# 1. INTRODUCTION

## 1.1 Project Summary:

The project entitled "*Cash In-Time*" is a useful in the time of emergency. Money is a verifiable record that is generally accepted as payment for goods and services and repayment of debts in a particular country or socio-economic context. This shows how important money is in our day to day life. Cash is required for many various activities.

There are many shops which doesn't have facility of card payment or any related facility. This can be a problem for those who do not have cash. At that point of time we have to find ATMs near us which are not always available. At times situation may arise that after waiting in the long queues the denominations are not available in the ATMs. This in turn leads to waste of time and also gets much tedious and irritating. Also banks are closed on national holidays or have strikes at time, so getting cash from bank is also not a regular option.

To solve problems like this the application *Cash In-Time* will help the people in need of cash where we won't be going here and there in search of ATMs and also ready cash would be available.

## 1.2 Purpose

**Goal**

The main goal of the system is to help people who are in emergency need of cash. The other goal is to save time of people who are waiting in long queues for withdrawal or to deposit of money.

**Objective**

The main purpose of this application is to let the users meet and exchange the cash. It helps the person who is in need of cash to meet the right person who can provide him/her with the cash. It also gives user with the facility of chatting and calling. User can decide the meeting place with the

help of map. It provides the user to set a range in which two can meet. The application also suggests the meeting place for the users nearest and convenient to them.

## 1.3 Scope:

It is useful in case of emergency of cash. It saves time as lot of time is wasted in queue in ATM and bank for withdrawal and deposit of money. It provides the facilities like chat and calling. The facility of map location is also provided to guide users and help them meet easily.

## 1.4 Literature Review:

- **Studied Systems Included:**

We studied the system *Online Payment in Auction* that includes the online payment for the items sold in any auction. By this system, the hefty amounts can be transferred in a secured way and also the involvement of the third party can be avoided.

We studied the system *Transaction Authorization* that involves series of notifications and OTPs for any activity noticed with the account of the registered users. Notification is sent to the account owner whenever any transaction is initialized by his/her account and also for the completion of the transaction the system sends OTP for secured transaction.

We studied the system *Automated Process for Transfer of Funds* that is used for the automatic transfer of certain amount of funds from one account to another periodically. This reduces the task of remembering the amount and the account of the other party to whom funds are to be transferred periodically. For example, Bank Loans, Medical Bills, Insurances, Mutual Funds.

# 2. SYSTEM REQUIREMENT STUDY

## 2.1 User Characteristics:

The project *Cash In-Time* system would be used by mainly two users:-

| USER | CHARACTERISTICS |
|---|---|
| **Cash Sender** | - View the users in need of cash<br>- Select the suitable user<br>- Confirm the request of the selected user<br>- Communication<br>- Exchange cash |
| **Cash Receiver** | - Fill in the cash amount required<br>- Fill in the range<br>- Send request<br>- Communicate<br>- Exchange cash |

## 2.2 Hardware and Software Requirement:

Hardware Requirements:

- Smart Phones (Version: Android 6.0)
- Laptop (8 GB RAM, i5 processor)
- Internet Connection

Software Requirement:

- Android Studio (Version 3.0.1)
- Genymotion Android emulator (Version 2.12.2 )
- MySql 2008 and above
- Microsoft Word
- Firebase

## 2.3 Constraints:

**Hardware Limitation:**

The installation of Android Studio requires the RAM of 8 GB, i5 and the speed as recommended in Hardware Requirements.

**Higher Order Language Requirements:**

The system needs Android Studio as it is a new Technology and there are many extra features in that. We need to have a good Graphical User Interface (GUI) with user friendly environment which visual studio provides.

**Reliability Requirements:**

The main reliability requirement is the validations used. Without proper validation the system would not allow to enter the value into the database. For e.g. in the name field only characters are allowed and user cannot enter any dummy values, the validations check all these things. Any null value is not allowed in the compulsory fields.

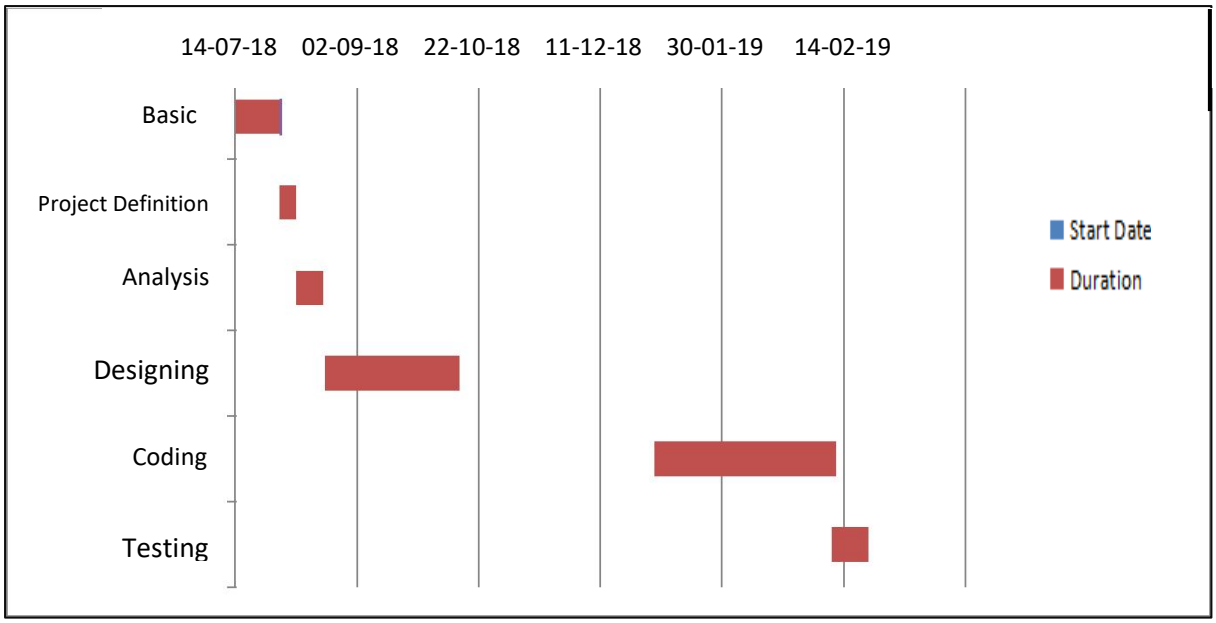
**Safety and Security Consideration:**

The username and password authentication has been provided for the login.

**Assumptions and Dependencies:**

The project design is mainly created by keeping in mind the dependency with Language. The Student wants to share the data to other student so he/she must have basic knowledge of computer. The currencies that are used in the transactions are real and not fake. The users are truly genuine.

## 2.4 Timeline chart:



# 3. SYSTEM ANALYSIS

## 3.1 Study of Current System:

In today's scenario, whenever we want to credit to our bank we need to go there and stand in the long queues. A lot of time is wasted and is also irritating to stand every time we want to deposit the cash in our accounts. Same is the problem with withdrawals. Outside ATMs, long lines are waiting for us. Many a times situation may arise that cash in ATMs is not available. Also for debit cards there are limits that can be swiped from other bank ATMs. Exceeding that limit, charges are levied on us. We generally receive our monthly income in the starting days of the month. And generally tend to spend it almost 1 week before the month ends.

There are also many applications available related to banks and ATMs which have proven to be useful. Many applications have been developed that helps us to locate the banks and the ATMs known as Bank Locaters and ATM Locaters. But they just help us to find the banks and ATMs nearby us. They are unable to tell whether the bank is working or not and whether the denominations are available or not.

Whenever in need of cash we can obviously go to the bank, but then what about after the working hours of bank. At that point of time ATMs can be an option but not always because even they run out of the denominations. So at that point of time this system can be very useful.

## 3.2 Problems and Weaknesses of Current System:

The existing systems are time consuming due to which many users avoid the usage of these systems. Waiting in long queues outside ATMs or in the banks for cash withdrawal or deposit is much irritating. Many a times denominations not available in ATMs and lack of cash in nationalized banks. There are restrictions on the number of transactions in ATM using debit/credit card. The problem of ATM Machine failures and taxes are also levied on exceeding the certain limit of card swipes from other bank ATMs.

## 3.3 Requirements of New System :

The main aim of this project is to provide people with the benefit of on the spot cash. Basically it can be described as an 'Uber / Ola for Cash'. If a person has to debit and another has to credit then they both can do the transaction themselves. There's no need to go till the bank.

If there a person who is in need of cash, he can ping that in the application. He can also set the radius of the area accordingly. Then the application will notify the users who have to credit their money. Those users accordingly can reply that person. Both the users can meet and can transfer the money either by online transfer, Paytm or any other convenient mode of payment.

The system also provides facility that suggests the nearest and convenient place for the two persons to meet. They can also manually decide the meeting place but the applications also suggests them the place.

There is also an additional security feature of blacklisting the numbers. As the transaction is going to take place so personal details are to be exchanged. So to maintain the privacy of the users and keeping security in mind, feature of blacklisting is also to be added.

## 3.4 Feasibility Study:

An Important part of initializing a project is justifying it. In other words, determine whether or not it should be built. The main goal of this study is to define the best implementation solution for the project undertaken and justify why it is best. The feasibility study uses technique that helps to evaluate a project and or compare it with other project.

**Objective of system feasibility analysis:**

The objective of system feasibility analysis is to identify the user's need and evaluate the system concept for feasibility. Economic and technical analysis is to be performed. Allocation of functions to hardware, software, people, database and other system analysis is to be done. Its objective is also to establish cost and schedule constrains and to create system definition forming the foundation for all subsequent

work. Feasibility study mainly focuses on whether the system is feasible or not. It concentrates on various kinds of feasibility study such as:

**a) Technical Feasibility:**

It can work for the project to be done with the present equipment, current procedures existing and software technology available This will be requiring a close examination of the present system.

**The Technical Feasibility should ask question related to:**

Adequacy of available technology, adequacy of Hardware, availability of computer, operating time and support facilities etc. This is mainly concerned with specifying equipment and software that will successfully satisfy the user requirement. We studied our project is technically feasible with the equipment's and software provided to us.

**b) Operational Feasibility:**

Operational feasibility can be described as Will the system be used, if it is implemented or about job security, loss of peer group, changes in job context and so on whenever new systems are proposed. Our project is feasible in this aspect because any user of the system can operate the system easily without much training needed and time wasted for it.

**c) Social Feasibility:**

This is concerned with the determination of whether the system will be acceptable to the people or not. Our project is acceptable to the user from all the aspects with its predefined factors.

**d) Economic Feasibility:**

The cost must include both onetime costs and recurring costs. Costs involved in Software packages are also to be included. This is most frequently used technique for evaluating the effectiveness of a proposed system. This is procedure to determine the benefits and savings that are expected from proposed system and compare them with cost.

**e) Management Feasibility:**

This is used for determination of whether our project is acceptable to management or not. If the management does not accept project, the analyst will tend to view the project has non-feasible.

**f) Legal Feasibility:**

This is determination of whether our project is proposed project infringes on knows as well as any pending legislation.

**g) Time Feasibility:**

This is mainly concerned with the determination of whether the system project can be implemented fully within stipulated period. Our project is feasible with aspects of time as it would be fully prepared and could be implemented in the given period of our project training.

## 3.5 Requirement Validation:

We use validations to add constraints to an entry field. We have certain fields in most of the activities where validation is required. We are using validations in the fields like password, contact number, email, amount, etc.

## 3.6 Functions of System:

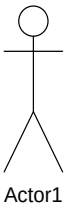
### 3.6.1 Use case diagrams:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior,

perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.


Actors: An actor is a person, organization, or external system that plays a role in one or more interactions with the system.




System boundary boxes: A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not.

1. **Cash Sender :**

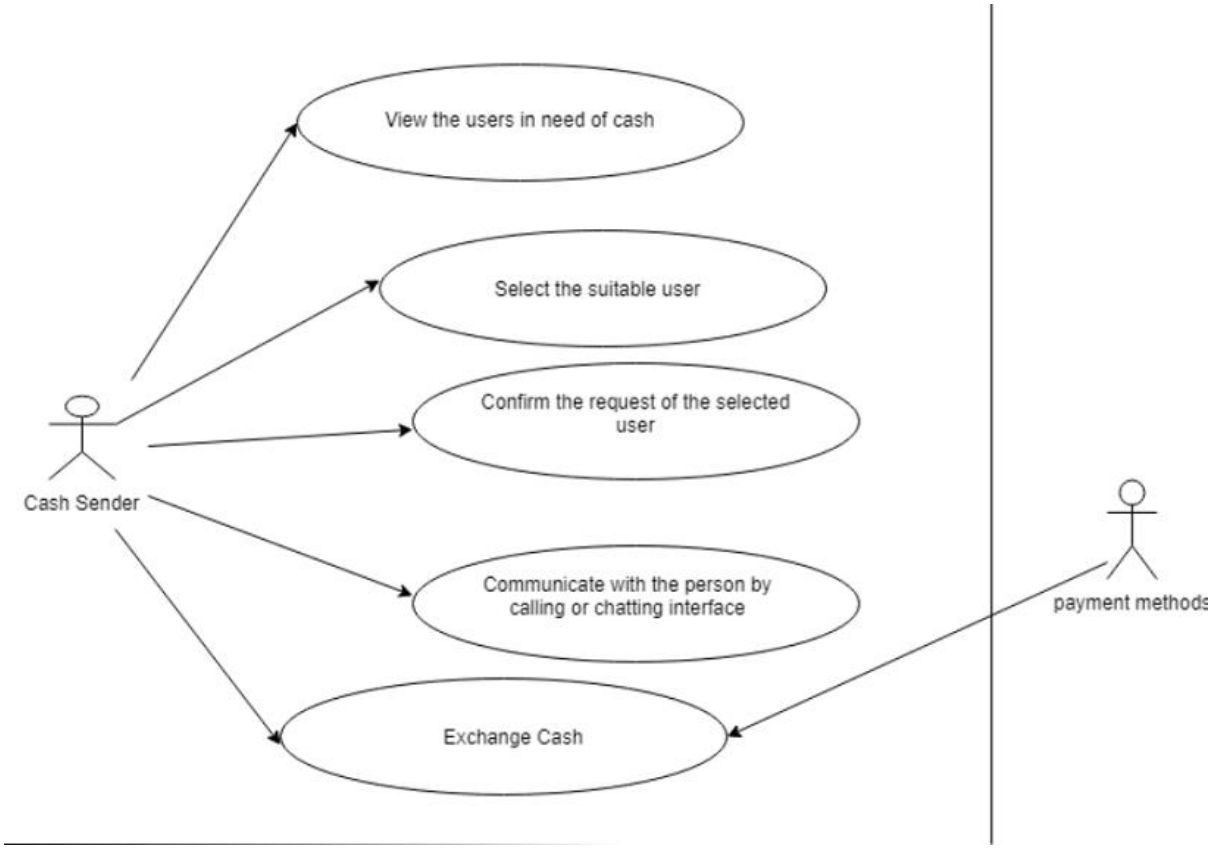



**Fig 1: Usecase for Cash Sender**

## 2. Cash Receiver :

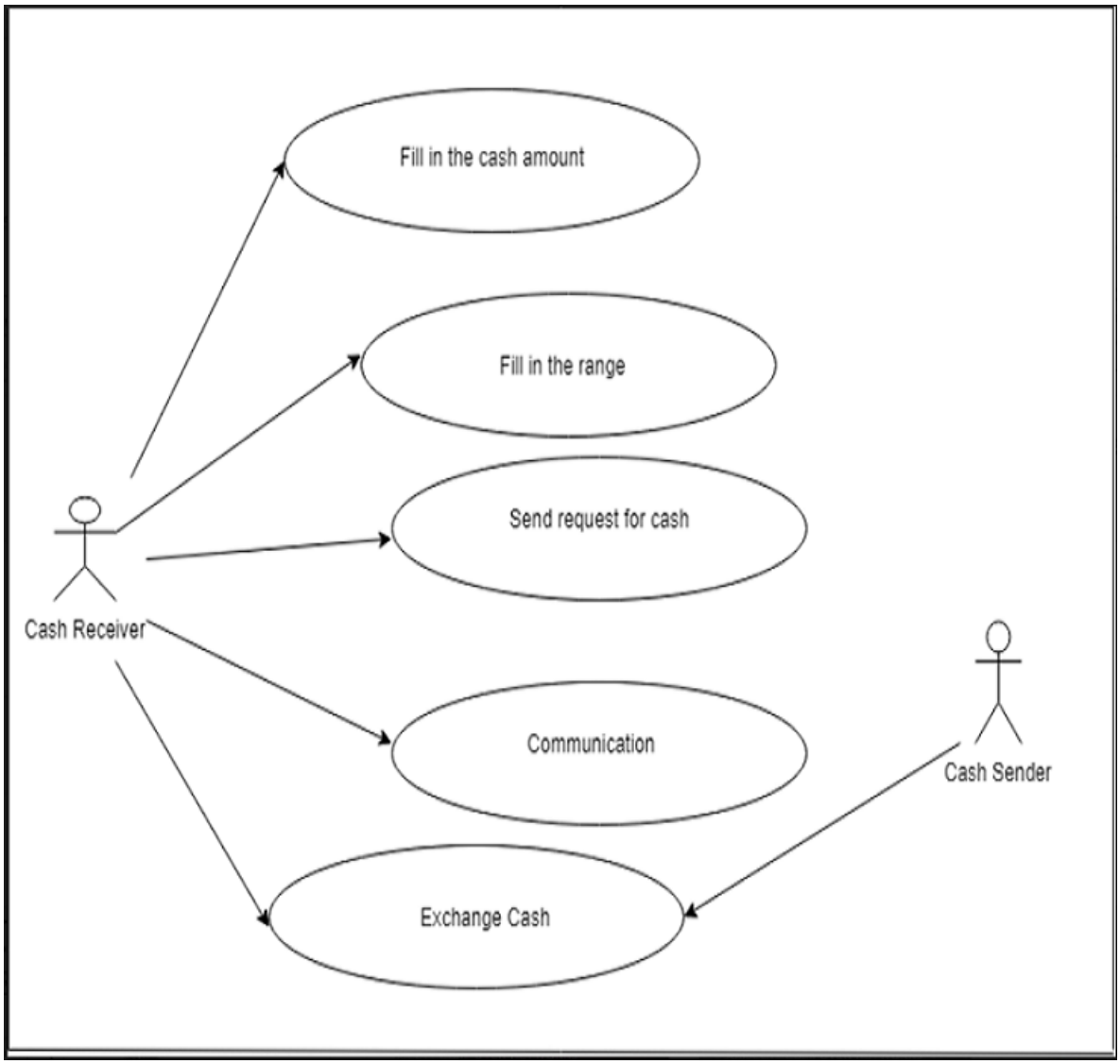



**Fig 2 : Usecase for Cash Receiver**

### 3.6.2 CLASS DIAGRAM:

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved.

In a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class. Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

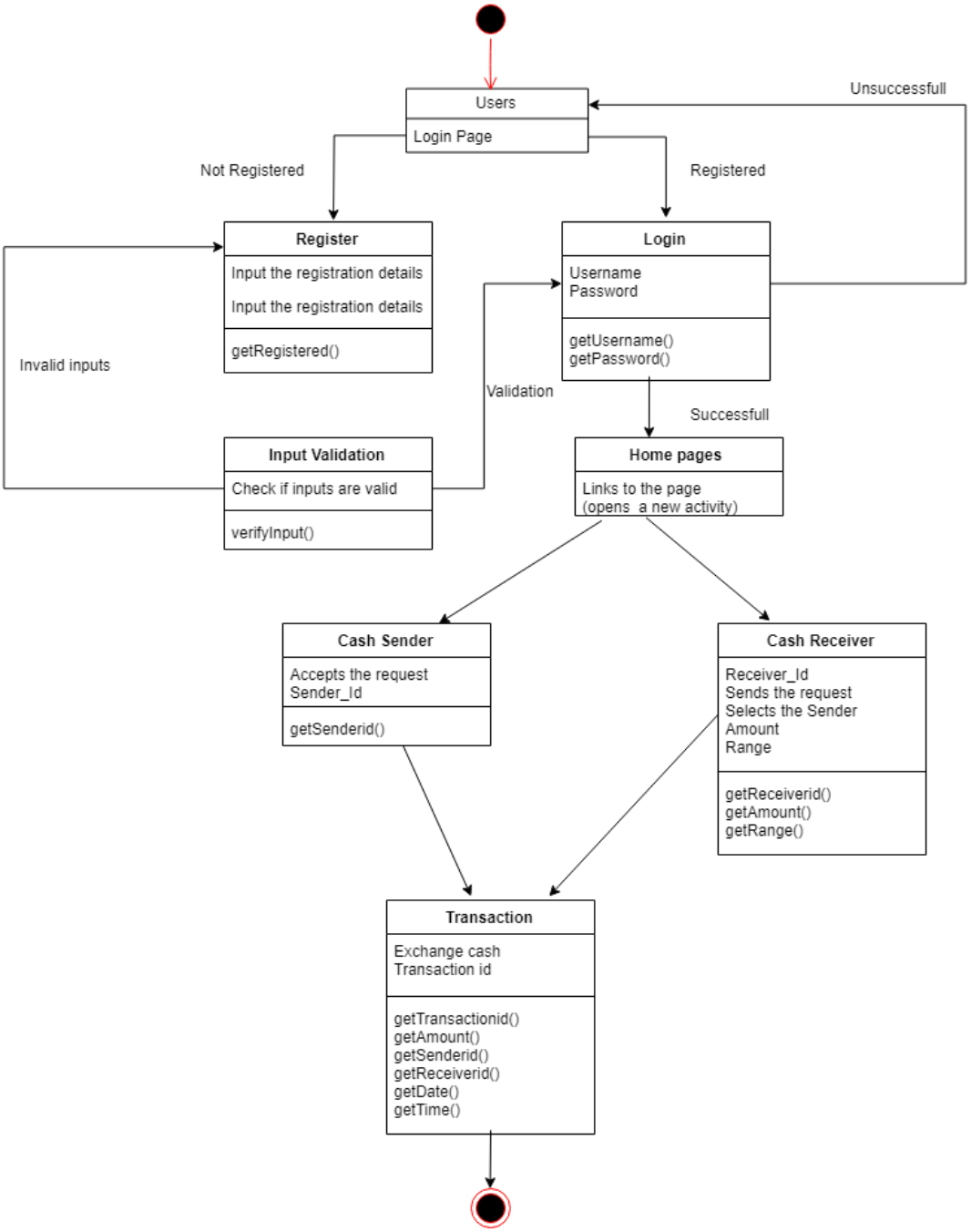



**Fig 3: Class Diagram**

### 3.6.3 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. Some systems have simple dynamic behavior that can be expressed in terms of specific sequences of messages between a small, fixed number of objects or processes. In such cases sequence diagrams can completely specify the system's behavior. Often, behavior is more complex, e.g. when the set of communicating objects is large or highly variable, when there are many branch points (e.g. exceptions), when there are complex iterations, or synchronization issues such as resource contention. In such cases, sequence diagrams cannot completely describe the system's behavior, but they can specify typical use cases for the system, small details in its behaviors, and simplified overviews of its behavior.

Usage: Some systems have simple dynamic behavior that can be expressed in terms of specific sequences of messages between a small, fixed number of objects or processes. In such cases sequence diagrams can completely specify the system's behavior. Often, behavior is more complex, e.g. when the set of communicating objects is large or highly variable, when there are many branch points (e.g. exceptions), when there are complex iterations, or synchronization issues such as resource contention. In such cases, sequence diagrams cannot completely describe the system's behavior, but they can specify typical use cases for the system, small details in its behavior, and simplified overviews of its behavior.

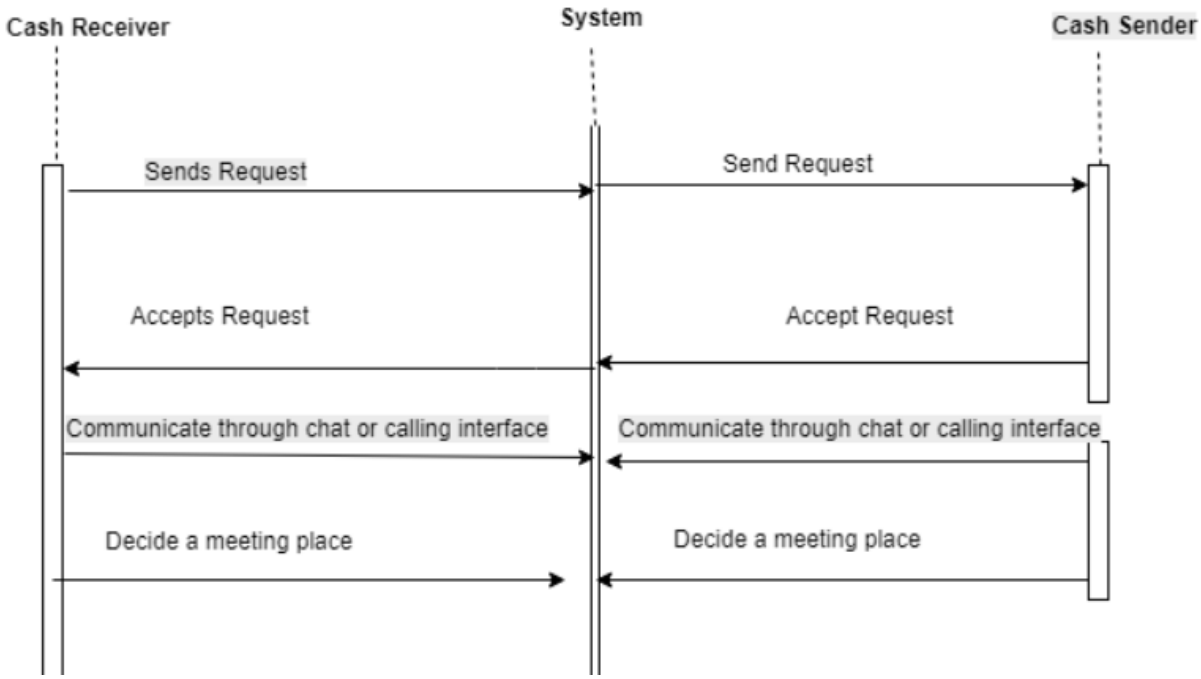



**Fig 4: Sequence diagram**

## 3.7 Data Modelling

### 3.7.1 E-R diagram

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often relational, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.

E-R Diagrams mainly consists of:

 Entity  Attributes  Relations

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of some domain. When we speak of an entity we normally speak of some aspect of the real world which can be distinguished from other aspects of the real world. Entity-relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets. A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Every entity (unless it is a weak entity) must have a minimal set of uniquely identifying attributes, which is called the entity's primary key. Entity-relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets.

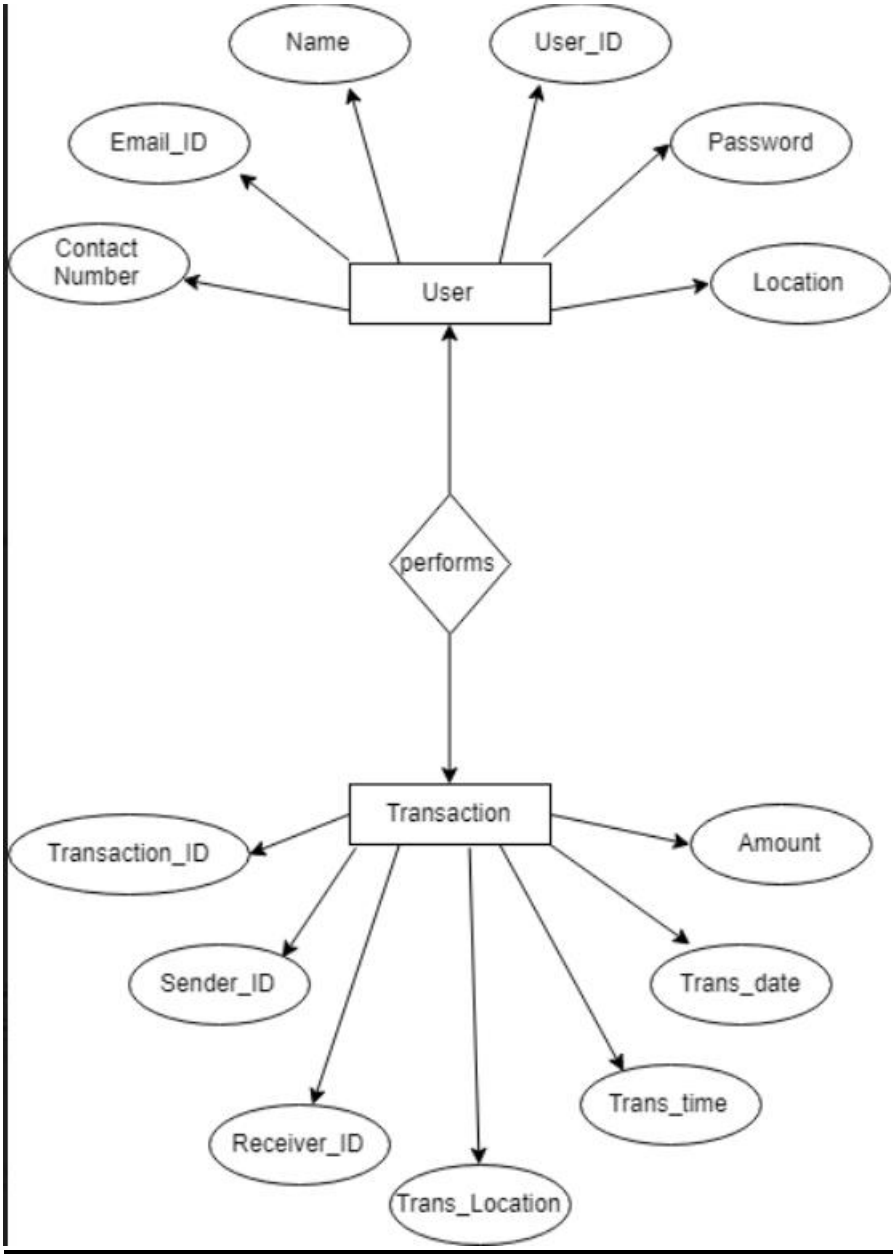



**Fig 5: E-R Diagram**

### 3.7.2 System Activity Diagrams:

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

Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows. The most important shape types are rounded rectangles that represent activities, diamonds representing decisions, bars that represent the start (split) or end (join) of concurrent activities, a black circle representing the start (initial state) of the workflow and an encircled black circle that represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen. Hence they can be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

1. **Login :**

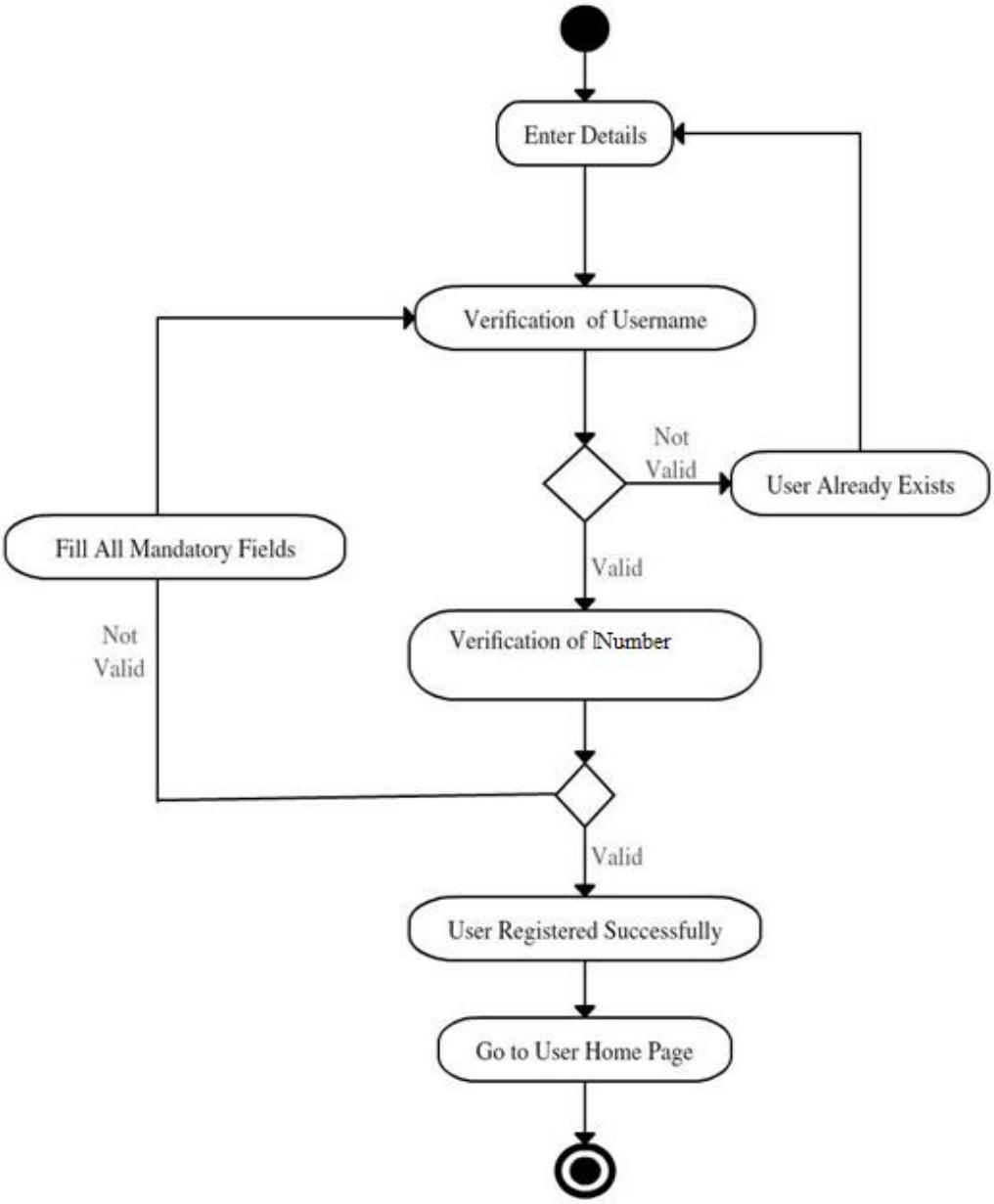



**Fig 6: Activity diagram for Login**

## 2. Transactions :

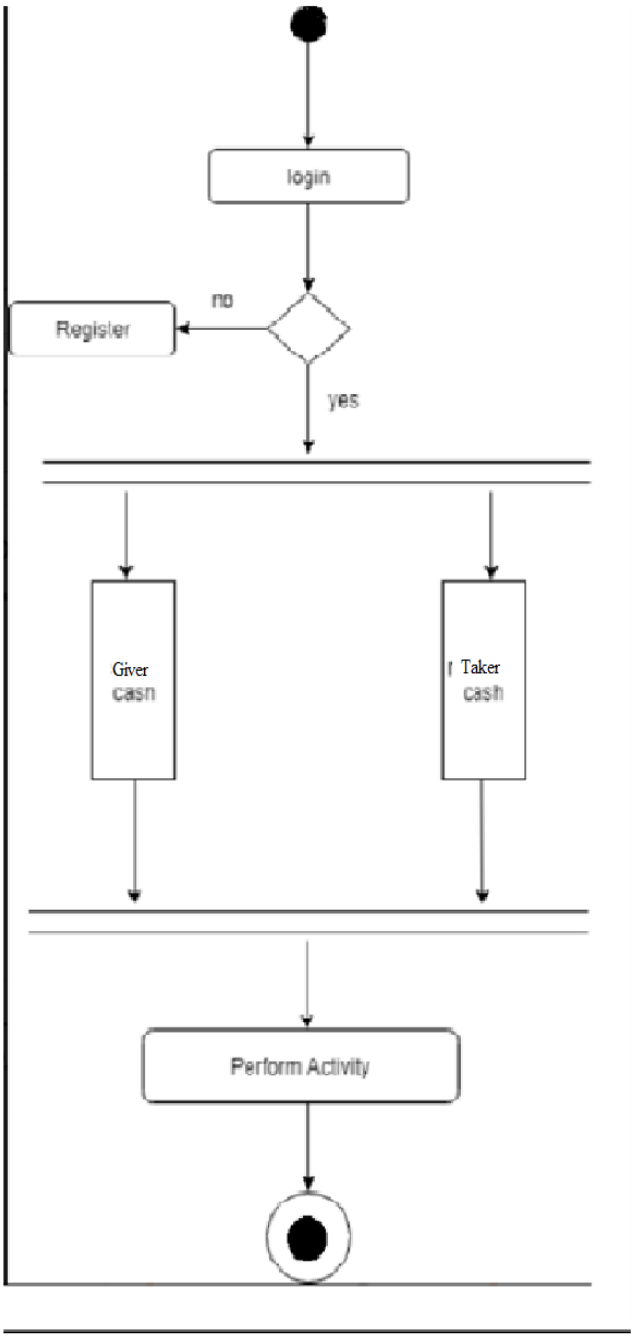



**Fig 7: Activity diagram for Transaction**

### 3. Send Cash :

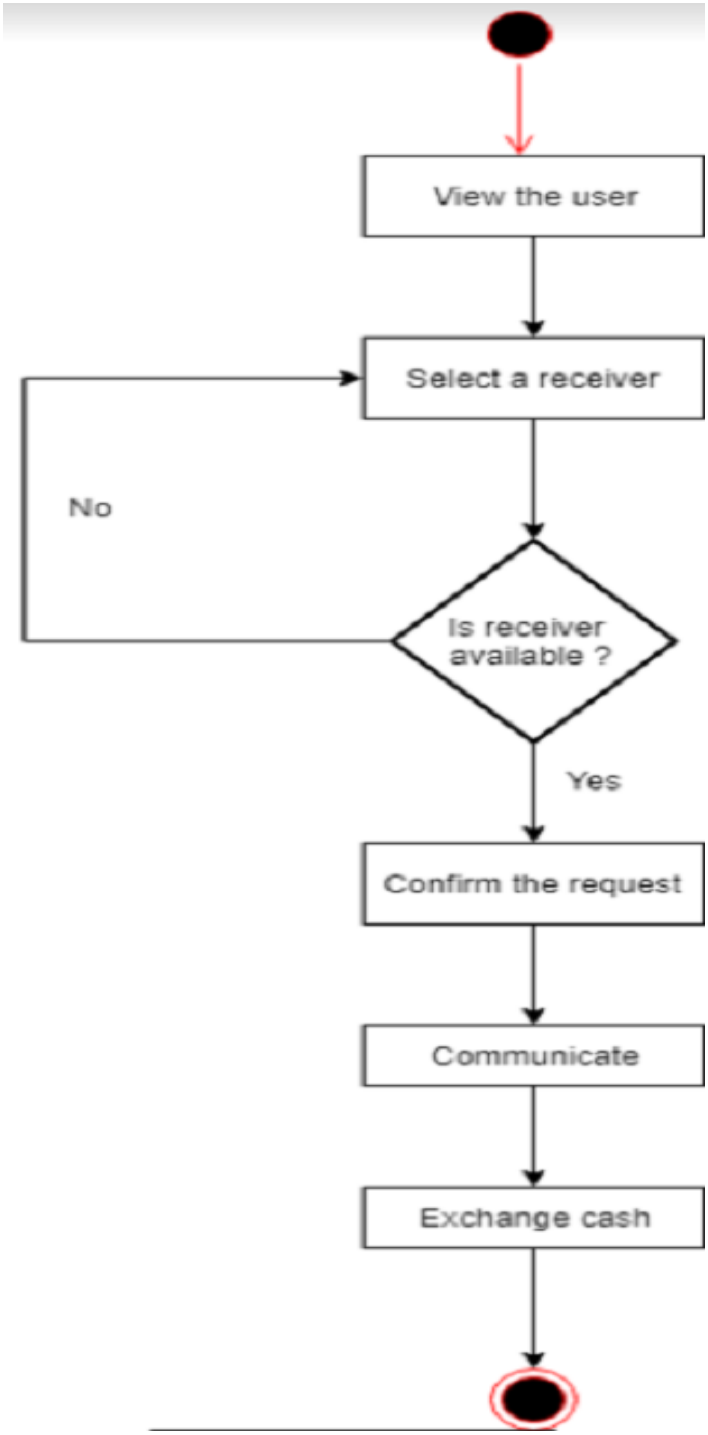



**Fig 8: Activity diagram for Sending Cash**

4. **Receiving Cash :**

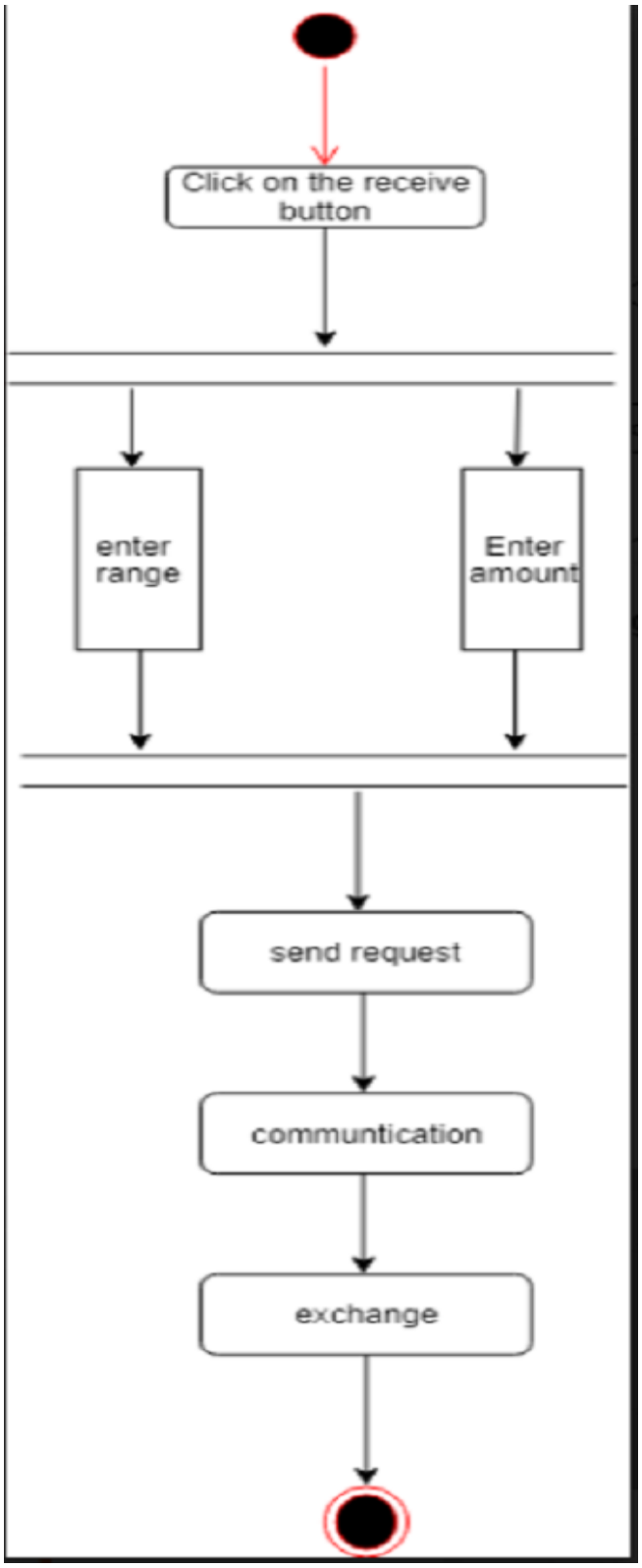



**Fig 9: Activity diagram for receiving cash**

### 3.7.3 Data Dictionary

**1. Table Name: User**

| SR. NO. | FIELD NAME | TYPE (SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | User_Id | Auto Number (10) | Primary Key | Id of user for unique identification of every user. |
| 2 | Username | Varchar (25) | Not Null | It is the username. |
| 3 | Password | Varchar (20) | Not Null | Password to login as user. |
| 4 | Mobile No | Int (10) | Not Null | Mobile Number of user. |
| 5 | Email_Id | Varchar (25) | Not Null | Email address of the user. |
| 6 | Address / Location | Number | Not Null | The residential address of the user is stored for security purpose. |

**2. Table Name: Transactions**

| SR. NO. | FIELD NAME | TYPE (SIZE) | CONSTRAINT | DESCRIPTION |
|---|---|---|---|---|
| 1 | Transaction_Id | Auto Number(10) | Primary Key | Unique id of each transaction. |
| 2 | Sender_ID | Number(10) | Foreign key | It shows that the user is a sender for the particular transaction. |
| 3 | Receiver_ID | Number(10) | Foreign key | It shows that the user is a receiver for the particular transaction. |
| 4 | Location | Number | Not Null | Location of both the users will performing transaction |
| 5 | Date | Date | Not Null | Date of transaction. |
| 6 | Time | Time | Not Null | Time of transaction. |
| 7 | Amount | Currency | Not Null | Amount transacted |

## 3.8 Functional and Behavioural Modeling

### 3.8.1 Context Diagram

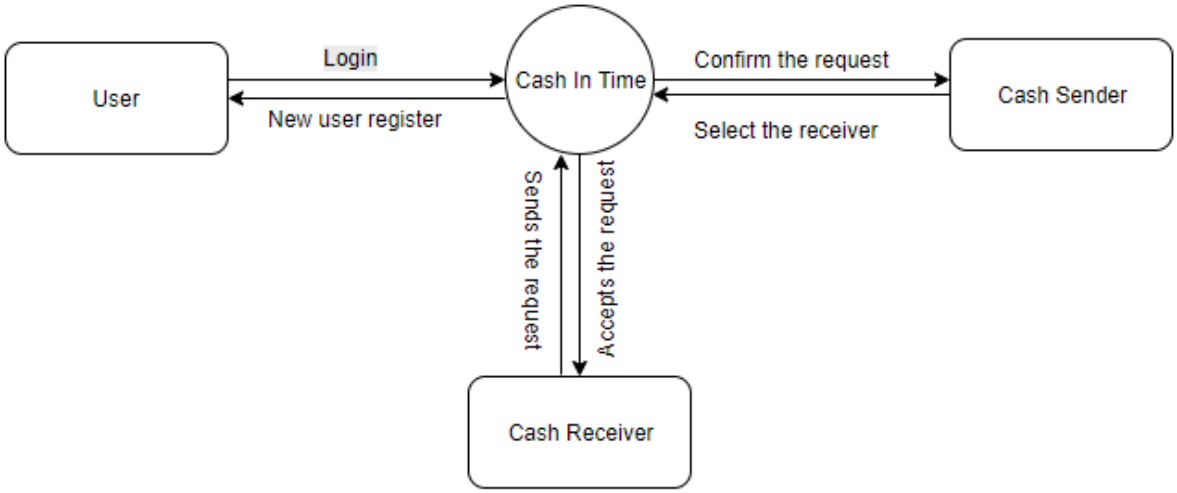


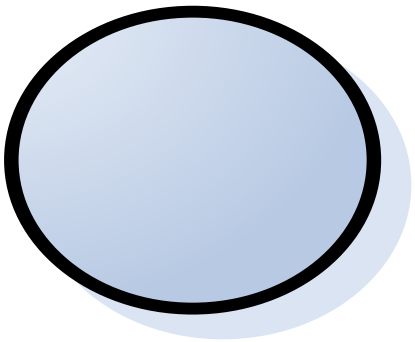

**Fig 10: Context Diagram**

### 3.8.2 Data flow Diagram:

Data flow diagram is a graphical tool used to describe and analysis the movement of data through a system-manual or automated including the Processes stores the data, and delays in the system. Data flow diagrams are the central tool and basis from which other components are developed.

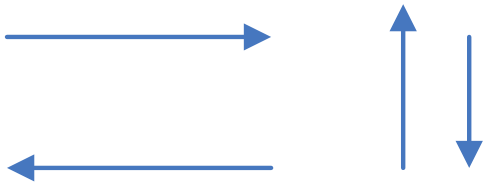
Symbols used in DFDs:

1) Process: Here flow of data is transformed. E.g. Registering Student, etc.

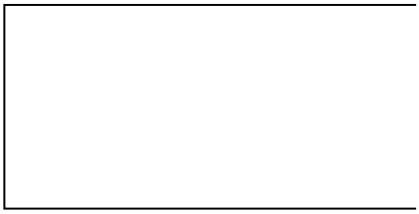


2) External Entity: A source or destination of data which is external to the system. E.g. Principal Etc.

3) A data flow: It is packet of data. It may be in the form of document, letter etc.



4) Data store: Any store data but with no reference to the physical method of storing.



LEVEL 1:-

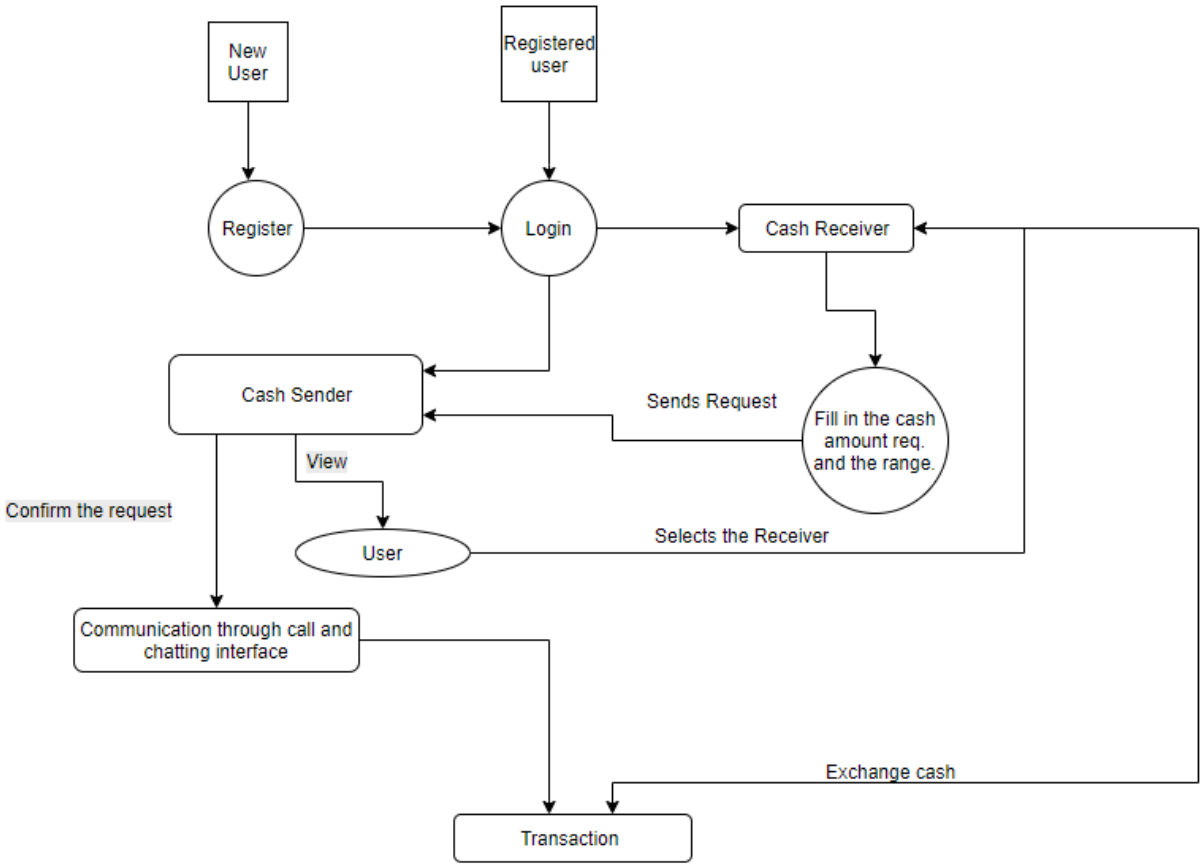



**Fig 11: Data Flow Diagram**

## 3.9 Main Modules of New System:

### User

The user can become sender or receiver but no both at a particular time User as receiver will be asked to enter the amount of money required and the range of area. Users as sender will be shown with the users who have requested for cash. Both the users will then be provided with the nearest distance where they can meet and perform the transaction.

## 3.10 Selection of Hardware and Software and Justification:

### Hardware Requirements:

The hardware requirements are minimum RAM of 8 GB, Internet connection and android phone with android version greater or equal to Marshmallow. The minimum RAM required is 8 GB so that the computer system can work efficiently. Internet connection is also required as our system is a web service internet connection is mandatory. Android Phones with android version greater or equal to marshmallow is the necessity because major support of firebase starts with the marshmallow version of android.

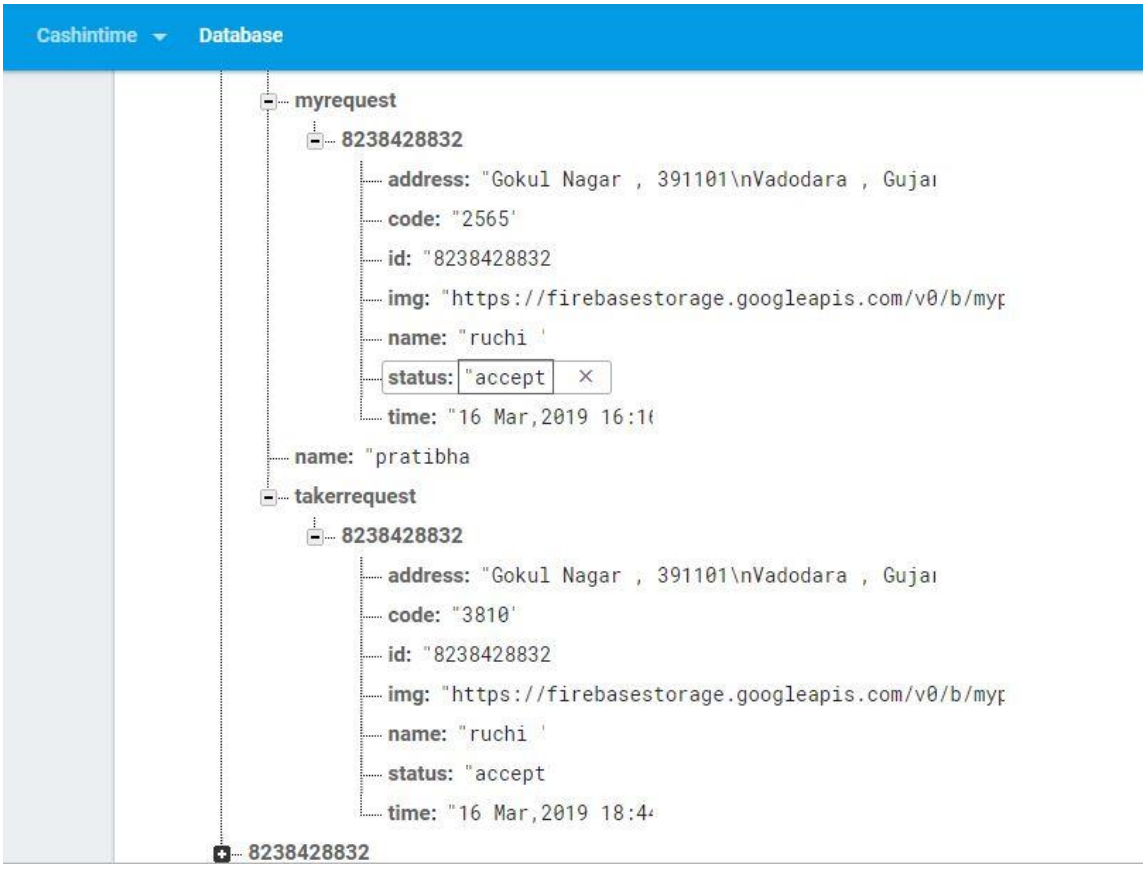
### Software Requirements:

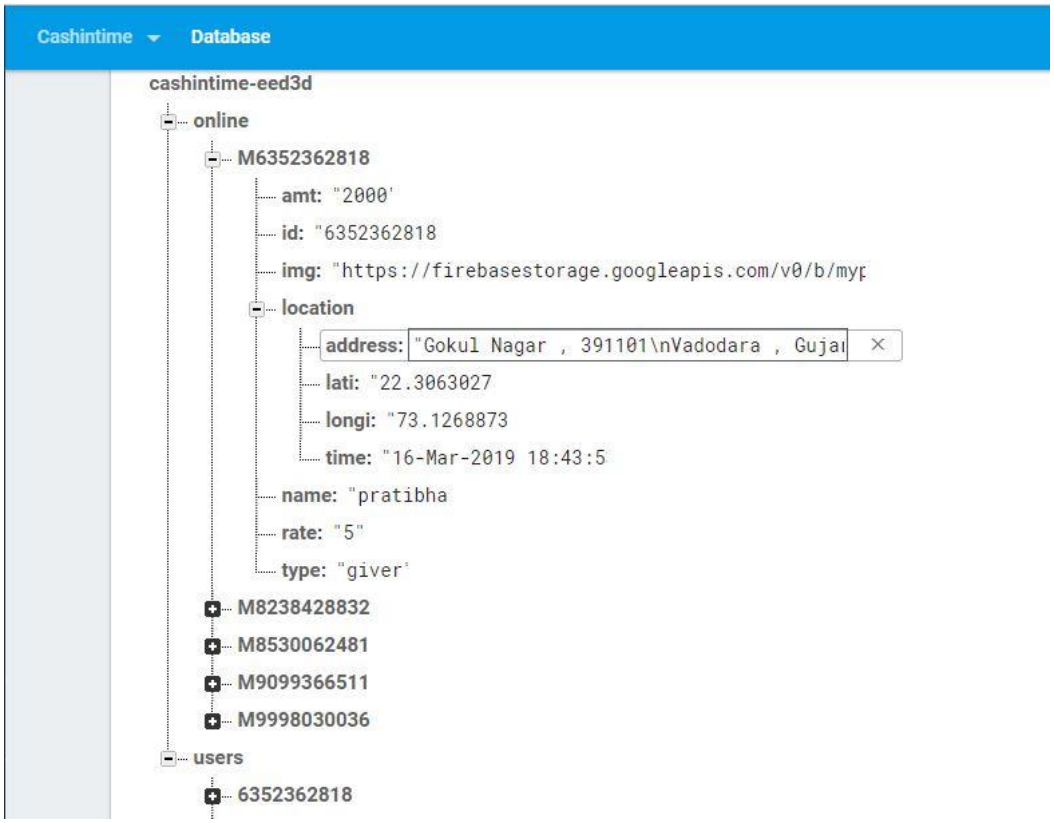
The software requirements are operating system, Android Studio, Microsoft Word, Genymotion and Android emulator. We require operating system Windows 7 or above so that the all software can work efficiently. Android Studio having version 3.0.1 is required as the latest version of android studio includes latest methods and those methods are not included in the older versions. Microsoft Word will be used for the documentation. Genymotion of version 2.12.2 is required as it works as a virtual emulator and an in-built android emulator that can be used as a virtual emulator is also required.

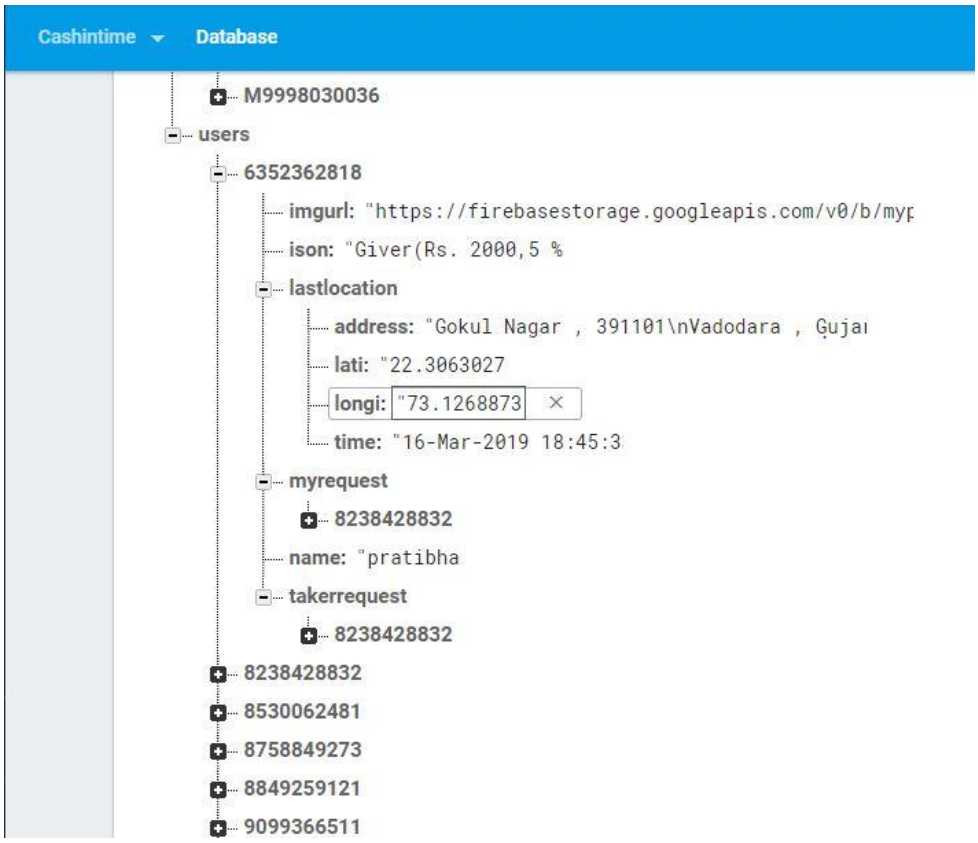
# 4. SYSTEM DESIGN

## 4.1 Database Design/Data Structure Design

**4.1.1 Tables and Relationship:** We have developed the interface using real time database i.e. firebase database which does not operate on table structure the structure followed by firebase is real time database. Hence the structure followed is shown below.




Cashintime
Database
cashintime-eed3d
online
M6352362818
amt: "2000"
id: "6352362818
img: "https://firebasestorage.googleapis.com/v0/b/myp
location
address: "Gokul Nagar , 391101\nVadodara , Gujar
lati: "22.3063027
longi: "73.1268873
time: "16-Mar-2019 18:43:5
name: "pratibha
rate: "5"
type: "giver"
M8238428832
M8530062481
M9099366511
M9998030036
users
6352362818


Cashintime
Database
M9998030036
users
6352362818
imgurl: "https://firebasestorage.googleapis.com/v0/b/myp
ison: "Giver(Rs. 2000,5 %
lastlocation
address: "Gokul Nagar , 391101\nVadodara , Gujar
lati: "22.3063027
longi: "73.1268873
time: "16-Mar-2019 18:45:3
myrequest
8238428832
name: "pratibha
takerrequest
8238428832
8238428832
8530062481
8758849273
8849259121
9099366511

## 4.2 System Procedural Design

### 4.2.1 Designing pseudo code for method and operation:-

**Algorithm:**

1. Login
2. Select whether to be a giver or a taker.
3. If a giver, then fill in amount and rate, he/she wishes to charge.
4. If a taker, then select any convenient giver appearing on the map and send request.
5. The giver may accept any taker's request.
6. As the two parties match, they can contact each other via the calling facility and also make use of live tracking facility available.
7. Hence, the parties meet and can exchange cash.

- For the **Android** application we have used One Time Password **(OTP)** for user verification for new user login.

- Using **OTP** for android application, the users do not have to remember password for his or her authentication and also there is no need for developers to develop an encryption or decryption mechanism to store and retrieve password for user authorization.

## 4.3 Input/Output and Interface Design

### 4.3.1 State Transition Diagram

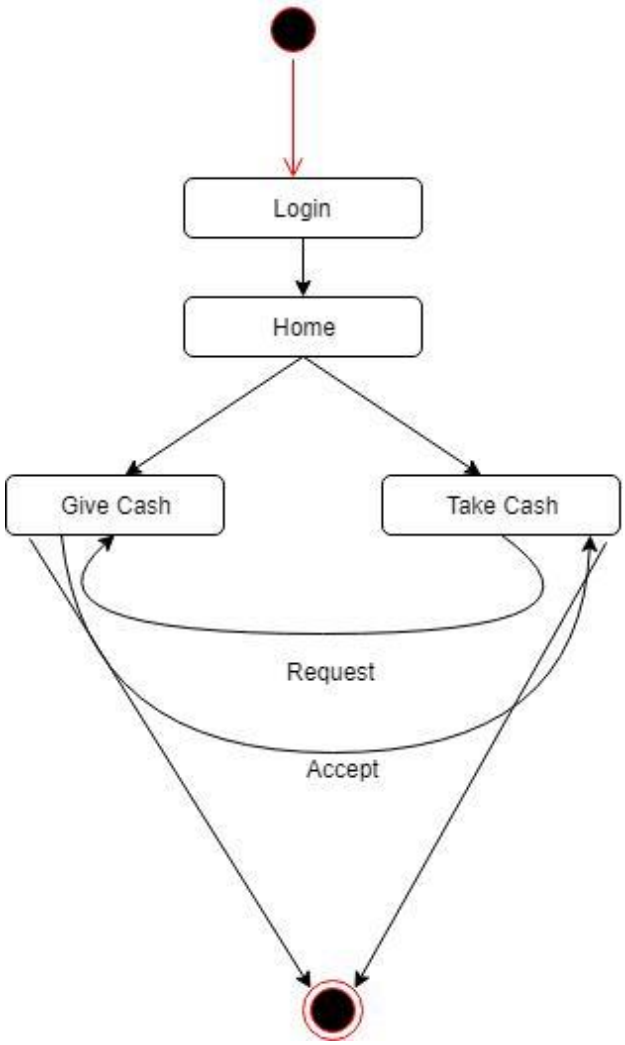



**Fig 12: State Transition Diagram**

# 5. IMPLEMENTATION PLANNING AND DETAILS

## 5.1 Implementation Environment (Single vs Multiuser, GUI vs Non GUI)

- Without a well-formed implementation plan enforced by a team, even justified software will probably fail to meet anticipation. Only when the implementation team has performed similar process within a constructed company then only can the experience be complete.
- As the application is based on making two people meet to exchange cash, it is a multiuser application in the time of emergency. Our execution methodology is based phased as Implementation.
- Phased implementation is basically an idea of developing an interface into several different phases. In our system implementation the interface is first developed considering several different modules that are already being present in market and hence we have considered the reasons and limitations that led to those system drawbacks. We can implement each subsystem by using any of the other different transformed methods. In this approach risk of errors or failures is limited to the implemented module with decreased expense than the full parallel operation.
- The parallel operation for implementing the interface has resulted in side by side development of several different modules present in the interface.

## 5.2 Program/Modules Specification

- Login
- Maps
- Button
- User profile
- Logout button
- History
- Calling

## 5.3 Security Features

- The login of the application is done by a valid mobile number only. An otp is received on the registered number and is auto verified. Now, when the giver accepts the requests, then the 4-digit code on the giver's and taker's equipment matches to ensure that the opposite party is same as selected.

## 5.4 Coding Standards

### 5.4.1 Pseudo Code Naming and Conventions

Following things happen when project is developed using common conventions

- Programmers can go into any code and figure out what's going on.
- New programmer can get up to speed quickly.
- Programmer new to Android are spared making the same mistakes over and over again.
- Programmers make fewer mistakes in consistent environments.
- We have used Android Studio to develop the application Cash in Time. Android Studio is based on java and xml coding.

### 5.4.2 Class Name

- Name the class after what it is. If you can't think of what it is that is a clue you have not thought through the design well enough.
- Avoid the temptation of bringing the name of the class a class derives from into the derived class's name. A class should stand on its own. It doesn't matter what it derives from.

### 5.4.3 Method and Functions Name

- On clicking the Button, the name of the method will be as per the event generated.
- For example: On clicking the Submit button the name of the method will be btnSubmit_Click().

## 5.5 Sample Coding

Home.java

```
1   package Cashintime.com.cit;
2
3   import ...
72
73  public class Home extends AppCompatActivity implements OnMapReadyCallback {
74      TextView tvstatus;
75      private GoogleMap mMap;
76
77      public static Context con;
78      CircleImageView iv;
79
80      TextView address;
81      double lati,longi;
82      Button btn;
83      RelativeLayout lic,lic1;
84
85      boolean isGiver;
86      ProgressDialog pd;
87      Location location;
88
89      static List<Userobject> giver,taker;
90
91      Button btntkr,btngvr;
92
93      @Override
94      protected void onCreate(Bundle savedInstanceState) {
95          super.onCreate(savedInstanceState);
```

# 6. TESTING

## 6.1 Testing Plan

Software testing is a captious element of software quality assurance and represents the ultimate review of specification, design, and coding. Testing portrays an interesting anomaly for the software. The testing phase involves testing of the system using various test data. Preparation of test data plays a vital role in system testing. After preparing the test data, the system beneath study is tested using those test data. Errors found were corrected and corrections were recorded for future sources. Thus, a series of testing is performed on the system before it is ready for implementation.

The development of the software systems involves a series of production activities where opportunities for injection of human fallibility are immense. Errors may begin to occur at very inception of the process where the objectives may be untrue or imperfectly specified as well as in later design and development stages. Because of human inability to perform and convey with perfection, software development is followed by quality assurance activity. While testing an application software product we first need to follow the principles of testing as they are the essential elements of testing.

We are testing our application on different mobile devices and discovering minute bugs appearing on those devices and try to solve them. Also testing for fetching location, crashing of the system, etc. is considered.

## 6.2 Testing Strategy

A standard screen in android is tested at four levels before it goes for production.

- Level 1 is generally the work to be tested by other formulator or other interns (this is typical first level of testing where focus is not on requirement but number of user testing) Ratio: 0% end user: 100% Technical.

- Level 2 is level where a senior programmer comes into the testing cycle of the screen that was unit tested by the developer in this phase the onus is to test software for technical requirements specified. Ratio: 80% Technical: 20% end user

- Level 3 is where a tester will come into picture. The tester will try and test the software for both end user as well as technical point of view. The ratio here is: 50% Technical: 50% end user.

- Level 4 is where we put together the code at Release-Ready. Here screen is tested to the core and each and every standard must be succeed and verified. Ratio here is: 80% User Testing – 20% Technical.

This allows us to text a screen at four levels and at the end of these procedure when the screen goes to production, it is generally bug free because more people have looked at this screen from variety of viewpoints.

1. **Unit Testing:**

   Unit testing focuses verification effort on the finest 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 bugs are limited by the constrained scope established for unit testing. The unit test is white-box oriented; the step can be directed in parallel for multiple components.

- Module interface is tested. It verifies the information flow. It checks that the information flows into or out of the program unit.
- Data structure is examined to ensure the storage of data. It maintains the integrity during all steps of execution.
- Testing of boundary conditions is also involved.
- It assures that the module operates properly at boundaries established to limit or restrict processing.

2. **Integrating Testing:**

   Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to disclose errors associated with interfacing. Objective is to take unit tested components and build a program structure that has been instructed in design.

- All components are combined in advance.
- The entire program is tested as a whole.
- A set of errors is encountered. Correction is difficult because isolation of causes is complicated by the enormous expanse of the entire program.
- Ones these errors are corrected, new ones appear and the Process continues in a seemingly boundless loop.
- The program is constructed and tested in small increments, where errors are easier to isolate and correct, interfaces are more likely to be tested completely and a systematic test approach may be applied.

**3. Validation Testing:**

- In validation testing, the software is assembled as a package. Validation Testing is completely associated with requirement satisfaction of customers. This testing checks weather all functional requirements of customer are satisfied or not. According to this test, the project is tested and found to be satisfactory for functional characteristics, behavioral characteristics and performance requirement. It is also found to have good documentation up to the final stage. So, the performance characteristics conform to specification and are accepted.

## 6.3 Testing Methods

### 6.3.1 Black Box Testing:

- It takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure.
- Black Box Testing is testing without knowledge of the internal workings of the item being tested.
- For example, when black box testing is applied to software engineering, the tester would only know the "legal" inputs and what the expected outputs should be, but not how the program actually arrives at those outputs. It is

because of this that black box testing can be considered testing with respect to the specifications.

- No other knowledge of the program is necessary. For this reason, the tester and the programmer can be independent of one another, avoiding programmer bias toward his own work. Due to the nature of black box testing, the test planning can begin as soon as the specifications are written.
- This method of test design is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance.

### 6.3.2 White Box Testing:

- As per our project, we have used white Box Testing Model because independent paths of ASP files and loops are the cornerstones of the vast majority of all algorithms implemented in the software.
- White-box testing sometimes called glass-box testing, where test data are derived from direct examination of the code to be tested. For glass box testing, the test.
- Cases cannot be determined until the code has actually been written. Both of these testing techniques have advantages and disadvantages, but when combined, they help to ensure thorough testing of the product. Software testing approaches that examine the program structure and derive test data from the program logic. Structural testing is sometimes referred to as clear-box testing since white boxes are considered opaque and do not really permit visibility into the code.
- All the possible combination of inputs is tested and works fine for each of the pages.
- All the decision testing are done on the both the true and false sides they work fine.
- All the simple, nested, concatenated, unstructured and continues loops are tested with the extreme values of the loop counter at maximum and minimum.
- All loops work properly.

# 7. SCREENSHOTS AND USER MANUAL

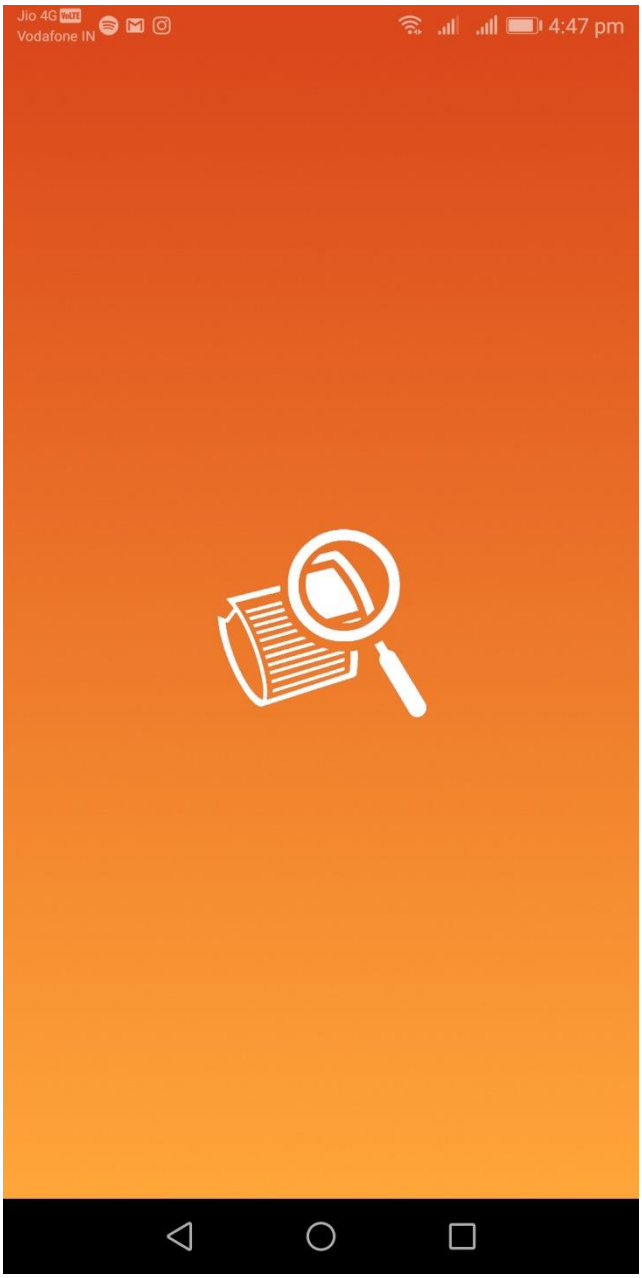



**Fig 7.1 Splash Screen**

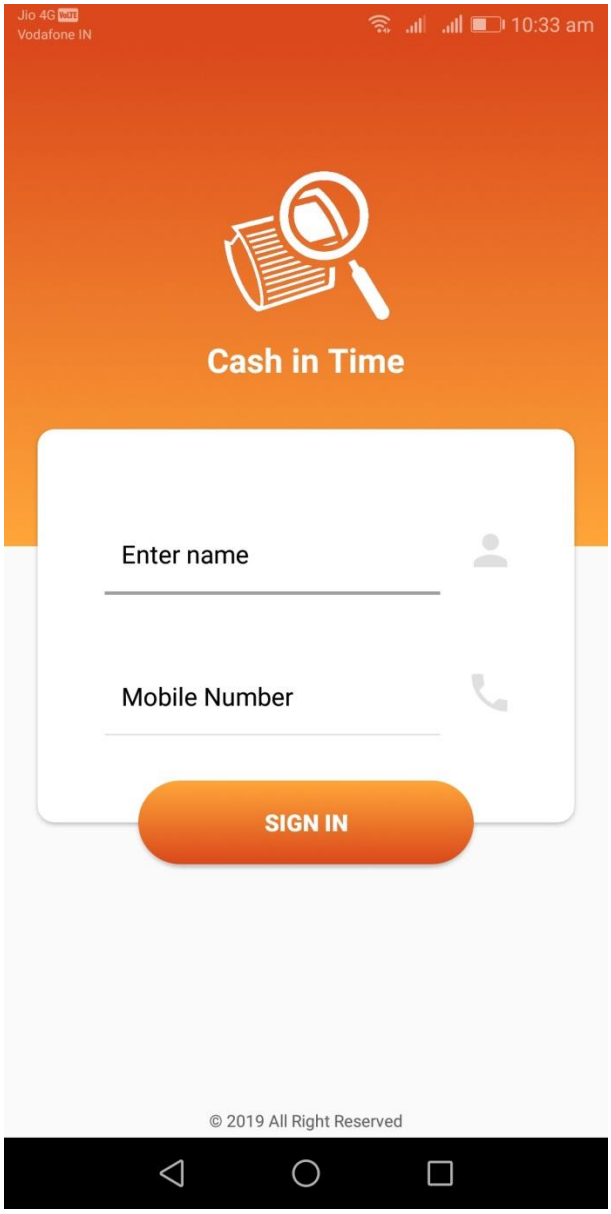



**Fig 7.2 Login Screen**

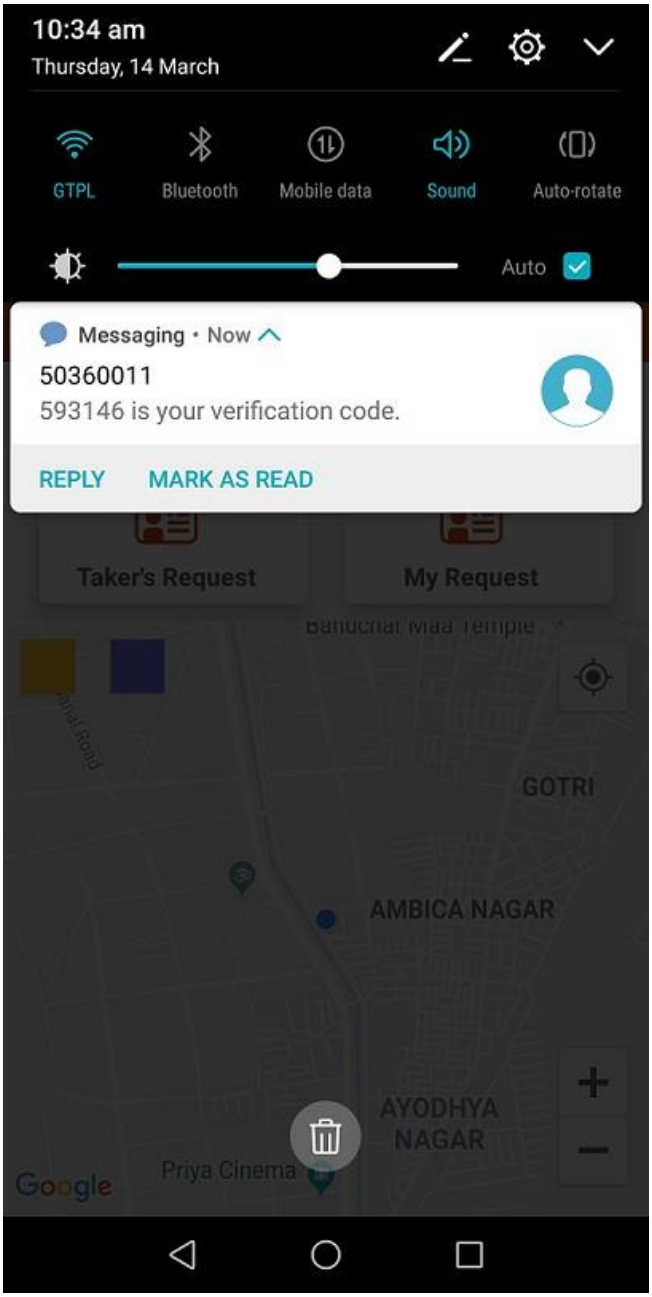



**Fig 7.3 OTP received**

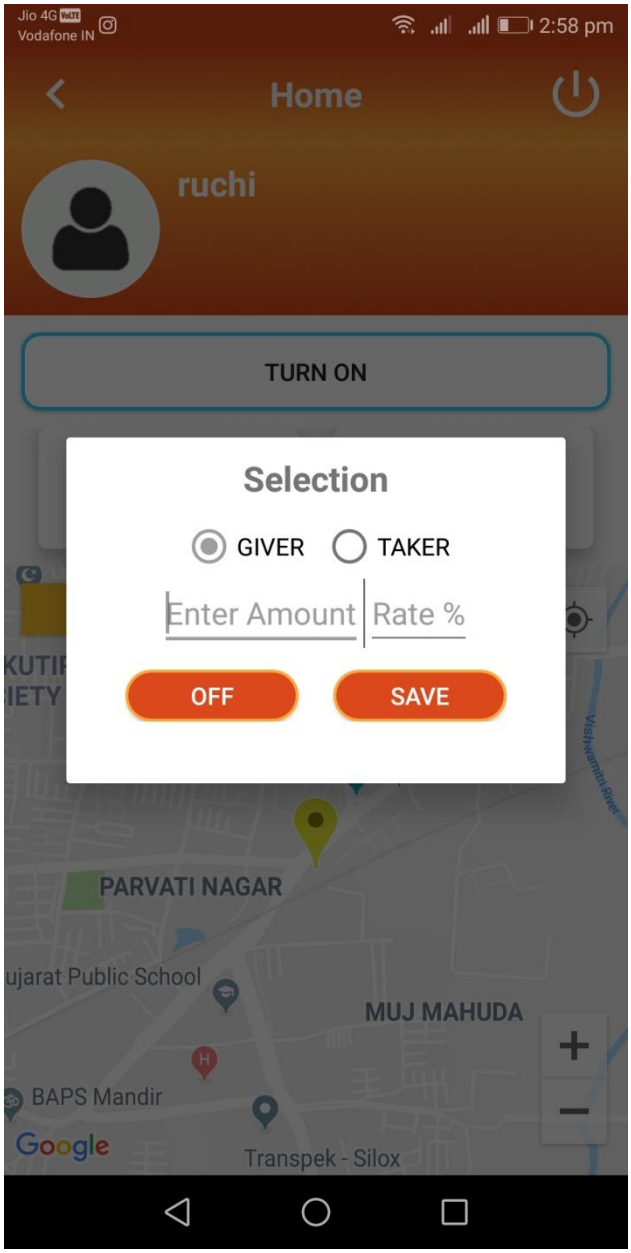



**Fig 7.4 Decide to be a giver or a taker**

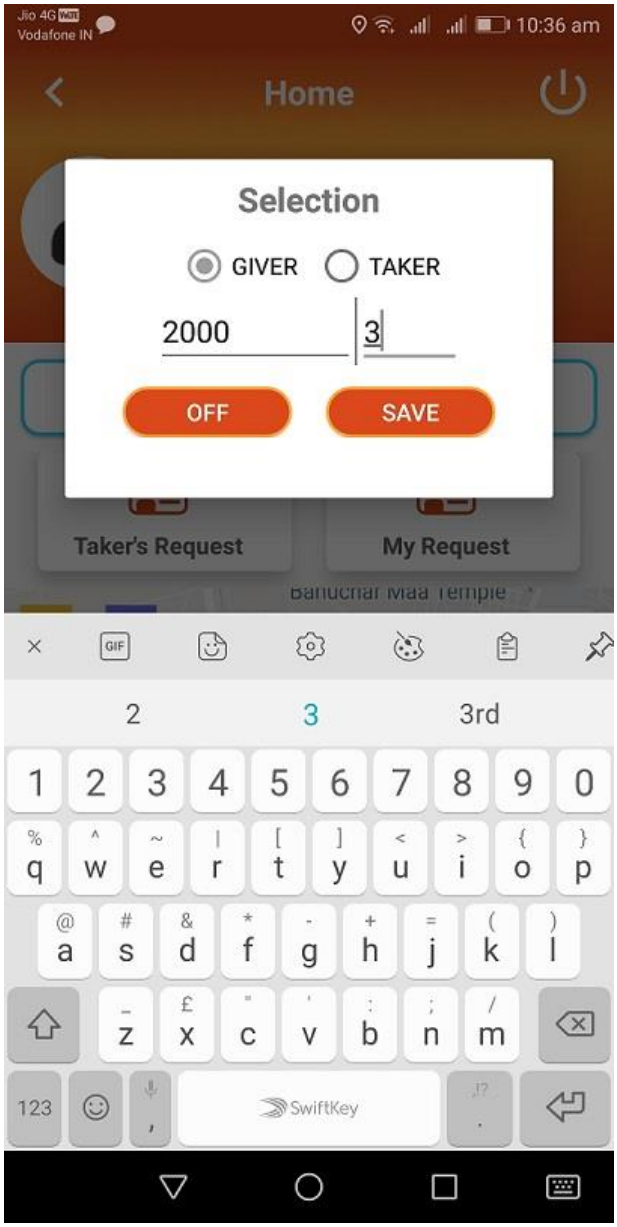



**Fig 7.5 Enter amount and rate**

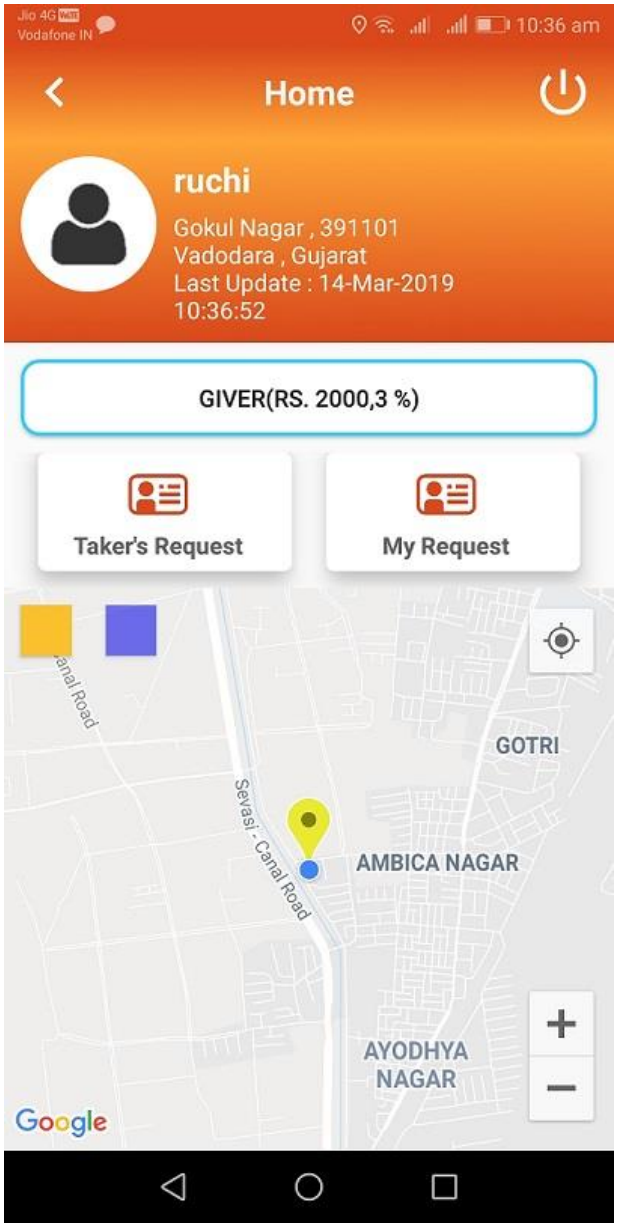

Jio 4G
Vodafone IN
10:36 am
Home
ruchi
Gokul Nagar , 391101
Vadodara , Gujarat
Last Update : 14-Mar-2019
10:36:52
GIVER(RS. 2000,3 %)
Taker's Request
My Request
GOTRI
Sevasi - Canal Road
AMBICA NAGAR
AYODHYA
NAGAR
Google

**Fig 7.6 Giver**

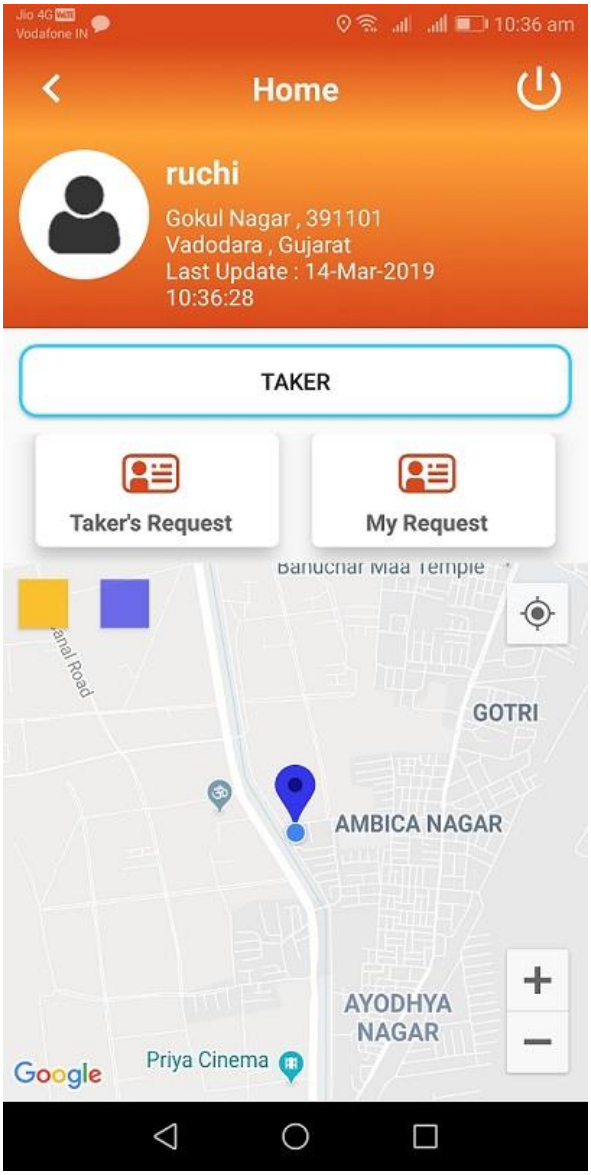

Jio 4G
Vodafone IN
10:36 am
Home
ruchi
Gokul Nagar , 391101
Vadodara , Gujarat
Last Update : 14-Mar-2019
10:36:28
TAKER
Taker's Request
My Request
GOTRI
AMBICA NAGAR
AYODHYA
NAGAR
Google
Priya Cinema

**Fig 7.7 Taker**

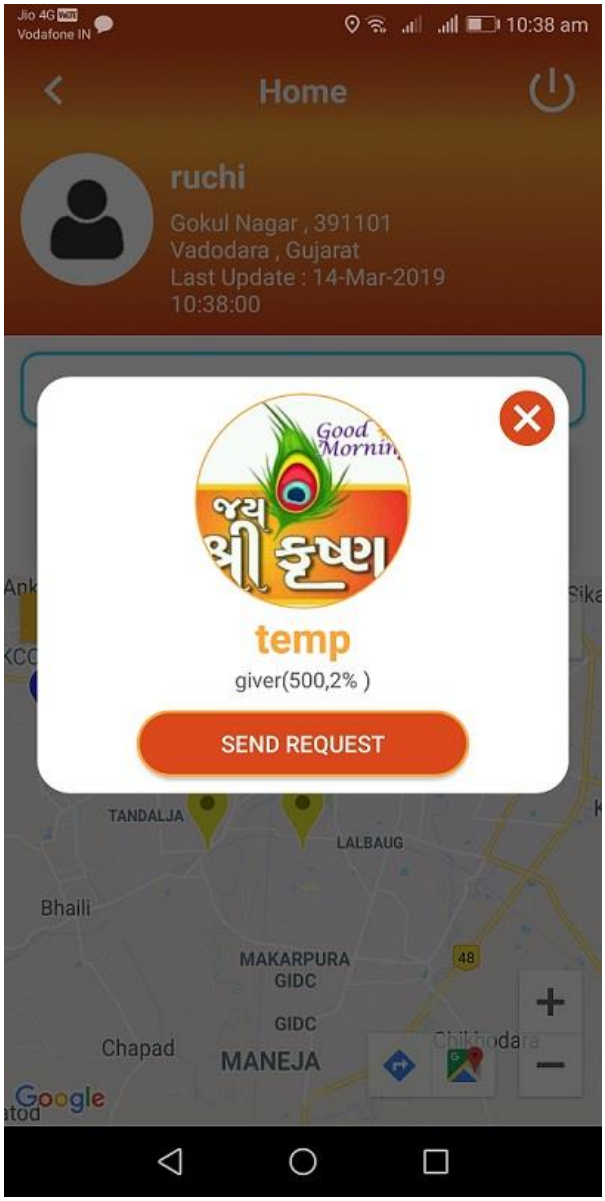



**Fig 7.8 Send request**

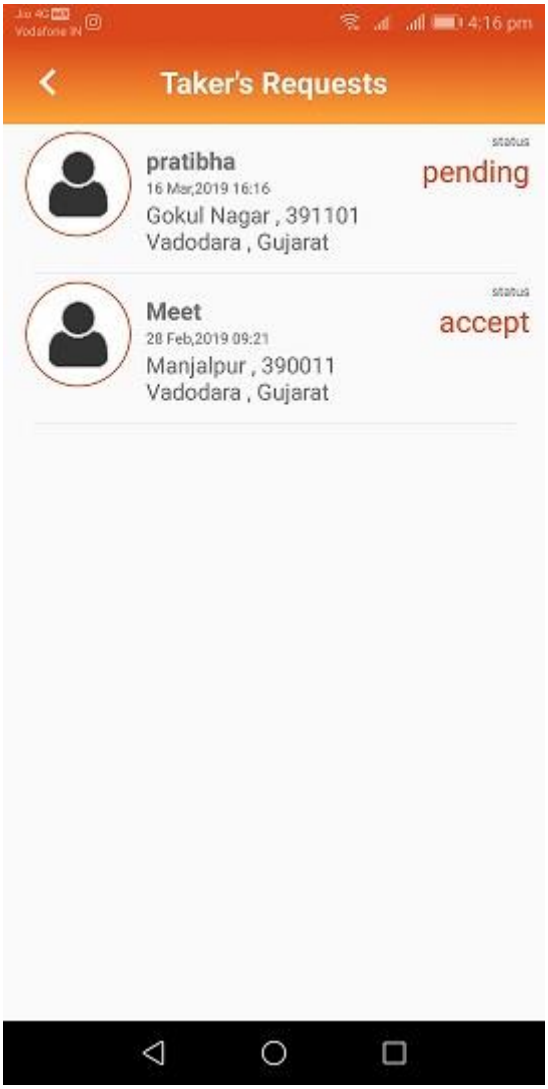



**Fig 7.9 Taker's request**

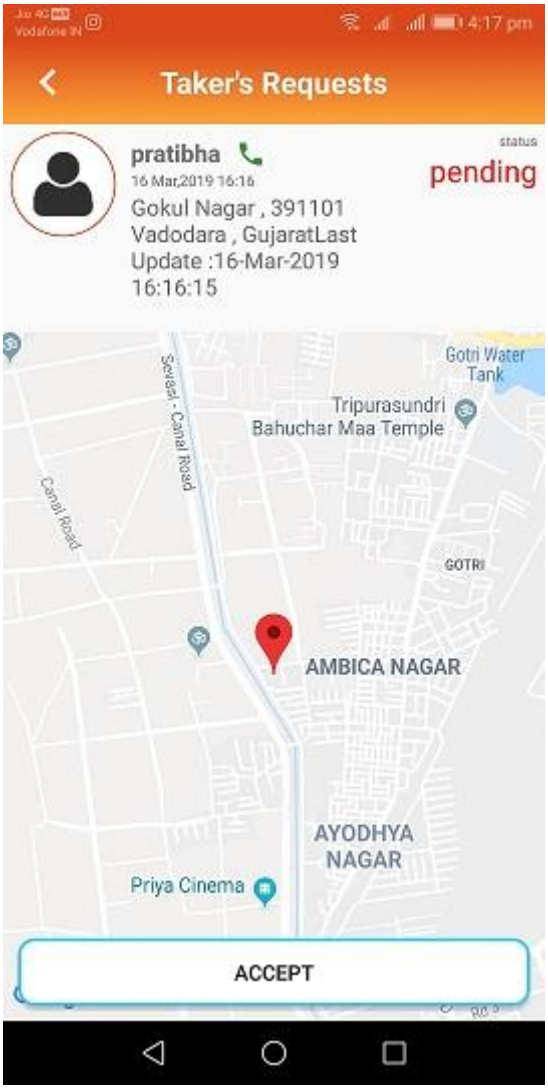



**Fig 7.10 Location and status of taker**


**Fig 7.11 Matching of code**

# 8. LIMITATIONS AND USER MANUAL

## Limitations:

- Application does not work in offline mode.
- Only calling facility is available in the mode of communication.

## Future Enhancement :

- For the future work, chatting facility would be availed.
- On each transaction, we would charge them some amount of money.

# CONCLUSION

Considering the practical problem of waiting in long queues for deposit or withdrawal of cash outside banks and ATMs. Lack of cash in nationalised banks and not enough denominations available in the ATMs. The problem when no ATMs or banks are found nearby. Also after a certain amount taxes are levied on transactions from other bank ATMs. So, in case of emergency, a person wanting ready cash can use the system ‘Cash In-Time’.

Here, the user has to feed the amount of cash he is in need of and can also set the radius within how much area he wants. So, a request will be sent to other users. The users willing to help can accept the request and meet at a place and can give the cash to the other person. The application also gives suggestion to the users about the nearest and convenient meeting place. They can select the place as suggested by the system or can also manually decide by communicating through chatting or calling facility available.

# REFERENCES

**https://patents.google.com/patent/US7765148B2/en?oq=US7765148B2** - Method and system for facilitating payment of an online auction transactions.

Patent No. : US7765148B2

Inventor: Dan A. German, Dominic J. Morea, Henry T. Tsui, John Duncan, Matt Golub

**https://patents.google.com/patent/EP0745961A2/en?oq=EP0745961A2** - Transaction authorization and alert system

Patent No. : EP0745961A2

Inventor: Greg E. Blonder, Steven Lloyd Greenspan, J. Robert Mirville, Binay Sugla

**https://patents.google.com/patent/US5825003A/en?oq=US5825003A** - Customer-directed, automated process for transferring funds between accounts using a holding account and local processing

Patent No. : US5825003A

Inventors: Horton Jennings, Nigel Pinnell, Khanh Do, Virendrakumar Shah, Marjorie Profumo, John Downing, Neil Goodhand, Marion Maino, Michael H. Thompson

**https://www.draw.io**

**https://en.wikipedia.org/wiki/Automated_teller_machine**

**https://www.bankrate.com/banking/checking/find-an-atm-with-these-smart-phone-apps**

**www.patents.google.com**