**Design Document**

Any Time Medicine

Alok Asok

Kapil Sharma

Sagar Dhamija

Varun Khandelwal

Ramanpreet Singh Khinda

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Version Date | Rathin’s Approval | Nikhil’s Approval |
| 1.0 | 10/18/2015 | NA | NA |
| 1.1 | 11/10/2015 | 11/13/2015 |  |

Contents

[1. Introduction 6](#_Toc435270688)

[1.1. Purpose 6](#_Toc435270689)

[1.2. Scope 6](#_Toc435270690)

[1.3. High Level Design Summary 6](#_Toc435270691)

[1.4. References 6](#_Toc435270692)

[1.5. Acronyms and Glossary 7](#_Toc435270693)

[2. Global Data Structures and Shared Data Functions 8](#_Toc435270694)

[2.1. Table Name: ATM\_USER\_PROFILE\_TB 8](#_Toc435270695)

[2.2. Table Name: ATM\_HEALTHCARE\_CENTER\_TB 8](#_Toc435270696)

[2.3. Table Name: ATM\_INSURANCE\_PROVIDER\_TB 8](#_Toc435270697)

[2.4. Table Name: ATM\_BODY\_TB 8](#_Toc435270698)

[2.5. Table Name: ATM\_REMEDY\_TB 9](#_Toc435270699)

[2.6. Table Name: ATM\_SYMPTOM\_TB 9](#_Toc435270700)

[2.7. Table Name: ATM\_HISTORY\_TB 9](#_Toc435270701)

[2.8. Table Name: ATM\_INSURANCE\_PRO\_HEALTHCARE\_CENTER\_JOIN\_TB 9](#_Toc435270702)

[2.9. Table Name: ATM\_OTC\_TB 9](#_Toc435270703)

[2.10. Table Name: ATM\_HOME\_REMEDY\_TB 10](#_Toc435270704)

[3. High Level Design 11](#_Toc435270705)

[3.1. Use case Diagram 11](#_Toc435270706)

[3.2. Sequence Diagram 12](#_Toc435270707)

[3.2.1 USR\_001: Register 12](#_Toc435270709)

[3.2.2 USR\_002: Login 13](#_Toc435270710)

[3.2.3 USR\_003: Logout 14](#_Toc435270711)

[3.2.4 USR\_004: Get Medicine 15](#_Toc435270712)

[3.2.5 USR\_005: Locate healthcare center: 16](#_Toc435270713)

[3.2.6 USR\_006 & USR\_007: Calculate BMI & Get Diet Plan 17](#_Toc435270714)

[3.2.7 USR\_008: View History 18](#_Toc435270715)

[3.2.8 USR\_009: Change Password: 19](#_Toc435270716)

[3.2.9 USR\_010: Reset Password: 20](#_Toc435270717)

[3.2.10 USR\_011: Update Profile: 21](#_Toc435270718)

[3.3. ER DIAGRAM 22](#_Toc435270719)

[3.4. DATA FLOW DIAGRAM 23](#_Toc435270720)

[4. User Interface Design 25](#_Toc435270721)

[4.1. USR\_001: 25](#_Toc435270722)

[4.1.1. Design Details 25](#_Toc435270723)

[4.1.1.1. Wireframe Design 25](#_Toc435270724)

[4.1.1.2. Processing Details 25](#_Toc435270725)

[4.1.1.3. Error to be checked for: 26](#_Toc435270726)

[4.1.1.4. Output: 26](#_Toc435270727)

[4.1.1.5. Error Capturing 26](#_Toc435270728)

[4.1.1.6. Boundary and normal Operating condition 28](#_Toc435270729)

[4.1.1.7. Global data structure references: 28](#_Toc435270730)

[4.1.1.8. Module specific data structure: 28](#_Toc435270731)

[4.1.1.9. External Interfaces: 28](#_Toc435270732)

[4.1.1.10. Assumptions: 28](#_Toc435270733)

[4.2. USR\_002: 29](#_Toc435270734)

[4.2.2. Design Details 29](#_Toc435270735)

[4.2.2.1. Wireframe Design 29](#_Toc435270736)

[4.2.2.2. Processing Details 29](#_Toc435270737)

[4.2.2.3. Error to be checked for: 29](#_Toc435270738)

[4.2.2.4. Output: 30](#_Toc435270739)

[4.2.2.5. Error Capturing 30](#_Toc435270740)

[4.2.2.6. Boundary and normal Operating condition 31](#_Toc435270741)

[4.2.2.7. Global data structure references: 31](#_Toc435270742)

[4.2.2.8. Module specific data structure: 31](#_Toc435270743)

[4.2.2.9. External Interfaces: 31](#_Toc435270744)

[4.2.2.10. Assumptions: 31](#_Toc435270745)

[4.3. USR\_003: 32](#_Toc435270746)

[4.3.1. Design Details 32](#_Toc435270747)

[4.3.1.1. Wireframe Design 32](#_Toc435270748)

[4.3.1.2. Processing Details 32](#_Toc435270749)

[4.3.1.3. Error to be checked for: 32](#_Toc435270750)

[4.3.1.4. Output: 32](#_Toc435270751)

[4.3.1.5. Error Capturing 33](#_Toc435270752)

[4.3.1.6. Boundary and normal Operating condition 33](#_Toc435270753)

[4.3.1.7. Global data structure references: 33](#_Toc435270754)

[4.3.1.8. Module specific data structure: 33](#_Toc435270755)

[4.3.1.9. External Interfaces: 33](#_Toc435270756)

[4.3.1.10. Assumptions: 33](#_Toc435270757)

[4.4. USR\_004: 34](#_Toc435270758)

[4.4.1. Design Details 34](#_Toc435270759)

[4.4.1.1. Wireframe Design 34](#_Toc435270760)

[4.4.1.2. Processing Details 34](#_Toc435270761)

[4.4.1.3. Error to be checked for: 35](#_Toc435270762)

[4.4.1.4. Output: 35](#_Toc435270763)

[4.4.1.5. Error Capturing 35](#_Toc435270764)

[4.4.1.6. Boundary and normal Operating condition 35](#_Toc435270765)

[4.4.1.7. Global data structure references: 35](#_Toc435270766)

[4.4.1.8. Module specific data structure: 35](#_Toc435270767)

[4.4.1.9. External Interfaces: 35](#_Toc435270768)

[4.4.1.10. Assumptions: 36](#_Toc435270769)

[4.5. USR\_005: 37](#_Toc435270770)

[4.5.1. Design Details 37](#_Toc435270771)

[4.5.1.1. Wireframe Design 37](#_Toc435270772)

[4.5.1.2. Processing Details 37](#_Toc435270773)

[4.5.1.3. Error to be checked for: 38](#_Toc435270774)

[4.5.1.4. Output: 38](#_Toc435270775)

[4.5.1.5. Error Capturing 38](#_Toc435270776)

[4.5.1.6. Boundary and normal Operating condition 38](#_Toc435270777)

[4.5.1.7. Global data structure references: 38](#_Toc435270778)

[4.5.1.8. Module specific data structure: 39](#_Toc435270779)

[4.5.1.9. External Interfaces: 39](#_Toc435270780)

[4.5.1.10. Assumptions: 39](#_Toc435270781)

[4.6. USR\_006 & USR\_007: 40](#_Toc435270782)

[4.6.1. Design Details 40](#_Toc435270783)

[4.6.1.1. Wireframe Design 40](#_Toc435270784)

[4.6.1.2. Processing Details 40](#_Toc435270785)

[4.6.1.3. Error to be checked for: 40](#_Toc435270786)

[4.6.1.4. Output: 40](#_Toc435270787)

[4.6.1.5. Error Capturing 41](#_Toc435270788)

[4.6.1.6. Boundary and normal Operating condition 41](#_Toc435270789)

[4.6.1.7. Global data structure references: 41](#_Toc435270790)

[4.6.1.8. Module specific data structure: 41](#_Toc435270791)

[4.6.1.9. External Interfaces: 41](#_Toc435270792)

[4.6.1.10. Assumptions: 41](#_Toc435270793)

[4.7. USR\_008: 42](#_Toc435270794)

[4.7.1. Design Details 42](#_Toc435270795)

[4.7.1.1. Wireframe Design 42](#_Toc435270796)

[4.7.1.2. Processing Details 42](#_Toc435270797)

[4.7.1.3. Error to be checked for: 42](#_Toc435270798)

[4.7.1.4. Output: 43](#_Toc435270799)

[4.7.1.5. Error Capturing 43](#_Toc435270800)

[4.7.1.6. Boundary and normal Operating condition 43](#_Toc435270801)

[4.7.1.7. Global data structure references: 43](#_Toc435270802)

[4.7.1.8. Module specific data structure: 43](#_Toc435270803)

[4.7.1.9. External Interfaces: 44](#_Toc435270804)

[4.7.1.10. Assumptions: 44](#_Toc435270805)

[4.8. USR\_009: 45](#_Toc435270806)

[4.8.1. Design Details 45](#_Toc435270807)

[4.8.1.1. Wireframe Design 45](#_Toc435270808)

[4.8.1.2. Processing Details 45](#_Toc435270809)

[4.8.1.3. Error to be checked for: 46](#_Toc435270810)

[4.8.1.4. Output: 46](#_Toc435270811)

[4.8.1.5. Error Capturing 46](#_Toc435270812)

[4.8.1.6. Boundary and normal Operating condition 48](#_Toc435270813)

[4.8.1.7. Global data structure references: 48](#_Toc435270814)

[4.8.1.8. Module specific data structure: 48](#_Toc435270815)

[4.8.1.9. External Interfaces: 48](#_Toc435270816)

[4.8.1.10. Assumptions: 48](#_Toc435270817)

[4.9. USR\_010: 49](#_Toc435270818)

[4.9.1. Design Details 49](#_Toc435270819)

[4.9.1.1. Wireframe Design 49](#_Toc435270820)

[4.9.1.2. Processing Details 49](#_Toc435270821)

[4.9.1.3. Error to be checked for: 50](#_Toc435270822)

[4.9.1.4. Output: 50](#_Toc435270823)

[4.9.1.5. Error Capturing 50](#_Toc435270824)

[4.9.1.6. Boundary and normal Operating condition 50](#_Toc435270825)

[4.9.1.7. Global data structure references: 50](#_Toc435270826)

[4.9.1.8. Module specific data structure: 51](#_Toc435270827)

[4.9.1.9. External Interfaces: 51](#_Toc435270828)

[4.9.1.10. Assumptions: 51](#_Toc435270829)

[4.10. USR\_011: 52](#_Toc435270830)

[4.10.1. Design Details 52](#_Toc435270831)

[4.10.1.1. Wireframe Design 52](#_Toc435270832)

[4.10.1.2. Processing Details 52](#_Toc435270833)

[4.10.1.3. Error to be checked for: 52](#_Toc435270834)

[4.10.1.4. Output: 53](#_Toc435270835)

[4.10.1.5. Error Capturing 53](#_Toc435270836)

[4.10.1.6. Boundary and normal Operating condition 53](#_Toc435270837)

[4.10.1.7. Global data structure references: 54](#_Toc435270838)

[4.10.1.8. Module specific data structure: 54](#_Toc435270839)

[4.10.1.9. External Interfaces: 54](#_Toc435270840)

[4.10.1.10. Assumptions: 54](#_Toc435270841)

[5. TRACEABILITY TO REQUIREMENTS 55](#_Toc435270842)

# Introduction

## Purpose

The document’s chief purpose is to provide a detailed description of the design for “Any Time Medicine” (ATM) application software. It elaborates the purpose and features of the application software, what and how the application does (description of User Interfaces), the design approach used, the programming constructs and their interactions, data design and the architecture design. The document is primarily intended for the stakeholders, developers and quality assurance team and will be proposed to the instructor, teaching assistant and quality assurance team for review and feedback.

## Scope

The “Any Time Medicine” is an android based mobile application that provides one stop medical solution to user’s health concerns/ medical issues by recommending Over the Counter (OTC) medicines and home remedies without having to visit the physician.

The scope of the project will be to provide the functionality as described in the requirements document. The application will be developed on a LINUX machine using Android Studio, Oracle 11g and JDBC/ ODBC.

Requirements:

1. User – Register, Login, Logout, Change password, Update Profile, Reset password, Search Remedy, Get BMI and Diet Plan, Locate Health care centre, View history.

## High Level Design Summary

Architecture: Client - Server

Front End: JAVA

Backend: Oracle

## References

* <https://en.wikipedia.org/wiki/Software_design_description>
* <http://www.iso-architecture.org/ieee-p1016/>
* <https://en.wikipedia.org/wiki/Data_flow_diagram>
* <https://en.wikipedia.org/wiki/Sequence_diagram>
* <http://www.justinmind.com/>
* <http://www.uml-diagrams.org/>
* <https://en.wikipedia.org/wiki/Class_diagram>
* <http://www.agilemodeling.com/artifacts/classDiagram.htm>
* <http://developer.android.com/guide/index.html>
* <https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model>
* <https://support.office.com/en-us/article/Create-a-data-flow-diagram-43623251-96c4-4ed5-8da3-70361afed88e>
* [https://erdplus.com/#/](https://erdplus.com/" \l "/)
* <https://msdn.microsoft.com/en-us/library/dd409416.aspx>

## Acronyms and Glossary

Abbreviation:

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| N/A | Not Applicable |
| GPS | Global Positioning System |
| DOB | Date of Birth |
| OTC | Over The Counter |
| FK | Foreign Key |
| PK | Primary Key |

Glossary:

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| FOREIGN KEY | A foreign key is a column (or columns) that references a column (most often the primary key) of another table. |
| PRIMARY KEY | A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records. |
| NOT NULL | The NOT NULL constraint enforces a column to NOT accept NULL values. |
| REFERENTIAL INTEGITY | Referential integrity is a property of data which, when satisfied, requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute in a different (or the same) relation (table). |
| CONSTRAINT | Constraints are the rules enforced on data columns on table. |
| UNIQUE | The UNIQUE constraint uniquely identifies each record in a database table. |
| VARCHAR | A varchar or Variable Character Field is a set of character data of indeterminate length. |
| FIELD | A field is part of a record and contains a single piece of data for the subject of the record |

# Global Data Structures and Shared Data Functions

## Table Name: ATM\_USER\_PROFILE\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| USER\_ID | NUMBER(10) | PRIMARY KEY(PK\_USER\_PROFILE) |
| FIRST\_NAME | VARCHAR2(200) |  |
| LAST\_NAME | VARCHAR2(200) |  |
| DOB | DATE |  |
| GENDER | VARCHAR2(1) | CHECK |
| CONSTRAINT |
| HEIGHT | NUMBER(3) |  |
| WEIGHT | NUMBER(5,0) |  |
| EMAIL\_ID | VARCHAR2(50) | UNIQUE & NOT NULL |
| PHONE\_NUMBER | NUMBER(10) | UNIQUE |
| INSURANCE\_PROVIDER\_ID | NUMBER(10) | FOREIGN KEY |
| PASSWORD | VARCHAR2(32) | NOT NULL |
| LOGIN\_STATUS | NUMBER(1) | 0 - Not Logged in |
| 1 - Already logged in |

## Table Name: ATM\_HEALTHCARE\_CENTER\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| HEALTHCARE\_PROVIDER\_ID | NUMBER(10) | PRIMARY KEY(PK\_HEALTHCARE\_CENTER) |
| HEALTHCARE\_PROVIDER\_NAME | VARCHAR2(200) | NOT NULL |
| ADDRESS | VARCHAR2(250) | NOT NULL |
| PHONE\_NUMBER | NUMBER(10) | NOT NULL |
| EMAIL\_ID | VARCHAR2(50) |  |
| LATITUDE | NUMBER(20,10) | NOT NULL |
| LONGITUDE | NUMBER(20,10) | NOT NULL |

## Table Name: ATM\_INSURANCE\_PROVIDER\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| INSURANCE\_PROVIDER\_ID | NUMBER(10) | PRIMARY KEY(PK\_INSURANCE\_PROVIDER) |
| INSURANCE\_PROVIDER\_NAME | VARCHAR2(200) | NOT NULL |

## Table Name: ATM\_BODY\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| BODY\_PART\_ID | NUMBER(10) | PRIMARY KEY(PK\_BODY) |
| BODY\_PART | VARCHAR2(200) | UNIQUE & NOT NULL |

## Table Name: ATM\_REMEDY\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| REMEDY\_ID | NUMBER(10) | PRIMARY KEY(PK\_REMEDY) |
| SYMPTOM\_ID | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_SYMPTOM) |
| BODY\_PART\_ID | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_BODY) |
| HOME\_REMEDY\_ID1 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_HOME\_REMEDY) |
| HOME\_REMEDY\_ID2 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_HOME\_REMEDY) |
| HOME\_REMEDY\_ID3 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_HOME\_REMEDY) |
| OTC\_ID1 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_OTC) |
| OTC\_ID2 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_OTC) |
| OTC\_ID3 | NUMBER(10) | FOREIGN KEY(FK\_REMEDY\_OTC) |

## Table Name: ATM\_SYMPTOM\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| SYMPTOM\_ID | NUMBER(10) | PRIMARY KEY(PK\_SYMPTOM) |
| SYMPTOM | VARCHAR2(255) | UNIQUE & NOT NULL |

## Table Name: ATM\_HISTORY\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| DATE | DATE | NOT NULL |
| USER\_ID | NUMBER(10) | FOREIGN KEY(FK\_HISTORY\_USER\_PROFILE) |
| REMEDY\_ID | NUMBER(10) | FOREIGN KEY(FK\_HISTORY\_REMEDY) |

## Table Name: ATM\_INSURANCE\_PRO\_HEALTHCARE\_CENTER\_JOIN\_TB

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| INSURANCE\_PROVIDER\_ID | NUMBER(10) | FOREIGN KEY(FK\_JOIN\_INSURANCE\_PROVIDER) |
| HEALTHCARE\_CENTER\_ID | NUMBER(10) | FOREIGN KEY(FK\_JOIN\_HEALTHCARE) |

## **Table Name: ATM\_OTC\_TB**

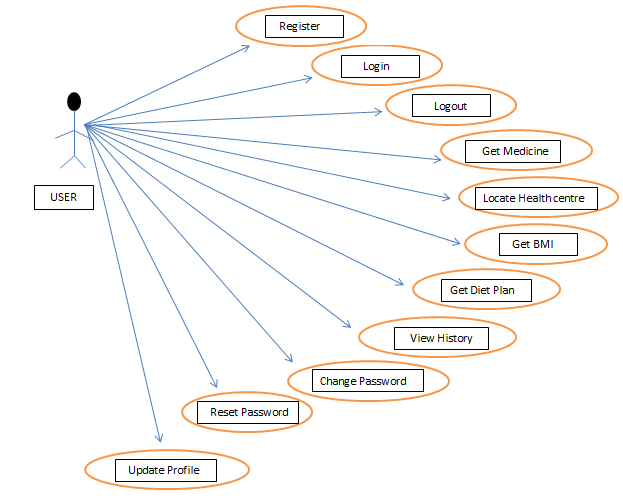
|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| OTC\_ID | NUMBER(10) | PRIMARY KEY(PK\_OTC) |
| OTC | VARCHAR2(255) | NOT NULL |

## **Table Name: ATM\_HOME\_REMEDY\_TB**

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATATYPE** | **CONSTRAINT** |
| HOME\_REMEDY\_ID | NUMBER(10) | PRIMARY KEY(PK\_HOME\_REMEDY) |
| HOME\_REMEDY | VARCHAR2(255) | NOT NULL |

# High Level Design

## Use case Diagram



## Sequence Diagram



### USR\_001: Register



### USR\_002: Login



### USR\_003: Logout



### USR\_004: Get Medicine



### USR\_005: Locate healthcare center:



### USR\_006 & USR\_007: Calculate BMI & Get Diet Plan



### USR\_008: View History



### USR\_009: Change Password:



### USR\_010: Reset Password:



### USR\_011: Update Profile:



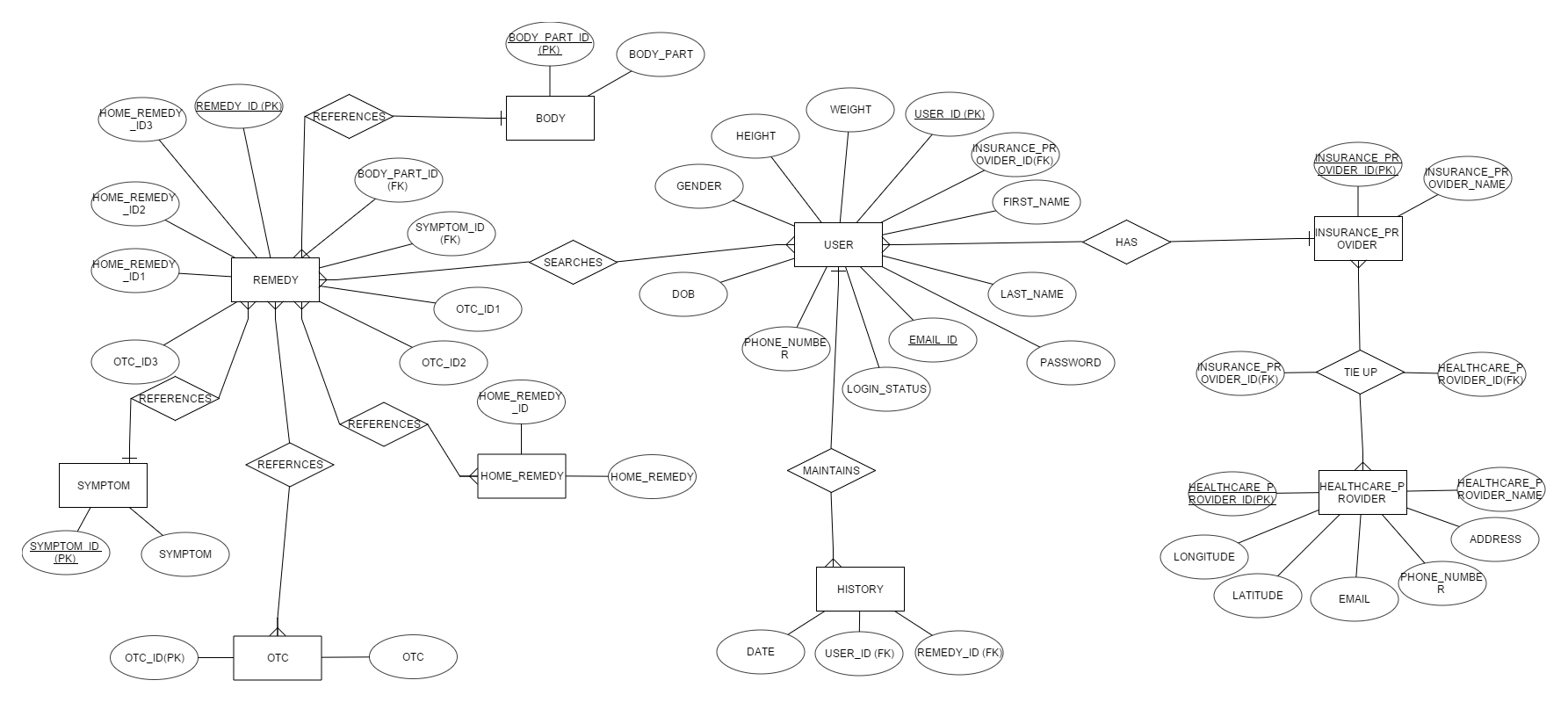
## ER DIAGRAM

Terms used in the ER diagram:

OTC: Over the Counter medicine

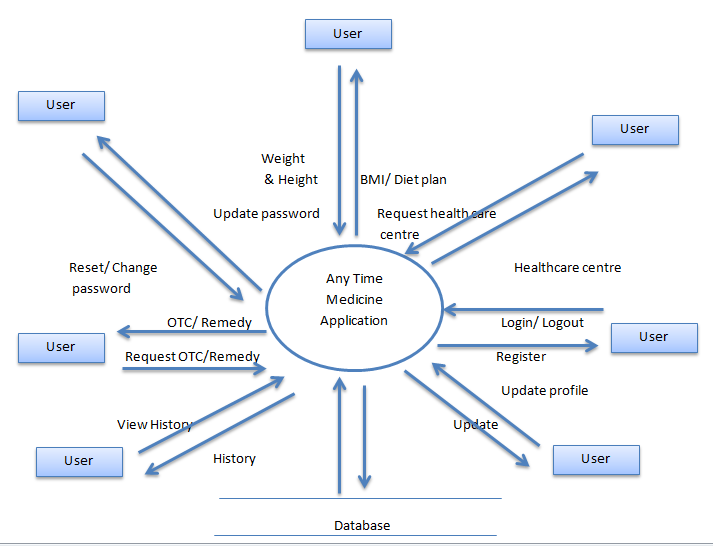
DOB- Date of Birth

Date- Date when the user searched for a remedy corresponding to the body part and symptom

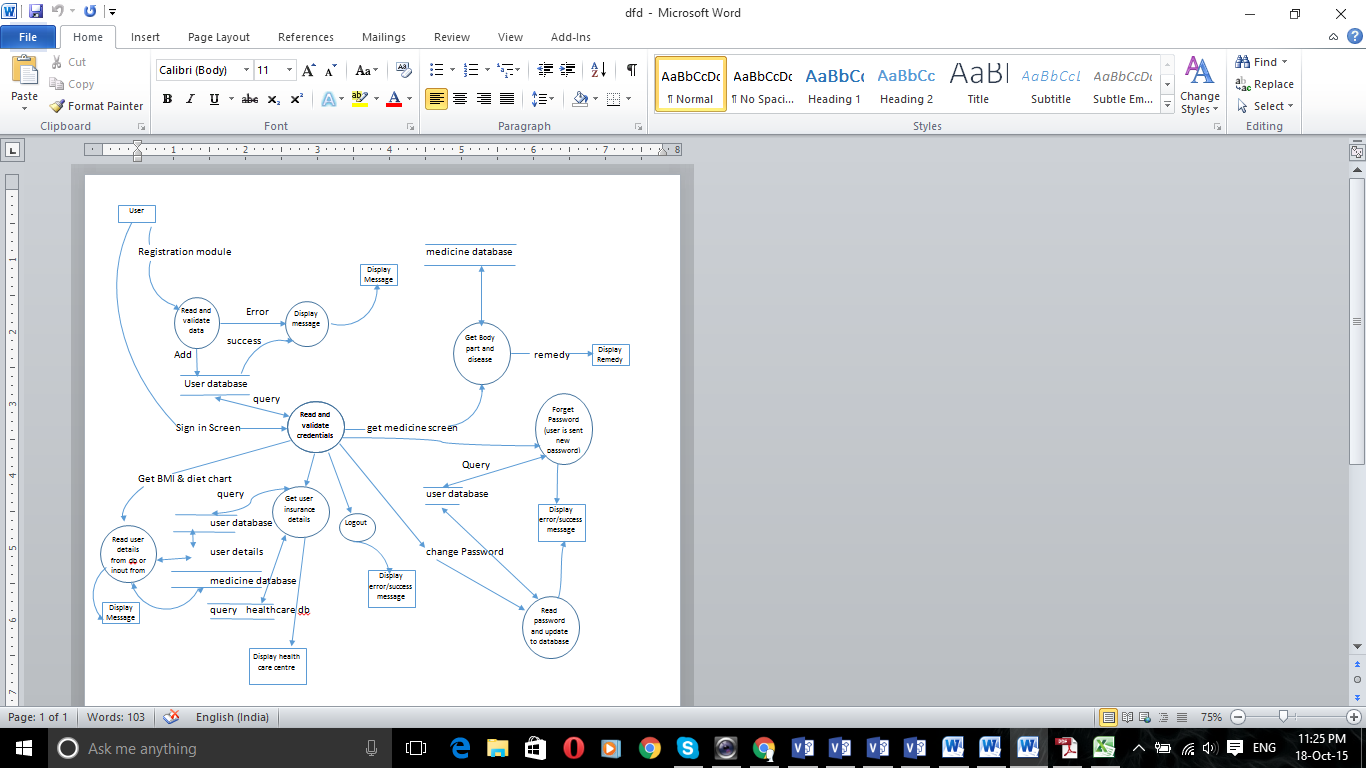


**Fig. 3.3.1 ER diagram**

## DATA FLOW DIAGRAM



**Fig. 3.4.1 Level-0 DFD**



**Fig. 3.4.2 Level-1 DFD**

# User Interface Design

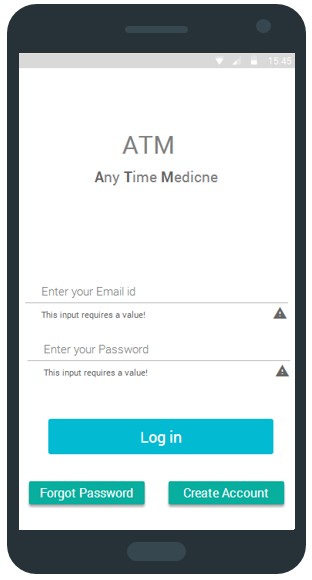
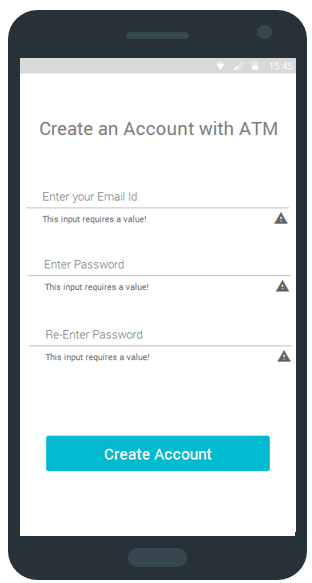
## USR\_001:

The user should be able to register with the application

# Design Details

# Wireframe Design

Log in Screen Create Account Screen

# Processing Details

Upon opening the application user is presented with the ‘Log in’ screen. If the user is not registered with the application then he can register with the application by simply clicking on the “Create Account” button. Upon clicking the button, user is redirected to the registration page where he can input his valid email id and password. Once the user has filled in the valid details, the “Create Account” button becomes available for clicking.

Once the user clicks on the “Create Account” button, the user information is stored on the database and a new entry for the user is created with the following fields:

|  |  |
| --- | --- |
| Field Name | Data Type |
| User\_id | Number |
| Email\_id | Varchar2 |
| Password | Varchar2 |

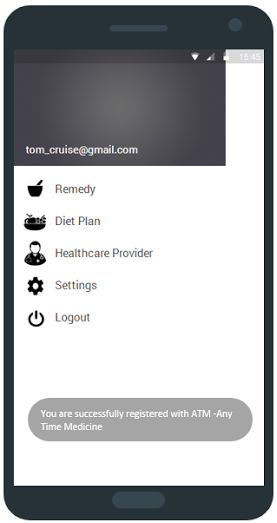
## 

# Error to be checked for:

We should check that no field is null. The “Create Account” button is enabled only when all the fields are filled.

# Output:

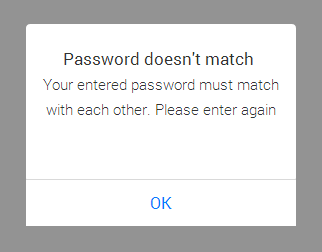
The user information is added to the database and is redirected to the homepage.

****

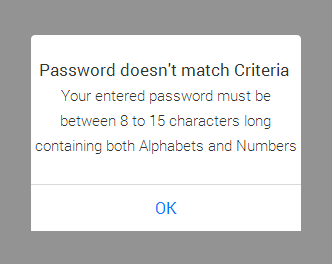
# Error Capturing

Try catch block for null values. Failure message will be displayed.

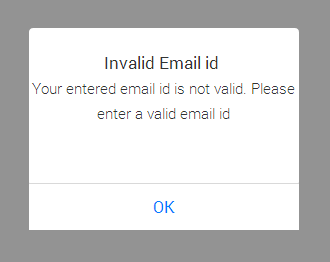
1. If upon re-entering the password, the password does not match with the previously entered password then the following error message is displayed:



1. If the password entered doesn’t match the criteria then the following error message is displayed:



1. If INVALID/ ALREADY REGISTERED Email id is entered then the following error message is displayed:



# Boundary and normal Operating condition

Email id length must not exceed 100 characters and Password length must be between 8 to 15 characters.

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android EditText and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User has downloaded the application

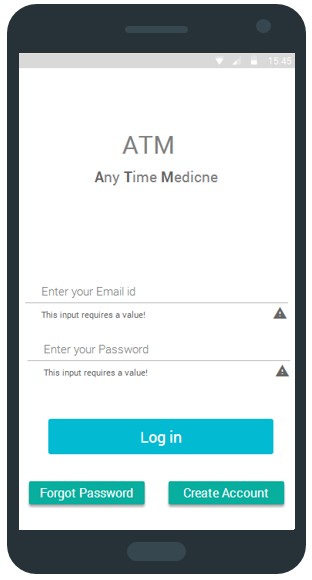
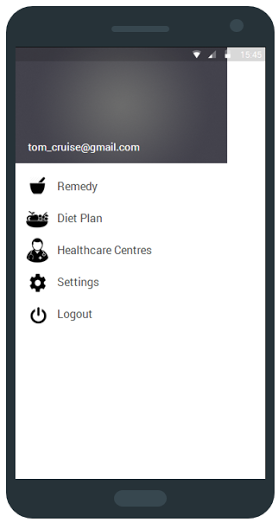
## USR\_002:

The user should be able to login to the application

# Design Details

# Wireframe Design

Log in Screen Homepage

# Processing Details

Upon opening the application the user is presented with the “log in” screen. Once the user enters valid email id and password the log in button is enabled for clicking.

On clicking the log in button a background service runs which connects with the remote database and validates the user’s email id and password.

If the validation is successful, the user is redirected to his/her homepage.

# Error to be checked for:

We should check that no field is null. The “Log in” button is enabled only when all the fields are filled.

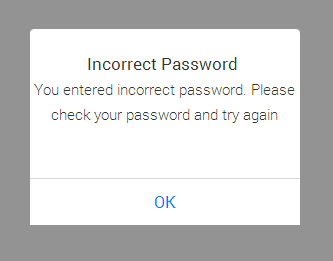
# Output:

System validates the details and redirects the user to the homepage.

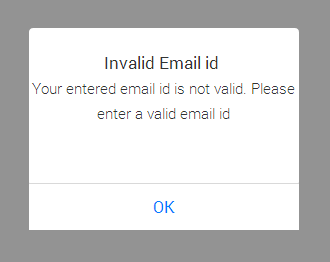
# Error Capturing

Try catch block for null values. Failure message will be displayed.

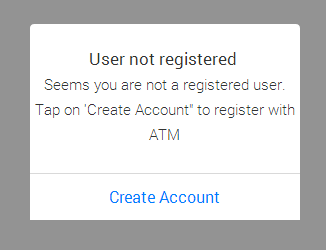
1. If invalid password is entered then the following error message is displayed:



1. If invalid email id is entered then the following error message is displayed :



1. If user tries to log in without registering with the application then the following error is displayed:



# Boundary and normal Operating condition

Email id length must not exceed 100 characters and Password length must be between 8 to 15 characters.

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android EditText and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User has downloaded the application

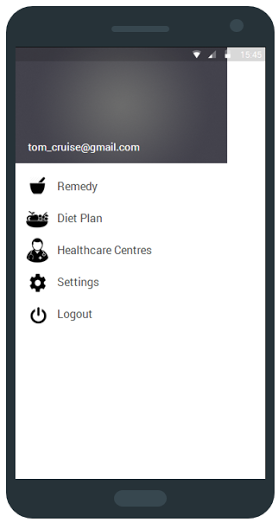
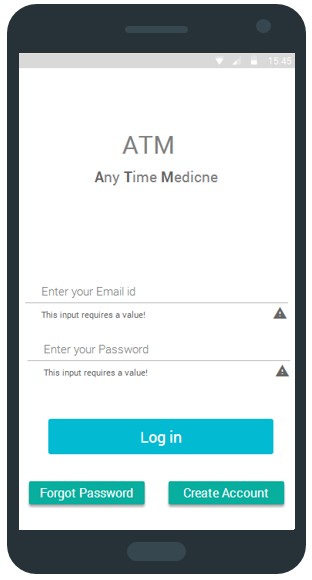
## USR\_003:

The user should be able to logout from the application

# Design Details

# Wireframe Design

Homepage Log in Screen

# Processing Details

Logout option is available on the homepage. Once the user clicks on the logout button, he is logged out and redirected to the login page.

# Error to be checked for:

N/A

# Output:

User is logged out and redirected to the login page

# Error Capturing

N/A

# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android drawer

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User has registered and logged into the application

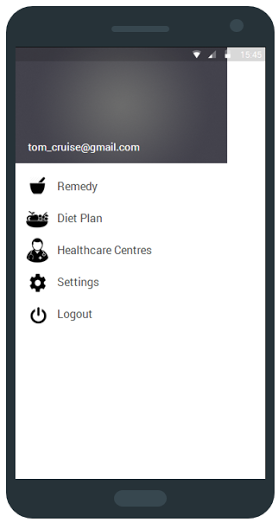
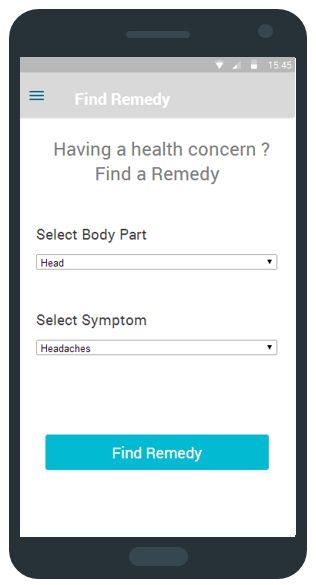
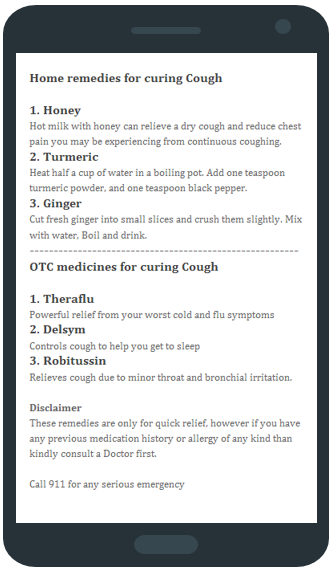
## USR\_004:

The user should be able to select a body part and problem associated with that body part and view medicines/remedies as per the selection

# Design Details

# Wireframe Design

Homepage Find Remedy Screen Remedy Screen

# Processing Details

When the user clicks on list item “Remedy” on the homepage, he is redirected to the “Find Remedy” screen where he can select a body part and the associated symptom from the drop down list.

Once the user clicks “Find Remedy” button a background service will run that connects with remote database and fetches the remedy details associated with the specific selection made. The data is parsed at the client end and presented to the user.

# Error to be checked for:

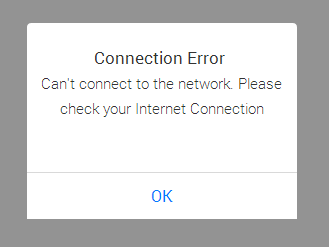
We should check for connection error.

# Output:

Based on the selections made by user, remedies are displayed.

# Error Capturing

If the system cannot connect to database due to connection error then the following error message is displayed:



# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_BODY\_TB | 2.5 |
| ATM\_REMEDY\_TB | 2.6 |
| ATM\_SYMPTOM\_TB | 2.7 |
| ATM\_OTC\_TB | 2.9 |
| ATM\_HOME\_REMEDY\_TB | 2.10 |

# Module specific data structure:

Android ListView and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User is logged into the application.

## USR\_005:

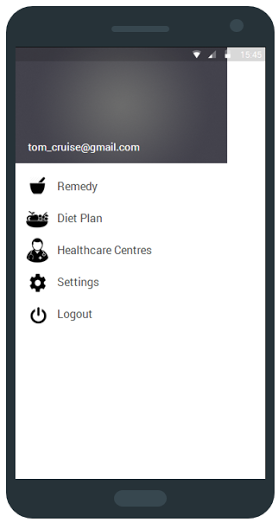
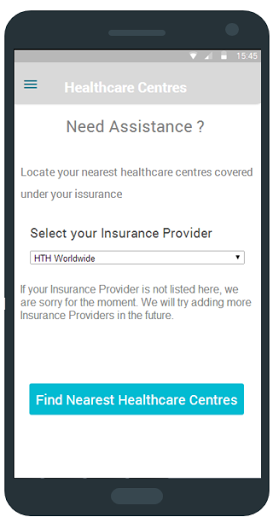
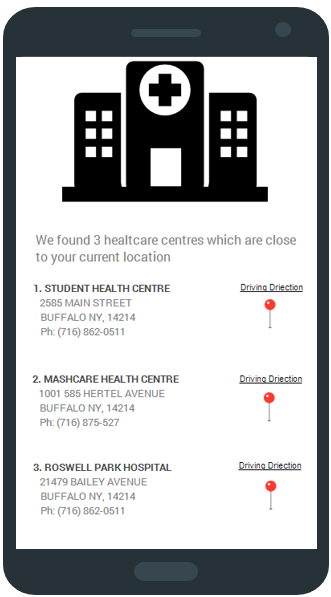
The user should be able to locate the nearest healthcare center covered under his insurance policy

# Design Details

# Wireframe Design

Homepage Search nearest Healthcare Provider Screen

Healthcare Center Screen

# Processing Details

When the user clicks on list item “Healthcare Provider”, he is redirected to the Healthcare Provider screen where by just providing his/her insurance provider user can search for the nearest healthcare center.

A background service runs which sends user’s current GPS location information to the remote database and fetches the nearest healthcare center based upon latitude and longitude. The received location information is parsed at client end and displayed to the user.

# Error to be checked for:

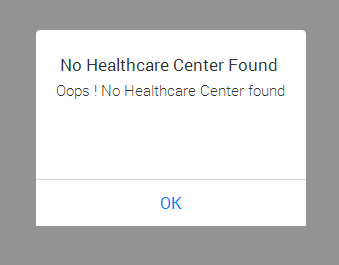
We should check for connection error. If user’s GPS cannot track his location due to any error then the nearest healthcare center information will not be fetched.

# Output:

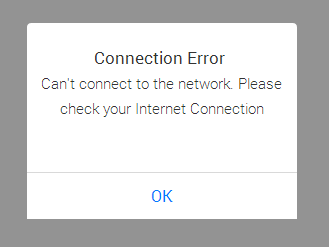
User is provided with the healthcare centers closest to his/her current location.

# Error Capturing

1. If no healthcare center is found nearest to the user’s location then the following error message is displayed



1. If the device cannot provide user’s current location due to lack of internet connection the following error message is displayed:



# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_HEALTHCARE\_CENTER\_TB | 2.3 |
| ATM\_INSURANCE\_PROVIDER\_TB | 2.4 |
| ATM\_INSURANCE\_PRO\_HEALTHCARE\_CENTER\_JOIN\_TB | 2.9 |

# Module specific data structure:

Android ListView and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User is logged into the application.

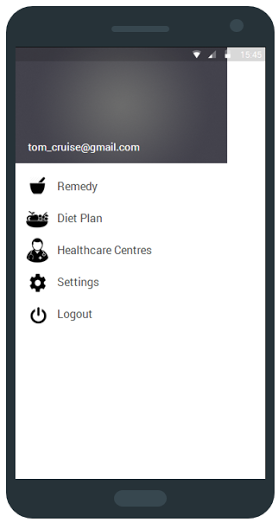
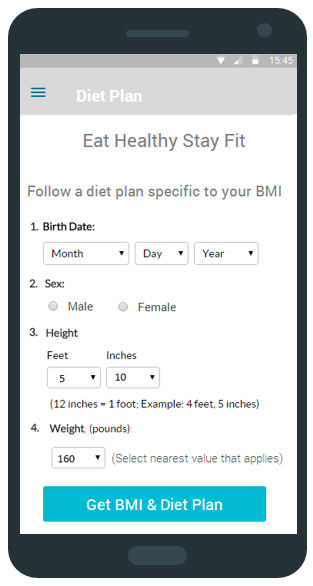
## USR\_006 & USR\_007:

The user should be able to get the diet plan as per his/her BMI (Body Mass Index)

# Design Details

# Wireframe Design

Homepage Diet Plan Screen Diet Plan

# Processing Details

When the user clicks on list item “Diet Plan” on the homepage, he is redirected to the Diet Plan screen where he can input his Date of Birth, Sex, Height and Weight. Once all the required details are filled, the “Get BMI & Diet Plan” button is enabled.

When the user clicks on the button, his/her BMI is calculated and a background service runs that connects with remote database and fetches the matching available diet plan based on user’s BMI. The received information is parsed at client end and displayed to the user.

# Error to be checked for:

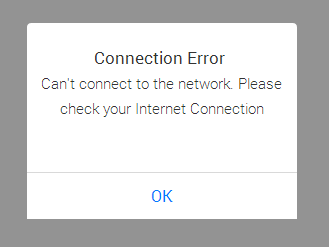
We should check for connection error.

# Output:

User is provided with the BMI and diet plan according to his/her BMI.

# Error Capturing

1. If the system cannot connect to database due to some connection error then following error message is displayed



# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android ListView and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User is logged into the application.

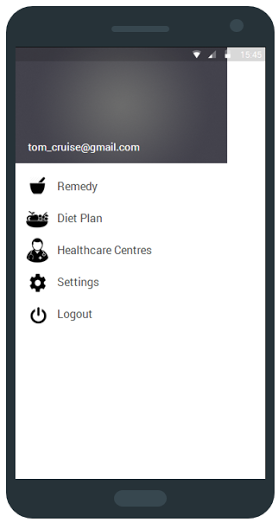
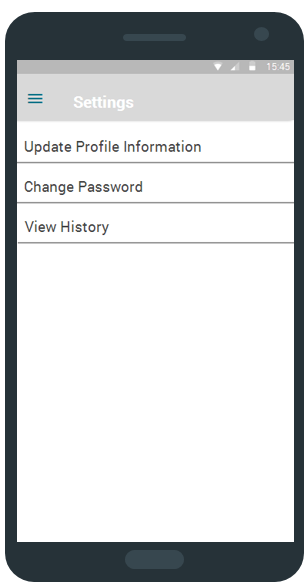
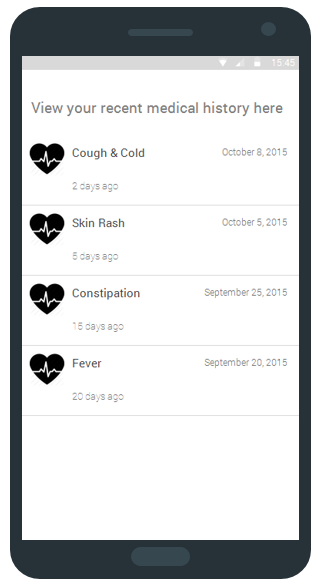
## USR\_008:

The user should be able to view his/her history of remedy searches

# Design Details

# Wireframe Design

Homepage Settings Screen Recent History Screen

# Processing Details

When the user clicks on list item “Settings”, he is redirected to the “Settings” screen where he can click on “View History” to view his recent history of searches.

Once the user click on “View History” a background service runs that connects with remote database and fetches user’s recent history of remedy searches. The received information will be parsed at client end and shown to the user.

# Error to be checked for:

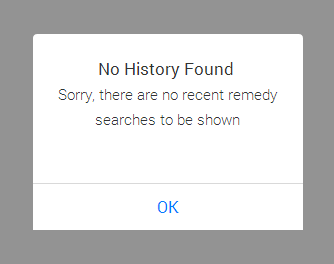
We should check for connection error.

# Output:

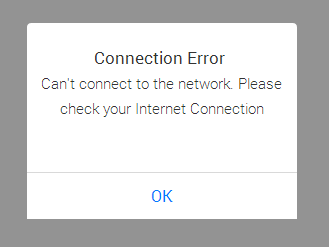
User is able to view his recent history of remedy searches.

# Error Capturing

1. If no recent history is found then the following error message is displayed:



2. If the system cannot connect to database due to some connection error then following error message is displayed



# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |
| ATM\_HISTORY\_TB | 2.8 |
| ATM\_REMEDY\_TB | 2.6 |

# Module specific data structure:

Android ListView and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User must be logged in to perform this task.

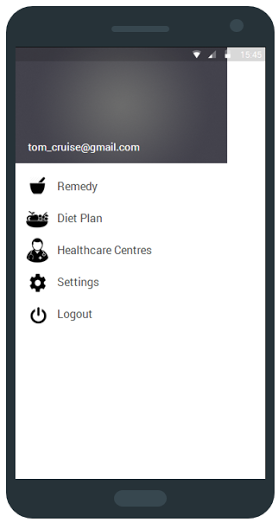
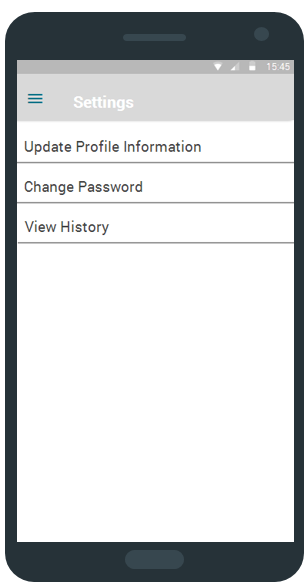
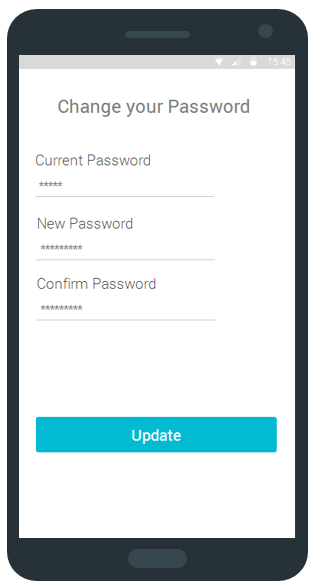
## USR\_009:

The user should be able to change his password

# Design Details

# Wireframe Design

Homepage Settings Screen Change Password Screen

# Processing Details

When the user clicks on list item “Settings” he is redirected to the “Settings” screen where he can click on “Change Password” to change his/her password. Once the user clicks on “Change Password”, he is redirected to change password page where he enters his current password and provides a new password.

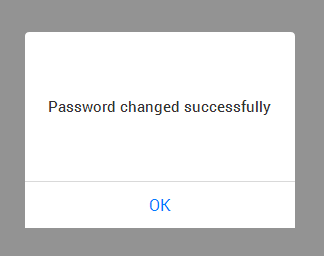
Once the user clicks on “Update” button a background service runs that connects with remote the database and validates the current password and updates the new password if validation is successful.

# Error to be checked for:

We should check for connection error.

# Output:

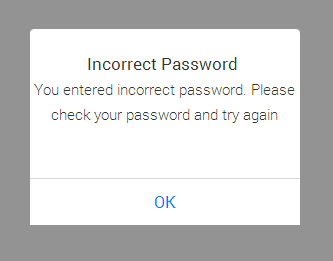
User is successfully able to change his/her password. Following success message is displayed:



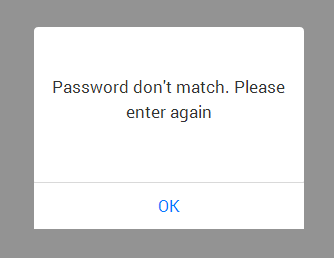
# Error Capturing

Try catch block for null values. Failure message will be displayed.

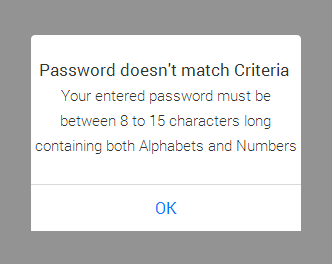
1. If INCORRECT “current password” is entered then the following error message is displayed:



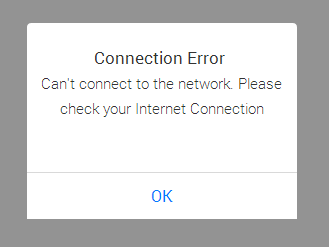
1. If upon re-entering the new password, the password does not match with the previously entered new password then the following error message is displayed:



1. If the new password doesn’t meet the password criteria then the following error message is displayed:



1. If the system cannot connect to database due to some connection error then the following error message is displayed:



# Boundary and normal Operating condition

Password length must be between 8 to 15 characters.

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android EditText and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User is logged into the application

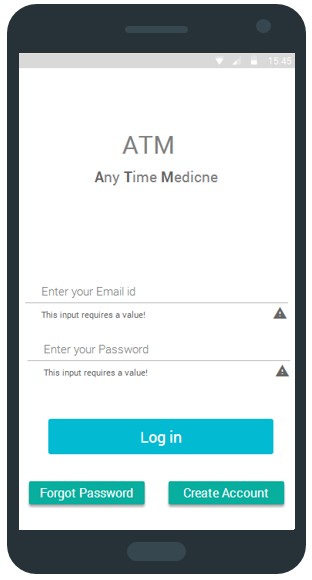
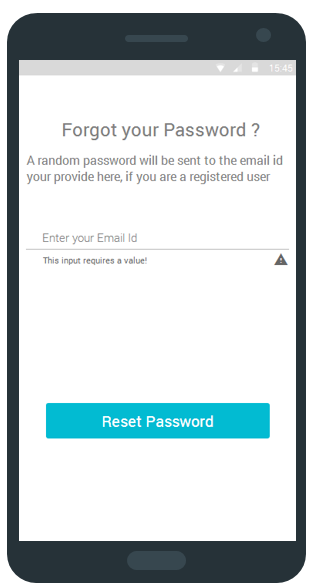
## USR\_010:

The user should be able to reset his password

# Design Details

# Wireframe Design

Log in Screen Forgot Password Screen

# Processing Details

When the user clicks on “Forgot Password” button, he is redirected to “Forgot Password” screen where he can enter his email id to reset the password. Once the user enters his/her valid email id, the “reset password” button is enabled.

As the user clicks on the button a background service runs that connects with remote database and sends a random password on his/her email id.

User is redirected to the log in Screen after email is sent to login again with the new password.

# Error to be checked for:

We should check for connection error.

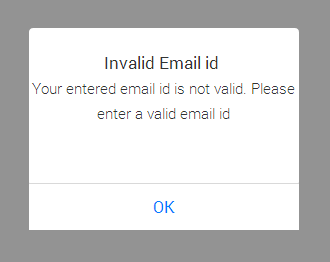
# Output:

User is able to login after receiving the random password sent on his/her email id.

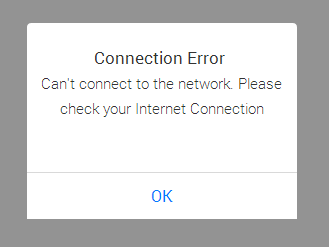
# Error Capturing

Try catch block for null values. Failure message will be displayed.

1. If email id entered is not same as the one used for registration then the following error message is displayed to the user



1. If the system cannot connect to database due to some connection error then the following error message is displayed:



# Boundary and normal Operating condition

Email id length must not exceed 100 characters.

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android EditText and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User knows his email id.

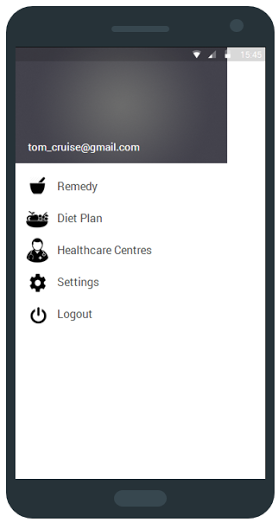
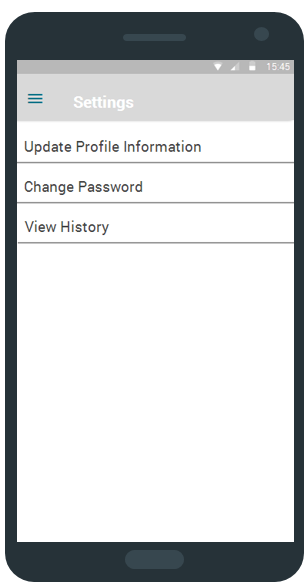
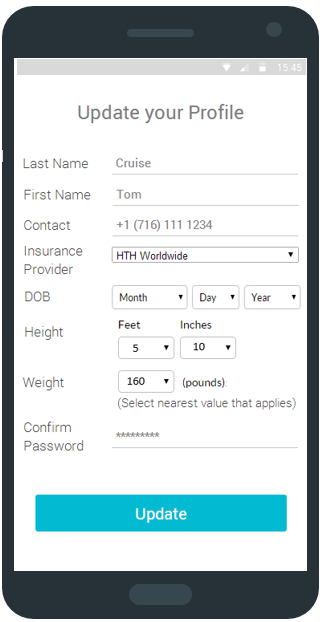
## USR\_011:

The user should be able update his profile information

# Design Details

# Wireframe Design

Homepage Settings Screen Update Profile Screen

# Processing Details

When user clicks on list item “Settings”, he is redirected to the “Settings Screen” where he has the option to select “Update Profile Information” and get redirected to the “Update Profile Screen”.

Once on the “Update Profile’’ screen user can update his details like Date of Birth, Height, Weight and Name. User needs to enter his/her password to update the information provided.

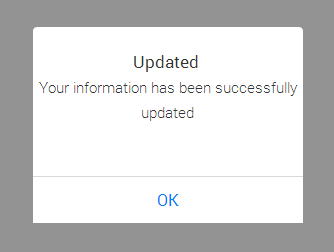
As user clicks on the “Update” button a background service runs that connects with remote database and updates user’s information.

# Error to be checked for:

We should check for connection error.

# Output:

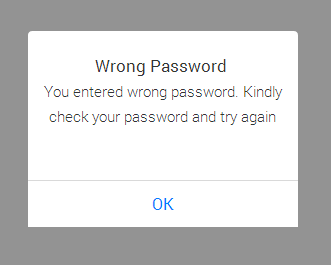
User is successfully able to update his/her information. A success message is displayed to the user:



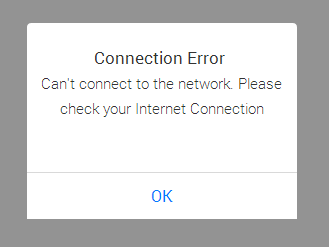
# Error Capturing

Try catch block for null values. Failure message will be displayed.

1. If incorrect password is entered then the following error message is displayed to the user:



1. If the system cannot connect to database due to some connection error then the following error message is displayed:



# Boundary and normal Operating condition

N/A

# Global data structure references:

|  |  |
| --- | --- |
| **TABLE\_NAME** | **SECTION** |
| ATM\_USER\_PROFILE\_TB | 2.1 |

# Module specific data structure:

Android EditText and Button

# External Interfaces:

Android – Remote Database connection

# Assumptions:

User is logged into the application.

# TRACEABILITY TO REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| **Document reference Id & Description: (Doc Id from which this document is derived)** | | |
| **Sl No.** | **Reference document**  **Requirement/Feature (Section ID/Name)** | **Design document**  **(Section ID/Name)** |
| 1 | USR\_001 | 4.1 |
| 2 | USR\_002 | 4.2 |
| 3 | USR\_003 | 4.3 |
| 4 | USR\_004 | 4.4 |
| 5 | USR\_005 | 4.5 |
| 6 | USR\_006 | 4.6 |
| 7 | USR\_007 | 4.6 |
| 8 | USR\_008 | 4.7 |
| 9 | USR\_009 | 4.8 |
| 10 | USR\_010 | 4.9 |
| 11 | USR\_011 | 4.10 |