

Kisan Network (Hindi for ‘Farmer Network’)

Aditya Agarwalla
Advisor - Dr Robert Dondero

Department of Computer Science
Princeton University
Princeton, NJ, USA

Abstract

Stakeholders in the agricultural ecosystem in India lack awareness about resources available beyond their immediate local community. As a result, farmers suffer from fair price deprivation for their crops and poor accessibility for labor and other agricultural resources like equipment and transportation. However, since the influx of inexpensive Android smartphones and 2G / 3G data services, it has become possible for people to get information at their fingertips at low costs. This paper looks at Kisan Network, an Android application in English and other Indian regional languages, that takes advantage of these developments by connecting all the stakeholders in the ecosystem to form a marketplace to buy, sell and rent agricultural produce and resources.

1. Introduction

Agriculture in developing nations employs 48% of their total population [1]. In India alone, the agricultural (agri) ecosystem is 255 million people strong [2]. It comprises of five major stakeholders -

1. **The farmer / cultivator** who grows the crop.
2. **The buyer / middleman** who purchases the crops, either to use or sell further.
3. **The agricultural worker / labor contractor** who assists the farmer in the field with tasks like weeding, irrigation and harvesting.

4. **The equipment provider** who rents / sells farm and agricultural equipment like tractors and harvesters to the farmer.
5. **The transportation provider** who provides trucks, lorries and other forms of transportation to the farmer for raw material and final produce movement.

The interactions between the above stakeholders currently rely on traditional approaches that have been present in the ecosystem for hundreds of years. These traditional approaches primarily involve meeting the stakeholders in the local community in person to get the job done (Section 2). For example, if a farmer wants to sell his crop that he harvested recently, his only way of doing so is by going to the local village market and selling the produce to local buyers / middlemen, who often exploit him. This is because he is unaware of buyers outside his village and adjoining areas who may be offering a better price. Therefore, in most cases, he ends up selling his produce for a price that is lower than that available outside his village and as a result, suffers from **fair price deprivation**. The farmer faces a similar problem when he needs labor, agricultural equipment and transportation, as once again, he is solely dependent on people he knows in his village. Here, he suffers from **poor accessibility** and **high prices** for these agricultural resources because of a **lack of awareness about possible better deals outside his local community**. This plight of the farmers has been present for generations and has received extensive media coverage across the country. The Times of India has called it “exploitation by middlemen” [3] while the Epoch Times labeled the system “inefficient” and said that it is “monopolized by trading and middlemen communities” [4] (More examples of press coverage can be found in the Appendix). Similarly, the other stakeholders, being part of the same ecosystem as the farmers, suffer because of not knowing farmers outside their village and adjoining areas.

So, what has changed recently that enables us to try and solve this problem faced by a traditional and inefficient system? It is the **emergence of the mobile data enabled low cost smartphone**. More specifically, it is the rise of local Indian and Chinese cell phone manufacturers like Micromax, Karbonn and Lava that produce low cost Android smartphones and widespread 2G / 3G services by telecom operators like Bharti Airtel, Vodafone and Idea. In fact, smartphone sales in India increased by 84 per cent in the second quarter of 2014, compared to the same period in 2013 with the cheapest smartphone, manufactured by the New Delhi based Jivi Mobiles, being sold for \$32 [5]. To go with this, 3G services have also picked up in the country with a 114 per cent year-on-year growth in 2014 compared to the year before [6].

Because of these recent developments, it is now possible to utilize the power of the smartphone to solve the problems faced by farmers and other stakeholders in the agri ecosystem in the country. Kisan Network does exactly that. It is a **mobile only platform on Android that connects all the stakeholders in the agri ecosystem to form a marketplace to buy, sell and rent agricultural produce and resources**. It allows the stakeholders to negotiate and eventually, strike deals. It has been developed keeping in mind the target market's needs, limitations and expectations as follows -

- It adheres to the current practices involved in selling / purchasing / renting produce and resources, and negotiating deals. This is based on an extensive customer discovery focus group conducted in the two Indian states of Punjab and Uttar Pradesh in December 2014 – January 2015 where the problem / solution fit was assessed i.e. whether the proposed solution solves the problems faced by the stakeholders (Section 4.2).
- It incorporates interface design principles that accommodate the requirements of semi-literate people and of infrequent smartphone users (Section 4.3 – 4.5).

- The technology, design and ease of use of the application have been evaluated during a minimum viable product (MVP) evaluation survey conducted with farmers and other stakeholders in the state of Uttar Pradesh in March 2015 (Section 4.4).
- It has been iterated upon based on input from the target market that was received during the MVP evaluation survey (Section 4.5).
- It has been developed to work on inexpensive Android smartphones with limited local storage capacity, limited data connectivity and low RAM (Section 3.4).
- It has multi language support. It supports not only English but also Hindi and has been set up so that other regional Indian languages can be integrated in the future (Section 3.4).

2. The Problem

Let us consider a hypothetical farmer, Vijay Agarwalla, from the agriculturally diverse Indian state of Uttar Pradesh. He cultivates potatoes twice a year, once harvested in April, and then in November. To help him grow his produce, he requires agricultural workers to work with him in the field, to assist in tasks like sowing, harvesting etc. To procure workers, he reaches out to labor contractors in his village and other adjoining areas. He needs five workers to help him and the two labor contractors he contacts quote rates of ₹200 per person per day and ₹225 per person per day respectively. Both these rates are high as every farmer in the village grows potato as well and therefore, requires labor at the same time, driving prices up. But, in a village 100 kilometers away, there is an excess of workers because the main crop grown there follows a different cycle and the farmers do not require help at that time. However, Vijay does not know about their availability and opts for the best local option to get his work done. He faces similar scenarios while trying to rent a tractor for his field from local equipment providers and while

organizing transportation from local providers to take his harvested produce to the village market.

When it is time for him to sell his produce in April and November, he takes his potatoes to the village market and sells it to known buyers via the village middlemen. The best deal he is offered is ₹100 per kg but, later in the day, he learns from his friends that potato is selling for a minimum of ₹150 per kg outside his village. Not knowing any buyers in other villages, he once again decides to stick to his local option and sells his produce at the lower price.

What we see above in the case of Vijay is that farmers in India **lack awareness about resources beyond their immediate local community**. This is because of the existing traditional system that has been in place for generations. As a result, they face the following two problems -

- Lack of **fair prices for their produce**. They do not have complete knowledge about the best way to sell their own produce and as a result, receive lower prices than what they could have received outside their village.
- **Poor accessibility** and **high cost** for agricultural resources (like equipment and transportation) and labor. Once again, they are restricted by their local community and eventually end up paying higher prices, despite better rates being available a few kilometers away.

3. The Solution

3.1 Overview

To solve Vijay's problem, it is necessary to provide him access to information, resources and people from other parts of the country without having to physically step outside. To tackle

this, I developed Kisan Network. It is an **Android based mobile application that connects the farmer, the center of the agricultural ecosystem, to all the other stakeholders – the buyer / middleman, the labor contractor, the equipment provider and the transportation provider.**

With it, farmers get access to better prices because of greater choice in buyers and increased accessibility to potentially cheaper agricultural resources. Buyers also get the ability to transact with farmers beyond their own village and this provides them with a chance to increase their income. Finally, labor, equipment and transportation providers simply get more customers and are able to earn more.

The application is capable of running on inexpensive smartphones with memory and RAM restrictions, which need not be connected to the Internet at all times (Section 3.4). In this paper, I describe the crop transaction between a farmer and a buyer / middleman.

3.2 Flows

3.2.1 Flows for the farmer

To better understand how Vijay would be able to sign up for this application and interact with buyers / middlemen, let us have a look at the application flows for a farmer in the following sections.

3.2.1.1 Sign up

Firstly, Vijay would need to sign up for Kisan Network to be able to use the application (see Figure 1). The sign up process consists of entering the following basic details – title (Mr./Mrs./Ms./Dr.), first name, middle name (optional), last name, sex, date of birth, age, phone number, email address (optional), role (farmer/buyer), location and finally, a maximum of four

crops that he cultivates from a list of crops that are associated with his district. If at any stage, he requires assistance in understanding what a particular step of the sign up process is asking for, he can click on the audio icon at the top right corner of the screen (circled in red in the first screen in Figure 1) to start a short audio clip that gives him relevant instructions.

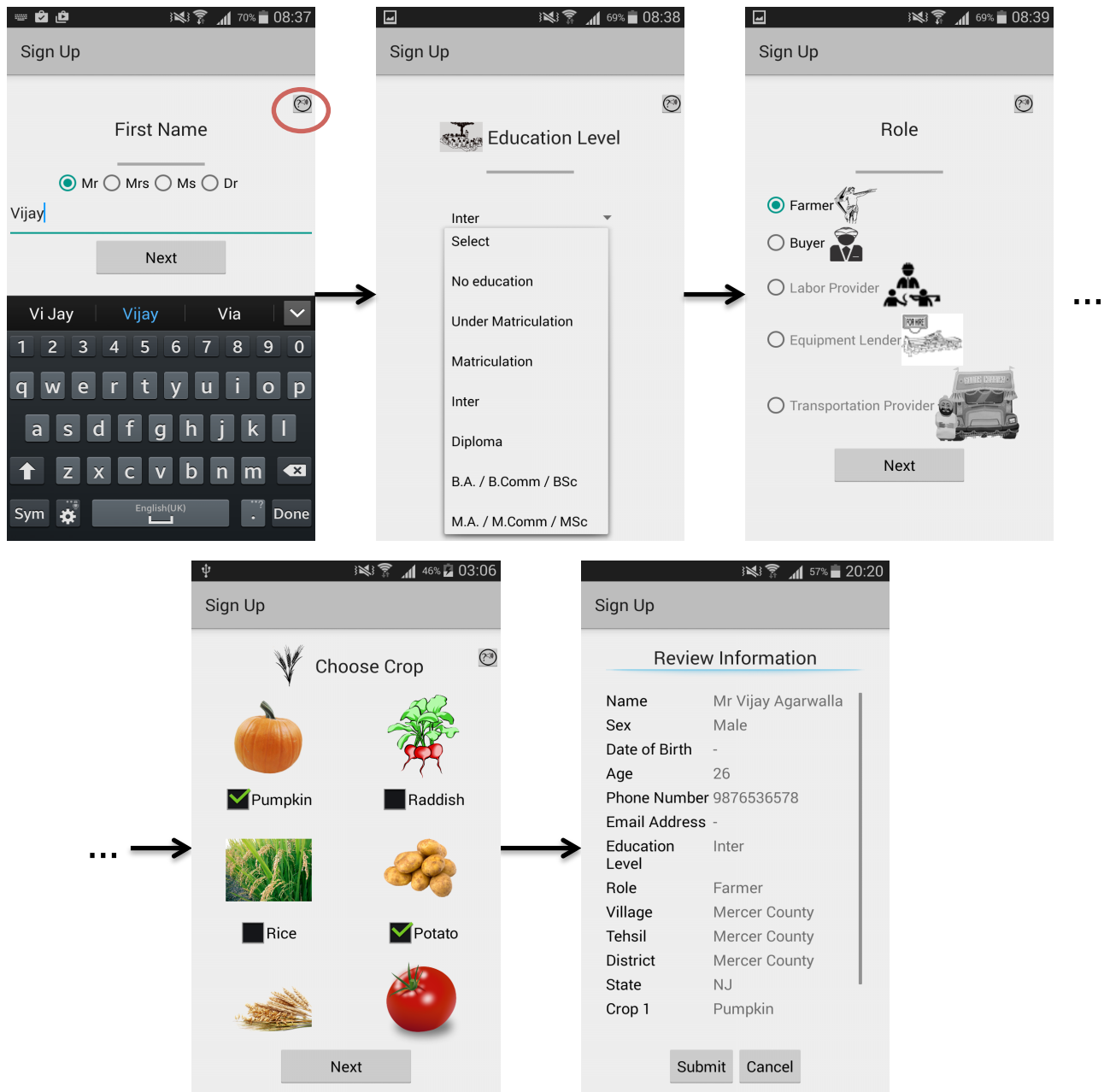
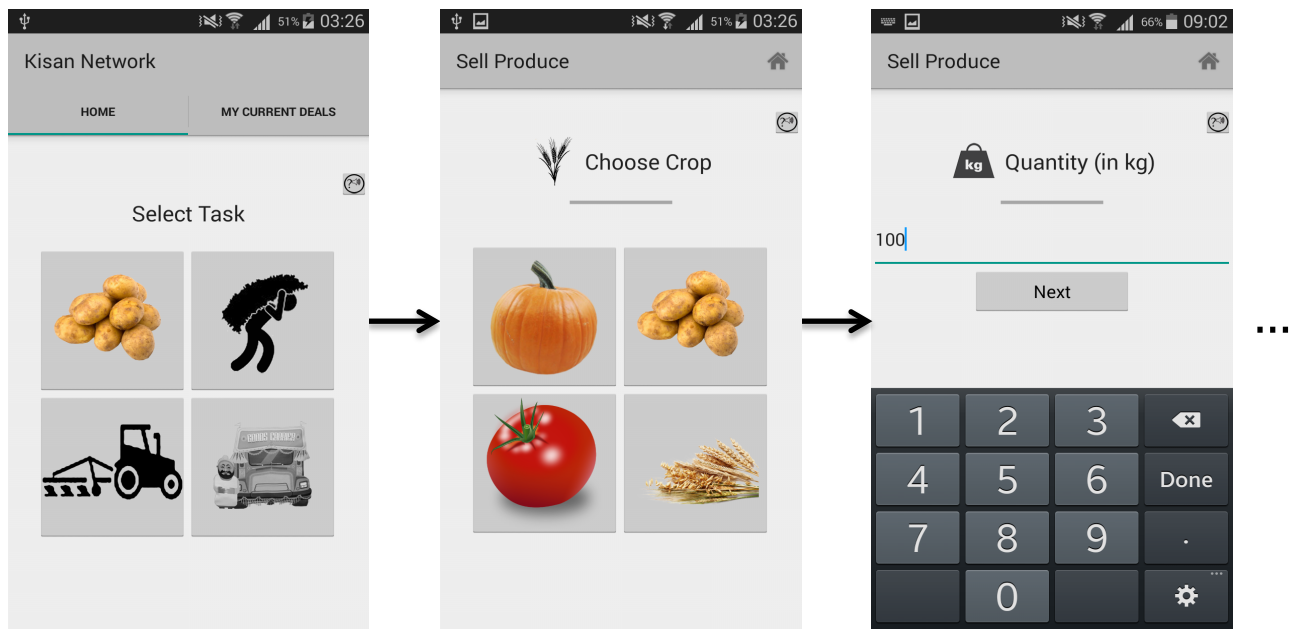


Figure 1 - Sign up Flow (all steps have not been shown)

3.2.1.2 Sale of Crop

After signing up, Vijay can select one of four tasks – put his crop up for sale (top left square in first screen of Figure 2), request for labor (top right square in first screen of Figure 2), request for equipment (bottom left square in first screen of Figure 2) or request for transportation (bottom right square in first screen of Figure 2). For this semester’s Independent Work, the first of the four tasks has been implemented. So to put his crop up for sale, Vijay selects the top left square and then proceeds to enter basic details about the crop – crop name (for example, potato), crop variety (for example, ‘Red Nanital’), crop quantity (in kg), crop rate per kg (in ₹), crop sale date and finally, any comments in addition to the information provided so far (optional). The final screen is a confirmation page that allows him to review the information filled in before submitting his sale offer.



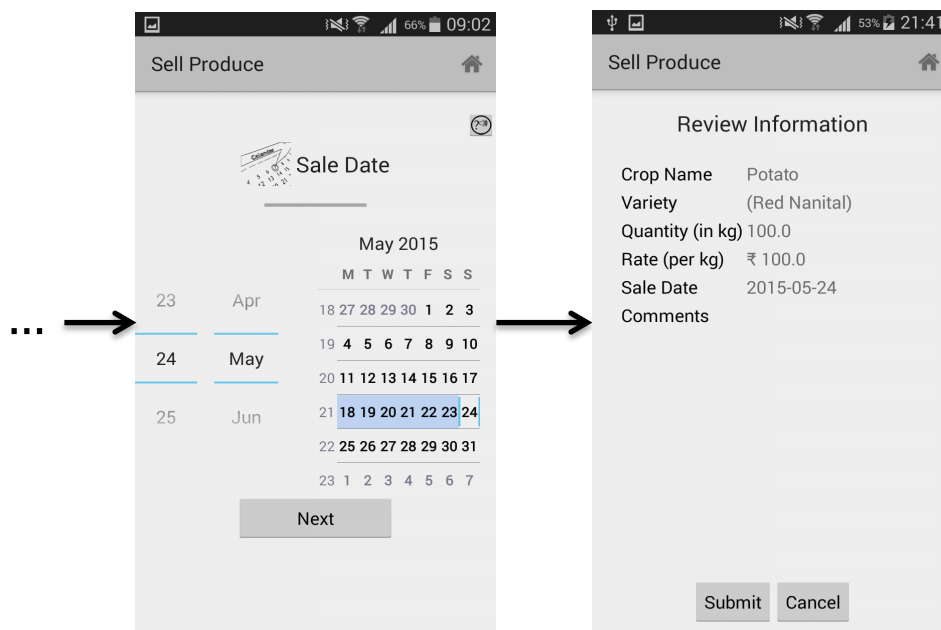


Figure 2 - Crop Sale Flow (all steps have not been shown)

3.2.1.3 Checking status of current deals

After putting up his crop for sale, the application takes Vijay back to the home page (first screen in Figure 2). Here, there are two tabs available to him – the ‘Home’ tab to initiate a crop sale offer and the ‘My Current Deals’ tab to look at the status of his current crop sale offers. The latter is where he will find his most recent sale offer for potato with details like crop variety, rate, quantity and sale date visible (first screen of Figure 3). Clicking on the deal takes him to a list of purchase responses received from buyers for that particular deal. If there aren’t any offers, a small popup, called a toast, is shown telling him so and he is taken back to the home screen. But, we can see in Figure 3 below that there are purchase responses. Each purchase response contains the following details – name of buyer, crop rate the buyer is willing to offer, the quantity that the buyer desires and finally, the date on which the buyer wants the crops (Section 3.2.2.2). Let us

assume that Vijay is interested in Vikas Das’ (a hypothetical buyer) response as he is offering the highest rate for his produce. So, he clicks on it and moves to the next screen that shows further details about the response like additional comments provided by Vikas and an option to either accept or negotiate the deal.

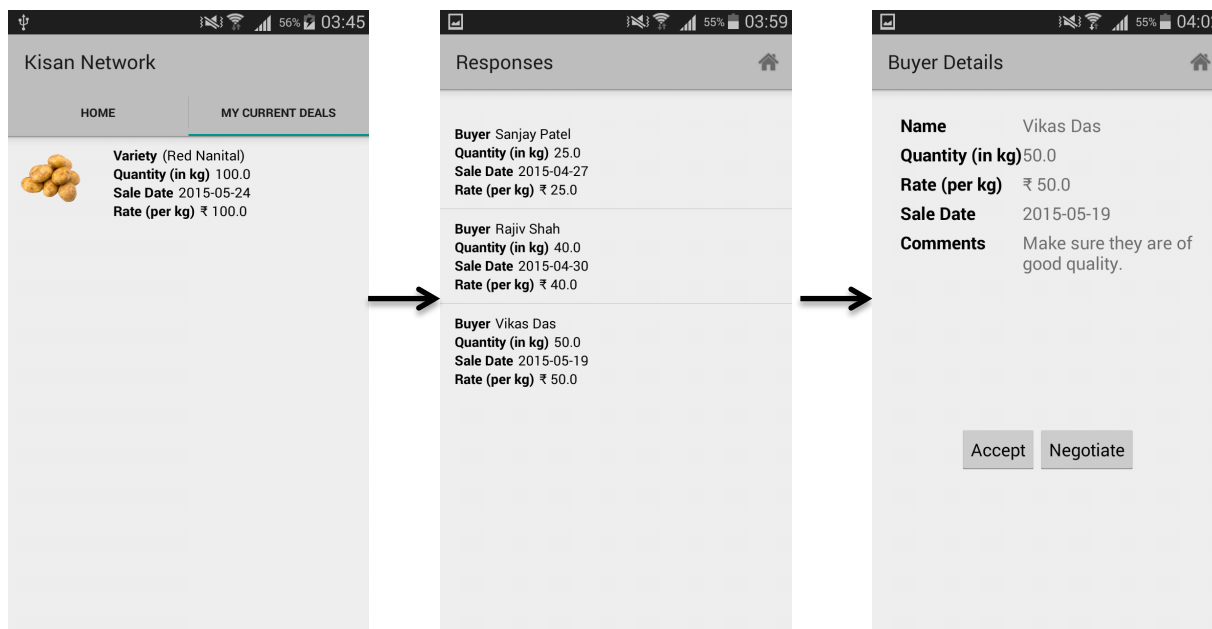


Figure 3 - Current deal status flow

3.2.1.4 Negotiating a deal

If Vijay would like to accept the deal offered by Vikas, he can click on the ‘Accept’ button (as seen in last screen of Figure 3). However, let us assume that he would like to negotiate a better deal. So, he clicks on the ‘Negotiate’ button. On doing so, he is taken to a series of negotiation screens – crop quantity, crop rate, sale date and comments (optional). Each screen displays the corresponding value that was last offered by Vikas. At the end of these steps, Vijay sees a confirmation screen with details about his updated offer that he can submit. If Vikas is still

not happy with Vijay’s latest offer and responds with another updated offer, Vijay can repeat the steps described in Section 3.2.1.3 and 3.2.1.4.

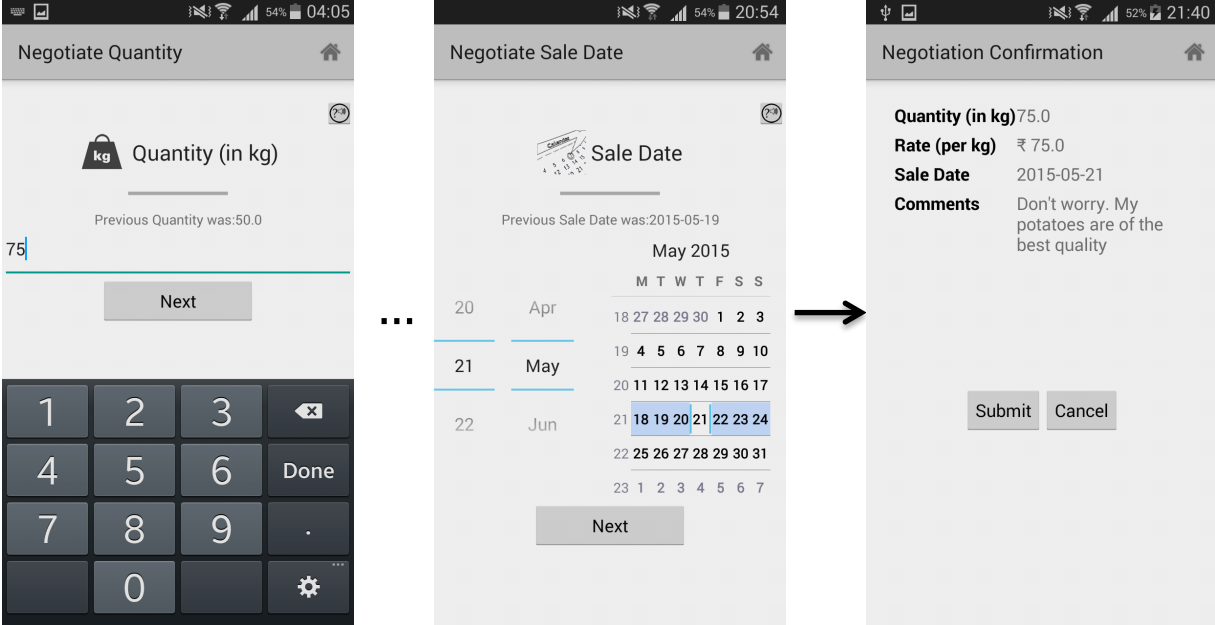


Figure 4 - Negotiation flow (all steps have not been shown)

3.2.2 Flows for the buyer

To understand how the buyers / middlemen get involved in this marketplace for agricultural produce, let us take the example of Vikas, who is a buyer of potatoes and learns about Vijay’s sale offer on the application (as mentioned earlier in Section 3.2.1.3).

3.2.2.1 Sign up

The sign up flow for Vikas is identical to that of Vijay (refer to Section 3.2.1.1) where the last step requires him to select a maximum of four crops that he buys, from a list of crops available in his district.

3.2.2.2 Viewing sale offers for crops

After sign up for Kisan Network, Vikas sees a grid with pictures of the four crops he chose during the sign up process on the home screen. Since he is interested in potatoes, he clicks on its image and is then shown a list of sale offers for potato where each offer displays the following details – crop variety, crop rate, quantity and sale date. He is interested in one of the offers and clicks on it to be taken to the next page to see further details about the offer, including the name of the seller, who happens to be Vijay in this case (as we learnt in Section 3.2.1.3). If he would like to accept the offer, he can click on the ‘Accept’ button. However, let us assume that he would like to negotiate. He then proceeds with the steps highlighted in Section 3.2.1.4 and submits his offer, which can be seen in Figure 3.

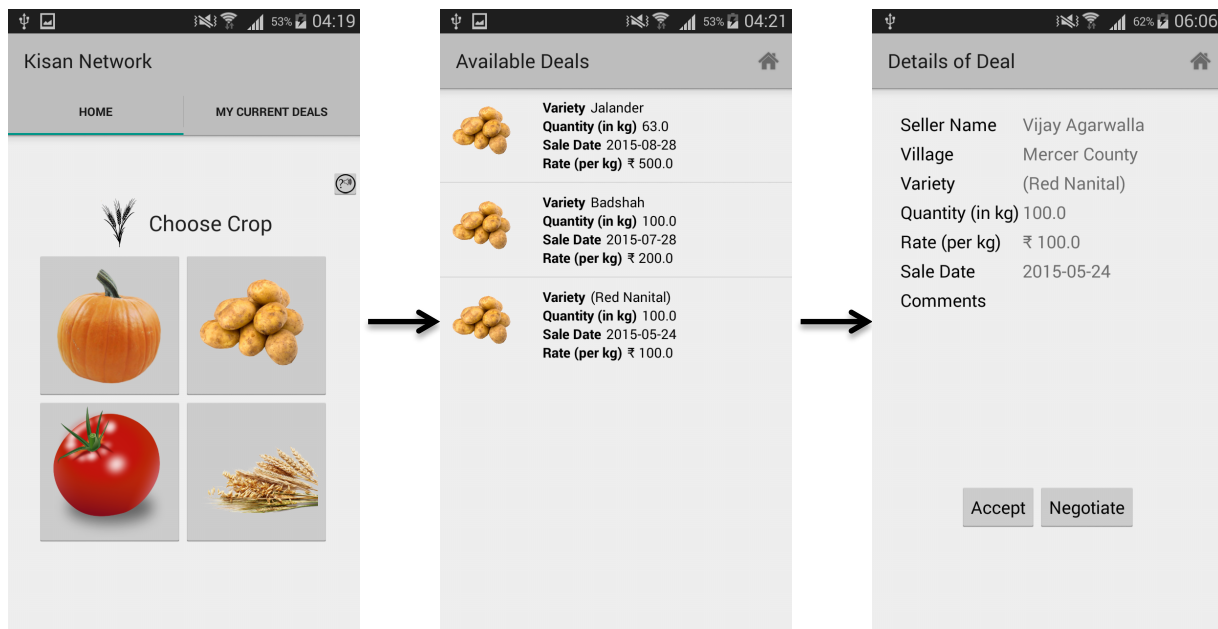


Figure 5 - Flow to view crop sale offers

3.2.2.3 Checking status of current deals and negotiating

Once a negotiation offer has been sent by Vikas, he can access this current deal from the home screen. On clicking the corresponding tab, he is shown a list of current deals he is involved in, where each deal is represented by the seller's name and crop variety. Clicking on Vijay's deal takes Vikas to a detailed view of the last offer by the seller for that deal. If Vijay has responded to his negotiation offer, he will see it on this screen (we know that Vijay responds with another offer from Section 3.2.1.4). If he hasn't, then the last offer from Vijay will be shown. Vikas can re-negotiate from this screen following the steps illustrated in Section 3.2.1.4.

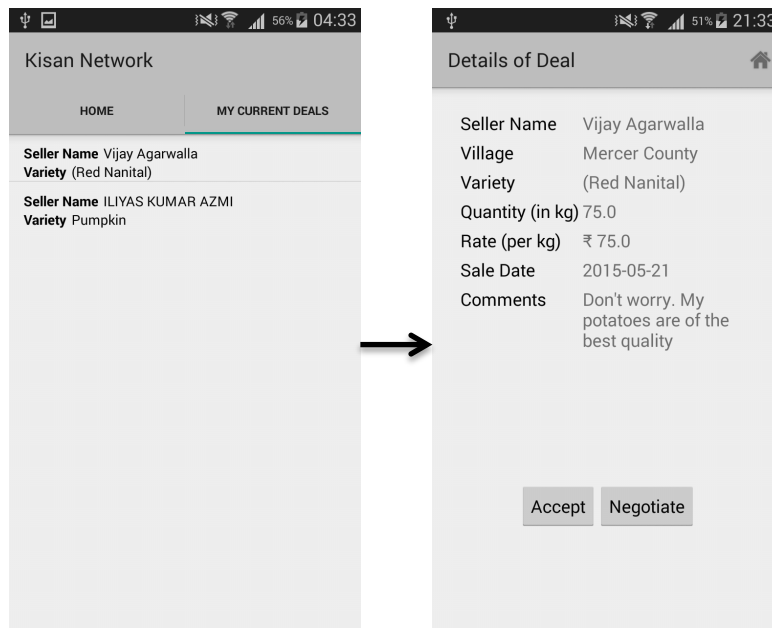


Figure 6 - Flow to check status of current deals

3.3 Implementation Details

3.3.1 Overview

Kisan Network has been developed for smartphones that run the Android operating system. Its backend runs on a Google App Engine (GAE) instance that is connected to Google

Cloud SQL, a MySQL database that lives in Google’s cloud. The connection between the Android client and the App Engine mobile backend is done using Google Cloud Endpoints (GCE) that enables generation of APIs and client libraries from the backend, thereby simplifying client access to data from other applications (in this case, Google Cloud SQL) [7]. GCE also enables the extension of the same API to iOS and mobile web as well, thereby providing the flexibility to expand to other platforms in the future (as seen in Figure 7).

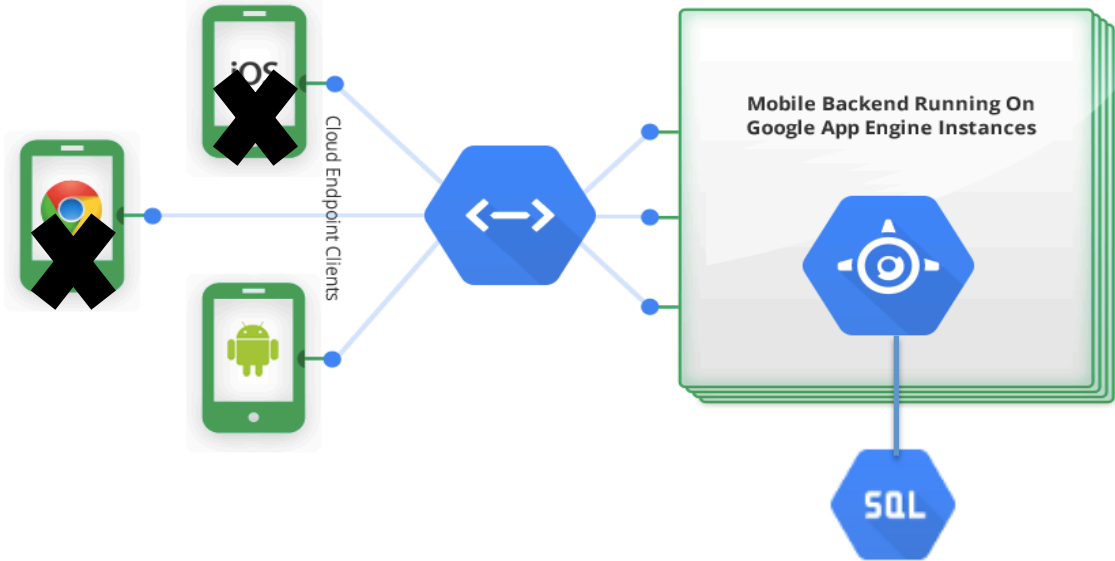


Figure 7 - System Design Overview [7]

The entire application consists of **~11,500 lines of code**, not including any auto generated libraries, comments and blank lines. This includes ~7000 lines of Java code (for the Android client and GAE backend) (Section 3.3.2 and 3.3.3), and ~4500 lines of XML code for the Android client (for screen layouts, menu layouts and strings XML) (Section 3.3.2). The Google Cloud SQL database currently consists of 30 tables (Section 3.3.4) of size ~275 MB.

(Because of its size, the code cannot be included as a part of this paper. It is available upon request. Please reach out to me at adi.agarwalla@gmail.com for more information.)

3.3.2 Android Client

The Android client primarily consists of the screens, called activities, which were discussed in Section 3.2. Each activity consists of the following three components -

- An XML file describing the layout of the elements on the screen.
- An XML file describing the layout of the menu bar appearing at the top of every screen.
- A Java class for the activity that has the following roles, including but not limited to -
 - Deciding which of the components specified in the XML files are displayed.
 - Assigning functionality to the user interface (UI) components on the activity screen.
 - Handling actions performed at any stage when the screen is being displayed such as click on a button, transition to another screen, display of a toast, playing a sound, etc.
 - Calling asynchronous tasks to connect to the App Engine backend in the background.

In addition, all the images and icons displayed in the application are included as a part of the client. Each image should be present in multiple densities (a maximum of six) in terms of dots per inch (dpi) values to adjust to different screen sizes and resolutions. For Kisan Network, the images are present in up to five densities, ranging from ldpi (low), which is ~120dpi, to xxhdpi (extra-extra-high), which is ~480dpi.

Lastly, all the textual information displayed in the activities is stored in an XML file with a separate key to identify each string. This allows easy updates to strings as instead of needing to change a string in every activity, we can just change the value for the key once in the XML file. It also enables the application to be supported in multiple languages, as a new XML file with the

same keys for the strings is all that is needed for a new language to be added. In the case of Kisan Network, the additional XML file supports the application in Hindi and consists of all Hindi content in the application, with keys identical to their equivalents in the English XML file. For instance,

```
<string name="fName">First Name</string>
```

and

```
<string name="fName">पहला नाम</string>
```

Here, the key “fName” enables the string “First Name” to be shown in the English version of the application, where as it allows “पहला नाम” to be shown in the Hindi version. This can be seen in the TextView component below -

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:text="@string/fName"  
    android:id="@+id/textView" />
```

3.3.3 Google App Engine mobile backend with Cloud Endpoints

Kisan Network is built on Google’s infrastructure and uses its Platform as a Service (PaaS) offering called Google App Engine (GAE). GAE distributes requests for an application running on it across multiple servers, and is built to scale to meet traffic needs [8]. Each application runs within its own secure environment that is independent of the hardware, operating system or physical location of the server. Its free quota per day allows 1GB of incoming and outgoing network traffic and 5GB of cloud storage [9], which is sufficient for Kisan Network at this stage in its development.

The Android client is connected to the GAE mobile backend using Google Cloud Endpoints (GCE). GCE helps create an API for the GAE backend application for the Android client to use. It works as follows -

- An entity class is created. For instance, User is an entity class for Kisan Network used during the sign up process which looks as follows -

```
public class User {  
    private String userId;  
    private String title;  
    private String firstName;  
    private String middleName;  
    ...  
    public String getUserId() { return userId; }  
    public String getTitle() { return title; }  
    ...  
    public void setUserId(String data) {userId = data; }  
    public void setTitle(String data) { title = data; }
```

It consists of private fields for the entity object and public methods to set values for the fields and also obtain the field values.

- The backend API is created, which exposes the entity objects to the client and provides API methods for adding, deleting, updating and searching entity objects. By creating this API layer, we create a RESTful (Representational State Transfer) service that enables us to use HTTP to perform basic CRUD operations (Create, Read, Update, Delete). Thereby, the client application can be completely agnostic to the server side code as HTTP binds the two together [10]. This provides the flexibility to extend the application to an iOS client or web client in the future, as all of them will need to invoke the same HTTP API. For Kisan Network, endpoints exist for the following classes -
 - **User** class for posting and retrieving user information during processes like sign up.

- **DistrictCommodity** class to display crops associated with a location during sign up.
 - **Offer2Sale** class to submit and retrieve crop sale offers by farmers.
 - **SalesResponseList** class to retrieve responses to a crop sale offer during the negotiation process, both from the buyer and the farmer.
- An AsyncTask is created in the Android client that connects to the backend API asynchronously. This allows performing background operations and publishing results on the UI thread as it runs separately. After connecting to the API, it can call methods in the API to perform CRUD operations.

3.3.4 Google Cloud SQL

Kisan Network's MySQL database resides in Google's Cloud via Google Cloud SQL, which is a relational database. It is currently running at the second entry level D1 tier that provides 0.5GB of RAM, 250 maximum concurrent connections and 1 GB included storage with a max storage capacity of 500GB. The relational database structure is as follows (only major tables discussed) -

- **UserMaster** table consisting of all the details entered during the sign up process (Section 3.2.1.1). A unique user identifier is generated for each user, which is used to relate it other tables like *UserLocation_IND*. The unique user identifier is generated using the following command -

```
SELECT CONCAT ('IN',LPAD((MAX(CONVERT(RIGHT(UniqueUserID,12),SIGNED
INTEGER))+1),12,'0')) FROM UserMaster;
```

First two characters are the country code followed by a sequential left 0 padded numeric

number of length 12. A 12 digit sequential number is sufficiently large to handle any number of users in the future. For example, IN000000000393 is a unique user identifier.

- ***UserLocation_IND*** table consisting of location details for each user. Columns include latitude and longitude values; and state, district, sub-district and town/village codes. This table is intentionally de-normalized to provide quicker querying for state, district, sub-district and village names using the codes mentioned above. These codes are used to link this table to *State*, *District*, *Subdistt* (i.e. Subdistrict) and *TownVillage* tables respectively, each of which contain the codes and associated names for all the four levels of location. If *UserLocation_IND* weren't de-normalized, it would not contain the codes for all the 4 levels of location. Rather, it would simply contain the town/village codes and this would result in multiple join operations for certain queries, which would be slow. For instance, to get the state name based on town/village code, we would first need to join *UserLocation_IND* with *TownVillage* and then further join it with *State*.

Currently, the tables have been populated for the state of Uttar Pradesh (UP) where the application will be rolled out first. This means that the *State* table has 1 row where as the *TownVillage* table has 1142 representing the number of villages/towns in UP.

- ***Crop2DistMapping*** table that links each district in the country to the crop varieties found there. This data has been collected from the Open Government Data (OGD) Platform India (data.gov.in) [11]. Currently, the table is updated for all the districts in UP and is used during the sign up process when the farmer/buyer selects crops he deals in from a list of crops in his district (Section 3.2.1.1).
- ***Offer2Sale*** table that stores details of each crop put up for sale by a farmer (Section 3.2.1.2). A unique transaction identifier is generated for each offer, which is used to

relate it to other tables like *SaleResponse*. The unique transaction identifier is generated using the following command -

```
SELECT CONCAT
('TI',CURDATE()+0,LPAD((MAX(CONVERT(RIGHT(TransactionID,6),SIGNED
INTEGER))+1),6,'0')) FROM Offer2Sale;
```

The first 2 characters are just “TI” followed by current date in the YYYYMMDD format, which is itself followed by 6 digit sequential number allowing close to 1 million transactions a day. For example, TI20150426000033 is a unique transaction identifier with the date 2015-04-26.

- ***SaleResponse*** table that stores details for each to and fro negotiation communication between buyers and sellers for all the deals. It consists of all the details about the negotiation, which are identical to the details about a sale offer, as present in the *Offer2Sale* table (Section 3.2.1.4).
- Some other tables include ***CommodityMaster*** that stores crop names (like potato, wheat etc.) along with unique crop identifiers, ***VarietyMaster*** that stores crop variety codes and crop commodity codes (linking it to *CommodityMaster*) associated with the crop variety names (like Badshah and Red Nanital), ***Role*** table that stores the role for each user along with the unique user identifier to associate it with *UserMaster* and ***FinalSaleDeal*** that will store details of the final sale agreed to by both the buyer and the seller. Tables analogous to the crop transaction tables mentioned above have been set up for the ‘Labor Request’ feature as well like ***LaborRequirement*** (~*Offer2Sale*) and ***LaborResponse*** (~*SaleResponse*). For the ‘Transport Request’ and ‘Equipment Request’ features, tables

like *EquipmentNameMaster* containing equipment names and unique codes and *TruckDetails* containing truck details and codes have also been set up.

3.4 Technical Challenges

Let us have a look at some of major technical challenges faced this semester during Kisan Network's development -

- **Deciding which mobile backend and database to use**

For the mobile backend development for Kisan Network, there were three options that I was considering - Parse, Amazon Web Services (AWS) Mobile Services and Google App Engine (GAE). Infrastructure as a Service (IaaS) platforms like AWS (Note - This is different from the AWS Mobile Services mentioned earlier) and Google Compute Engine were not being considered as these are self service models for managing remote datacenter infrastructures [19] and require extensive knowledge and effort to build and maintain. Now, let us have a detailed look at the three options that were considered -

- **Parse** is a mobile app platform, owned by Facebook that is based on the Backend as a service (BaaS) model. Its documentation is excellent and it also has a large developer community. However, despite these benefits, Parse was ruled out first because of certain limitations that could potentially adversely impact development in the near future. One is that it can fetch a maximum of 1000 results in one query which can be an issue once the application scales even moderately. In addition, it does not allow distinct queries that retrieve unique values which are an integral part of every relational database [12]. Finally, being a BaaS model, it provides a

lot more abstraction than a Platform as a Service (PaaS) offering, which can also be a problem as the application scales.

- **AWS** Mobile Services provides the AWS SDK for Android, which is an open-source development kit, distributed under an Apache Open Source license [13]. It is also based on the BaaS model that allows the developer to access Amazon's extensive cloud infrastructure and services and was launched only recently in July 2014. This was the primary reason it was ruled out. It is a new product that has not matured yet and being based on the BaaS model (that provides a significant amount of abstraction), there were concerns about opting for it.
- **GAE** is a PaaS offering that provides access to Google's cloud and infrastructure. It was the best option not only because it meant lesser abstraction than Parse and AWS Mobile Services, but also because it provided the option of using GCE that simplified generation of APIs and client libraries.

For the database, the two options that were being considered were a SQL relational database and a NoSQL database. The latter is a more recent option that has been made available to developers and has gained a lot of popularity quickly as it is known to scale better than a relational database. However, it models data differently than tabular relations used in a traditional relational database and therefore, is a steep learning curve. As a result, a relational database is the best option because of my prior experience in using it and also because it is a mature concept. For a relational database, Google Cloud SQL was selected because it works well with GAE using GCE.

- **Supporting old Android versions**

Kisan Network has been developed such that it can work on older Android OS versions, dating back to Android Gingerbread (API Level 9) that was launched in December 2010; thereby, covering 99.5% of all Android devices connected to the Google Play Store [14]. Currently, the most recent version is Android Lollipop (API Level 22). This vast difference in the latest API level and the minimum API level the application supports resulted in some latest functionalities not being supported. Let us consider one of the major ones –

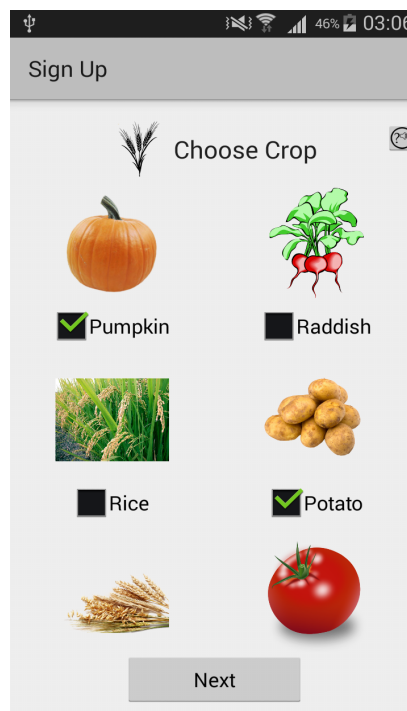


Figure 8 - Crop Selection Grid View

This is the activity screen during the sign up process that requires the farmer and the buyer to select a maximum of four crops they deal in. The underlying structure behind

this activity is a *GridView* component that allows multiple selection. The most common way to achieve this task is using the following *GridView* method –

setChoiceMode(GridView.CHOICE_MODE_MULTIPLE)

However, this method is not supported by versions below API 11. To get around this, I had to essentially incorporate my own multiple selection option in the following manner -

- a. Convert each grid item to a custom class containing an *ImageView*, a *CheckBox* and an identifier.
- b. Maintain a separate Boolean array (of dynamic size equal to the number of crops available in the user's district) that keeps track of whether the checkbox for a grid item is checked or not.
- c. Create a custom adapter called *ImageAdapter* that not only populates the grid with the values returned from the database call, but also treats each grid item individually and updates the Boolean array when the checkbox associated with a grid item is checked.
- d. Use the Boolean array to keep the checkbox associated with each grid item checked when the user scrolls up or down the grid view. This is key as the *getView()* method of the custom adapter reloads the grid each time the user scrolls and therefore, it is needed to explicitly keep track of which items the user had previously selected. If this is not taken care of, random checkboxes in the grid view end up being selected when the user scrolls.
- e. Later, when the options selected by the user need to be re-loaded on the screen using Shared Preferences, after exiting the application or moving to another screen (discussed later in this Section), the UI thread needs to wait for an

AsyncTask calling the database to conclude before loading the previously selected items. If this is not done, then a possible NULL pointer exception could occur if the database call takes longer than the UI thread to load.

- **Multi language support**

To support multiple languages (currently English and Hindi), the application required modifications in the Android client (discussed in Section 3.3.2) as well as in the database.

The database changes can be divided into the following categories -

- **Storing user entered text in the language of choice, with everything else stored in English**

In the *UserMaster* table that contains each user's sign up information, the only columns that require the user to input text are first, middle and last name. For the remaining sign up steps, the user selects values from lists, drop downs, radio buttons and checkboxes and therefore, the selected values from these steps are pre-defined in the application in the strings XML (Section 3.3.2). These values are then converted from the language on the user's phone to English, and then stored in the database. For the name fields, the data is stored in the language in which the user enters them.

- **Storing a record multiple times in a table, each in a new language**

For certain tables whose records need to be present in multiple languages, each row is stored multiple times, each time in a new language. To understand this, let us consider the *CommodityMaster* table that contains a collection of crops grown

in the country. Now, every state in the country has its own regional language (sometimes multiple). Therefore, for a state where Hindi is spoken, we want the crops grown in that state to be available in English and Hindi as users from that state will interact with the application in either of these two languages. But, for another state where the regional language is Tamil, we want the crops to be available in English and Tamil (and not Hindi). Thus, we can see that for a crop growing in both these states, it will need to be available in English, Hindi and Tamil. To achieve this, *CommodityMaster* has a crop present in all the languages of the states it grows in, each in a new row. So for example,

CT129 | L001 | Cucumber

and

CT129 | L002 | ककम्वर

represent two records for the same crop in two languages. The first column is the unique crop identifier, the second is the language identifier (“L001” for English and “L002” for Hindi) and the third is the name of the crop in a particular language. So, if an Android phone’s language is set to Hindi, it will use the second of the two records. This also provides a platform with which the application can scale to include more languages as it will solely involve adding a new row for the crop with a new language identifier.

- **Multiple columns for multiple languages**

While crops can span across many states and therefore need to be available in multiple languages, names of villages, districts and sub-districts in a state only

need to be present in two languages – English and the main regional language of the state. As a result, they are stored in the database tables as follows -

09 | 132 | *Saharanpur* | सहारनपुर

The third and fourth columns are the names of the location in the two required languages, and we can see, this structure is more optimal (in terms of memory usage) for this use case than the one used for crops, as illustrated in the previous point.

- **Inexpensive smartphone memory restrictions**

Inexpensive smartphones, which will be used by the target population of this application, have memory constraints. For instance, Videocon ZEST Lite, which was used by me during this semester, costs a mere \$40 and comes with only 0.5 GB of storage, a significant portion of which is taken by the Android OS. Therefore, it has been observed that as the application size increases, the number of downloads of that application decreases. For instance, a blog by Lightspeed Venture Partners India observed that 10 -15 MB is the ideal application size globally [15]. However, this size drops for countries like India.

To keep the size as low as possible, Kisan Network reuses multiple activities across flows. This is done by using the *SharedPreferences* class, which provides a general framework to save and retrieve persistent key-value pairs of primitive data types. For example, the negotiation flow (Section 3.2.1.4) occurs three times in the application – a buyer sending his negotiation offer to a new sale offer he has discovered, a buyer sending his negotiation offer to an existing deal of his and finally, a farmer sending a negotiating

offer to an interested buyer. Each of these situations has a distinct flow of screens, thereby making it essential to know how we arrive at the common negotiation screen. Shared Preferences allows the application to do exactly that as it keeps track of whether the user is a farmer or a buyer, and if a buyer, it further stores whether this is the initial negotiation offer. This allows the application to reuse the same activities and saves memory space, as a key value pairing takes up kilobytes of space, whereas a new set of activities, as can be seen from its composition in Section 3.3.2, would take up space in the hundreds of kilobytes or even megabytes.

- **Offline Support**

Using Shared Preferences, Kisan Network also allows the application to run offline i.e. without being connected to the Internet, for a majority of the screens. For instance, the sign up process consists of eleven steps and no Internet connection is required up until the ninth step. Even the crop sale flow only requires an Internet connection for listing the varieties for a crop and then finally submitting the sale offer. This is done using Shared Preferences, which stores the values entered by the user at each step locally, so that there is no need to connect to the online database till the last step.

- **User Location Determination**

During the Sign up process, we retrieve the user's location by utilizing Google Play Services' Location API's Fused location provider that optimizes the device's use of battery power. After obtaining the latitude and longitude values from the previous step, I call an external API that uses the open source GIS Server Software called MapGuide.

The service then calls a functionality of MapGuide to find a polygon that intersects the provided latitude-longitude point. The polygon derived represents the boundaries of villages in India and contains information such as the name, district name, sub-district name codes, etc. that is used by the application.

4. The Process

4.1 The Timeline

To make Kisan Network work, it is essential to look beyond the technology and the product. It should be seen as an attempt to bring about socio-economic change in one of the most traditional ecosystems in India. To make such an attempt, I had to understand the problems faced by the stakeholders in the ecosystem, and also the manner in which they operate phones and interact with technology. As a result, a lot of research, surveys and evaluation were done by me, along with others, so that I was well informed about the realities of the target users, both before and during the application development process. A timeline of the process followed by me is as follows –

Customer Discovery Focus Group (Problem / Solution Fit) in the Indian states of Punjab and Uttar Pradesh (Section 4.2)	December 2014 – January 2015
Incorporated aspects of the paper – <i>Exploring suitable interfaces for agriculture based smartphones apps in India</i> [16] (Section 4.3)	February 2015

Minimum Viable Product (MVP) Technology, Usability and Design Evaluation Survey (Section 4.4)	March 2015 (Spring Break)
Iteration based on Spring Break survey results (Section 4.5) Testing of application by Professor Dondero and me (Section 4.6)	April 2015

4.2 Customer Discovery Focus Group

4.2.1 Objective

Over Winter Break, I conducted a focus group study in the Indian states of Punjab and Uttar Pradesh with my father, and under the guidance of MART, a leading knowledge based consulting firm on emerging markets (wireframes and pictures from the study can be found in the Appendix). The objective of the study was as follows –

- To understand the current practices with respect to the logistics of selling and acquiring produce, agricultural workers, equipment and transportation.
- To understand the inefficiencies and shortcomings in the existing system.
- To identify willingness to use and buy smartphones.
- To introduce the Kisan Network concept and understand its perceived benefits.

4.2.2 Approach

The survey involved target user group discussion at two levels – the first was to understand the current ecosystem in which the stakeholders function and identify latent needs. The second was to introduce the concept, take feedback and thereby, validate (or invalidate) the concept. For this, the surveyee segmentation was done in the following manner -

- **Farmer (Wheat)** – In 6-8 person discussion groups. Large farms > 20 acres.
- **Farmer (Potato / Vegetables)** – In 6-8 person discussion groups. Small farms < 4 acres.
- **Middleman** – 5-10 years of experience in the profession. Young to Middle aged. One on one interaction.
- **Transportation Provider** – Fleet and single vehicle owner. One on one interaction.
- **Labor Contractor** – Provider of agricultural workers to farmers. One on one interaction.

4.2.3 Study Details

- A total of approximately 35-40 people were surveyed.
 - 4 farmer groups
 - 6 middlemen and labor contractors
 - 4 transportation providers
- Format of questions was as follows (in sequential order) -
 - a. Understanding the current ecosystem
 - i. How does the system currently work? For instance, farmers were asked how they currently secure sufficient funds to begin cultivation and then once produce is ready, where do they go to sell?
 - ii. How do you procure equipment and people to help in the field?

- iii. Who provides transportation?
- b. Understand inefficiencies
 - i. Are you happy with the way sale of produce happens currently? Do you get a fair deal?
 - ii. Don't all of you in the same village need agricultural workers and equipment at the same time? Who gets it then?
 - iii. To transportation and equipment providers – Do your equipment and vehicles lie unused for large parts of the year?
- c. Introduce the Kisan Network concept
 - i. What if you could know that someone wants the produce at a higher rate in a village far away and wants to buy from you?
 - ii. What if you could get workers from villages a bit farther away, but at cheaper rates?
- d. Show wireframe designs of the mobile application and explain it step by step
 - i. Do you understand each step?
 - ii. Are all the parameters of a deal covered? If not, what is missing?
 - iii. What smartphones do you have? If not, will you buy one for ₹2,500 if it comes with this application?
- e. Receive feedback on the concept, followed by the application flow
 - i. Is any step unclear?
 - ii. Would you like the entire application in Hindi?
 - iii. What step(s) was unnecessary?

- f. Rank the 4 flows in terms of their value to you – crop, labor, equipment and transportation.

4.2.4 Study Response Summary

- Farmers were excited about the pricing and labor concepts. They believe it will help them connect to buyers directly and reduce the amount of money they lose to middlemen.
- They want to eliminate the need for middlemen.
- Labor contractors were excited about the concept as they believe they can always arrange labor if there is a need.
- Middlemen were excited at the prospect of gaining access to more farmers beyond their territory, though some were skeptical of losing their existing farmers.
- Equipment providers and transportation providers appreciated the concept as well, especially considering their equipment went unused for those parts of the year when crops were not being grown in their own and adjacent villages.
- Ranking of four flows - crop > labor > equipment > transportation. For crops, the following table was observed -

Parameter	Farmer	Buyer / Middleman
Appeal	High	High
Relevance	High	High
Customer Value perceived	High	High
Willingness to pay	High	Medium
Overall Attraction	High	High



Figure 9 - Response to Crop Transaction Concept

4.3 Findings from agricultural application interface design paper

Kisan Network's interface design needs to be well suited to its target user, the rural Indian population. Most people in this segment of the population use smartphones infrequently and in addition, are semi-literate as well. To better understand how this would impact the interface design of the application, the paper *Exploring suitable interfaces for agriculture based smartphone apps in India* by Rakshit Agrawal, Mridu Atray and Krishna Sudnari Sattiraju [16] was helpful. It highlighted the following important observations -

- The application should require minimal text input. Also, it should have minimal text and instead use images and icons extensively.
- The application should include voice instructions wherever possible to help the user understand what is being displayed on the screen.
- Scrolling is not preferred as users may have difficulty in understanding how they are able to see more content on the screen than the physical dimensions permit.
- Home buttons should be simple in functionality and consistent throughout the application in terms of their location on the screen. If they are not required on a particular screen, the space should be left empty and another button should not be put in its place as it causes confusion.
- For activities requiring keyboard based input, there should only be one input field per screen to reduce confusion in terms of making the keyboard appear and disappear while moving from one input field to another.

In February 2015, I incorporated some of these observations in the interface design. Let us take a look at one of the screens in detail and the various elements incorporated in it.

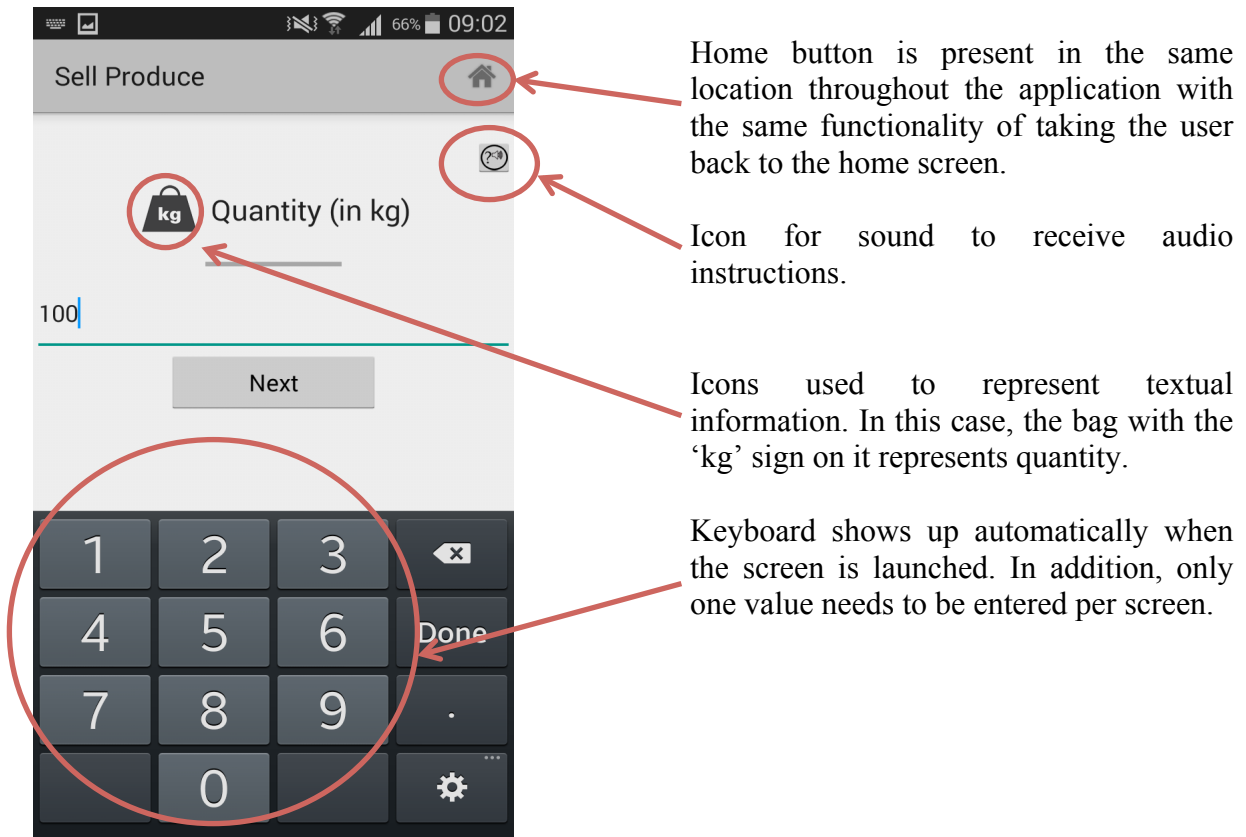
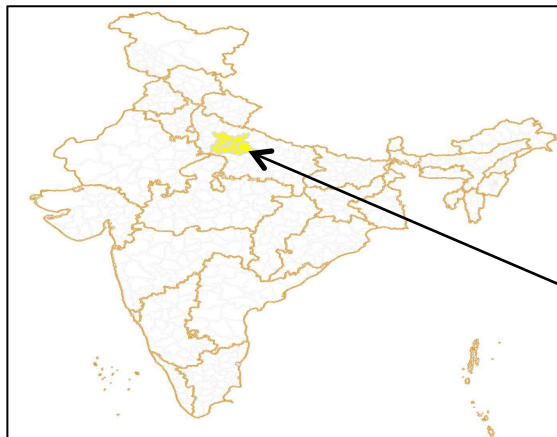


Figure 10 - Interface design implementation

4.4 MVP Technology, Usability and Design Evaluation Survey

4.4.1 Objective



Over Spring Break, I conducted a technology, design and usability survey using a part of the prototype I built over the course of the semester. This was done in the Indian state of Uttar Pradesh with my father and under the guidance of MART.

Figure 11 - Survey Location

The objective of the survey was as follows –

- To evaluate the ease of use of the application, in terms of flow and design consistency, with the target users and test those observations made in the paper (in the previous section) that had been implemented.
- To assess the user’s ability to use a smartphone, with and without guidance.
- To assess the user’s familiarity with icons and images used in the application.
- To assess any performance issues while running the application on different Android devices, from inexpensive phones to high-end Nexus and Samsung Galaxy devices.
- To inspect issues, if any, while dealing with slower 2G speeds prevalent in Indian rural areas.

4.4.2 Approach

We visited 4 villages in western Uttar Pradesh to carry out individual MVP testing with 18 – 20 farmers, spending 30 – 60 minutes with each. Each conversation’s audio was recorded and observations were written down. For this, the surveyee segmentation was done in the following manner -

- **Based on education**

Three kinds of farmers were included in the survey – educated (with Bachelor and Master degree), semi-literate (with high school degree) and illiterate (with no high school education i.e. middle school and below).

- **Based on phones used**

Once again, there were three kinds of users under this segmentation – smartphone owners, standard phone owners, and people with no phones.

4.4.3 Survey Details

Each individual survey with the farmer was divided into the following sections (in sequential order) -

1. Introduce the Kisan Network concept briefly and explain why we are building the application.
2. Show a list of icons used in the application and ask the farmer to identify them. If the farmer is unable to identify them, then ask them to describe what they see in the icon and then attempt to connect the components together (icons can be found in the Appendix).
3. Hand a smartphone with the application open to the farmer and only mention that the first few steps involves filling up a form with basic details about himself.
4. Depending on previous smartphone experience, provide assistance if the farmer struggles at any step for longer than a few minutes.
5. Explain the crop sale and negotiation steps and guide the farmer step by step through them.
6. Conclude the survey by asking the farmer about his experience using the application and inquire about specific pain points faced in the previous steps.

4.4.4 Study Feedback Summary

- Implement an audio button on each step, with instructions supported by an example.
- Set up environment variables like unit of measurement (kg, ton for quantity) at the start of the app as it varies from user to user.
- Explicitly indicate where the user needs to click.

- Make icons dynamic to make it more personalized for the user (for instance, instead of a regular crop icon for the homepage, put his crop's icon that he entered during sign up).
- Make design even more consistent and limit the vocabulary for better understanding.
- Farmers are comfortable with scrolling after doing it once, contrary to what the paper in the previous section mentioned.
- Default android calendar is preferred to drop down lists for calendars, as it enables the farmer to see a calendar view, similar to real physical calendars.
- 2G connections and low-end smartphones handled the application and its database calls (via the GAE) with no visible problems.

4.5 Iteration based on Spring Break Survey Results

As the application did not encounter problems with respect to technology during the evaluation survey, I focused on the interface responses received in April 2015 and iterated upon them. It resulted in some changes to the application. The important ones are mentioned below –

- **Making icons and images dynamic**

During the survey, farmers indicated that the home screen, where the four task options were available, was unclear as the icon meant for crop transactions was that of a generic crop (in this case wheat). As the farmers I met were potato cultivators, they did not feel that they should press the image, as they weren't wheat growers. Therefore, to personalize the experience, the icon on the task page has been made dynamic. Now, it shows the first crop that the farmer deals in (filled in during the sign up process) rather than a generic crop. This change can be seen in the figure below –

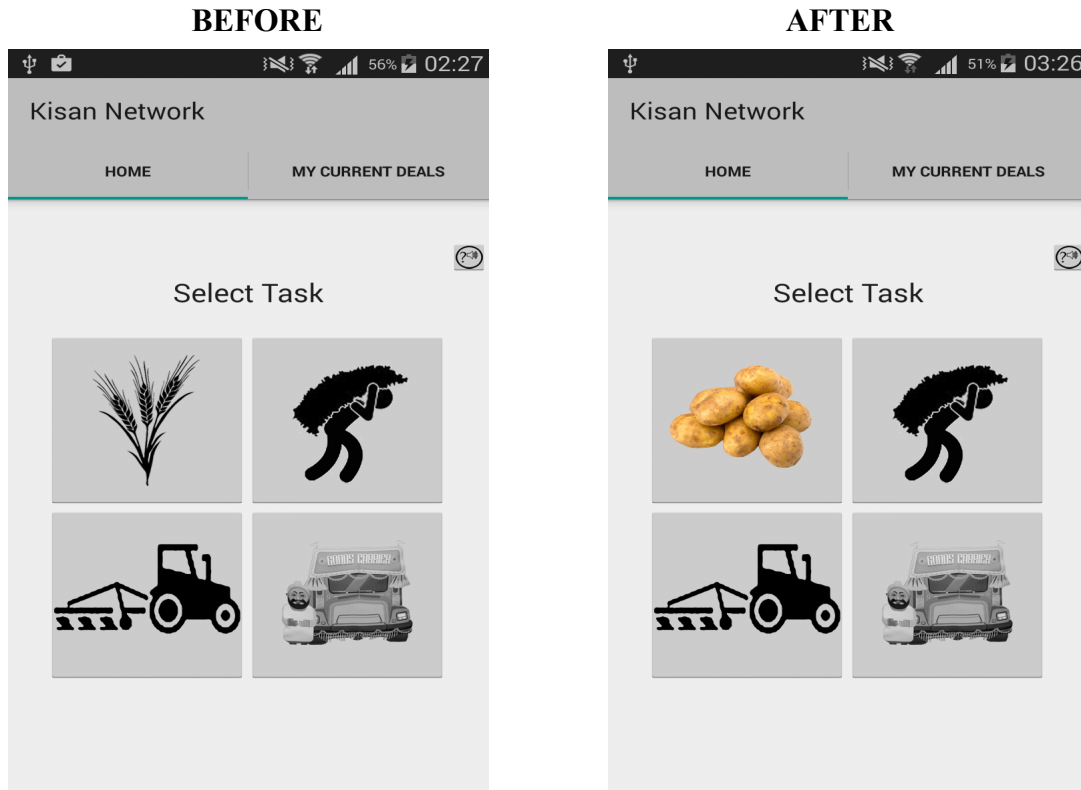


Figure 12 - Dynamic Icon on Farmer Home Screen

- **Even more consistent design**

The survey responses also indicated that the design needed to be a bit more consistent at certain places. One major aspect of the design that came up during my conversations was the importance of the ‘Home’ icon on the menu bar to go back to the home screen and the use of the physical keys on the phone to go back to the previous screen. Farmers were comfortable with this idea. However, they weren’t clear about what the ‘Up’ functionality in Android meant which could be enabled by clicking on the backward facing arrow button on the menu bar (‘Up’ functionality in Android takes you back to the hierarchical parent of the current screen. For instance, if you click on an email from your inbox and then move to another email by swiping left or right, and then press the ‘Up’ button, you

are not taken back to the previous email you viewed. Instead, you are taken back to your inbox as that is the hierarchical parent).

As a result, I decided to remove this option in the application and just keep the ‘Home’ button on the menu bar to go back to the home screen, with the physical back key always being there. This does go against the normal way Android applications are designed, but it was needed to reduce confusion for the target users.

- **Scrolling can be incorporated in the application**

Contrary to what the paper mentioned in Section 4.3 highlighted, infrequent users of smartphones were comfortable with scrolling after using it a few times. This made things simpler in the interface design, especially for confirmation screens, as it allowed me to incorporate a *ScrollView* component into the layout that enabled scrolling.

4.6 Testing

My advisor, Dr Robert Dondero, and I did implementation testing for the application. Our tests looked at whether the application performed as expected and noted bugs, if any. The process involved going through each flow of the application and then, simulating a farmer – buyer interaction using two Android devices. The application **performed as expected** in both our tests, with the only error arising (at times) while transitioning from the location screen to the custom made crop selection screen during the Sign up process (Section 3.4 – Point on Supporting old Android versions). I believe the error arises from a concurrency issue between the UI thread creation and asynchronous call to the database, considering the entire multiple selection feature was created from scratch due to its non-availability below API Level 11. As it occurs very rarely, the debugging for this error will be done before the pilot launch in July 2015 (Section 6).

5. Related Work

The current landscape for applications in the agri sphere can be shown succinctly using Figure 13. We can see that on the y-axis we have solutions that provide generic information on weather and soil at one end, while on the other, ones that provide transactional value to the stakeholders. On the x-axis, we look at how the solution is marketed – whether it just has an online presence (like on the Google Play Store) or does it involve grassroots level marketing like for traditional products such as phones and soaps, which will be done for Kisan Network during

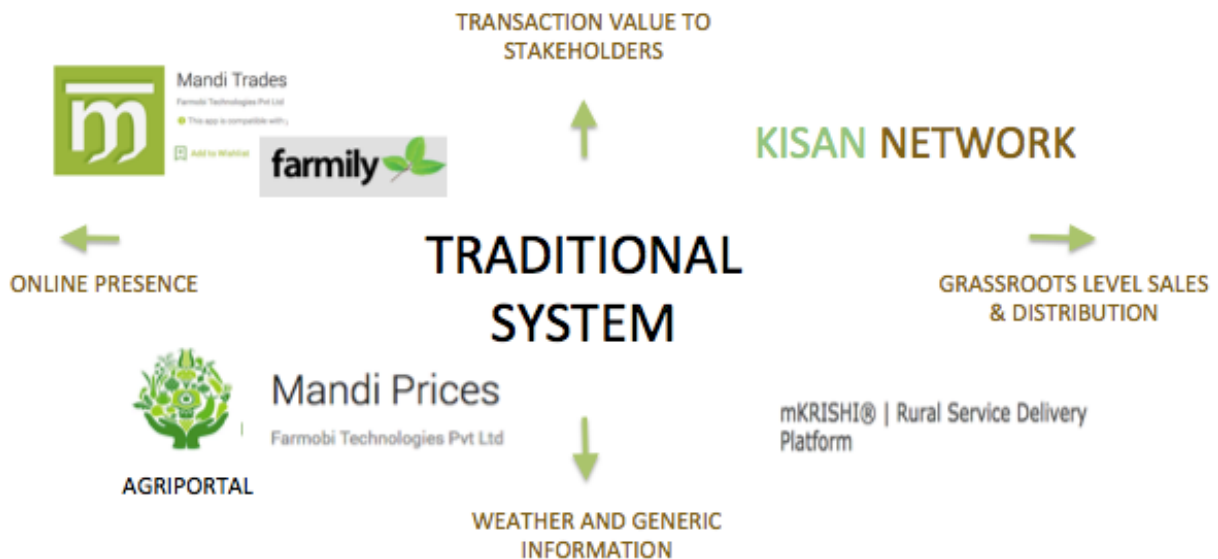


Figure 13 - Existing Solutions

its pilot launch in July 2015. Most applications like mKrishi, Agriportal and Mandi Prices only provide generic information about weather and soil. While these are useful to the farmer, they do not directly play a role in helping him secure better prices.

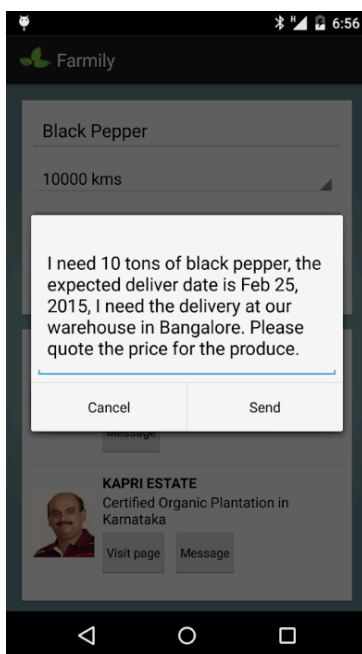
Therefore, let us focus on two solutions - Mandi Trades & Farmily that are similar to Kisan Network in that they provide transactional value to its users. The following problems exist with these solutions -

- **They are not India specific**

Farmily has expanded to other countries before creating a stronghold in India. Since they just have one application in the Google Play Store, independent of the country, their application is not India specific in terms of understanding the requirements and preferences of the Indian farmer.

- **Interface design inaccuracies**

In Section 4.3, we had a look at some of the design considerations incorporated while developing this application. Since the application is being developed for a semi-literate population predominantly, these are key to making it user-friendly. However, existing solutions do not appear to have kept these design principles in mind. For instance, here is a screenshot from the Farmily application.

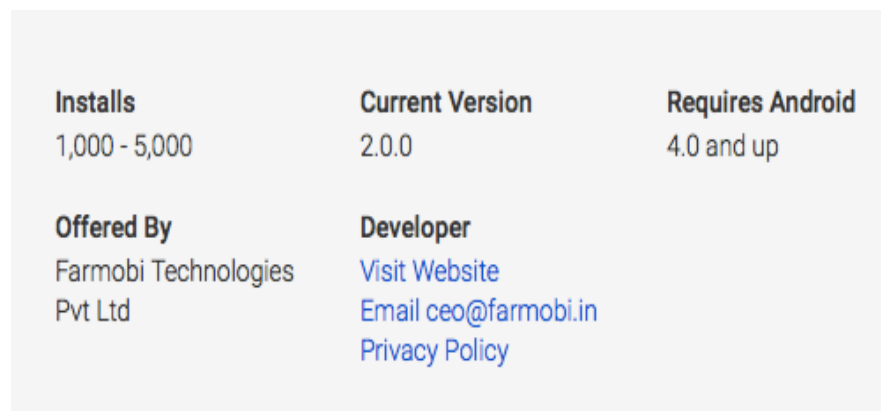


In the figure, we can see that a significant amount of explicit text input is required to use the application. This is in stark contrast to the observations mentioned in the paper (from Section 4.3) and those made during the evaluation survey over Spring Break (from Section 4.4).

Figure 14 - Farmily Explicit Text Input [17]

- **Support for old Android versions**

Existing solutions like Mandi Trades do not support Android versions before Android 4.0 (as can be seen in Figure 15). However, more than 9% of Android devices connected to the Google Play Store still run on pre Android 4.0 versions [14]. This is significant, as some of the cheapest Android phones sold in India have low-end phone specifications, which support Gingerbread at best. In addition, many people who buy Gingerbread smartphones will not update the OS on their devices because OS updates are significantly large in size, something that is only advised on Wi-Fi networks and not while using 2G / 3G. Therefore, by only supporting the latest Android versions, existing solutions possibly miss out on a large proportion of the target market.



Installs 1,000 - 5,000	Current Version 2.0.0	Requires Android 4.0 and up
Offered By Farmobi Technologies Pvt Ltd	Developer Visit Website Email ceo@farmobi.in Privacy Policy	

Figure 15 - Mandi Trades Android Version Requirement [18]

6. Future Work

Though a significant amount of progress was made this semester, a lot more needs to be done to solve this real problem faced by nearly a quarter of the nation's population [2]. Firstly, the 'Labor Request' features needs to be implemented for the farmer and a labor contractor login needs to be included as well (which is analogous to that of a buyer). This will be done by June

2015 so that I can go ahead and launch a pilot phase later in the summer from July 2015. The other two features, for requesting equipment and transportation, will be developed only after the pilot phase, as it was evident from the Customer Discovery Study in December 2014, that both of these rank the lowest in the desired features list. Other important milestones that need to be reached for the pilot phase in July 2015 are –

- Keeping the app size to below 15 MB which would require further reuse of existing activities as the size is at ~ 13.5 MB currently.
- Incorporating text message support for the application where the users get a text message when someone responds to their offer / negotiation deal. This is better than push notifications, as the latter require continuous Internet connection, something that is not prevalent at all times in rural India. On the other hand, incoming text messages in India do not count against your text message limit and are therefore, more economically viable.
- Incorporating online banking payment methods (using external APIs) for the payment step after a deal is agreed upon.
- Supporting extremely small screen sizes (4 inches and low) that are present on many low-end Android smartphones.
- Including a full negotiation thread, which the user can refer to while making a new offer. This will require extensive interface design research in order to convey the information in the simplest possible manner to an audience not familiar with how threads work (like in email conversations and comments on blogs).
- Providing the ability for a user to login to the application even after changing his smartphone or accidentally deleting the application data from his phone, without losing his existing data.

- Adding the audio clips with instructions for all the screens that require user input.

7. Conclusion

We have seen that Kisan Network is an attempt to incorporate technology into one of the most traditional industries that exists in India. It is being developed for the stakeholders in the agri ecosystem so that they can break away from the inefficient system they are a part of currently that deprives them of potentially cheaper resources and more profitable deals they can secure from beyond their limited local community. It uses the **data enabled low cost smartphone** to connect farmers to potential buyers for their crops and also provides them access to agricultural workers, equipment lenders and transportation providers for other agricultural needs. As a result, farmers will be in **total control of the sale of their own produce** and not suffer from fair price deprivation. In addition, they will have **greater access to agricultural resources** that will help make the market for labor, equipment and transportation competitive, equal and fair.

The application is being developed keeping the target user's requirements and preferences in mind. It has multi language support and runs on inexpensive Android smartphones with memory restrictions. Its user interface design is also optimized for use by semi-literate people who are infrequent users of smartphones. All these features have been developed after an extensive focus group study conducted in December 2014 – January 2015 and have also been evaluated as a part of the development cycle during a technology, usability and design survey in March 2015. Both these surveys were conducted in Indian villages with farmers and the other stakeholders. In addition, the feedback from the survey in March was utilized to iterate on the application in April so that it better suited the needs of the user.

Though a lot of application development has taken place, a significant amount needs to be done in the coming months to launch the pilot phase for the application in July 2015 with the ‘Crop Transaction’ and ‘Labor Request’ features. The response from the potential users of the application has been overwhelmingly positive as they too realize the importance of having access to useful information about their ecosystem at their fingertips.

Acknowledgments

First and foremost, I would like to thank my advisor, Dr Robert Dondero, for his guidance and insightful feedback throughout the semester, as I tackled both technical challenges and the best way to evaluate them with real users. A big thank you to my parents as well for advising me at every step as I approached this tough problem and attempted to find ways to solve it. I would also like to take this opportunity to thank my girlfriend Aakansha for her constant support and encouragement throughout the semester, most importantly at times when I needed it the most. Last but definitely not the least, my two closest friends at Princeton have been vital for me not just this past semester but also throughout my time here – my roommate Bizuwork for hearing me out whenever I was perplexed by a design or technical challenge and Saumya for her effort and time spent in reading my draft and providing detailed feedback.

Thank you to all of you.

Works Cited

- [1] Mouvement pour une Organisation Mondiale de l'Agriculture. (2015).
http://www.momagri.org/UK/agriculture-s-key-figures/With-close-to-40-%25-of-the-global-workforce-agriculture-is-the-world-s-largest-provider-of-jobs-_1066.html#ANCRTOP. Retrieved on 21 April 2015.
- [2] Census of India. (2011). <http://censusindia.gov.in>. Retrieved on 21 April 2015.
- [3] The Times of India, News Service. (2010, Dec 23). Farmers suffer as middlemen call the shots. <http://timesofindia.indiatimes.com/city/bengaluru/Farmers-suffer-as-middlemen-call-the-shots/articleshow/7147673.cms>. Retrieved on 28 April 2015.
- [4] Epoch Times. (2013, Apr 23). Indian farmers boosted by eliminating middlemen. <http://www.theepochtimes.com/n3/22844-direct-agricultural-marketing-no-middlemen-more-farmer-friendly/>. Retrieved on 28 April 2015.
- [5] The National. (2014, Nov 1). Smartphone growth accelerates in India. <http://www.thenational.ae/business/technology/smartphone-growth-accelerates-in-india>. Retrieved on April 27 2015.
- [6] The Hindu Business Line. (2015, Feb 18). 3G drove mobile data traffic growth in 2014. <http://www.thehindubusinessline.com/features/smartbuy/tech-news/3g-drove-mobile-data-traffic-growth-in-2014-study/article6909498.ece>. Retrieved on April 28 2015.
- [7] Google Cloud Endpoints Overview. (2015). <https://cloud.google.com/appengine/docs/java/endpoints/>. Retrieved on April 22 2015.
- [8] What is Google App Engine? (2015). <https://cloud.google.com/appengine/docs/whatisgoogleappengine>. Retrieved on April 26 2015.

- [9] Google App Engine Quotas. (2015). <https://cloud.google.com/appengine/docs/quotas>. Retrieved on April 26 2015.
- [10] Google Cloud Endpoints Tutorial – Part 1. (2014). <http://rominirani.com/2014/01/10/google-cloud-endpoints-tutorial-part-1/>. Retrieved on April 23 2015.
- [11] Open Government Data (OGD) Platform India. (2015). <https://data.gov.in>. Retrieved on April 28 2015.
- [12] Parse SDK Review. (2014). <http://profi.co/parse-sdk-review>. Retrieved on April 27 2015.
- [13] What is the AWS SDK for Android? (2014). <http://docs.aws.amazon.com/mobile/sdkforandroid/developerguide/Welcome.html>. Retrieved on April 27 2015.
- [14] Google Android Studio New Project Creation Setup.
- [15] Unbundling Mobile Apps for the Emerging Markets. (2014). <https://lightspeedindia.wordpress.com/2014/06/12/unbundling-mobile-apps-for-the-emerging-markets/>. Retrieved on April 25 2015.
- [16] Agrawal, R., Atray, M., & Sundari, S.K. (2013). Exploring suitable interfaces for agriculture based smartphone apps in India. (APCHI'13, September 24 – 27 2013, Bangalore, India. Copyright 2013 ACM 978-1-4503-2253-9/13/09).
- [17] Family on Google Play Store. (2015). <https://play.google.com/store/apps/details?id=com.family.android>. Retrieved on April 28 2015.

- [18] Mandi Trades on the Google Play Store. (2015).
<https://play.google.com/store/apps/details?id=com.manditrades&hl=en>. Retrieved on April 28 2015.
- [19] IaaS, PaaS, SaaS (Explained and Compared). <http://appenda.com/library/paas/iaas-paas-saas-explained-compared/>. Retrieved on 30 April 2015.

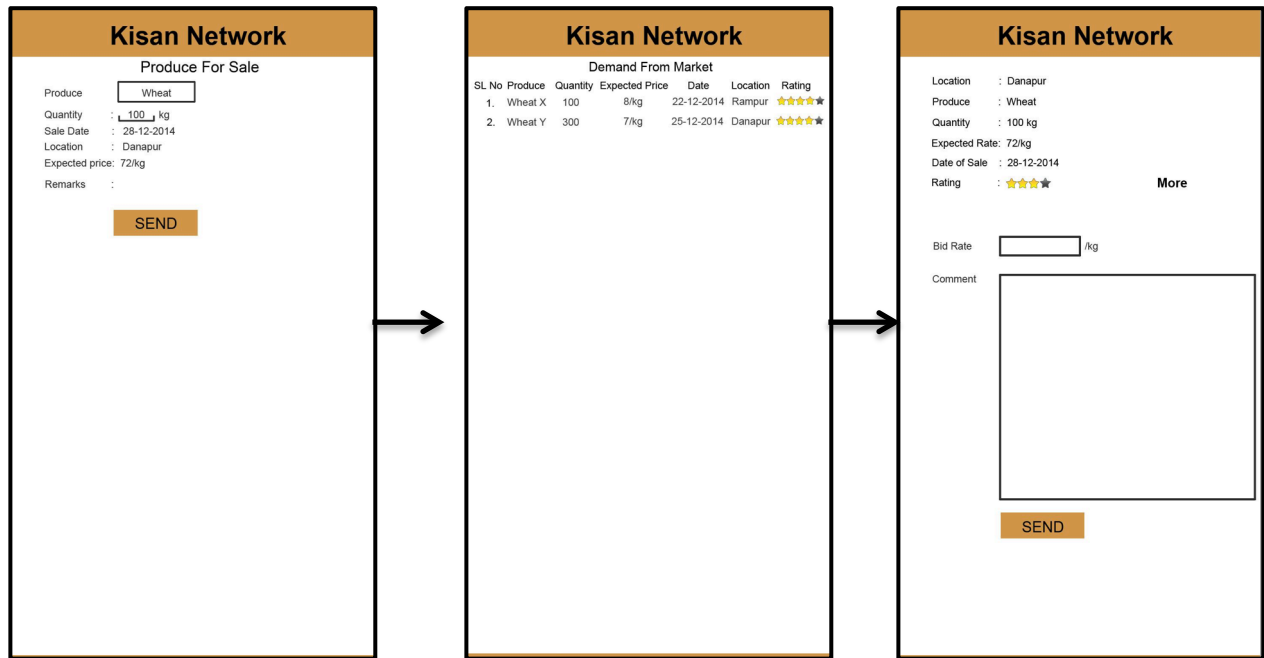
Appendix

Additional Media Coverage on the Farmer's Plight

The problem faced by the farmers in the agri ecosystem is a real one. I have always known about it. My parents have known about it as well. In fact, most people in the country know about it. As a result, it has received widespread media coverage, both domestic and international. In addition to the two articles referenced in the paper, I would like to draw your attention to another couple. The first is from the Wall Street Journal (WSJ) (2014, Feb 26) where it is mentioned that “The intermediaries add value but they increase the cost. You need to provide the farmer an alternative avenue to sell their produce”. This is exactly what I am attempting to do with Kisan Network – provide an alternative to sell produce while they are better informed about buyers. The second article is from the Inter Press Service News Agency (2012, Mar 9) (IPS News) where they talk about the fact that "During 1995-2010, over 250,000 poor farmers in India committed suicide, according to national statistics, mainly attributed to their inability to pay debts incurred on agricultural inputs”. This, once again, alludes to the gravity of the situation in this ecosystem. You can find the complete articles at - <http://blogs.wsj.com/indiarealtime/2014/02/26/farmers-struggle-to-escape-middlemen/> (WSJ) and <http://www.ipsnews.net/2012/03/indian-farmers-hostage-to-middlemen/> (IPS News).

Customer Discovery Focus Group Materials

During the Customer Discovery Focus Group (Problem / Solution Fit) in December 2014 – January 2015, wireframe designs of the application were used to introduce the Kisan Network concept to the stakeholders (application development began only in February 2015). Below, I have attached a few of these initial wireframes –

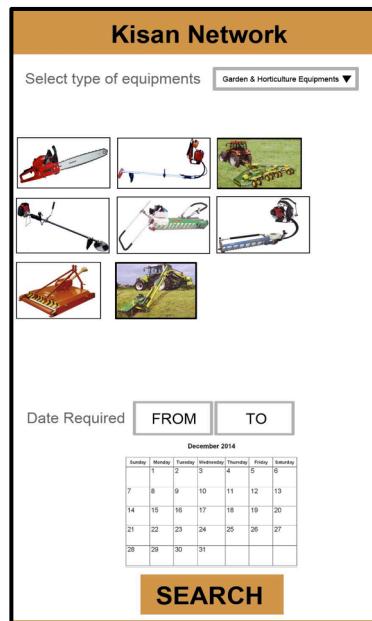


Farmer puts up produce for sale

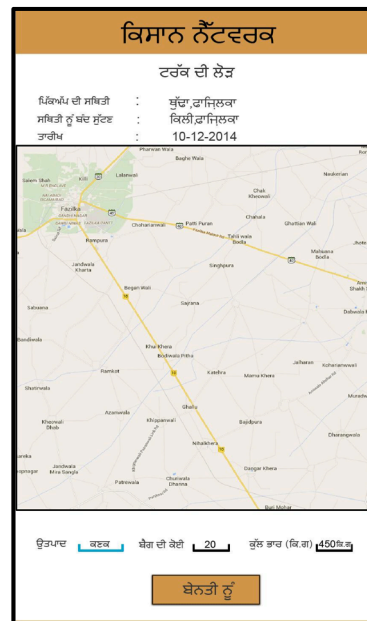
Buyer sees relevant information

Buyer accepts or negotiates

The above flow of wireframes was used to introduce the ‘Crop Transaction’ concept to the farmers and the buyers / middlemen.



Farmer requesting equipment / labor



Request for trucks (in Punjabi)

The above two wireframes were used to introduce the ‘Equipment / Labor Request’ concept and ‘Transportation Request’ concept respectively. The transportation wireframe was in Punjabi, the regional language of one of the states we visited so that the stakeholders knew that the application would be customized for them.

I have also attached a few pictures taken during the survey below. I hope they give you a good idea of what rural life in India looks like.





Pictures from the focus group study

MVP Technology, Usability and Design Evaluation Survey

The icons used in the application were evaluated during the survey where a sheet full of icons was handed out to a farmer and then he was asked to say what he thought they meant. Either the farmer understood an icon's meaning immediately. If he did not, he was asked to describe what he saw in the icon and split it into its components. Then, he was able to join the components together to understand the icon in most cases. In the cases where he couldn't, we'd mark a cross near the icon. One of these icon sheets used during the evaluation can be found below -

KISAN किसान नेटवर्क NETWORK

1. Name *Nehraj Parmar.*
 2. Age *38*
 3. Education Qualification *B.com*
 4. Crops : *फसलें (Kufri Bakhur 3797).*
 5. Observations

Parameters	Scale				
i. On Your Own	1	2	3	4	5
ii. Registration	1	2	3	4	5
iii. Sale Process	1	2	3	4	5
iv. Negotiation	1	2	3	4	5

6. Where does he get stuck?
 7. Other Observation

poor name instead of first name
 confusion about first & last
 dropdown list confusion - can scroll
 all clear till now
 location ✓

main activity
 geku
 no idea what to do if clickable
 'select task' not clear
 'per quintal' not kg
 camga ✓
 kabos task clear ✓
 area ✓
 rate - not clear if to give rate on own

50 acre - 80%
 15-20 acre - 70%
 farmer - business - mid man / buyer

active deals doesn't seem clickable
 quality negotiation
 scroll ✓
 more comfortable with audio date

slow with is clear
 compatible with bank acct & exchange company

An example of an icon sheet used during the evaluation

The interviews with the farmers were recorded during the evaluation. One such recording can be found at the link below (password: *networkkisan*) (NOTE: The language being used is Hindi) –

https://www.dropbox.com/s/qnv3vj63qlz8429/Record_0018.wav?dl=0

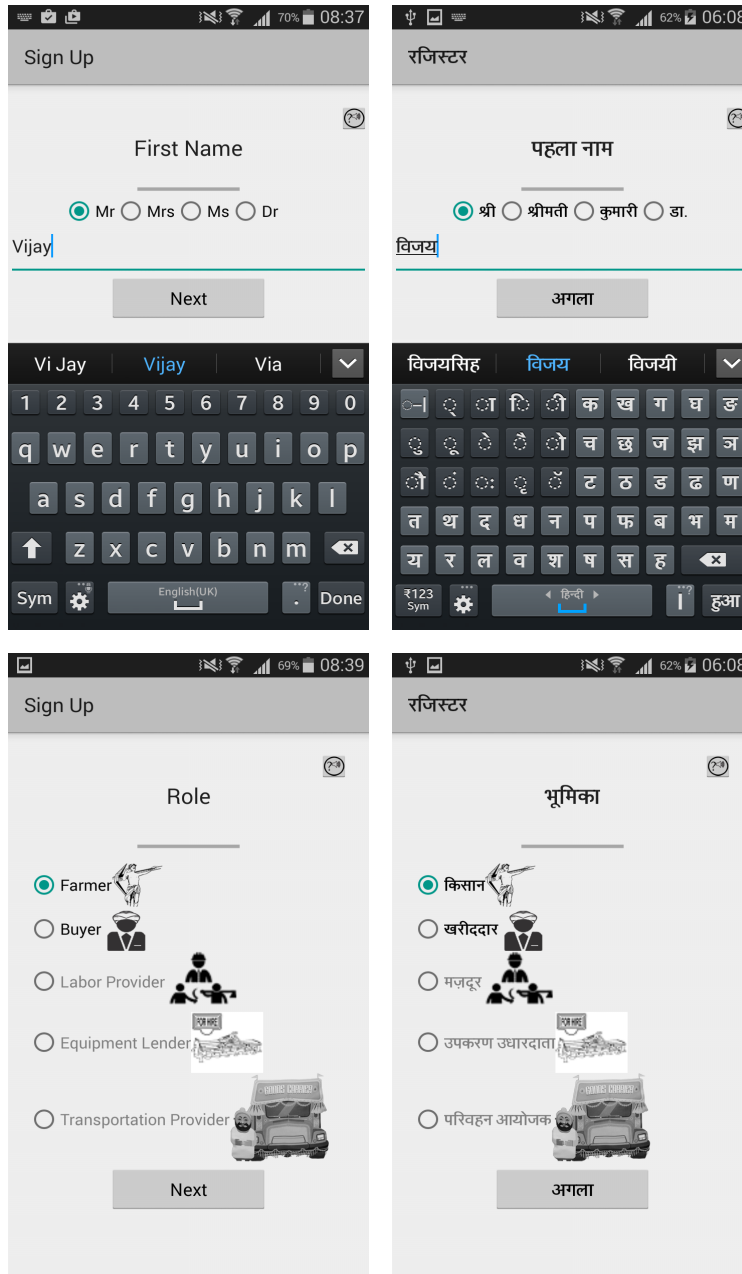
I have also attached a few pictures taken during the evaluation below -



Pictures from the evaluation survey

Multi language support examples

Since the paper is written in English, I refrained from adding any flows that consisted of text in Hindi. However, below are a couple of examples of multi language support at work –



Multi Language Support Examples

“This paper represents my own work in accordance with University regulations.”

Aditya Agarwalla

Aditya Agarwalla