CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The main aim of developing this application is to keep our loved once safe and always have eye on them, from all possible threads of today's world. This project help to keep track of child with the help of GPS technologies. The application uses two main services that is GPS and telephonic services. For location services is GPS , telephony services is to get live voice as per parents need and notify parents with strict HIGH- VOICE alarm if child leaves a specific geographical location which parent marked .Internet is used for communicating between children and parent side. The System can be designed in a simple way. The application developed to make user-friendly approach on both sides. The parents and children both should have GPS Based smart phones. The application is used to track the Child's location as well as to check if the child leaves a specific geographical location.

1.2 PROJECT SPECIFICATION

The proposed system is an android app in which user can buy monitor their child online. Also the customers can monitor multiple child users who installed the application on their devices .

The system includes 3 modules. They are:

1. Registration
2. Login
3. child
    - Get help if need
    - Send location

4.parent
- Get child location
- Limit child movements

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

Child monitoring is a complete solution for ensuring the safety of kids. Now a day the crimes against children are widely increasing. While a crime happens against the child we didn’t have any option to find the current location of our child. This system provides the facility to find the location of child simply with the help of an android mobile. Child monitoring system gives a facility to monitor the motion of a child .The system will provide some facilities such as find the current location of the child. This system will provide exact address of current location with great accuracy. Now a day many crimes are there against children. Each parent can monitor the movements of their children by using this android application. In this paper, I use GPS system to trace the location of the child. There are mainly two modules and a database is there for my system. One is parent module which can be an android mobile phone with the application in it. The child module can be an android mobile. The database is used to monitor the location of the child. Which can be connected to monitoring the movements of children with in an area.

A System that help to locate the child in real time basis is require for ensuring the safety of kids while they are not with the parents. Because now a day’s crime against kids are increasing very badly. It’s very important to monitor the movements of kids while they are going to school. The kids those who are attending the school need a security system that can be monitored by the parents from there place itself. While a crime against kid happens, like kidnapping I did not have a provision to know their situation. We need a system that also help in these types of situations too. The attendance monitoring of kids became a great issue for their parents. The attendance monitoring can be done only by contacting the authorities in every day. It is not possible to do so. An efficient system is required to monitor the attendance even without the help of any authority.

2.2 EXISTING SYSTEM

There are several systems that are used for child tracking. All that system used to track the kid separately or in a group. Those systems are mainly used by the school authority to trace the child groups. That means to track the school bus or system that help to track a group their child's. The control over the system will belongs to the school authority. Some of the existing will help the parents to trace the child. Parent can view the location of the child.

DRAWBACKS OF EXISTING SYSTEM

- Low security.
- Limited range.
- Parents have no control over the system.

2.3 PROPOSED SYSTEM

The proposed system is defined to meets all the disadvantages of the existing system. It is necessary to have a system that is more user friendly and user attractive for growth of service center; on such consideration the system is proposed. In our proposed system there is admin who can view all the customers. It allows customers to meet their service .Users of this proposed system are admin and customer. The software application which avoids more manual hours that need to spend in record keeping and generating reports. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the distributors to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance. It is very easy to record the information of online into databases.

2.4 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources. It has got following features:

- **Better security: -**

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

- **Ensure data accuracy: -**

The proposed system eliminates the manual errors while entering the details of the users during the registration.

- **Better service: -**

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility analysis is the procedure for identifying the candidate system, evaluating and electing the most feasible system. This is done by investigating the existing system in the area under investigation or generally ideas about a new system. It is a test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources. The following are its features: -

3.1.1 Economical Feasibility

Economic feasibility study presents tangible and intangible benefits from the project by com- paring the development and operational cost. The technique of cost benefit analysis is used as a basis of attaining economic feasibility.The proposed system done in java web and android we don’t make use of any external devices only the internet service is needed for the system. So the proposed system is economically feasible.

3.1.2 Technical Feasibility

It was found that we can easily done bank payments using our mobile phones by high security. This system can support the later versions of Android operating system. So the proposed system is technically feasible.

3.1.3 Behavioral Feasibility

One of the main problems faced during the development of the new system is getting the acceptance from the user. People are inherently resistant to change and so estimate should be made of how strong a reaction the user is likely to have towards the developing system. The system is much user friendly and the maintenance and working needs much less human effort.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel core i5
RAM	-	8 GB
Hard disk	-	1 TB

3.2.2 Software Specification

Front End	-	Android Studio IDE
Backend	-	FIREBASE
Operating System	-	Windows 8 , Android 5 lolipop
Technologies	-	java,xml

3.3 SOFTWARE DESCRIPTION

3.3.1 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

3.3.2 Firebase

A Firebase project is the top-level entity for Firebase. In a project, you create Firebase apps by registering your Apple, Android, or web apps. After you register your apps with Firebase, you can add the Firebase SDKs for any number of Firebase products, like Analytics, Cloud Firestore, Performance Monitoring, or Remote Config.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term "design" is defined as "the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization". It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.

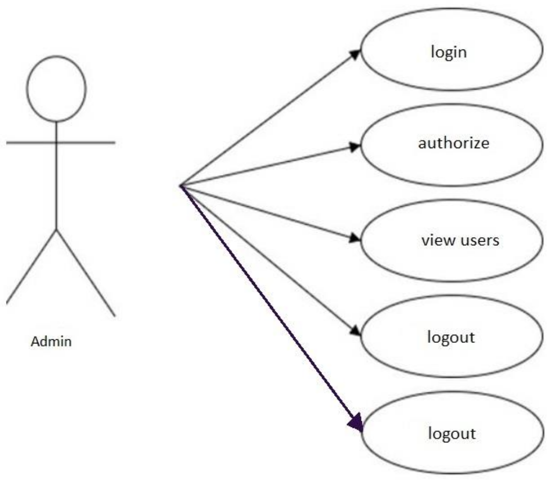
System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

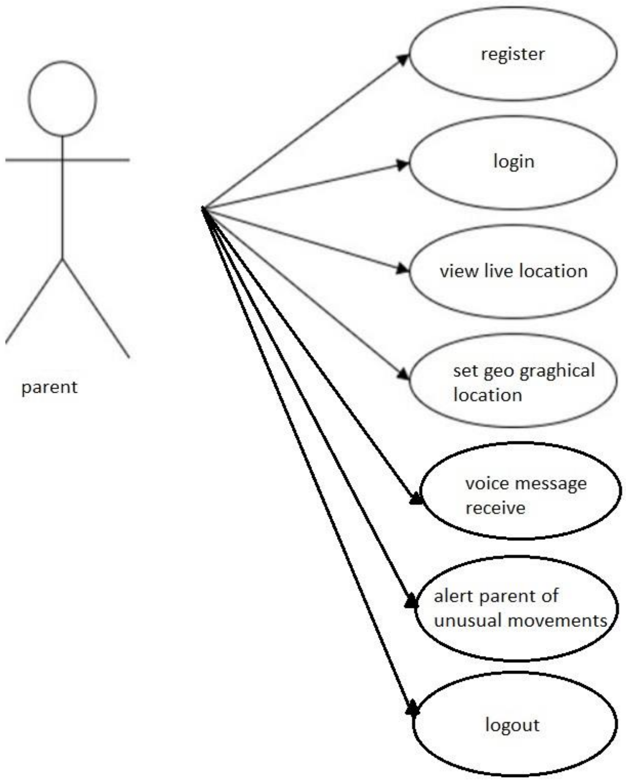
- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.

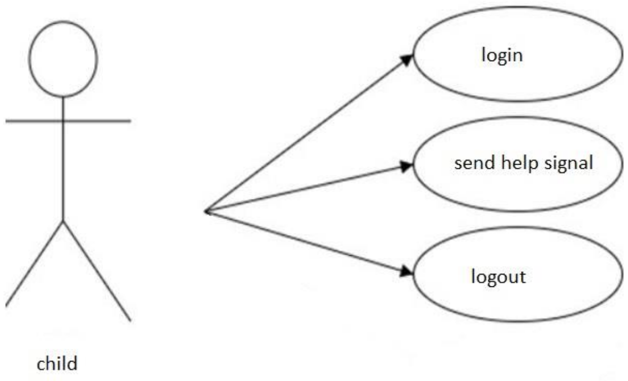
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.


Admin
login
authorize
view users
logout
logout


parent
register
login
view live location
set geo graghical location
voice message receive
alert parent of unusual movements
logout


child
login
send help signal
logout

4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

ii. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram

iii. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

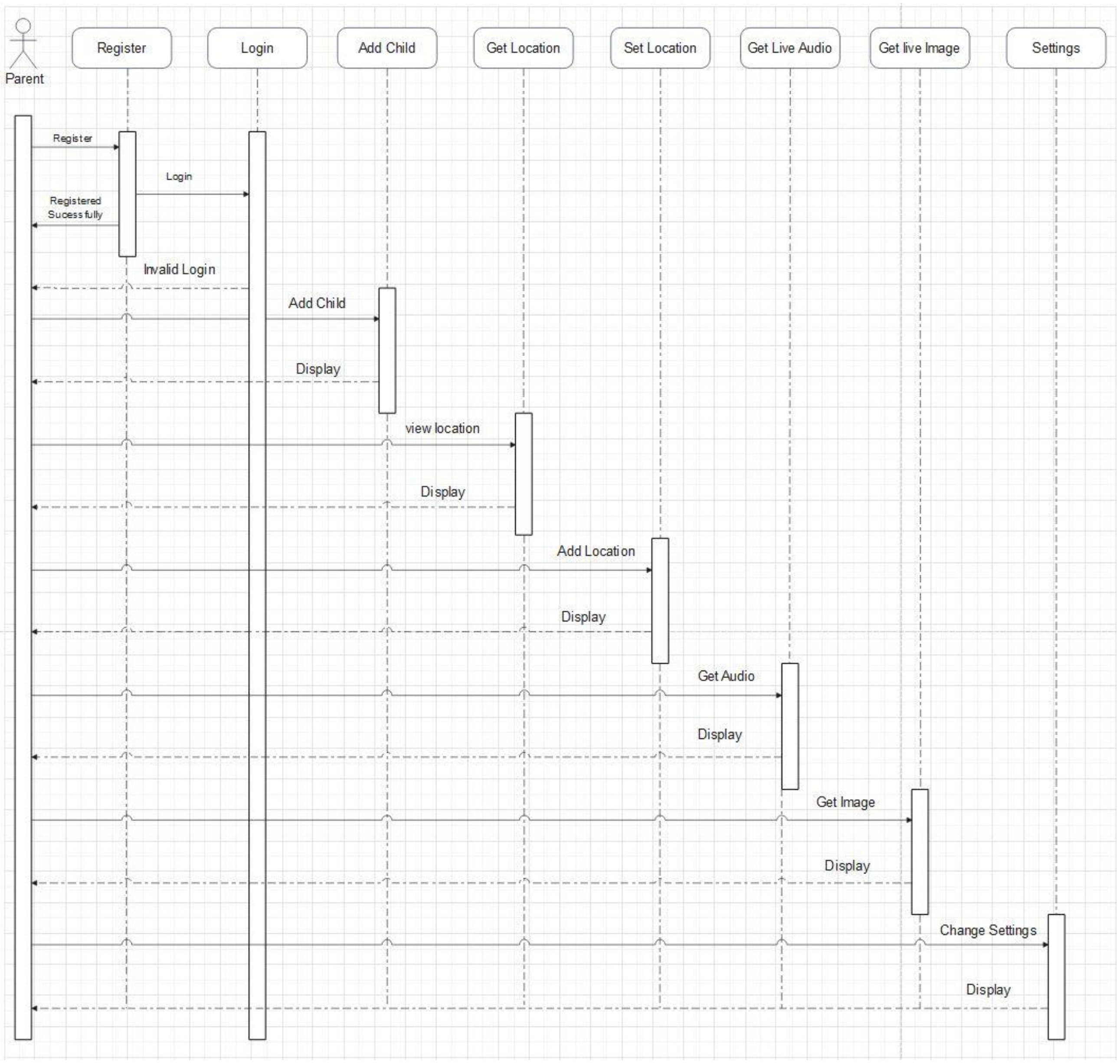
- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message

- Lost Message

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

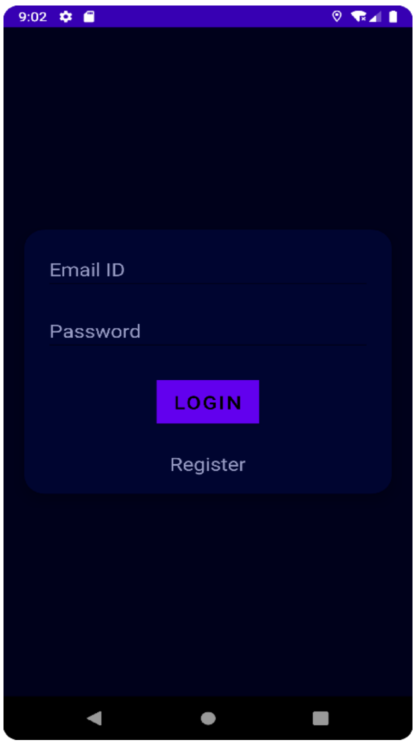
- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.


Parent
Register
Login
Add Child
Get Location
Set Location
Get Live Audio
Get live Image
Settings
Register
Login
Registered Sucess fully
Invalid Login
Add Child
Display
view location
Display
Add Location
Display
Get Audio
Display
Get Image
Display
Change Settings
Display

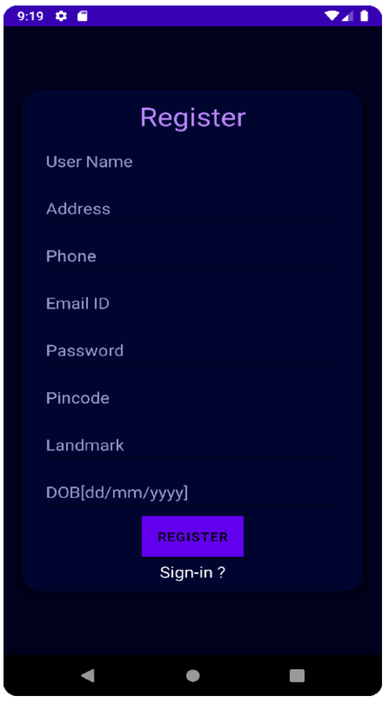
4.5 USER INTERFACE DESIGN

4.5.1 parent

User Login

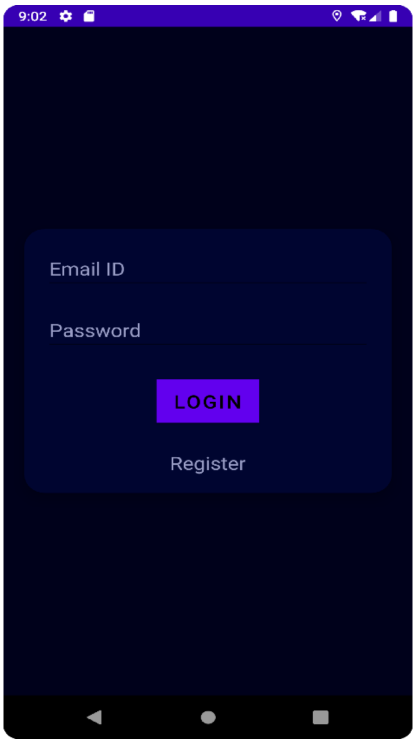


User Registration

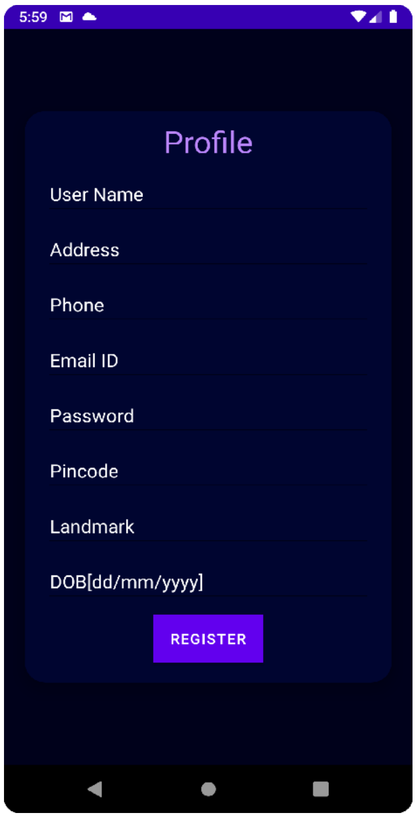


4.5.2 child

User Login



User Registration



4.6. DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

4.6.1 Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values.

Every value in a relation is atomic, that is not decomposable.

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.

4.6.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be donor by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

TABLE DESIGN

Table 1 : User_Parent

Field_name	**Type**	**Width**	**Constrains**
P_User_ID	Varchar	20	Primary key
Name	Varchar	30	Not null
Phone_no	Integer	15	Not null
Password	Varchar	30	Not null
Email	Varchar	30	Not null
Pincode	Integer	15	Not null
Date Of Birth	Varchar	5	Not null
Landmark	Varchar	50	Not null
Photo	File		Not null
Id Proof_F	File		Not null
Id Proof_B	File		Not null

Table 2 : User_Admin-user

Field_name	**Type**	**Width**	**Constrains**
AU_User_ID	Varchar	20	Primary key
Name	Varchar	30	Not null
Phone_no	Integer	15	Not null
Password	Varchar	30	Not null
Email	Varchar	30	Not null
Pincode	Integer	15	Not null
Date Of Birth	Varchar	5	Not null
Landmark	Varchar	50	Not null
Photo	File		Not null
Id Proof_F	File		Not null
Id Proof_B	File		Not null

Table 3 : User_Child

Field_name	**Type**	**Width**	**Constrains**
C_User_ID	Varchar	20	Primary key
Name	Varchar	30	Not null
Phone_no	Integer	15	Not null
Email	Varchar	30	Not null
Gender	Integer	15	Not null
Date Of Birth	Varchar	5	Not null
Photo	File		Not null
Id Proof_F	File		Not null
Id Proof_B	File		Not null

Table 4 : Relation_id

Field name	Type	Width	Constrains
Relation ID	Varchar	15	Primary Key
P_User_ID	Varchar	20	Not null
C_User_ID	Varchar	20	Not null

Table 5 : Location set

Field name	Type	Width	Constrains
L_User_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
longitude	Float	10	Not null
Latitude	Float	10	Not null
Boundary	Int	10	Not null

Table 6 : Live location

Field name	Type	Width	Constrains
LL_User_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
Longitude	Float	10	Not null
Latitude	Float	10	Not null

Table 7 : Alert ,Audio ,Image log

Field name	Type	Width	Constrains
AAI_User_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
Alert log*	Boolean		Not null
Uninstall Log*	Boolean		Not null
Audio log*	Boolean		Not null
Image log*	Boolean		Not null
Live Monitor Log*	Boolean		Not null
Last Seen Log*	Boolean		Not null

Table 8 : Audio

Field name	Type	Width	Constrains
A_User_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
Audio	file		Not null
A_Timestamp	Timestamp	15	Not null

Table 9 : Image

Field name	Type	Width	Constrains
I_User_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
Audio	file		Not null
I_Timestamp	Timestamp	15	Not null

Table 10 : Email

Field name	Type	Width	Constrains
E_ID	Varchar	20	Primary key
Relation id	Varchar	20	Not null
Email	Varchar	30	Not null
Password	Varchar	20	Not null
Type	varchar	5 [P,C,A]	Not null
Timestamp	date		Not null
Validity log	varchar	5 [T,GP,RP,DP]	Not null

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code where removed and ensured that all modules are working, and gives the expected result.

5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of theirrole in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The current system working technology is old fashioned and there is no scope for parents to directly monitor their child’s situation. The proposed system introduces facility for customer to keep aware of their situation always

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- https://android-arsenal.com/
- https://firebase.google.com/docs
- https://developer.android.com/guide

CHAPTER 9

APPENDIX

9.1 Sample Code

9.1.1 login

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".login"
    android:orientation="vertical"
    android:background="@color/Blackblue"
    android:gravity="center">

    <androidx.cardview.widget.CardView
        android:backgroundTint="@color/Blackbluelite"
        app:cardElevation="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        app:cardCornerRadius="20dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <EditText
                android:id="@+id/email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textEmailAddress"
                android:textSize="20dp"
                android:hint="Email ID "
                android:textColorHint="@color/grey"
                android:textColor="@color/grey"
                tools:ignore="Autofill,HardcodedText,SpUsage"
                android:layout_marginHorizontal="20dp"
                android:layout_marginTop="20dp"
                />

            <EditText
                android:id="@+id/password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textPassword"
                android:textSize="20dp"
                android:textColorHint="@color/grey"
                android:textColor="@color/grey"
                android:hint="Password"
                android:layout_marginTop="20dp"
                tools:ignore="Autofill,HardcodedText,SpUsage"
                android:layout_marginHorizontal="20dp" />

            <Button
                android:id="@+id/Login"
                android:layout_gravity="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Login"
                tools:ignore="HardcodedText,SpUsage"
                android:textSize="20dp"
                android:textColor="@color/black"
                android:background="@color/Blackbluelite"
                android:layout_marginVertical="30dp" />
            <TextView
                android:id="@+id/create"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Register ?"
                android:textColorHint="@color/grey"
```

```
            android:textColor="@color/white"
            tools:ignore="HardcodedText,SpUsage"
            android:layout_gravity="center"
            android:textSize="20dp"
            android:layout_marginBottom="20dp"
            />


      </LinearLayout>

    </androidx.cardview.widget.CardView>

      </LinearLayout>
```

Java code :

```
package com.example.myproject;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class login extends AppCompatActivity {

    EditText mail,passwd ;
    Button Login;
    String inmail, inpasswd;
    TextView Create;

    //firebase connection
    FirebaseDatabase database;
    FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);



        mail = findViewById(R.id.email);
        passwd = findViewById(R.id.password);
        Login = findViewById(R.id.Login);
        Create =findViewById(R.id.create);
        //binding

        //firebase connection
        auth = FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance();

        getSupportActionBar().hide();

        if(auth.getCurrentUser() != null)
        {
            startActivity(new Intent(login.this,navigatecenter.class));
            finish();
        }

        Create.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(login.this,Register.class));
                finish();
            }
        });

        Login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                inmail = mail.getText().toString().trim();
                inpasswd = passwd.getText().toString().trim();



                if (!validateEmail(mail, getString(R.string.lbl_null_value),
        getString(R.string.lbl_invalid_value)))
                {
                    Toast.makeText(login.this, "Registerion failed 2",
        Toast.LENGTH_SHORT).show();
                } else if (TextUtils.isEmpty(inpasswd) || inpasswd.length() < 6)
                {
                    passwd.setError("password should contain 6 or more character");
                    Toast.makeText(login.this, "Registerion failed 3",
        Toast.LENGTH_SHORT).show();
                }
                else {

        auth.signInWithEmailAndPassword(inmail,inpasswd).addOnCompleteListener(new
        OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if(task.isSuccessful()){
                                Toast.makeText(login.this, "SignIn Sucessfull",
        Toast.LENGTH_SHORT).show();
                                //for id
                                String id = task.getResult().getUser().getUid();
                                //shared preference
                                SharedPreferences sharedPreferences =
        getSharedPreferences("mineid",MODE_PRIVATE);
                                SharedPreferences.Editor mine = sharedPreferences.edit();
                                mine.putString("userid",id);
                                mine.apply();

                                startActivity(new Intent(getApplicationContext(),navigatecenter.class));
                                finish();
                            }
                            else
                            {
                                Toast.makeText(login.this, "password/email is invalid",
        Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                }


            }
        });

       }

       private boolean validateEmail(EditText p_editText, String p_nullMsg, String p_invalidMsg)
        {
        boolean m_isValid = false;
        try {
            if (p_editText != null) {
                if (validateForNull(p_editText, p_nullMsg)) {
                    Pattern m_pattern = Pattern.compile("([\\w\\-]([\\.\\w])+[\\w]+@([\\w\\-]+\\.)+[A-
        Za-z]{2,4})");
                    Matcher m_matcher = m_pattern.matcher(p_editText.getText().toString().trim());
                    if (!m_matcher.matches() & p_editText.getText().toString().trim().length() > 0) {
                        m_isValid = false;
                        p_editText.setError(p_invalidMsg);
                    } else {
                        m_isValid = true;
                    }
                } else {
                    m_isValid = false;
                }
            } else {
                m_isValid = false;
            }
        } catch (Throwable p_e) {
            p_e.printStackTrace();
        }
        return m_isValid;
       }

       private boolean validateForNull(EditText p_editText, String p_nullMsg) {
        boolean m_isValid = false;
        try {
            if (p_editText != null && p_nullMsg != null) {
                if (TextUtils.isEmpty(p_editText.getText().toString().trim())) {
                    p_editText.setError(p_nullMsg);
                    m_isValid = false;
                } else {
                    m_isValid = true;
                }
            }
        } catch (Throwable p_e) {
            p_e.printStackTrace();
        }
        return m_isValid;
       }

        }
```

9.1.2 Register

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
tools:context=".Register"
android:orientation="vertical"
android:background="@color/Blackblue"
android:gravity="center">

<androidx.cardview.widget.CardView
    app:cardElevation="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    app:cardCornerRadius="20dp"
    android:backgroundTint="@color/Blackbluelite">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Register"
            android:layout_gravity="center"
            android:textColor="@color/purple_200"
            android:textSize="30dp"
            android:layout_marginTop="10dp"/>
        <EditText
            android:id="@+id/Username"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="User Name"
            android:inputType="text"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage"
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"/>

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Address"
            android:hint="Address"
            android:inputType="textPostalAddress"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Phone"
            android:hint="Phone"
            android:inputType="phone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="Email ID "
            tools:ignore="Autofill,HardcodedText,SpUsage"
            android:layout_marginHorizontal="20dp"
            android:layout_marginTop="10dp" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Password"
            android:layout_marginTop="10dp"
            tools:ignore="Autofill,HardcodedText,SpUsage"
            android:layout_marginHorizontal="20dp" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Pincode"
            android:hint="Pincode"
            android:inputType="number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Landmark"
            android:hint="Landmark"
            android:inputType="text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage" />

        <EditText
            android:textColorHint="@color/grey"
            android:textColor="@color/grey"
            android:textSize="18dp"
            android:id="@+id/Dateofbirth"
            android:hint="DOB[dd/mm/yyyy]"
            android:inputType="date"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginHorizontal="20dp"
            tools:ignore="HardcodedText,SpUsage" />



        <Button
            android:id="@+id/Register"
            android:layout_gravity="center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Register"
            tools:ignore="HardcodedText,SpUsage"
            android:textColor="@color/black"
            android:background="@color/Blackbluelite"
            android:layout_margin="5dp" />

        <TextView
            android:textColorHint="@color/white"
            android:textColor="@color/white"
            android:textSize="18dp"
            android:id="@+id/signup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Sign-in ?"
            tools:ignore="HardcodedText,SpUsage"
```

```
            android:layout_gravity="center"
            android:layout_marginBottom="10dp"
            />

        </LinearLayout>

    </androidx.cardview.widget.CardView>

</LinearLayout>
```

Java Code :

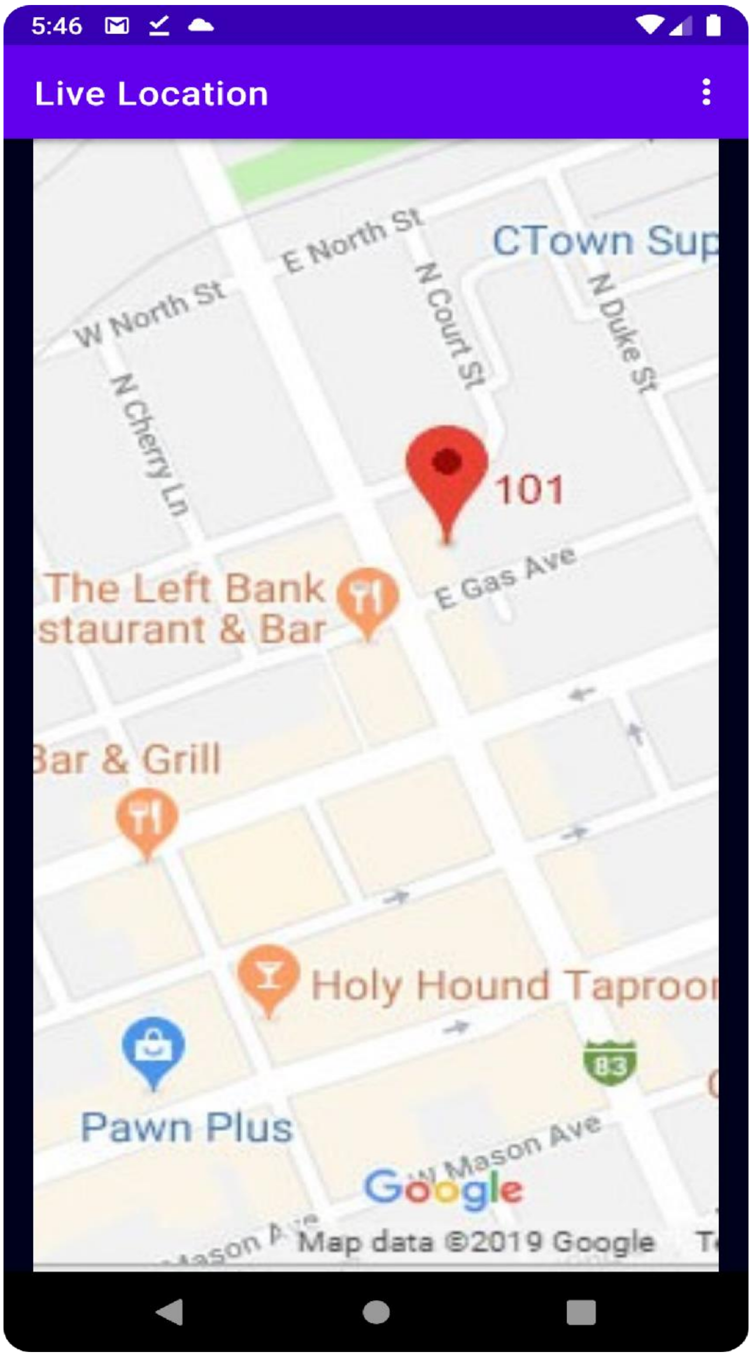
```
package com.example.myproject;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.example.myproject.models.users;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Register extends AppCompatActivity {

    EditText username,address,phone,mail,passwd,pincode,landmark,dateofbirth;
    Button Register;
```

```
TextView Signup;
String inusername,inaddress,inphone,inmail,inpasswd,inpincode,inlandmark,indateofbirth;

FirebaseAuth auth;
FirebaseDatabase database;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    username = findViewById(R.id.Username);
    mail = findViewById(R.id.Email);
    passwd = findViewById(R.id.Password);
    address = findViewById(R.id.Address);
    phone = findViewById(R.id.Phone);
    pincode = findViewById(R.id.Pincode);
    landmark = findViewById(R.id.Landmark);
    dateofbirth = findViewById(R.id.Dateofbirth);

    //buttons
    Register = findViewById(R.id.Register);
    Signup = findViewById(R.id.signup);

    auth = FirebaseAuth.getInstance();
    database = FirebaseDatabase.getInstance();

    getSupportActionBar().hide();

    Signup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(Register.this,login.class));
        }
    });
    Register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            inusername = username.getText().toString().trim();
            inmail = mail.getText().toString().trim();
            inpasswd = passwd.getText().toString().trim();
            inaddress = address.getText().toString().trim();
            inphone = phone.getText().toString().trim();
            inpincode = pincode.getText().toString().trim();
            inlandmark = landmark.getText().toString().trim();
            indateofbirth = dateofbirth.getText().toString().trim();

            Toast.makeText(Register.this, "user"+inusername+" mail"+inmail+"
    password"+inpasswd+" address"+inaddress+" phone"+inphone+" pin"+inpincode+"
    landmark"+inlandmark+" dateofbirth"+indateofbirth, Toast.LENGTH_SHORT).show();

            if (TextUtils.isEmpty(inusername) || inusername.length() < 3)
            { username.setError("username must be of 3 or more character");
                Toast.makeText(Register.this, "Registerion failed username",
    Toast.LENGTH_SHORT).show();
            }
            else if (TextUtils.isEmpty(inaddress) || inaddress.length() < 10) {
                address.setError("Enter a valid address");
                Toast.makeText(Register.this, "Registerion failed address",
    Toast.LENGTH_SHORT).show();
            }
            else if (TextUtils.isEmpty(inphone) || inphone.length() < 10) {
                phone.setError("Enter a valid phone number");
                Toast.makeText(Register.this, "Registerion failed phone",
    Toast.LENGTH_SHORT).show();
            }
            else if (!validateEmail(mail, getString(R.string.lbl_null_value),
    getString(R.string.lbl_invalid_value))) {
                Toast.makeText(Register.this, "Registerion failed mail",
    Toast.LENGTH_SHORT).show();
            }
            else if (TextUtils.isEmpty(inpasswd) || inpasswd.length() < 6) {
                passwd.setError("password should contain 6 or more character");
                Toast.makeText(Register.this, "Registerion failed password",
    Toast.LENGTH_SHORT).show();
            }else if (TextUtils.isEmpty(inpincode) || inpincode.length() < 6) {
                pincode.setError("enter a valid pincode");
                Toast.makeText(Register.this, "Registerion failed pincode",
    Toast.LENGTH_SHORT).show();
```

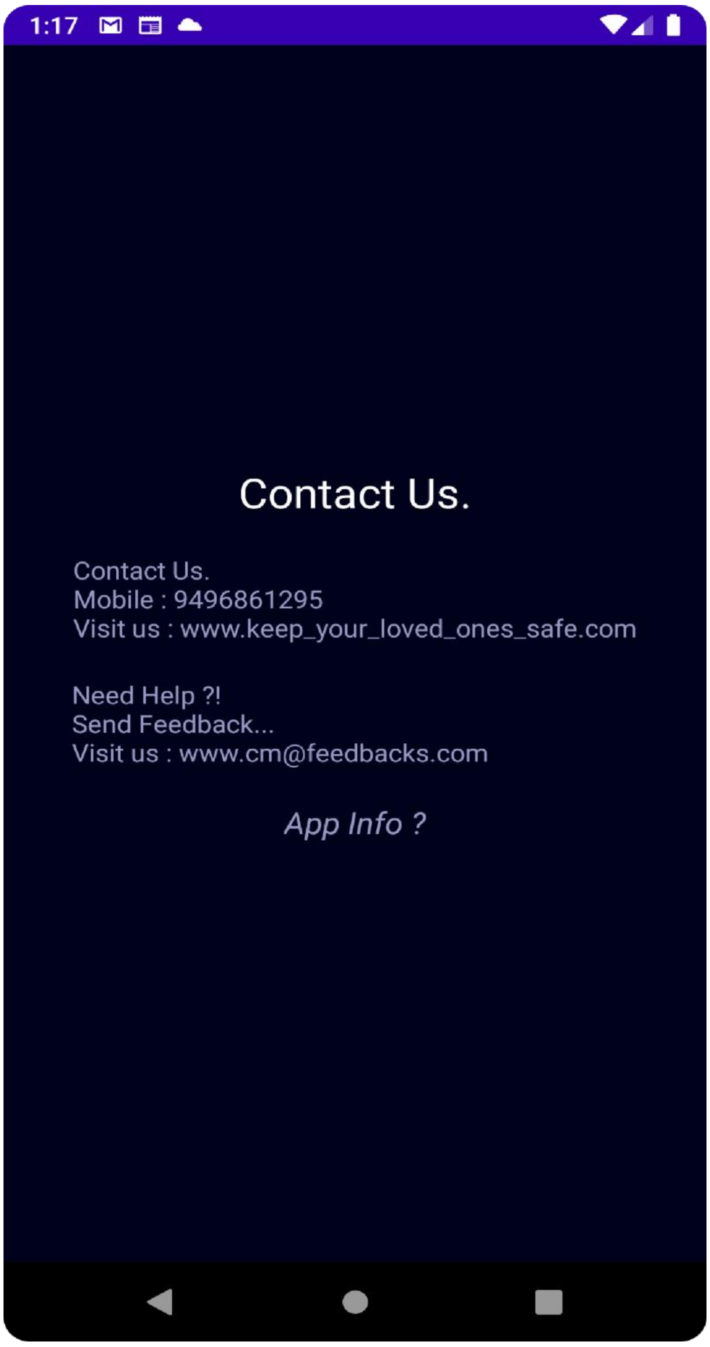
```
        }else if (TextUtils.isEmpty(inlandmark) || inlandmark.length() < 3) {
            landmark.setError("Enter a valid landmark");
            Toast.makeText(Register.this, "Registerion failed landmark",
  Toast.LENGTH_SHORT).show();
        }else if (TextUtils.isEmpty(indateofbirth) || indateofbirth.length() < 10) {
            dateofbirth.setError("Enter a valid DOB");
            Toast.makeText(Register.this, "Registerion failed DateOfBirth",
  Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(Register.this, "all set", Toast.LENGTH_SHORT).show();
            auth.createUserWithEmailAndPassword(inmail,
  inpasswd).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    //database user creation
                    String status = new String();
                    status = "1";
                    String id = task.getResult().getUser().getUid();
                    users user = new users(inusername, inmail,
  inpasswd,id,status,inaddress,inphone,inpincode,inlandmark,indateofbirth);
                    Toast.makeText(Register.this, "id : " + id, Toast.LENGTH_SHORT).show();
                    database.getReference().child("users").child(id).setValue(user);


                    Toast.makeText(Register.this, "User Created",
  Toast.LENGTH_SHORT).show();
                }
            });
            startActivity(new Intent(getApplicationContext(), navigatecenter.class));
        }
    }
   });

 }
 private boolean validateEmail(EditText p_editText, String p_nullMsg, String p_invalidMsg)
   {
   boolean m_isValid = false;
   try {
       if (p_editText != null) {
           if (validateForNull(p_editText, p_nullMsg)) {
               Pattern m_pattern = Pattern.compile("([\\w\\-]([\\.\\w])+[\\w]+@([\\w\\-]+\\.)+[A-
   Za-z]{2,4})");
               Matcher m_matcher = m_pattern.matcher(p_editText.getText().toString().trim());
               if (!m_matcher.matches() & p_editText.getText().toString().trim().length() > 0) {
                  m_isValid = false;
                  p_editText.setError(p_invalidMsg);
               } else {
                  m_isValid = true;
               }
            } else {
               m_isValid = false;
            }
         } else {
            m_isValid = false;
         }
      } catch (Throwable p_e) {
         p_e.printStackTrace();
      }
      return m_isValid;
   }
   private boolean validateForNull(EditText p_editText, String p_nullMsg) {
      boolean m_isValid = false;
      try {
         if (p_editText != null && p_nullMsg != null) {
            if (TextUtils.isEmpty(p_editText.getText().toString().trim())) {
               p_editText.setError(p_nullMsg);
               m_isValid = false;
            } else {
               m_isValid = true;
            }
         }
      } catch (Throwable p_e) {
         p_e.printStackTrace();
      }
      return m_isValid;
   }


}
```

9.2 Screen Shots

Home page : parent.



About Us page [both]


1:17

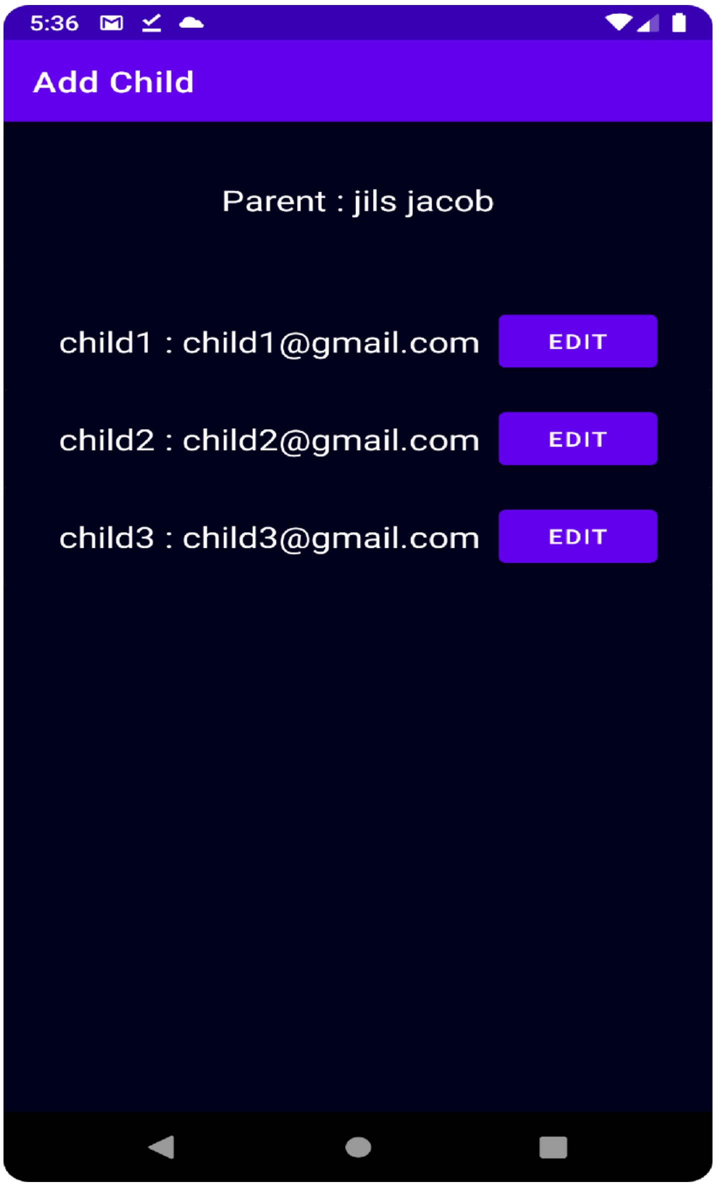
Contact Us.

Contact Us.
Mobile : 9496861295
Visit us : www.keep_your_loved_ones_safe.com

Need Help ?!
Send Feedback...
Visit us : www.cm@feedbacks.com

App Info ?

Parent add_child page


5:36

Add Child

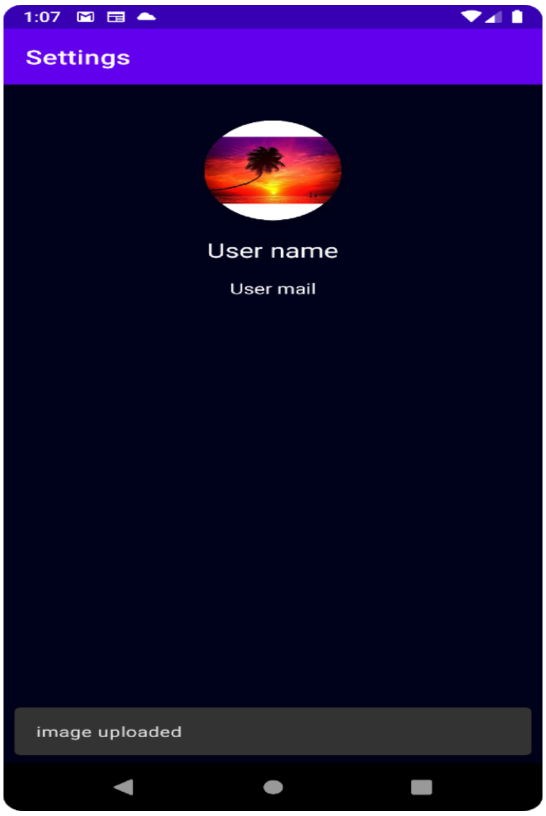
Parent : jils jacob

child1 : child1@gmail.com EDIT

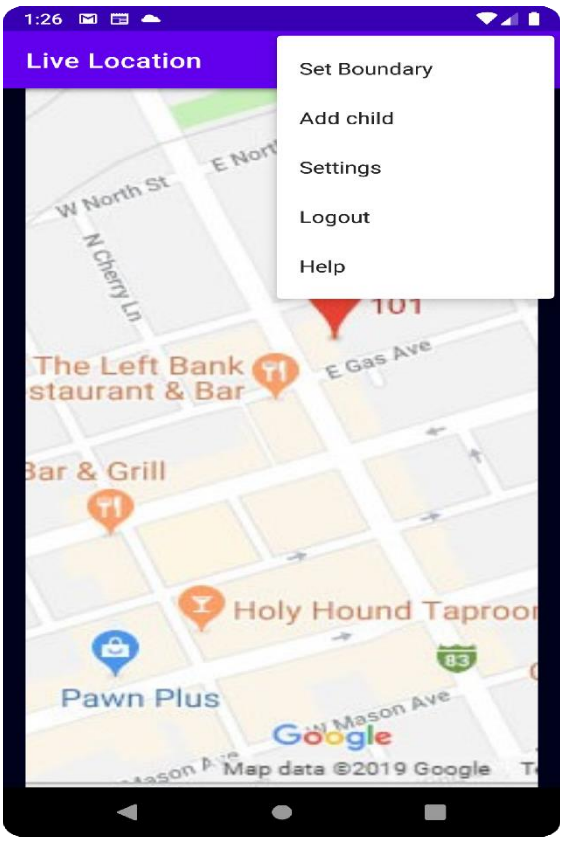
child2 : child2@gmail.com EDIT

child3 : child3@gmail.com EDIT

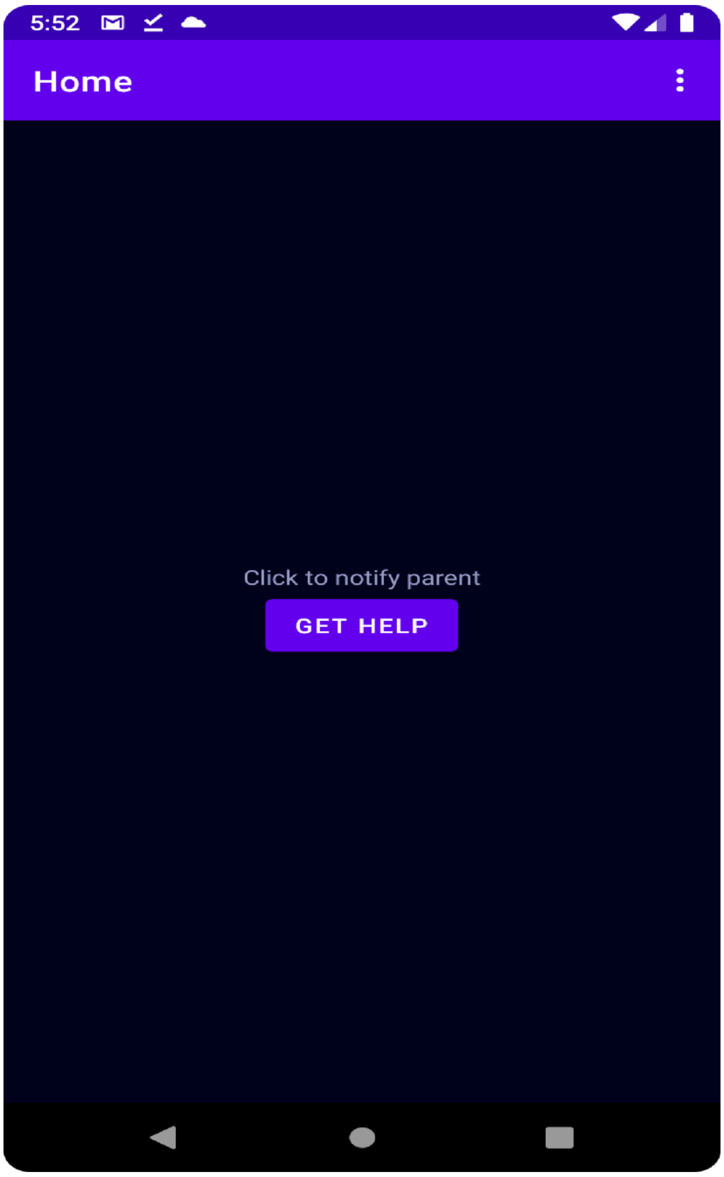
Parent profile page :



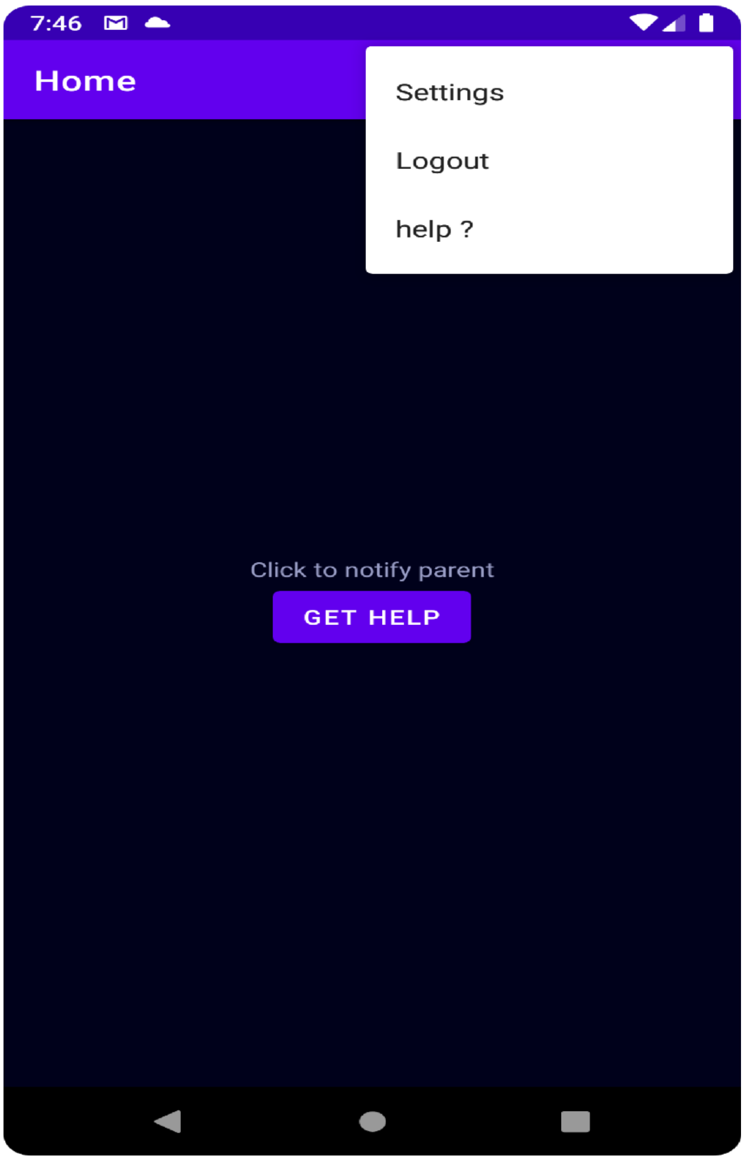
Parent option menu :



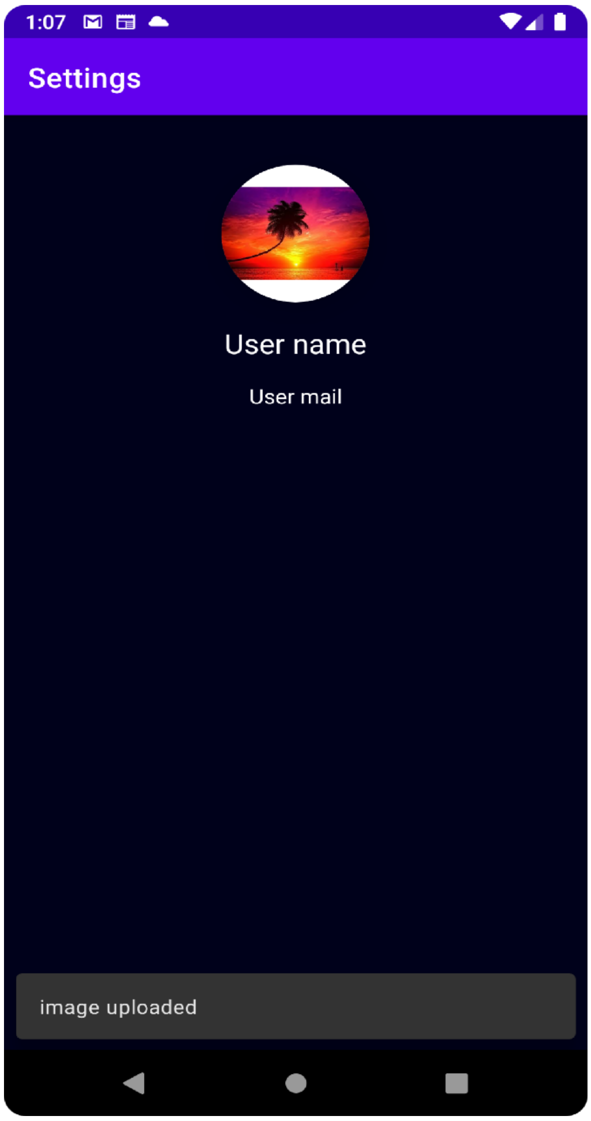
Home page : child



Child option menu



Child profile page :


1:07
Settings
User name
User mail
image uploaded