical advice, and emergencies.

Especially for those who are elderly, this feature provides users with peace of mind. Due to their extreme simplicity and minimalism, we think that IoT-based solutions are the way of the future for technology.

A medicine reminder system has been developed, that helps self-reliant seniors take their prescribed medicines at the right time. Additionally, it is also designed to help doctors/caretakers monitor their patients around the clock. The application is developed such that the user can store their medical details in IBM Cloudant DB. The user’s device will receive the medicine name via cloud and notify the user at the right time using voice commands.

**12.FUTURE SCOPE**

It is more useful for the aged person who usually forget to take medicine and are prone to serious medication. The Medicine Remainder for Elderly People System is a useful resource for those who need technological help in completing or need help in working through day-to-day tasks and taking care of their health.

Most people can expect to live into their seventies and beyond. According to the UN, the number of people aged 60 years or older is projected to grow by 56 percent worldwide by 2030. Most of these senior citizens may not have family support or cannot afford to hire personal caretakers.

Hence an application is developed for these self-reliant seniors to help them take their proper medications at the right time. The application also helps doctors/caretakers to monitor their patients. Since there is an increase in the number of potential users for the application and there is a necessity for self-reliant seniors, the medicine reminder system is developed to cater to such needs.

**SOURCE CODE:**

**CODE FOR APP INTERFACE:**

import java.util.List;

public interface AddMedicineContract {

interface View extends BaseView<Presenter> {

void showEmptyMedicineError();

void showMedicineList();

boolean isActive();

}

interface Presenter extends BasePresenter{

void saveMedicine(MedicineAlarm alarm, Pills pills);

boolean isDataMissing();

boolean isMedicineExits(String pillName);

long addPills(Pills pills);

Pills getPillsByName(String pillName);

List<MedicineAlarm> getMedicineByPillName(String pillName);

List<Long> tempIds();

void deleteMedicineAlarm(long alarmId);

}

}

**CODE FOR TIMER AND DATE :**

import static android.content.Context.ALARM\_SERVICE;

public class AddMedicineFragment extends Fragment implements AddMedicineContract.View {

public static final String ARGUMENT\_EDIT\_MEDICINE\_ID = "ARGUMENT\_EDIT\_MEDICINE\_ID";

public static final String ARGUMENT\_EDIT\_MEDICINE\_NAME = "ARGUMENT\_EDIT\_MEDICINE\_NAME";

@BindView(R.id.edit\_med\_name)

EditText editMedName;

@BindView(R.id.every\_day)

AppCompatCheckBox everyDay;

@BindView(R.id.dv\_sunday)

DayViewCheckBox dvSunday;

@BindView(R.id.dv\_monday)

DayViewCheckBox dvMonday;

@BindView(R.id.dv\_tuesday)

DayViewCheckBox dvTuesday;

@BindView(R.id.dv\_wednesday)

DayViewCheckBox dvWednesday;

@BindView(R.id.dv\_thursday)

DayViewCheckBox dvThursday;

@BindView(R.id.dv\_friday)

DayViewCheckBox dvFriday;

@BindView(R.id.dv\_saturday)

DayViewCheckBox dvSaturday;

@BindView(R.id.checkbox\_layout)

LinearLayout checkboxLayout;

@BindView(R.id.tv\_medicine\_time)

RobotoBoldTextView tvMedicineTime;

@BindView(R.id.tv\_dose\_quantity)

EditText tvDoseQuantity;

@BindView(R.id.spinner\_dose\_units)

AppCompatSpinner spinnerDoseUnits;

private List<String> doseUnitList;

private boolean[] dayOfWeekList = new boolean[7];

private int hour, minute;

Unbinder unbinder;

private AddMedicineContract.Presenter mPresenter;

private View rootView;

private String doseUnit;

static AddMedicineFragment newInstance() {

Bundle args = new Bundle();

AddMedicineFragment fragment = new AddMedicineFragment();

fragment.setArguments(args);

return fragment;

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

FloatingActionButton fab = Objects.requireNonNull(getActivity()).findViewById(R.id.fab\_edit\_task\_done);

fab.setImageResource(R.drawable.ic\_done);

fab.setOnClickListener(setClickListener);

}

@Nullable

@Override

public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {

rootView = inflater.inflate(R.layout.fragment\_add\_medicine, container, false);

unbinder = ButterKnife.bind(this, rootView);

setCurrentTime();

setSpinnerDoseUnits();

return rootView;

}

@Override

public void setPresenter(AddMedicineContract.Presenter presenter) {

this.mPresenter = presenter;

}

@Override

public void showEmptyMedicineError() {

// Snackbar.make(mTitle, getString(R.string.empty\_task\_message), Snackbar.LENGTH\_LONG).show();

}

@Override

public void showMedicineList() {

Objects.requireNonNull(getActivity()).setResult(Activity.RESULT\_OK);

getActivity().finish();

}

@Override

public boolean isActive() {

return isAdded();

}

@Override

public void onDestroyView() {

super.onDestroyView();

unbinder.unbind();

}

@OnClick({R.id.every\_day, R.id.dv\_monday, R.id.dv\_tuesday, R.id.dv\_wednesday,

R.id.dv\_thursday, R.id.dv\_friday, R.id.dv\_saturday, R.id.dv\_sunday})

public void onCheckboxClicked(View view) {

boolean checked = ((CheckBox) view).isChecked();

/\*\* Checking which checkbox was clicked \*/

switch (view.getId()) {

case R.id.dv\_monday:

if (checked) {

dayOfWeekList[1] = true;

} else {

dayOfWeekList[1] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_tuesday:

if (checked) {

dayOfWeekList[2] = true;

} else {

dayOfWeekList[2] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_wednesday:

if (checked) {

dayOfWeekList[3] = true;

} else {

dayOfWeekList[3] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_thursday:

if (checked) {

dayOfWeekList[4] = true;

} else {

dayOfWeekList[4] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_friday:

if (checked) {

dayOfWeekList[5] = true;

} else {

dayOfWeekList[5] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_saturday:

if (checked) {

dayOfWeekList[6] = true;

} else {

dayOfWeekList[6] = false;

everyDay.setChecked(false);

}

break;

case R.id.dv\_sunday:

if (checked) {

dayOfWeekList[0] = true;

} else {

dayOfWeekList[0] = false;

everyDay.setChecked(false);

}

break;

case R.id.every\_day:

LinearLayout ll = (LinearLayout) rootView.findViewById(R.id.checkbox\_layout);

for (int i = 0; i < ll.getChildCount(); i++) {

View v = ll.getChildAt(i);

((DayViewCheckBox) v).setChecked(checked);

onCheckboxClicked(v);

}

break;

}

}

@OnClick(R.id.tv\_medicine\_time)

void onMedicineTimeClick() {

showTimePicker();

}

private void showTimePicker() {

Calendar mCurrentTime = Calendar.getInstance();

hour = mCurrentTime.get(Calendar.HOUR\_OF\_DAY);

minute = mCurrentTime.get(Calendar.MINUTE);

TimePickerDialog mTimePicker;

mTimePicker = new TimePickerDialog(getContext(), new TimePickerDialog.OnTimeSetListener() {

@Override

public void onTimeSet(TimePicker timePicker, int selectedHour, int selectedMinute) {

hour = selectedHour;

minute = selectedMinute;

tvMedicineTime.setText(String.format(Locale.getDefault(), "%d:%d", selectedHour, selectedMinute));

}

}, hour, minute, false);//No 24 hour time

mTimePicker.setTitle("Select Time");

mTimePicker.show();

}

private void setCurrentTime() {

Calendar mCurrentTime = Calendar.getInstance();

hour = mCurrentTime.get(Calendar.HOUR\_OF\_DAY);

minute = mCurrentTime.get(Calendar.MINUTE);

tvMedicineTime.setText(String.format(Locale.getDefault(), "%d:%d", hour, minute));

}

private void setSpinnerDoseUnits() {

doseUnitList = Arrays.asList(getResources().getStringArray(R.array.medications\_shape\_array));

ArrayAdapter<String> adapter = new ArrayAdapter<>(Objects.requireNonNull(getContext()), android.R.layout.simple\_dropdown\_item\_1line, doseUnitList);

spinnerDoseUnits.setAdapter(adapter);

}

@OnItemSelected(R.id.spinner\_dose\_units)

void onSpinnerItemSelected(int position) {

if (doseUnitList == null || doseUnitList.isEmpty()) {

return;

}

doseUnit = doseUnitList.get(position);

Log.d("TAG", doseUnit);

}

private View.OnClickListener setClickListener = new View.OnClickListener() {

@Override

public void onClick(View v) {

int checkBoxCounter = 0;

String pill\_name = editMedName.getText().toString();

String doseQuantity = tvDoseQuantity.getText().toString();

Calendar takeTime = Calendar.getInstance();

Date date = takeTime.getTime();

String dateString = new SimpleDateFormat("MMM d, yyyy", Locale.getDefault()).format(date);

/\*\* Updating model \*/

MedicineAlarm alarm = new MedicineAlarm();

int alarmId = new Random().nextInt(100);

/\*\* If Pill does not already exist \*/

if (!mPresenter.isMedicineExits(pill\_name)) {

Pills pill = new Pills();

pill.setPillName(pill\_name);

alarm.setDateString(dateString);

alarm.setHour(hour);

alarm.setMinute(minute);

alarm.setPillName(pill\_name);

alarm.setDayOfWeek(dayOfWeekList);

alarm.setDoseUnit(doseUnit);

alarm.setDoseQuantity(doseQuantity);

alarm.setAlarmId(alarmId);

pill.addAlarm(alarm);

long pillId = mPresenter.addPills(pill);

pill.setPillId(pillId);

mPresenter.saveMedicine(alarm, pill);

} else { // If Pill already exists

Pills pill = mPresenter.getPillsByName(pill\_name);

alarm.setDateString(dateString);

alarm.setHour(hour);

alarm.setMinute(minute);

alarm.setPillName(pill\_name);

alarm.setDayOfWeek(dayOfWeekList);

alarm.setDoseUnit(doseUnit);

alarm.setDoseQuantity(doseQuantity);

alarm.setAlarmId(alarmId);

pill.addAlarm(alarm);

mPresenter.saveMedicine(alarm, pill);

}

List<Long> ids = new LinkedList<>();

try {

List<MedicineAlarm> alarms = mPresenter.getMedicineByPillName(pill\_name);

for (MedicineAlarm tempAlarm : alarms) {

if (tempAlarm.getHour() == hour && tempAlarm.getMinute() == minute) {

ids = tempAlarm.getIds();

break;

}

}

} catch (Exception e) {

e.printStackTrace();

}

for (int i = 0; i < 7; i++) {

if (dayOfWeekList[i] && pill\_name.length() != 0) {

int dayOfWeek = i + 1;

long \_id = ids.get(checkBoxCounter);

int id = (int) \_id;

checkBoxCounter++;

/\*\* This intent invokes the activity ReminderActivity, which in turn opens the AlertAlarm window \*/

Intent intent = new Intent(getActivity(), ReminderActivity.class);

intent.putExtra(ReminderFragment.EXTRA\_ID, \_id);

PendingIntent operation = PendingIntent.getActivity(getActivity(), id, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);

/\*\* Getting a reference to the System Service ALARM\_SERVICE \*/

AlarmManager alarmManager = (AlarmManager) Objects.requireNonNull(getActivity()).getSystemService(ALARM\_SERVICE);

/\*\* Creating a calendar object corresponding to the date and time set by the user \*/

Calendar calendar = Calendar.getInstance();

calendar.set(Calendar.HOUR\_OF\_DAY, hour);

calendar.set(Calendar.MINUTE, minute);

calendar.set(Calendar.SECOND, 0);

calendar.set(Calendar.MILLISECOND, 0);

calendar.set(Calendar.DAY\_OF\_WEEK, dayOfWeek);

/\*\* Converting the date and time in to milliseconds elapsed since epoch \*/

long alarm\_time = calendar.getTimeInMillis();

if (calendar.before(Calendar.getInstance()))

alarm\_time += AlarmManager.INTERVAL\_DAY \* 7;

assert alarmManager != null;

alarmManager.setRepeating(AlarmManager.RTC\_WAKEUP, alarm\_time,

AlarmManager.INTERVAL\_DAY \* 7, operation);

}

}

Toast.makeText(getContext(), "Alarm for " + pill\_name + " is set successfully", Toast.LENGTH\_SHORT).show();

showMedicineList();

}

};

}

**CODE FOR ALARM:**

Package com.gautam.medicinetime.alarm;

import android.content.Intent;

import android.os.Bundle;

import androidx.appcompat.app.ActionBar;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import android.view.MenuItem;

import com.gautam.medicinetime.Injection;

import com.gautam.medicinetime.R;

import com.gautam.medicinetime.utils.ActivityUtils;

import butterknife.BindView;

import butterknife.ButterKnife;

public class ReminderActivity extends AppCompatActivity {

@BindView(R.id.toolbar)

Toolbar toolbar;

ReminderPresenter mReminderPresenter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_reminder\_actvity);

ButterKnife.bind(this);

setSupportActionBar(toolbar);

ActionBar actionBar = getSupportActionBar();

if (actionBar != null) {

actionBar.setDisplayHomeAsUpEnabled(true);

}

Intent intent = getIntent();

if (!intent.hasExtra(ReminderFragment.EXTRA\_ID)) {

finish();

return;

}

long id = intent.getLongExtra(ReminderFragment.EXTRA\_ID, 0);

ReminderFragment reminderFragment = (ReminderFragment) getSupportFragmentManager().findFragmentById(R.id.contentFrame);

if (reminderFragment == null) {

reminderFragment = ReminderFragment.newInstance(id);

ActivityUtils.addFragmentToActivity(getSupportFragmentManager(), reminderFragment, R.id.contentFrame);

}

//Create MedicinePresenter

mReminderPresenter = new ReminderPresenter(Injection.provideMedicineRepository(ReminderActivity.this), reminderFragment);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

if (mReminderPresenter != null) {

mReminderPresenter.finishActivity();

}

}

return super.onOptionsItemSelected(item);

}

@Override

public void onBackPressed() {

if (mReminderPresenter != null) {

mReminderPresenter.finishActivity();

}

}

}

**GITHUB LINK:** https://github.com/IBM-EPBL/IBM-Project-9573-1659021703

**PROJECT DEMO LINK:**

https://drive.google.com/file/d/1ZbmSbU9YZC4GVrgCM7jhjX\_KYzrMe8ce/view?usp=share\_link

**REFERENCES**

1. Waykole, M., Prakash, V., & Nalini, N. H. S. (2016). Ardumed-smart medicine reminder for old people. Int J Sci Eng Res, 7,pp. 650-654.
2. Ahmad, S., Hasan, M., Shahabuddin, M., Tabassum, T., & Allvi, M. W. (2020). IoT based pill reminder and monitoring system. International Journal of Computer Science and Network Security, 20(7),pp.152-158.
3. Annamalai, M., & Jesintha, D. X. M. (2021). Smart iot system based patient monitoring and medicine reminder,7, pp.2710-2721
4. Ghogare, P. P., & Jahagirdar, M. V. (2018). Medication reminder and monitoring using IoT.
5. Madankar, A., Agrawal, A., & Yede, V. (2021, November). IoT based Advance Pill Reminder System for Distinct Patients.(IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) pp. 161-165. IEEE.
6. Zanjal, S. V., & Talmale, G. R. (2016). Medicine reminder and monitoring system for secure health using IOT. Procedia Computer Science, 78, pp.471-476.