INVIGOR

Project Report

Submitted by

VINIT SURTI, DEVASHIS TAMHANE, VAIBHAV VISHAL, MIT VAIDYA

Under the Guidance of

ALPA RESHAMWALA

in partial fulfilment for the award of the degree of

B. Tech.

in Computer Engineering

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING, MUMBAI

APR 2016

Annexure-II

DECLARATION

We, Vinit Surti, Devashis Tamhane, Vaibhav Vishal and Mit Vaidya Roll Nos. E054, E055, E059 and E062 respectively B.Tech (Computer Engineering), VII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source:IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Student:

Name: Vinit Surti, Devashis Tamhane, Vaibhav Vishal, Mit Vaidya

Roll No.: E054, E055, E059, E062

Place: MPSTME, Mumbai

Date: 28-04-2016

Annexure-III

CERTIFICATE

This is to certify that the project entitled “Invigor: Adaptive Environment for Android” is the bonafide work carried out by Vinit Surti, Devashis Tamhane, Vaibhav Vishal and Mit Vaidya of B. Tech. (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIII semester of the academic year 2016, in partial fulfilment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

Alpa Reshamwala

Internal Mentor

Examiner 1

Examiner 2

Dean

Dr. S.Y. Mhaiskar

Annexure-IV

Acknowledgement

First and foremost, the four of us would like to express our gratitude to our families for being ready to let go of a substantial amount of family time so that work on the project could be done. Secondly, we thank the countless developers and users at XDA Developers; without their valuable insights and research, we wouldn’t have had a starting point for our project. Thirdly, we thank our mentor, Prof. Shailja Sumeet, for always making time for us and accommodating us whenever we were pressed for time ourselves, and for lending us her learned ear and giving us advice on how to proceed each step of the way.

Annexure V

Table of contents

I. List of Figures

II. List of Tables

ABSTRACT

Using an Android launcher is a great way to customize your Android experience without the need of rooting the android device. Android launchers generally change the look and feel of the Android device. We are able to change animations and graphics, install custom widgets or give the app drawer a whole new look.

Invigor offers a simple and effective user interface allowing different kinds of users from different age groups to gain maximum benefits from the system as it learns and adapts according to their needs and daily requirements from the system. After a detailed survey, here is proposed a systemwhere we can build our own adaptive environment with improvements to the existing system.

The increasing complexity in today's applications, e.g. the number of available options, often leads to a decreased usability of the user interface. This effect can be countered with intelligent user interfaces (IUIs) that support the user in performing his/her tasks by facilitating the interaction as much as possible. IUIs facilitate information retrieval by suggesting relevant information or they support the system use, e.g. by providing explanations, performing tasks for the user, or adapting the interface. The main focus is on user-adaptive IUIs which are able to adapt their behaviour to individual users. It is a key feature for IUIs as the support that should be provided by an IUI heavily depends on the needs and preferences of each user.

We have proposed a framework that adapts the user interface (UI) of mobile computing devices like smartphones or tablets, based on the context or scenario in which user is present, and incorporating learning from past user actions. This will allow the user to perform actions in minimal steps and also reduce the clutter. The user interface in question can include application icons, menus, buttons window positioning or layout, colour scheme and so on. The framework profiles the user device usage pattern and uses machine learning algorithms to predict the best possible screen configuration with respect to the user context. The prediction will improve with time and will provide best user experience possible to the user. To predict the utility of our model, we measure average response times for a number of users to access certain applications randomly on a smartphone, and on that basis predict time saved by adapting the UI in this way.

It is a framework that supports the monitoring and analysis of mobile application usage patterns with the goal of updating user models for adaptive services and user interface personalisation.

Thus, we have introduced a model of launcher that adapts different kinds of users and accordingly changes the user interface allowing the user to use the launchers effectively and operate the system smoothly. We have included the literature survey consisting of different algorithms used in the design and adaptive functionality of the system.

1. INTRODUCTION

1.1 Project Review

In this technological savvy era, the smartphone is the corner stone of life or the basic necessity pivotal to the functioning of human life in a smooth and efficient manner. Though the modern generation becomes virtuosos and exponents as far as the usage and the handling of the phone is concerned, there exist various strata within the human society who are not empowered with these luxuries. Take for example the old, hoary and the aged or the ones suffering from disabilities. Hence we aim at catering to the needs of these people and ameliorating their scenario.

Our solution is an adaptive android launcher built for android smartphones that caters to these people. What we seek to do is to customize and revamp the current available launcher. We are intent on modifying the layout, completely transmuting the current design so as to transcend the previously established design. Our ardent goal is to enrich the living experience. The current smartphone is pretty intricate and convoluted. We seek to eradicate the struggle and make the smartphone experience like never before. For this, we propose Invigor, an Adaptive Android launcher, an application that acts as a go to tool. We aim at customizing it in such a way that the user is not burdened or weighed down. The functioning will be smooth. User preferences are taken into consideration.

The average user has a number of apps installed on their smartphone. Finding the application of interest is cumbersome and takes time as the applications are categorized either alphabetically or based on the type of thcurently e application. According to research by Weidenbeck, some users prefer to find their desired apps by looking at icons rather than search the labels by name. Therefore, the search by name option currently available on some mobile devices is not much use. In an ideal scenario, the user should be able to see only the specific apps on their screen which they actually want to access. Currently, such a system is not available.

We propose a framework to improve the user experience by providing context specific modifications to the user interface (UI). The framework uses machine learning to learn the usage patterns depending on the context, when accessing different applications. This is then used to make the predicted applications or elements of the user interface more prominent and accessible, thus saving time and smoothening the functionality of the system.

In all, we aim at changing the history of modern smartphones. Invigor will act as an assistant providing critical and valuable responses.

1.2 Hardware Specifications

The main hardware required for any user to adapt and infuse the elegant Invigor are:

1) Android device
2) Minimum 512MB RAM
3) Minimum 100 MB free space
4) A computer with a minimum of 2 GB RAM and 100GB of free memory on hard disk

1.3 Software Specifications

If a person has the android device with minimum android version 4.0, then the software to run the Invigor are:

1) Minimum Android version 4.0 [KitKat]
2) Windows OS
3) Eclipse IDE
4) Java

2. LITERATURE SURVEY

The home screen in Android phones is a highly customizable user interface where the users can add and remove widgets and icons for launching applications. This customization is currently done on the mobile device itself and will only create static content. The content, buttons and widgets, are shown in the same position with the same icons until the user manually changes it. Our work takes the concept of Android home screen one step further and incorporates this in regular Android applications. We add flexibility to the user interface by making it context-aware and integrated with the Google cloud. Cloud computing is focused on sharing data and computation resources over a scalable network of nodes. It has gained popularity over the last few years and large companies like Microsoft, Google and IBM all have initiatives promoting cloud computing [1]. The configuration of the application home screen is stored in the Google cloud and the server will push events to the registered mobile devices that are executed on the phone. Mark Weiser [2] argued that machines should fit the human environment and not force the humans to enter theirs. We try to follow this idea by registering the context of the user from many different sources and automatically adapting the home screen based on the predefined user configuration.

The brief overview on adaptivity can be referred from the following diagram:

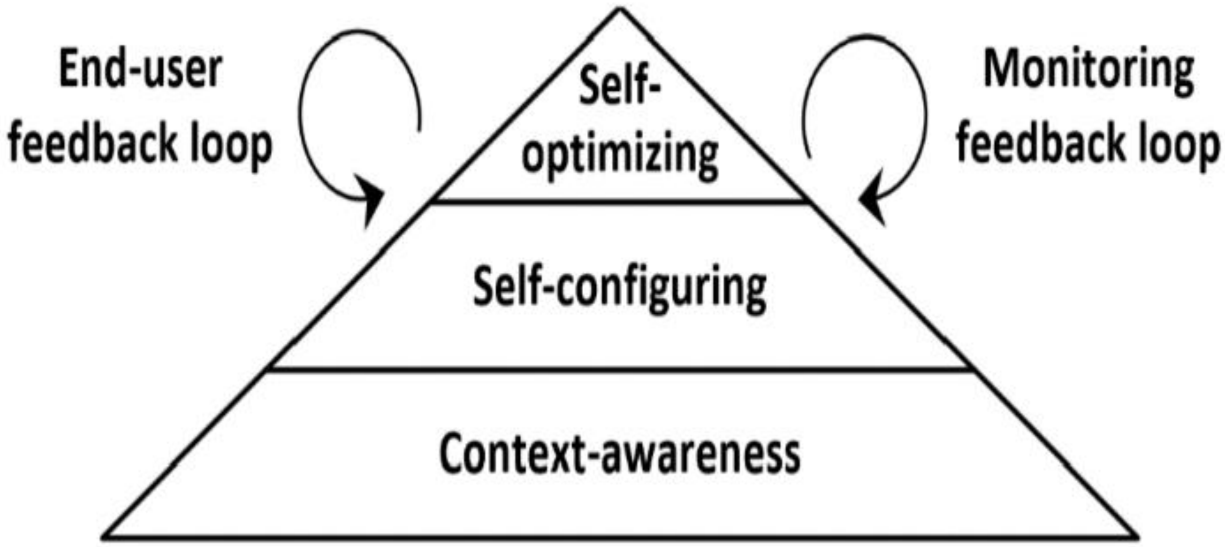


Figure 2.1: Adaptivity Implementation Cycle

2.1 Contextual Factors

As mentioned, we are proposing to modify the user interface of a smartphone in response to certain contextual factors. In this section we look at some of these factors.

Device sensor readings: Output of other sensors on the device including the ambient light sensor (to infer whether the user is indoors or outdoors), accelerometer and gyroscope (to say if the user is stationary or moving) can be used to derive additional contextual information in order to better predict the users chosen application and modify the UI appropriately.

User location: One of the factors to adapt the UI is the location of the user. This is based on the premise that the type of applications a user is expected to access when at home is different from the type of applications accessed when the user is at work. The location is determined by means of the GPS sensor on the mobile device.

Duration of contact: Based on how long a call takes, the apps accessed by the user may vary. A long call or chat conversation might require extensive use of the browser application, while a call of short duration might only require the appointments. The mobile device can access the call, text or chat logs to find out the average duration of the contact and map it with the applications accessed.

Category of the person being contacted: In a mobile users contact list, the contacts may be of different categories, including work colleagues, family, friends and casual acquaintances. In case of a call coming from a colleague at work, one may need to access certain kind of apps during the call e.g. productivity apps. In case of calls from friends, one may need to access other apps like maps or calendar. The user can specify the category, or the device from other factors including user location, time of contact, duration of call and so on infer itself.

Day and time: The type of applications accessed on weekdays might be different from the applications accessed on a weekend or on holidays. Similarly, in the morning the user may access different apps than the ones they do at night. A logging service running in the device would have to log the types of apps accessed at specific times of day or day or the week, and use it to make the appropriate UI modifications.

Application usage logs: Logs of the past application usage, the frequency at which the particular app was accessed and the user actions and interactions while using the app can act as another source of contextual information.

2.2 Components

The aim of the contextual adaptive user interface is to store the context of various user actions, predict the next user actions based on the context and on that basis to modify the user interface.

Figure 2 illustrates the architecture of the system and the relation between the different modules/ components of the proposed framework. The components include the following:

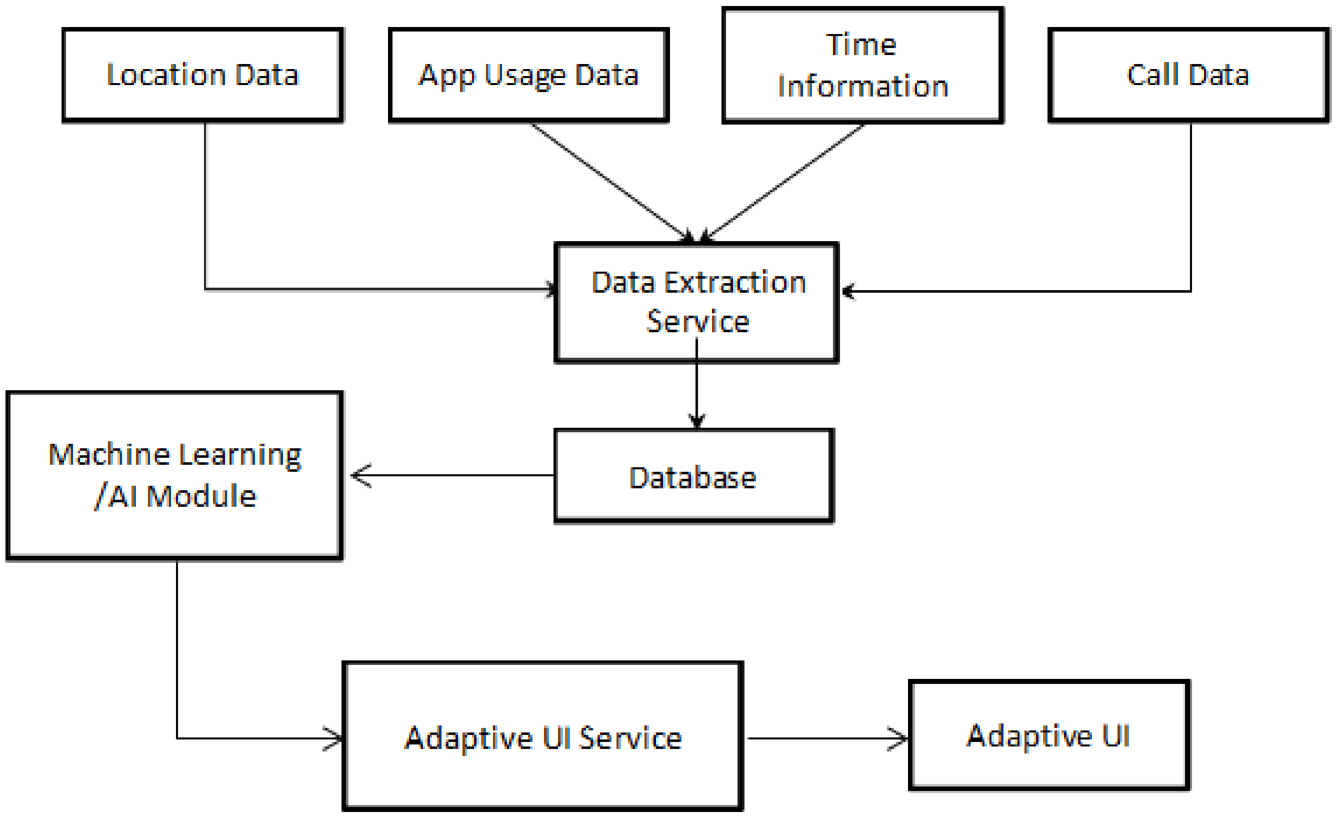


Figure 2.2.1: Module diagram for an Adaptive UI System

Data extraction service: This service runs in the background and extracts and logs various user data including location data, app usage data, call data and time information.

Database: The database stores the contextual user data collected by the data extraction service. The database can reside wholly on the mobile device, or partly in the cloud.

Machine learning module: The learning module uses data stored in the database and appropriate machine learning algorithms to derive rules associating the context with user actions regarding interaction with the device. The rules themselves are then stored in the database. The module also uses the stored rules to make real time predictions for future user actions based on the present context.

User interface adaptation module: Based on the predictions of the machine learning module, this component adapts the user interface in real time to make it easier for the user to perform the predicted actions. This adaptation can take a number of forms, such as displaying the icons of the predicted actions prominently, changing the menus to list the predicted items at the top, changing the relative sizes individual UI elements such as buttons to prominently display the element which the user is expected to click. Some of the possible adaptations are discussed in more detail in a later section.

In the following section we look at the rules derived by the machine learning module in more detail.

As mentioned in the previous section, the machine learning module takes the data stored by the data extraction service and derives rules associating the specific context with user actions. The rules are then each given a weight, which itself changes with time. The weighted average of the rules is then used to predict the UI adaptation to be made.

The rules are in the format of context parameter associated with a certain user action or UI adaptation. Some example rules are given below: <Currently running application> å <Menu item predicted to be chosen> <Currently running application> å <User interface element predicted to be clicked> <Caller or contact person id> å <Application icon predicted to be invoked during the call or chat conversation> <Caller or contact person id> å <Application predicted to be invoked just after the call or chat within a threshold of time T> <Id of person being called> å <Application predicted to be invoked during the call> <Time of day, Caller Id> å <Application invoked> <Location of user> å <Application invoked>

<Application being invoked currently> å <Next application predicted to be invoked>

Along with the database, the learning can also take place either locally on the device, or else on a remote cloud server. Each of these has their advantages. Having the database and algorithm running locally would lead to faster response times, especially if the network connection is slow. However, this might slow down the device at the same time, so a local storage and processing of learning rules is more appropriate for high end devices. Storing and processing the algorithms on a remote cloud server, on the other hand, has the advantage of incorporating data from various devices owned by the user, as well as freedom from constraints of memory and processing speed.

The learning algorithm can be chosen from any of the standard supervised learning algorithms including Support Vector Machines (SVMs), Hidden Markov Models (HMMs), Bayesian algorithms or Associative Memory Neural Networks. Existing open source tools or libraries, such as ghmm for HMMs or SVMTool for SVMs can be used for the purpose. In all the cases, regardless of the learning algorithm used, the model is trained from some past user behaviour data collected by the data extraction service running on the mobile device.

In the following section, we look at some of the ways in which the user interface can be modified.

The user interface adaptations possible in response to the present context and applying the learned rules are many. In this section we look at some of the adaptations.

A. Changing ordering of the menu items. The menu item that is predicted to be chosen are ordered as per the probability of the prediction. The items most likely to be chosen are displayed at the top of the menu and the least likely ones at the bottom.

B. Prominently displaying predicted UI elements. The user interface elements (such as buttons) that are expected to be next chosen as per the user's current context are displayed prominently or towards the centre of the application, and the ones less likely to be chosen are temporarily hidden or displayed towards the end.

C. Selectively displaying application icons. In this approach the number of application icons to be displayed on the mobile device screen is reduced, and only the icons of applications predicted to be invoked are displayed. The advantage of this approach is to reduce clutter and make it easier for the user to access their applications of choice. The disadvantage is that the user might be annoyed if a prediction goers wrong and the application of the user's choice is not displayed.

D. Changing positioning of icons. In this approach, the application icons predicted to be invoked are positioned on the present screen of the user in a more prominent way, along with other icons on that screen. The increased prominence would make it easier for the user to invoke the applications of choice. The advantage of this approach is that even if the prediction goes wrong, the user is able to select manually the icon they would like to invoke.

A number of ways can accomplish changing the icon positioning to give increase in prominence to predicted application icons:

- Positioning the predicted icons at the centre of the screen
- Increasing the size of predicted application icons
- Highlighting the predicted icons by special effects or animation, such as flashing them or adding glowing edges to the icons
- Having special gestures for the user to invoke the predicted applications for the particular screen

E. Overlaying the predicted icons on the current application screen. In this approach the icons for the predicted application are superimposed on the current screen of the user. For example, during a phone call, the icons to be invoked are displayed along with the call screen. Figure 2 illustrates how the modified adaptive user interface looks like during an incoming phone call.

This method has the advantage of saving the user time, since they do not have to leave the current screen to access the applications predicted to be invoked.

The above methods can also be used in combination to provide optimal results and decrease the time taken for the user to access the application of their choice.

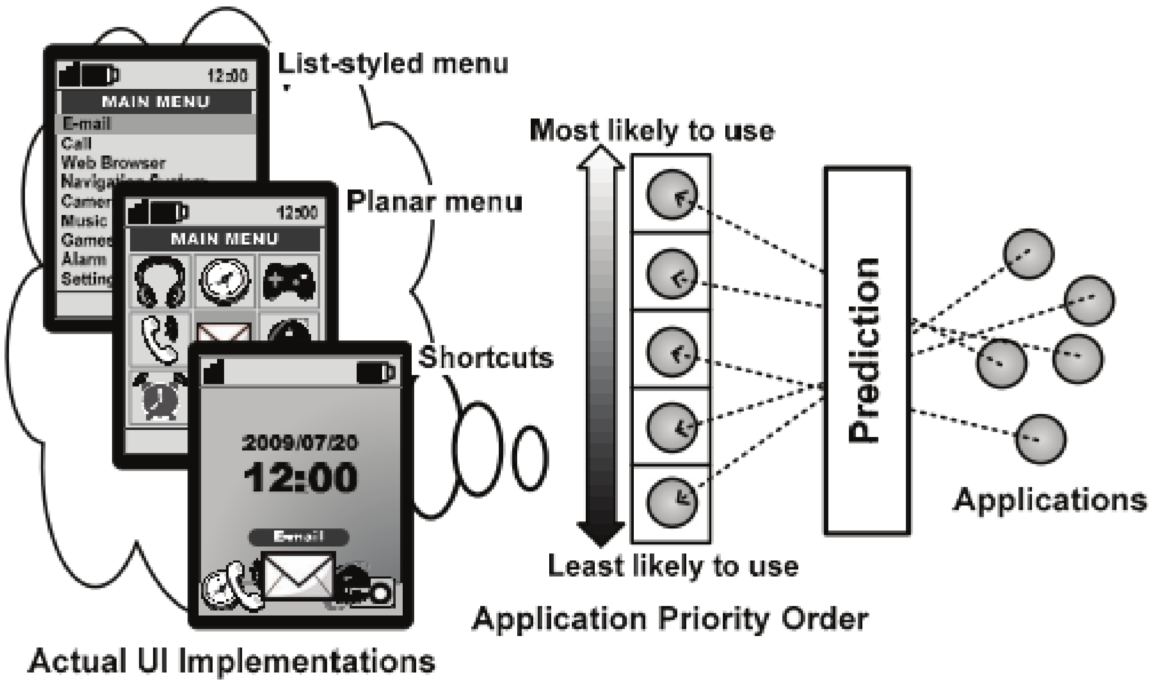


Figure 2.2.2 Application Priority Order

2.3 User-Adaptive Intelligent User Interfaces

The area of IUIs is one of the most heterogeneous research subjects, covering all kinds of different disciplines, which makes it difficult to give a common definition. IUIs try to solve the standard user interface question how the interaction between user and computer can be facilitated by means of artificial intelligence. In contrast to traditional human computer interaction (HCI), IUIs do not only focus on enabling the user to perform intelligent actions but on ways to incorporate knowledge to be able to assist the user in performing actions. In contrast to traditional research in artificial intelligence (AI), IUIs do not focus on making the computer smart by itself but to make the interaction between computer and human smarter. The goal of IUIs is to make the interaction itself as well as the presentation of information more effective and efficient to better support the user's current needs. The way to achieve this ranges from supporting a more natural interaction, e.g. by allowing multimodal or natural language input to intelligent tutoring systems and recommender systems. Based on the definition by Maybury and Wahlster [1998], we define IUIs as follows: Intelligent User Interfaces are human-machine interfaces that aim to improve the efficiency, effectiveness and naturalness of human machine interaction by representing, reasoning and acting on models of the user, domain, task, discourse, context, and device. In this paper, we focus on user-adaptive IUIs which are a subset of IUIs. User-adaptive IUIs hold a model for each individual user to be able to adapt its behaviour accordingly, which is not necessarily the case for all IUIs as often a generic user model suffices.

2.4 Challenges in Developing IUIs

The main goal in developing IUIs is that they should be usable, useful and trustable. This aligns with the main challenges identified by: Presentation, Competence and Trust. Presentation is concerned with the human computer interaction part of IUIs, whereas Competence focuses on the artificial intelligence techniques that can be applied. The development of IUIs has to take special care of Trust, as the user is not willing to delegate tasks to an IUI she does not trust, thus rendering the IUI useless. However, for IUIs it is much more challenging to induce user's trust in the system than for traditional user interfaces, because IUIs apply artificial intelligence techniques whose results can often not be directly foreseen by the user and thus reduce the user's feeling of being in control of the system.

In the following, we point out for each of these issues which challenges have to be faced when developing user-adaptive IUIs and describe possible ways to cope with them.

1. Presentation

For the presentation of IUIs, we at first need to consider how to design the interaction between the user and the IUI. Many IUIs are also augmentations of existing user interfaces, thus they have to be integrated into the existing layout offering the user a way to communicate with the IUI itself. Thereby, the IUI should not hamper the normal usage of the application. The interaction should also support some kind of forgiveness that is allowing the user to easily correct previously performed actions using an undo capability. Further, the design of the interaction tackles whether and how the user can instruct the IUI or whether an anthropomorphic agent is used for allowing the user to communicate with the IUI. These issues are closely related to trust issues that will be discussed later, i.e. how the user can control the system and which expectations are raised by the IUI. Another important factor regarding the presentation is unobtrusiveness. The intelligent support should not distract the user from normal usage of the application. A counterexample for this factor is Microsoft's Office Assistant that is constantly moving and thus drawing the user's attention to it without providing any relevant help for the user's current task. Wexelblatt and Maes [1997] propose to reduce the distraction of the user by minimizing the amount of interruptions and deferring interruptions until they are less disruptive. Another way to cope with this issue is to support different levels of obtrusiveness (or proactivity) depending on the information importance or the certainty in the action (e.g. applied by [Maes, 1994; Horvitz, 1999; Hartmann et al., 2009]). Furthermore, the user-adaptive IUI should be able to adapt its presentation to different users, devices and situations. For example, a novice user needs more explanations than an expert user and voice output is perhaps suitable for mobile usage, but not if she is sitting in a library. Further, as the interaction costs for interacting with applications via a mobile phone are much higher than in a traditional desktop setting, more support may be desirable in these settings. However, adaptivity does not only influence the presentation of an IUI, but also how much the user trusts the system or which demands it puts on the underlying algorithms (i.e. affecting the competence of the IUI).

2. Competence

The competence of an IUI is determined by the underlying algorithms. However, as we focus in this paper on human computer interaction issues, we only provide here a short overview of the main challenges that have to be faced for the competence of a system. At first, many user-adaptive IUIs cannot rely on a huge amount of training data at the beginning, especially if they need training data for each individual user. Thus, the algorithms used for user-adaptive IUIs mostly have to be able to deal with few usage data. For that reason, systems that just learn from observation are usually not of great aid at the beginning ("slow-start problem"). This problem can be faced e.g. by relying on predefined models or by using a default model that is inferred from the models of other users. However, the former requires great modelling effort by a developer and the latter can cause privacy problems. A second problem that arises is that the user's behaviour changes over time. Especially when she starts interacting with an application as novice, her usage patterns as expert will later dramatically differ from the initial patterns. For that purpose, ageing can be used that weighs older interactions as less important than more recent interactions. Finally, in order to be beneficial for the user, the artificial intelligence of course needs to produce correct results with a high accuracy. This is especially important as erroneous support can easily lead to losing the user's trust.

3. Trust

The trust the user puts in an IUI is influenced by many factors especially by presentation issues as discussed before. In the following, we state the main challenges identified in literature that have to be considered when building trustable IUIs. At first, it is essential that the user feels in control of the system. The user should be able to correct and adjust the IUI's actions and to control its autonomy. One possibility to control the system's actions is to require the user to approve or disapprove the system's actions or by letting the user specify confidence thresholds for actions. However, giving the user the maximal control at all times is usually not desirable as the users differ in their desire for control and too much control may lead to distraction and time-wasting. The amount of control should also depend on the criticalness of the task, e.g. for non-critical tasks, like prefilling data in input fields, a lower level of control is needed than for automatically buying goods. Another factor that influences how much control the user wants to exert is her trust in the system that (hopefully) evolves over time. For all those reasons, an IUI should support variable levels of control that can also be adjusted by the user. Another important issue for establishing the user's trust in the system is intelligibility, i.e. to enable the user to understand the system's actions. As stated by Maes, a user more likely trusts an IUI if she sees in advance what the agent would do. One way to achieve intelligibility is transparency, i.e. that the IUI helps the user understand its actions. Transparency can be realized by an IUI for example by giving feedback of its actions by being able to justify its actions, or by making the user aware of automatic adaptations. Another way of increasing the system's intelligibility is to give the user access to the knowledge source that was used for providing support. The intelligibility of the system's actions should thus support the user to develop an appropriate model of the IUI's behaviour. Thereby, it is not necessary to mediate a complete model of the IUI, as "understanding comes from a careful blend of hiding and revealing system state and functioning". Another factor influencing the intelligibility of the system is how predictable the system's actions are perceived by the user and finally which expectations she poses in the system. Erroneous higher expectations can easily lead to disappointing the user and thus stopping the user from using the system. This is also one of the main reasons why many researchers argue against using anthropomorphic agents for communicating with IUIs, as they are perceived by the user to be similar to a human being and that they thus could also take responsibility for their actions. Finally, for IUIs that share information between users, privacy has to be regarded. Thereby the requirements that are posed on privacy differ between applications. For example, the users of FireFly1, an application for sharing preferences for music or movies, did not perceive this sharing as critical, whereas users of the Doppelganger system that provides personalized news which also considered the news that a colleague is usually reading, had strong privacy concerns against the system. This might be the case as the data differed in their level of importance to the user and as the data was not anonymized in the Doppelganger system in contrast to the Firefly system. Besides anonymizing the data, another solution proposed for this problem is to split the user model in a private and a public part.

4. Adaptivity Challenges

There has been a debate for years whether automatic adaptation optimizes the user interface or disorients the user. The IUI community often favours automatic adaptivity, whereas the HCI community tends towards adaptable approaches that allow the user to customize her interface herself without any automatism. However, many studies have shown that users often fail to use the offered adaptation mechanisms, and when they do, they often do not re customize it if their working habits change. The value of an adaptation for a user is usually measured as the user interface's usability, i.e. the user's efficiency, effectiveness and her satisfaction.

An increased efficiency can lead to a decreased awareness of advanced features, probably because the adaptivity allows them to focus more on the task itself and not on the available menu elements. Thus it can hamper the learning of novel user interfaces. However, for seldom used applications or applications in which the user is already an expert, no awareness of advanced features is needed. In this section, we focus on the usability aspect and summarize the main results from user studies reported in literature. They investigate when an adaptation is useful and how adaptation has to be designed to improve the usability and to avoid confusion. The identified factors thereby comprise presentation as well as competence issues and also influence the trust in the system. An overview of these factors can be found in the table below.

Presentation	Spatial Stability, Locality
Competence	Accuracy , Predictability
Further factors	Interaction frequency, Task Complexity, Average interaction costs

Table 2.4.1: Factors influencing the value of an adaptation for a user

Regarding the presentation of the adaptation, spatial stability increases the user satisfaction and that high locality improves discoverability of the adaptation, i.e. that the promoted user interface element appears close to its original position. The spatial stability is required to enable the user to maintain a mental model of the application. An example for spatial in/stability are the Smart Menus used in Microsoft Office, in former versions they hid infrequently used items from view, which caused many negative reactions among users due to their spatial instability, whereas in Office 2007 the menus contain predefined adaptive parts (e.g. displaying the most recently used items) which seems to have much more supporters. Another important factor is the behaviour of the algorithm that adapts the UI, i.e. its accuracy and its predictability (which of course relates closely to the accuracy issue discussed before). It is found that an increase in each of the factors leads to a strongly improved user satisfaction. An increased accuracy moreover leads to improved user performance and more frequent use of the adaptive part. The increase in accuracy thereby had stronger effects on user performance, on how often they utilized it, and on some satisfaction ratings. Further, users are more likely able to build mental models for applications which are of low complexity or with which they frequently interact. These mental models can reduce the positive effect of adaptive parts if the interaction costs for using the unadapted version and the adapted version do not differ much, e.g. the amount of required clicks is about the same.

3. ANALYSIS AND DESIGN

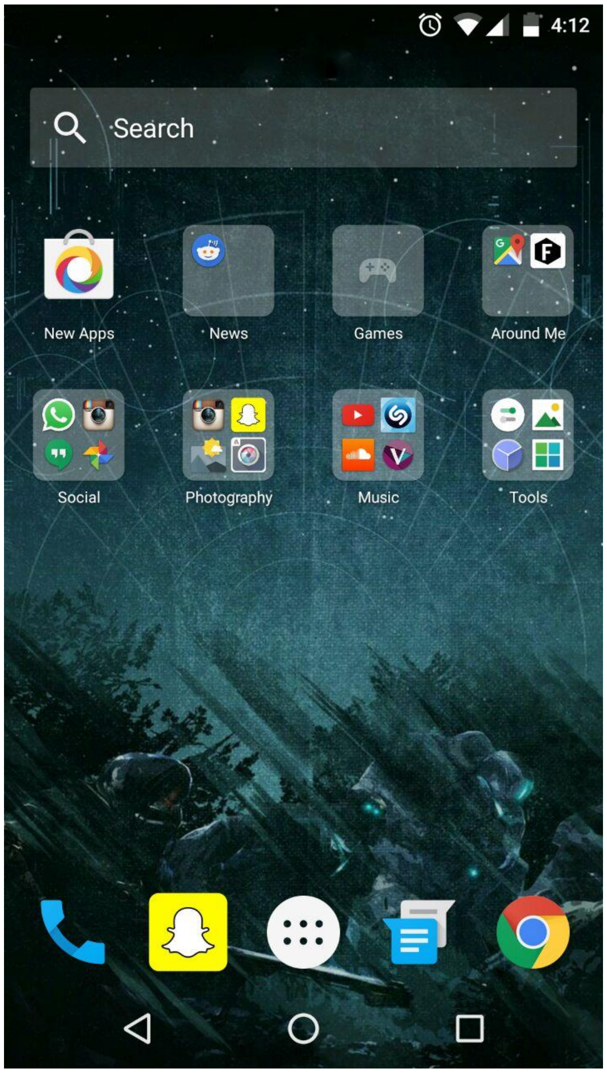


Figure 3.1

3.1 SPECULATION

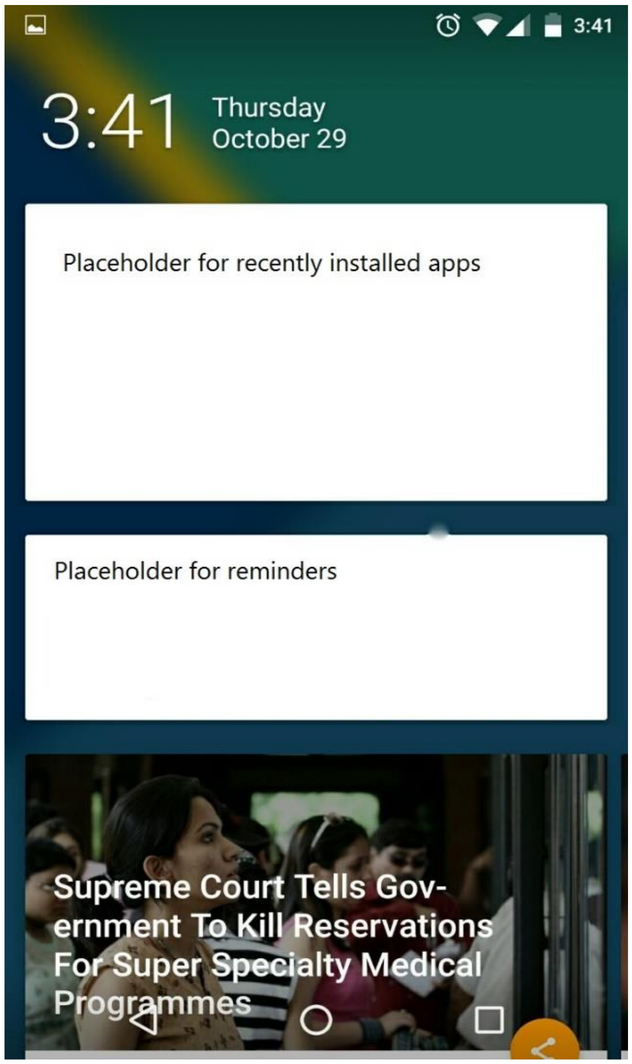
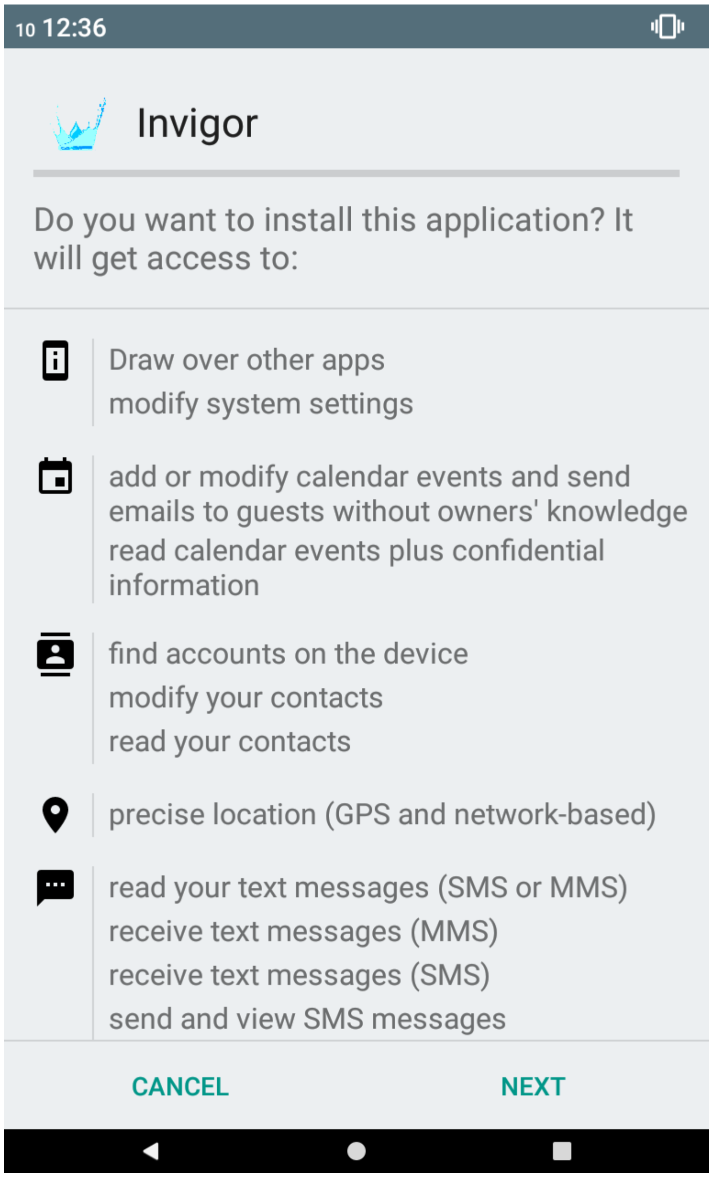

3:41
3:41
Thursday
October 29
Placeholder for recently installed apps
Placeholder for reminders
Supreme Court Tells Government To Kill Reservations For Super Specialty Medical Programmes

Figure 3.1.1

3.2 IMPLEMENTATION



10 12:36

Invigor

Do you want to install this application? It will get access to:

Draw over other apps
modify system settings

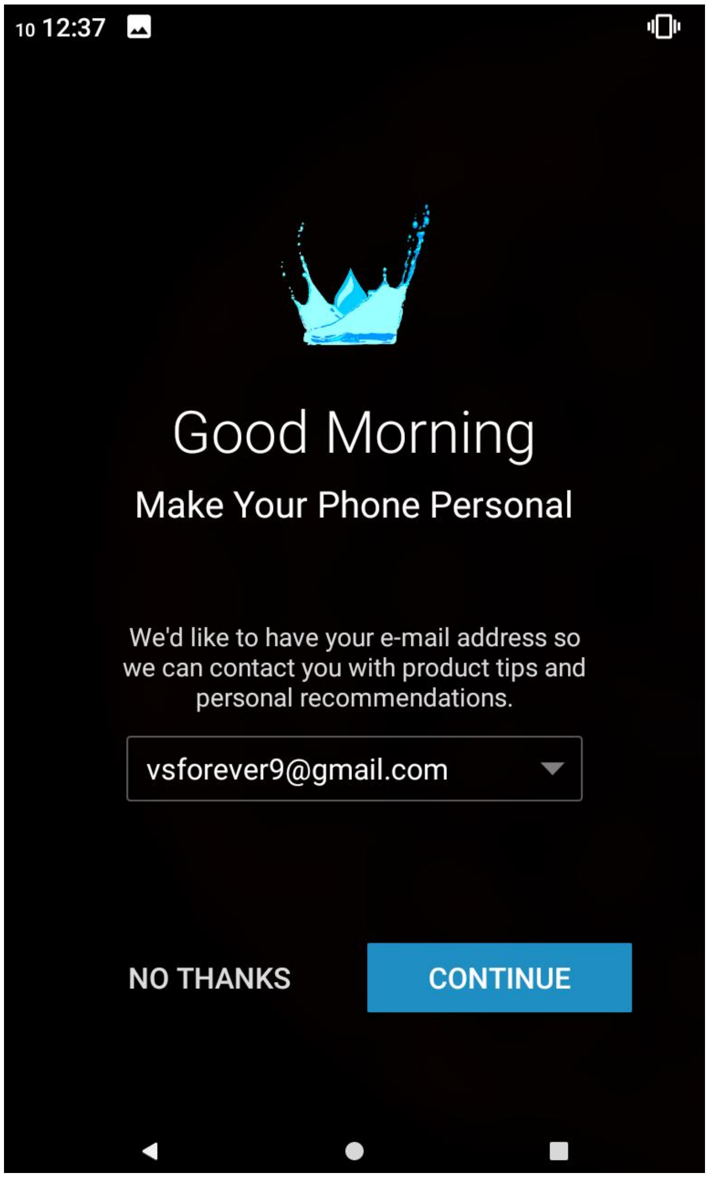
add or modify calendar events and send emails to guests without owners' knowledge
read calendar events plus confidential information

find accounts on the device
modify your contacts
read your contacts

precise location (GPS and network-based)

read your text messages (SMS or MMS)
receive text messages (MMS)
receive text messages (SMS)
send and view SMS messages

Figure 3.2.1



10 12:37

Good Morning

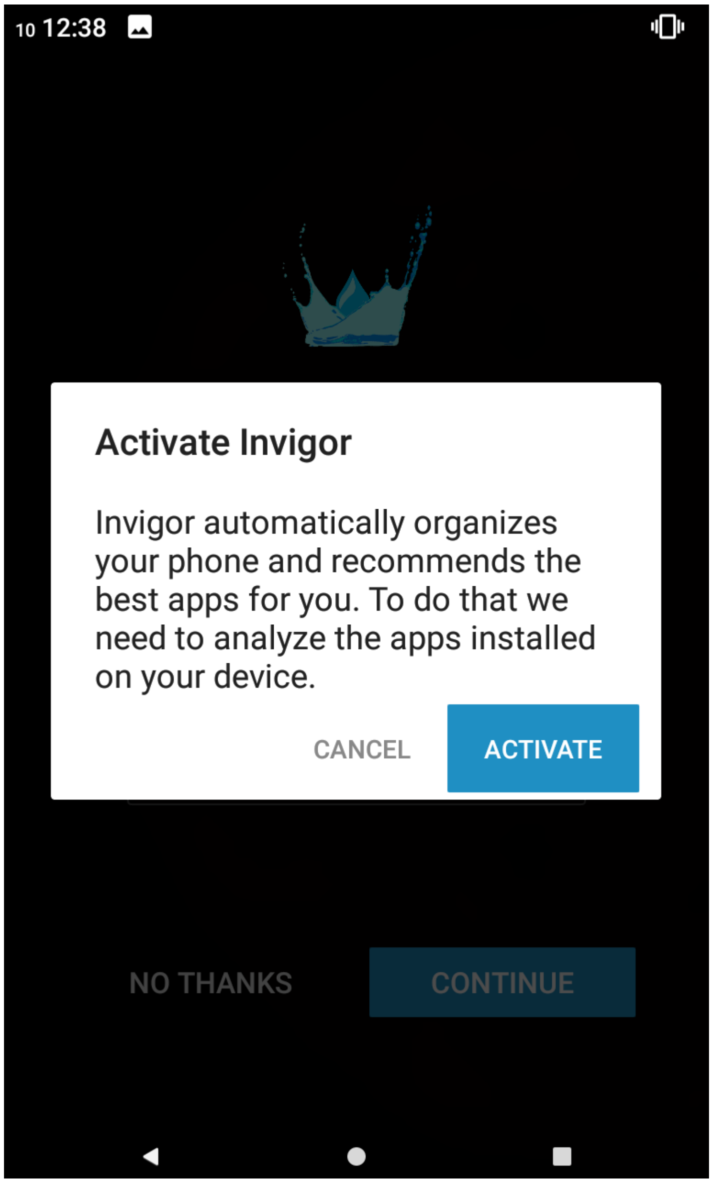
Make Your Phone Personal

We'd like to have your e-mail address so we can contact you with product tips and personal recommendations.

vsforever9@gmail.com

Figure 3.2.2

Figure 3.2.3

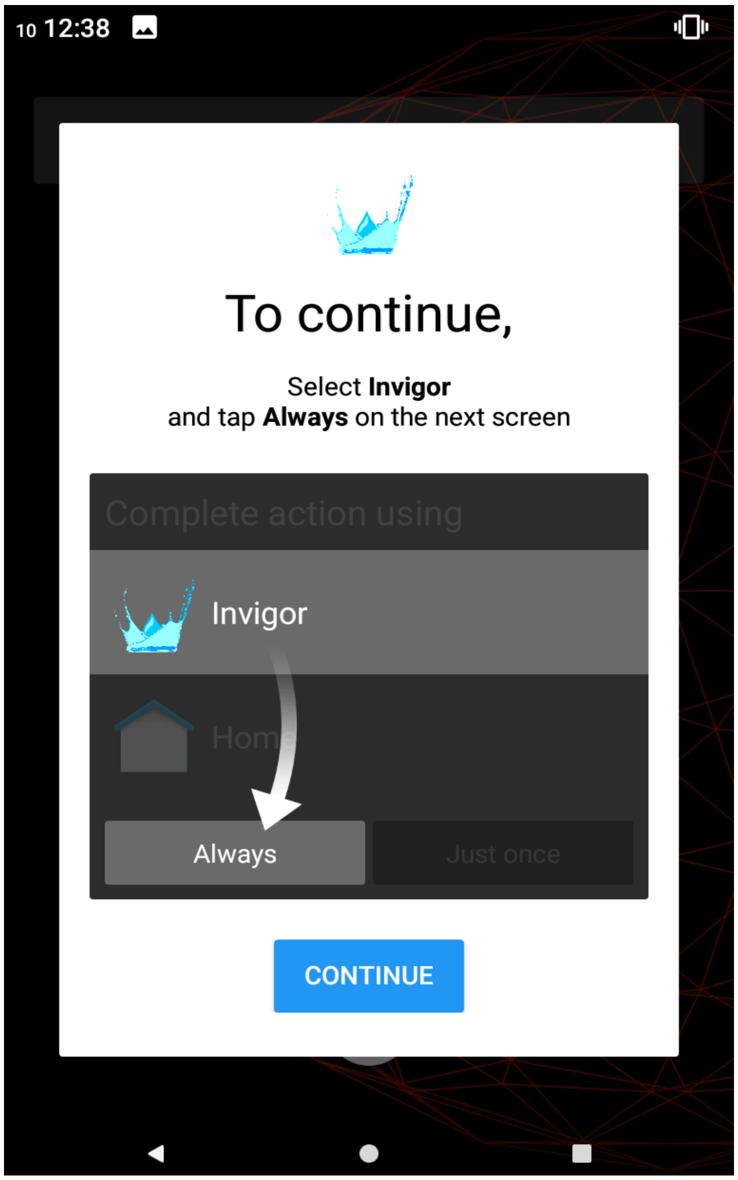


10 12:38

Activate Invigor

Invigor automatically organizes your phone and recommends the best apps for you. To do that we need to analyze the apps installed on your device.

Figure 3.2.4



10 12:38

To continue,

Select **Invigor**
and tap **Always** on the next screen

Complete action using

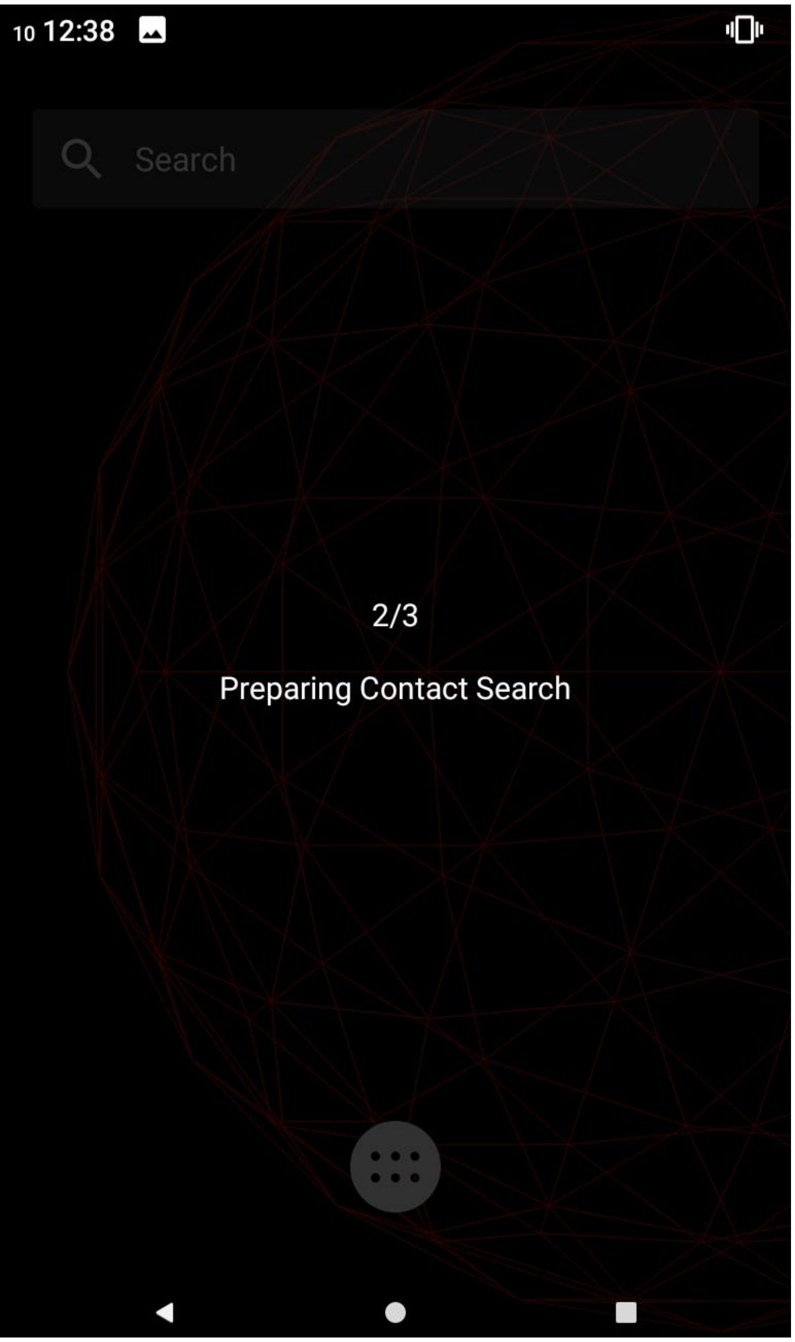


Figure 3.2.5

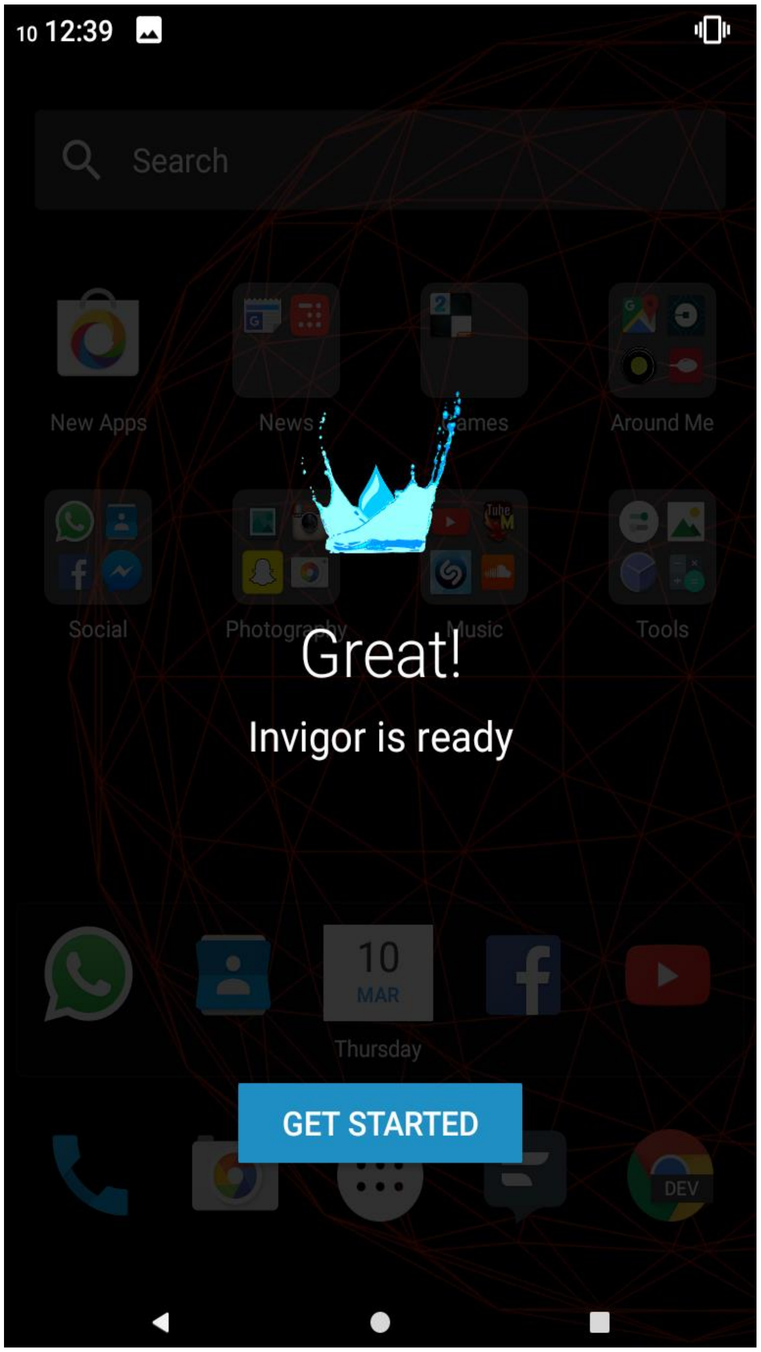


Figure 3.2.6

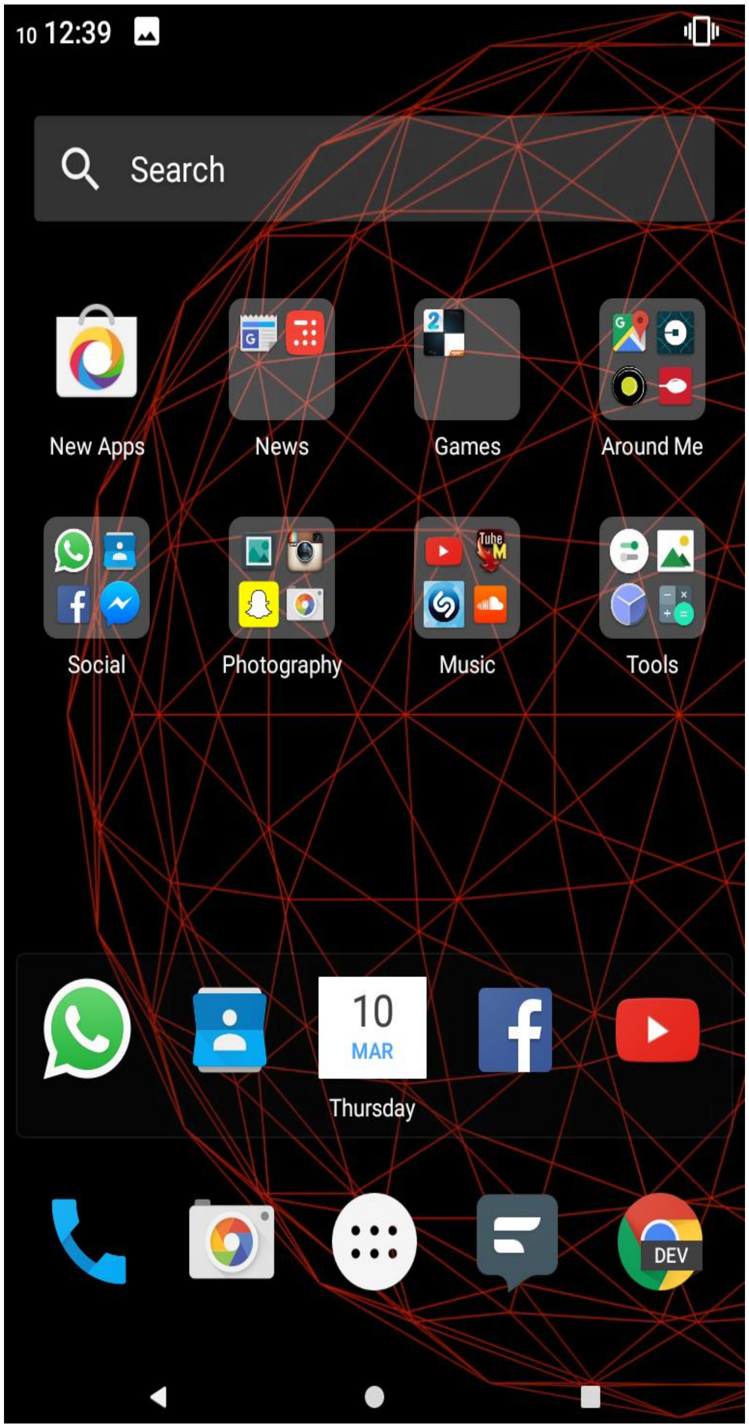


Figure 3.2.7

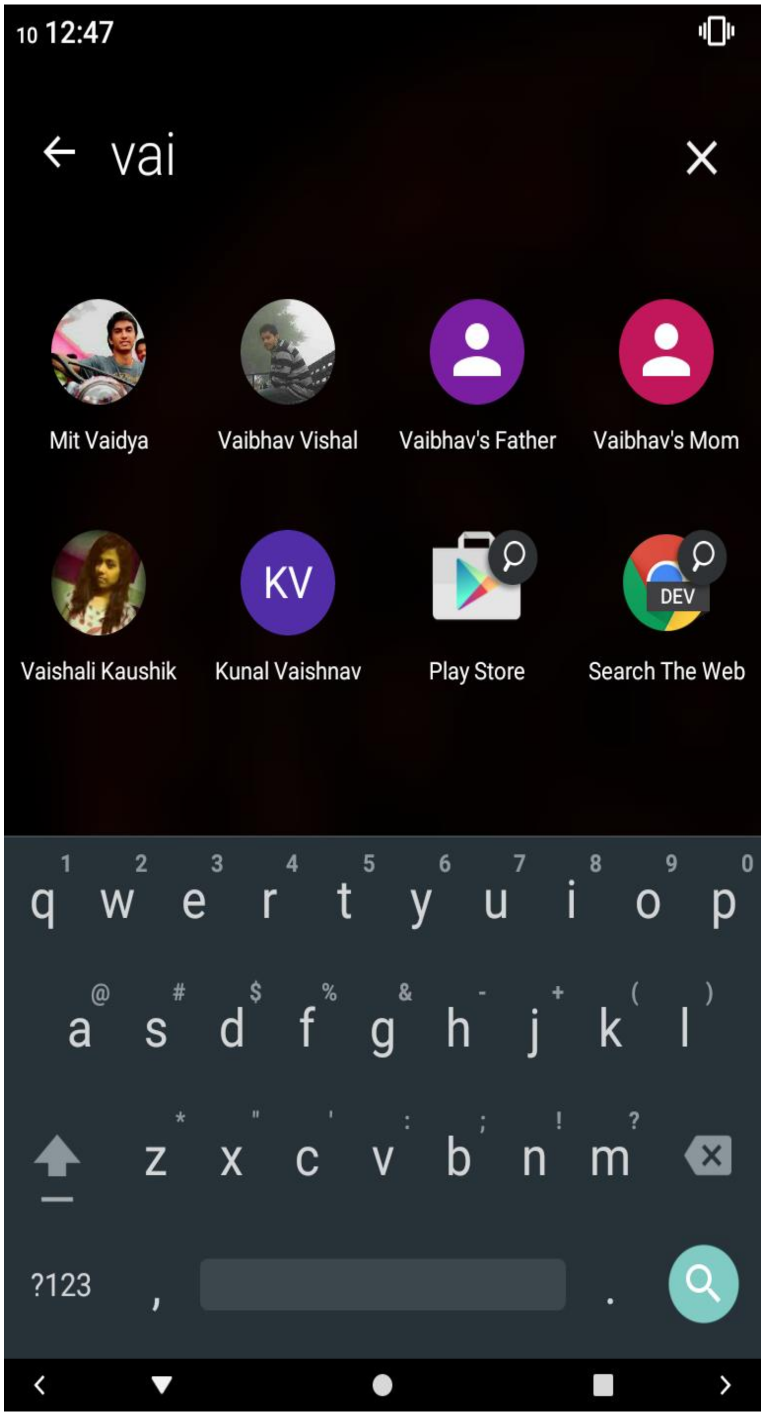


Figure 3.2.8

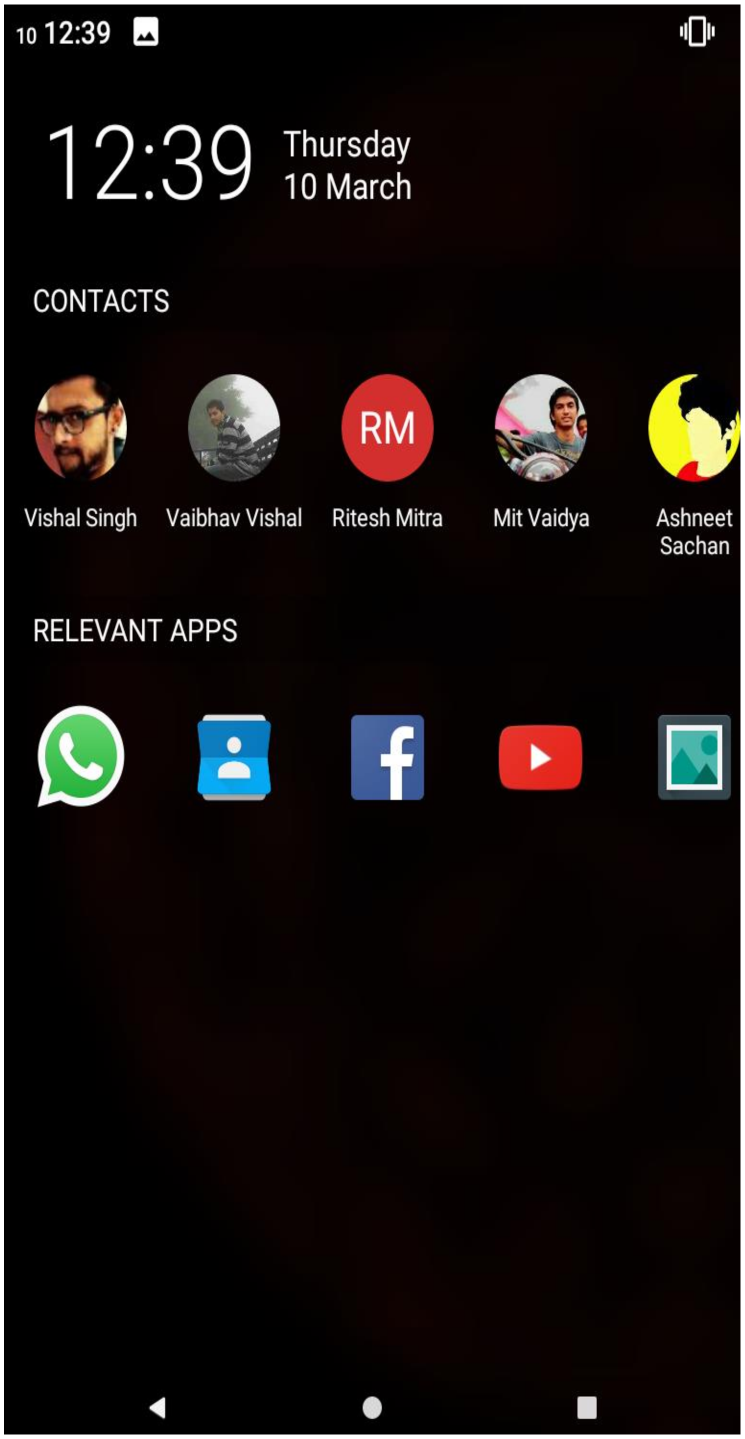
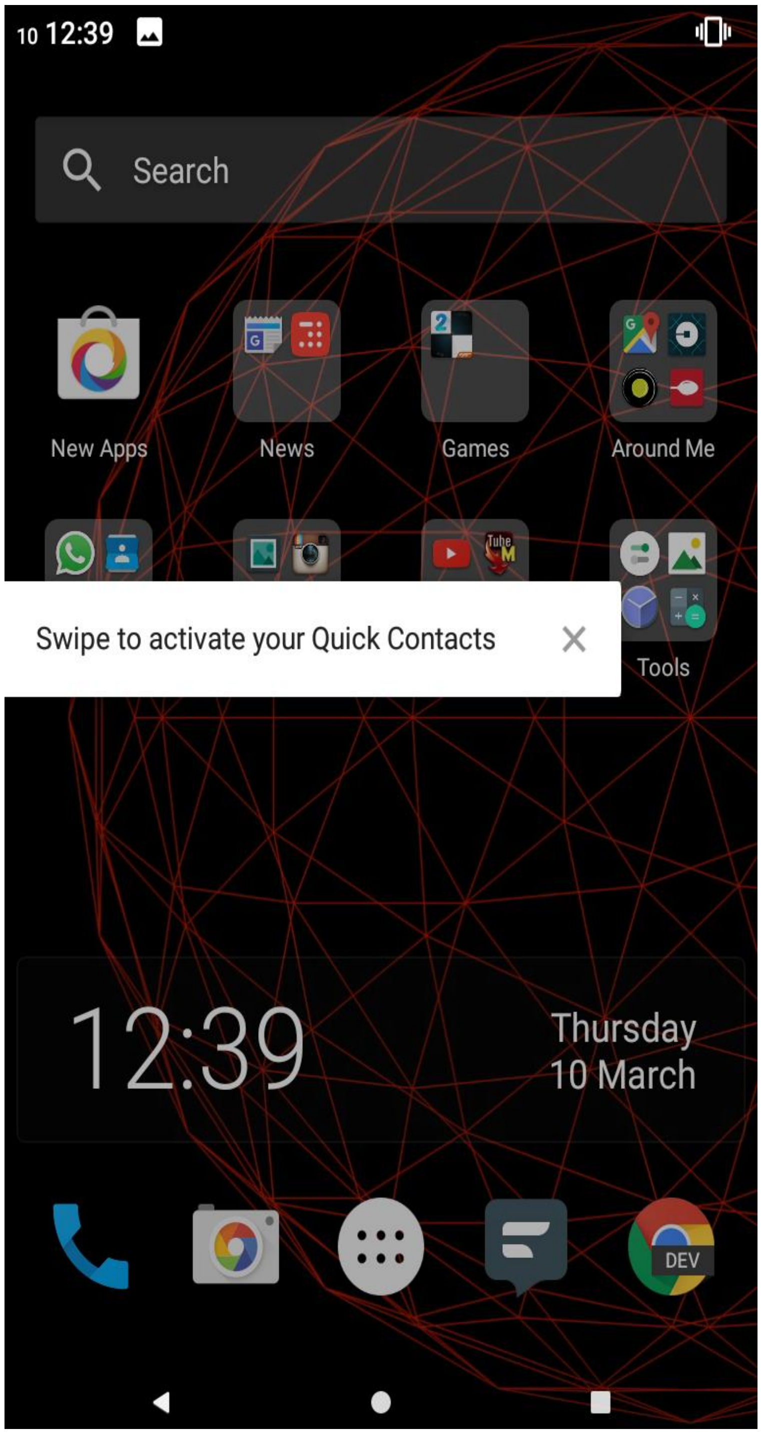
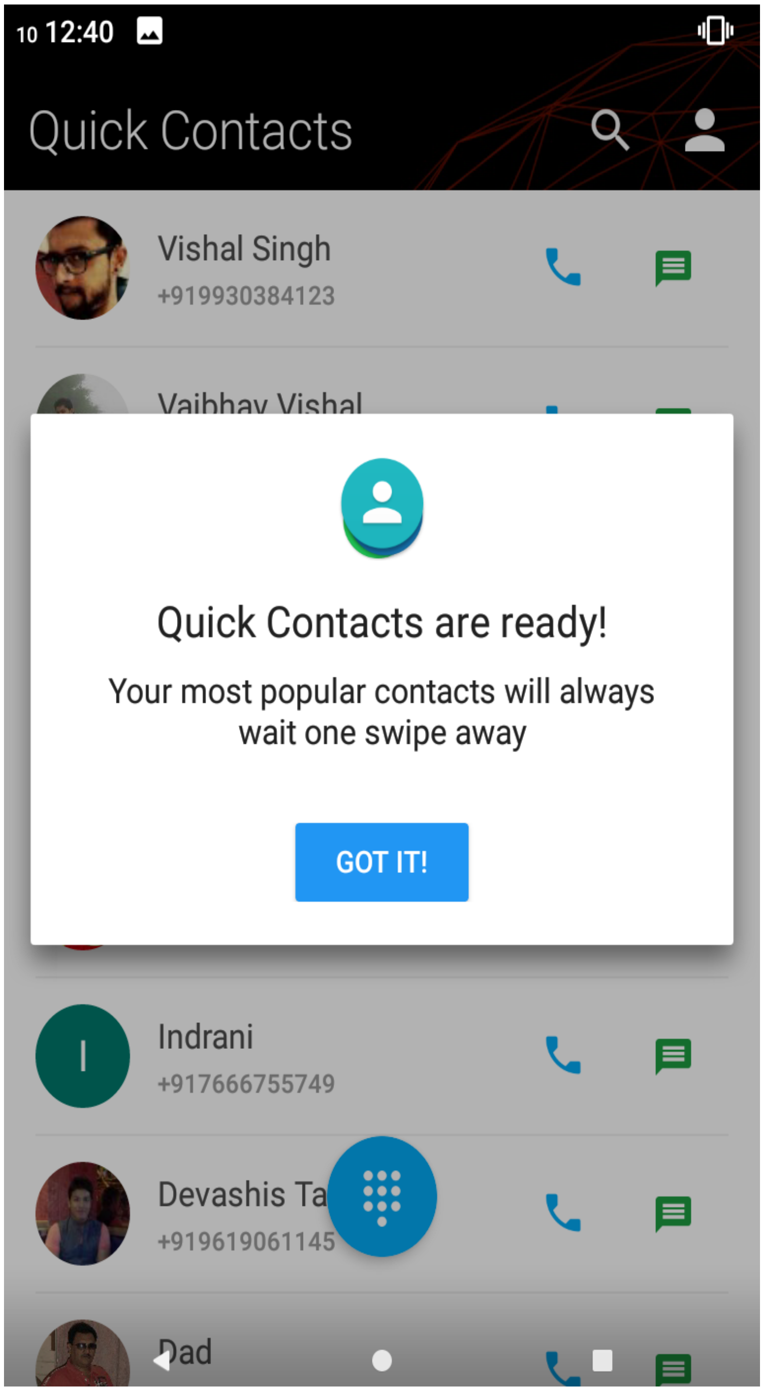


Figure 3.2.9



Swipe to activate your Quick Contacts

Figure 3.2.10



10 12:40

Quick Contacts

Vishal Singh
+919930384123

Vaibhav Vishal

Quick Contacts are ready!

Your most popular contacts will always wait one swipe away

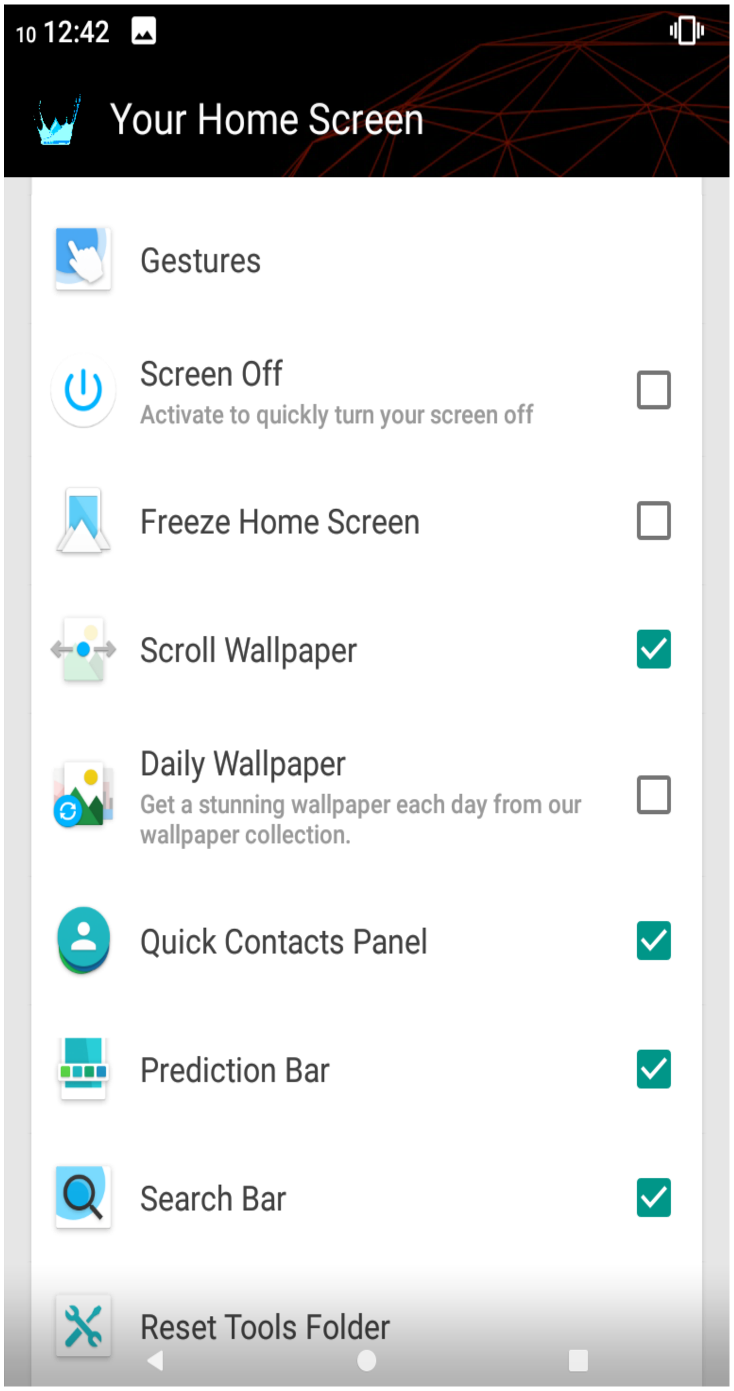
GOT IT!

Indrani
+917666755749

Devashis Ta
+919619061145

Dad

Figure 3.2.11



10 12:42

Your Home Screen

Gestures

Screen Off ☐
Activate to quickly turn your screen off

Freeze Home Screen ☐

Scroll Wallpaper ✓

Daily Wallpaper ☐
Get a stunning wallpaper each day from our wallpaper collection.

Quick Contacts Panel ✓

Prediction Bar ✓

Search Bar ✓

Reset Tools Folder

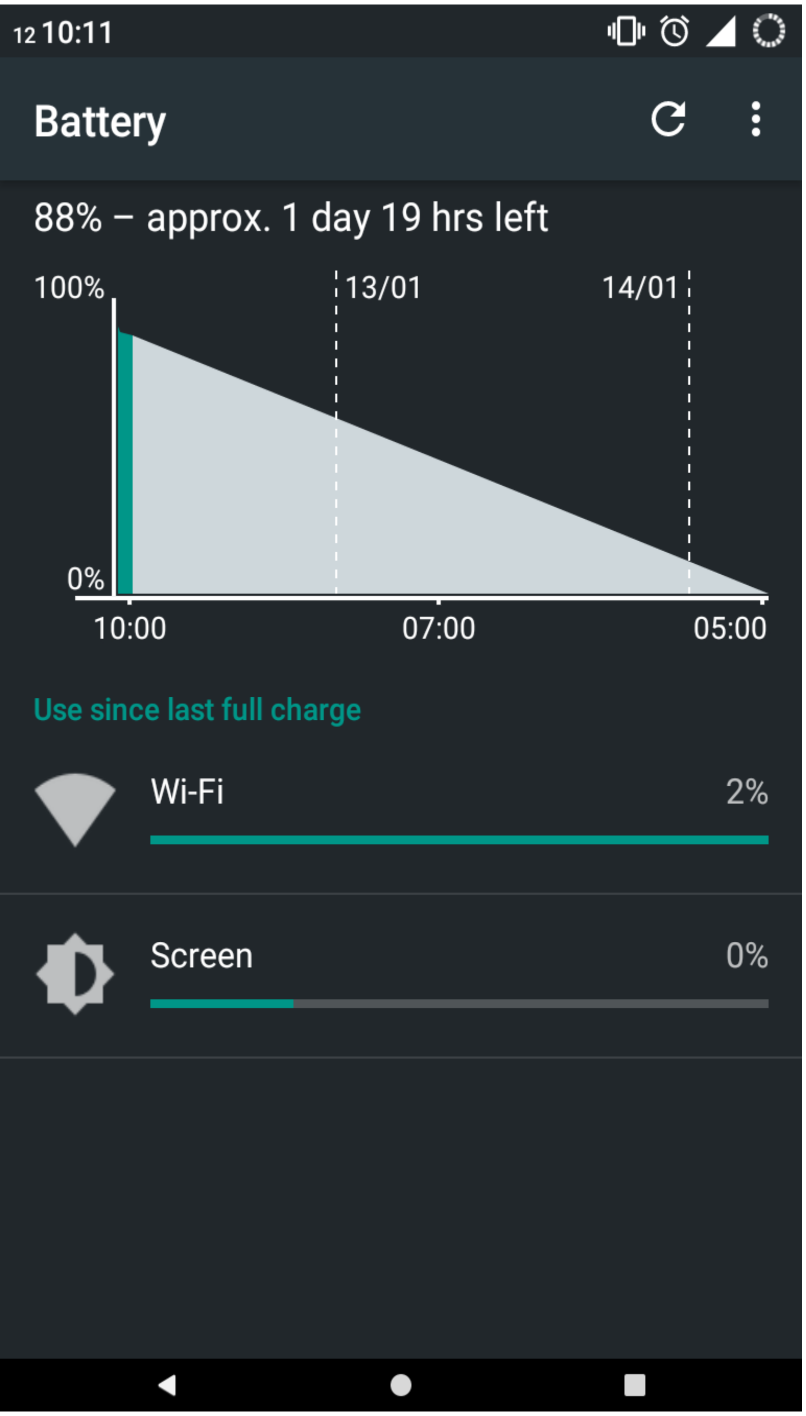
Figure 3.2.12

4. BATTERY OPTIMISATION

In order to perform context-aware battery management, we need to detect various data from the device's operating system, including battery status, list of processes running on the device, the calls made on the device (under the assumption that this will be regarded as a "crucial" application), and finally, some context information to allow us to predict the next charging opportunity. The *Process Monitor* is responsible for keeping track of the processes running on the device and informing the viceroy whenever a new process is detected. The *Battery Monitor* probes the battery periodically and enquires about remaining charge and voltage level. The *Context Monitor* is responsible for sensing and storing context information (such as location) on the phone. The *Call Monitor* logs communication (incoming/outgoing calls, incoming/outgoing SMSs).

We regard telephony as the "crucial application" for mobile phones, in that users always want to be able to use this application (e.g. for emergency calls or rendezvous with friends). The "non-crucial applications" should not be allowed to drain the battery to the stage where the user is deprived of telephony service. One obvious and oft-used method of predicting battery lifetime is to monitor the rate of drain of the battery and extrapolate this linearly to exhaustion.

One question that might be asked is: can we predict the discharge speedup factor for a new set of applications that have not previously been observed together? This would allow us to warn the user immediately as they attempted to start a new application, rather than having to wait until that application consumed some power. However, we have discovered that this is not easily possible, because the discharge speedup factors of applications cannot simply be added. Intuitively, for applications that do not make use of the same system resources (e.g movie player and web server), energy usage can be added, while for applications that share system resources (e.g movie player and audio player), it is not possible to add energy usage profiles. However, once the node has an opportunity to measure the discharge speedup factor, it can be cached and later used to flag immediate warnings to the user when they attempt to start a new application.

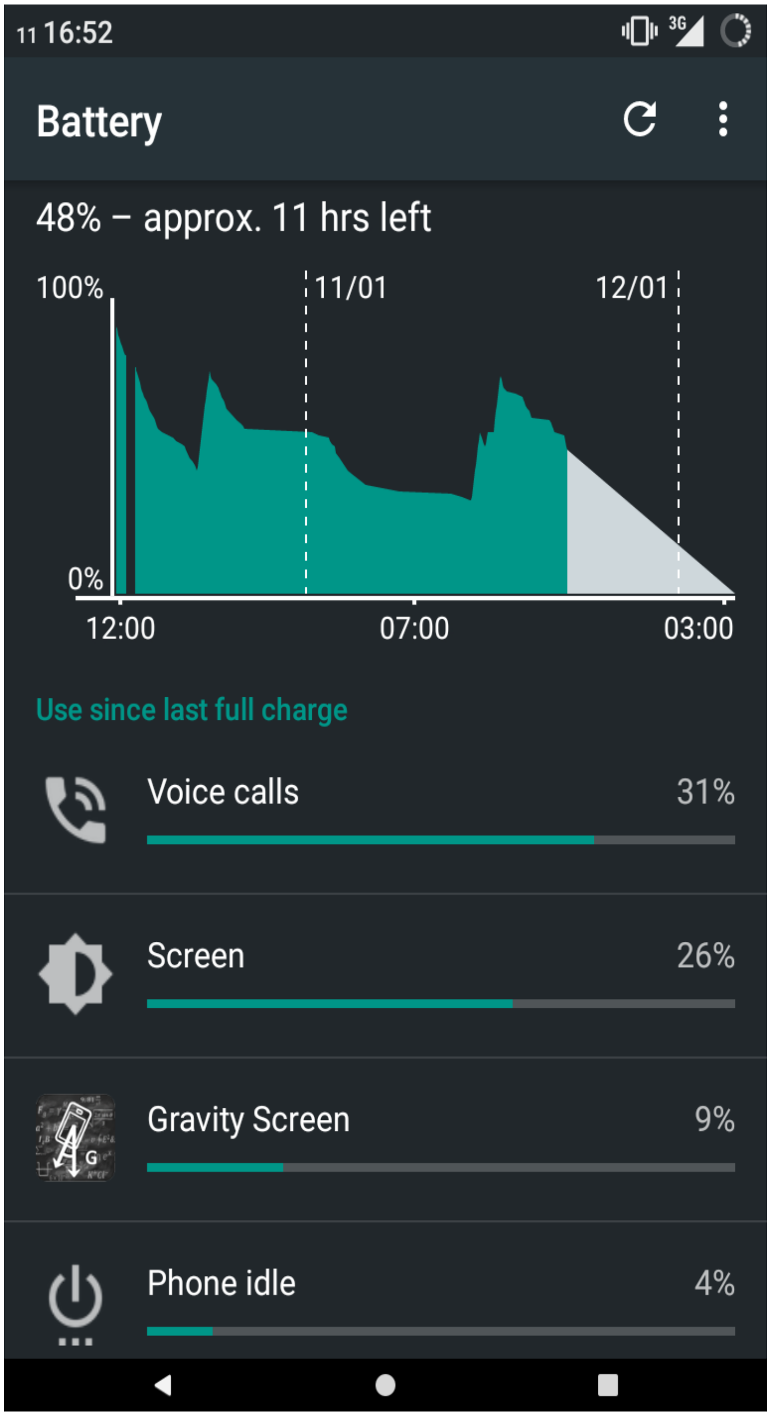


88% – approx. 1 day 19 hrs left

Use since last full charge

Wi-Fi	2%
Screen	0%

Figure 4.1



48% – approx. 11 hrs left

Use since last full charge

Voice calls	31%
Screen	26%
Gravity Screen	9%
Phone idle	4%

Figure 4.2

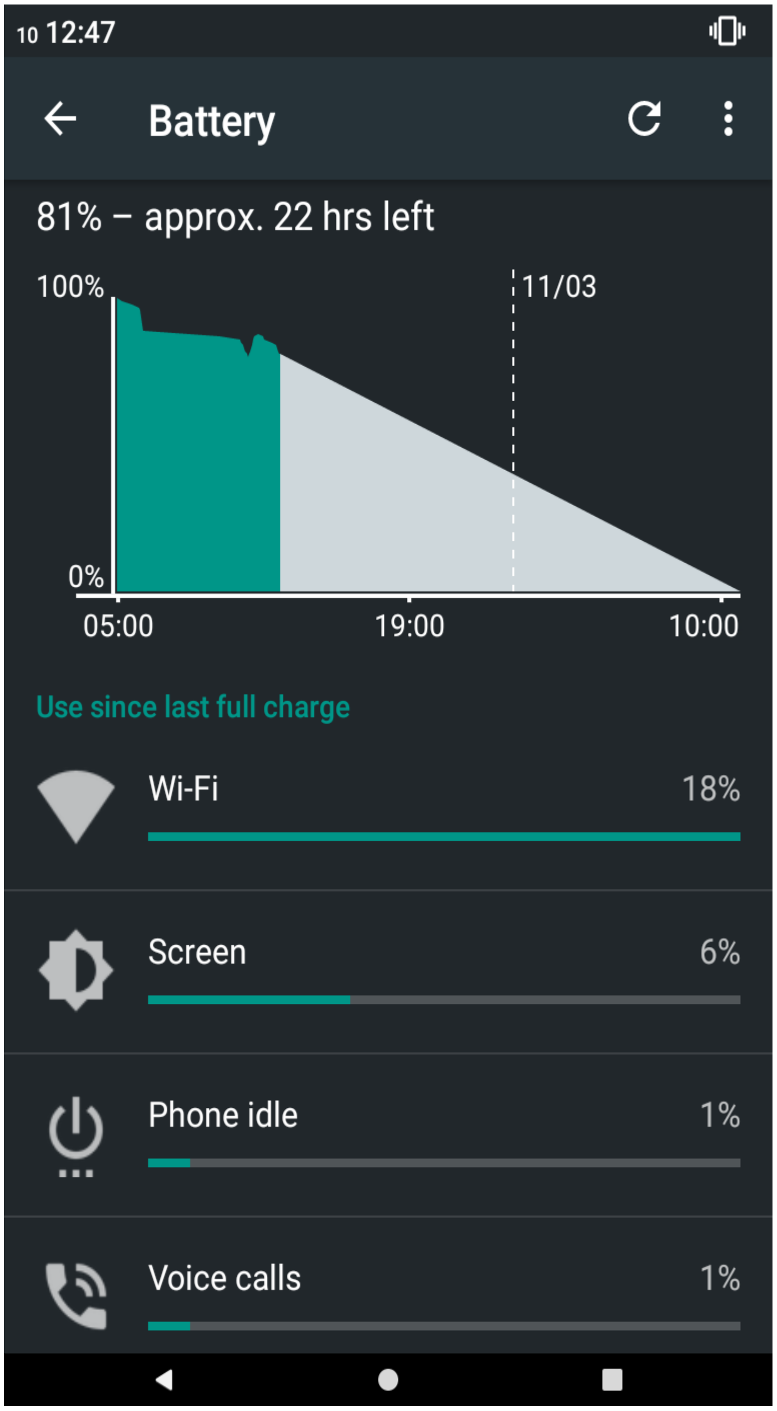


Figure 4.3

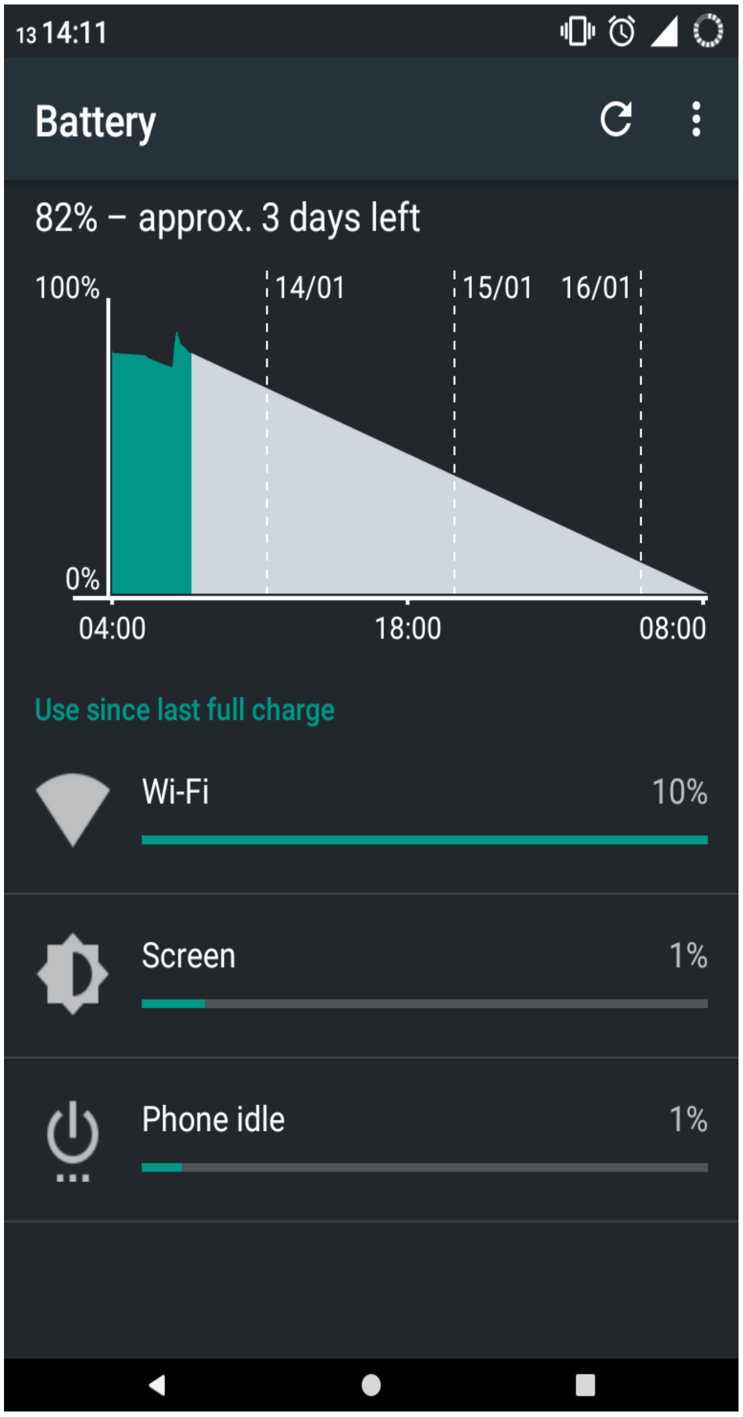


Figure 4.4

5. METHODS IMPLEMENTED

Following are the algorithms implemented in the designing of the Invigor's User adaptive Interface.

i. Service Usage algorithm

```
get all Usage Log data
get Interaction Level "serviceEvent"
for each "serviceDetails"
count each Service (Search, Travel etc.)
if count <= x AND number of items > y
Update UI & User Profile (Remove service from screen)
```

ii. Favorites Algorithm

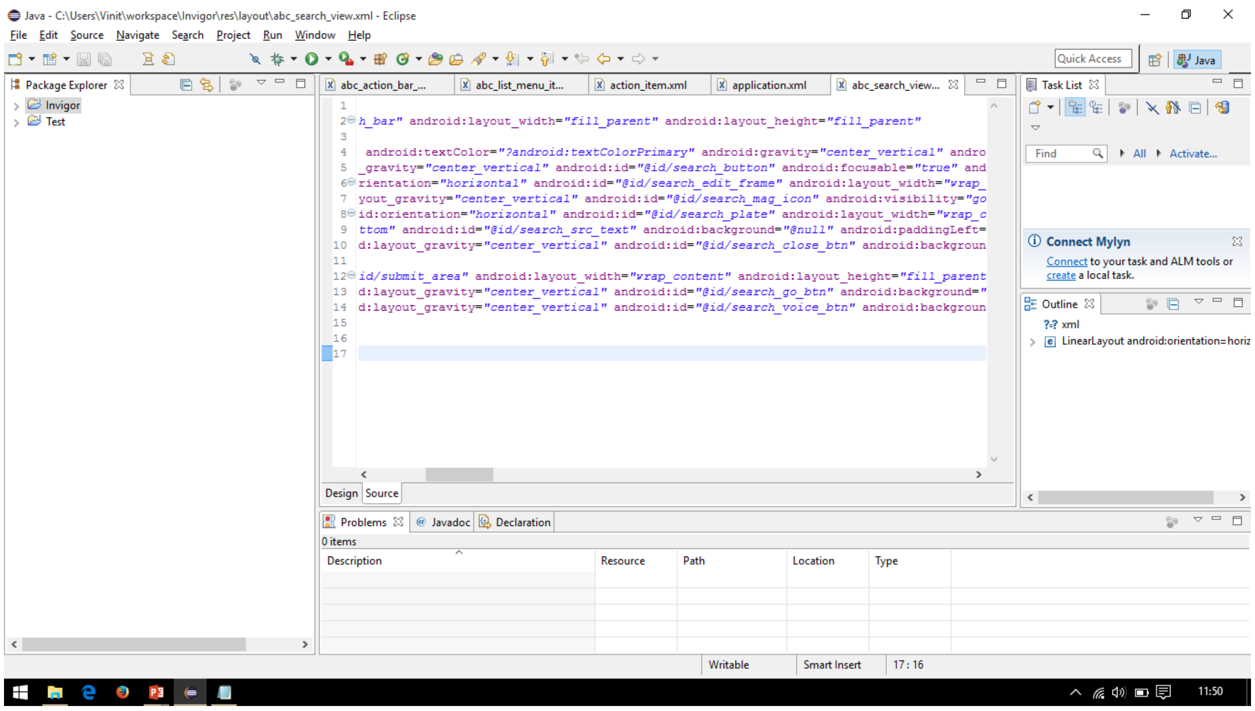
```
get all Usage Log data
get Interaction Level "serviceEvent"
for each "serviceDetails"
for each Service (Search, Travel etc.)
add value of "userInput" into hashmap
count duplicate values
if count >= x
update favorites table
```

iii. Personalize Request Algorithm

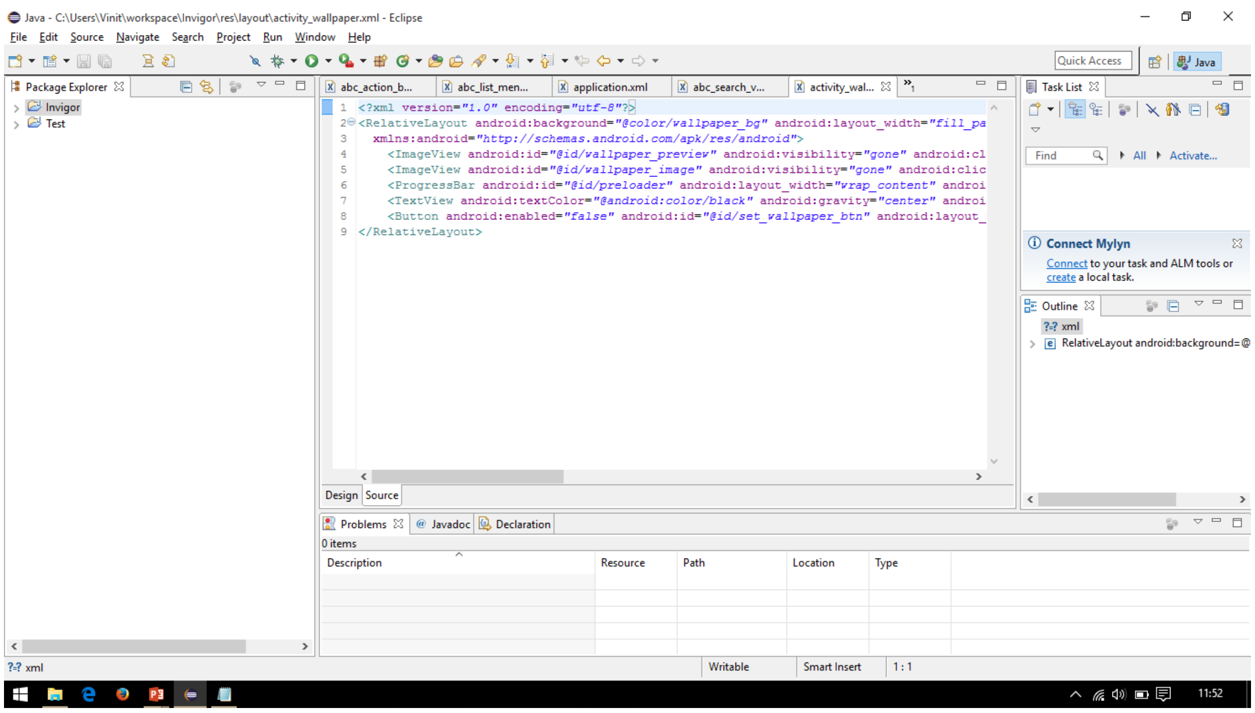
```
get SignalStrength
get BatteryLevel
if feedbackMode == "Video"
if SignalStrength < 30
or
if Battery Level < 20
prompt user to change feedbackMode
```

iv. Update User Profile Algorithm

```
get all Usage Log data
get Interaction Level "serverComm"
if feedbackMode = "Video" or "Audio"
for each serverResponse
 if next row serviceDetails =
"KEYCODE_VOLUME_UP"
While count < x
If next row serviceDetails =
"KEYCODE_VOLUME_UP"
Update User Profile Volume Parameter
```

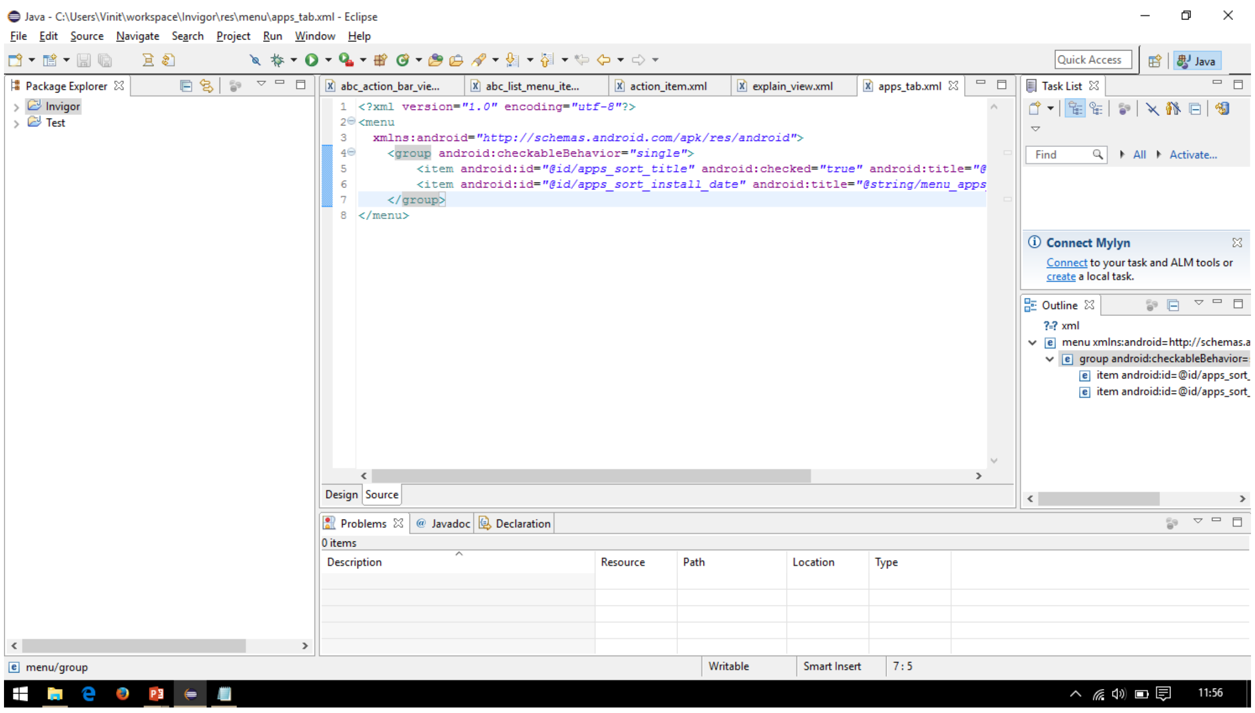


```

h_bar" android:layout_width="fill_parent" android:layout_height="fill_parent"

 android:textColor="?android:textColorPrimary" android:gravity="center_vertical" andro
_gravity="center_vertical" android:id="@id/search_button" android:focusable="true" and
rientation="horizontal" android:id="@id/search_edit_frame" android:layout_width="wrap_
yout_gravity="center_vertical" android:id="@id/search_mag_icon" android:visibility="go
id:orientation="horizontal" android:id="@id/search_plate" android:layout_width="wrap_c
ttom" android:id="@id/search_src_text" android:background="@null" android:paddingLeft=
d:layout_gravity="center_vertical" android:id="@id/search_close_btn" android:backgroun

id/submit_area" android:layout_width="wrap_content" android:layout_height="fill_parent
d:layout_gravity="center_vertical" android:id="@id/search_go_btn" android:background="
d:layout_gravity="center_vertical" android:id="@id/search_voice_btn" android:backgroun
```



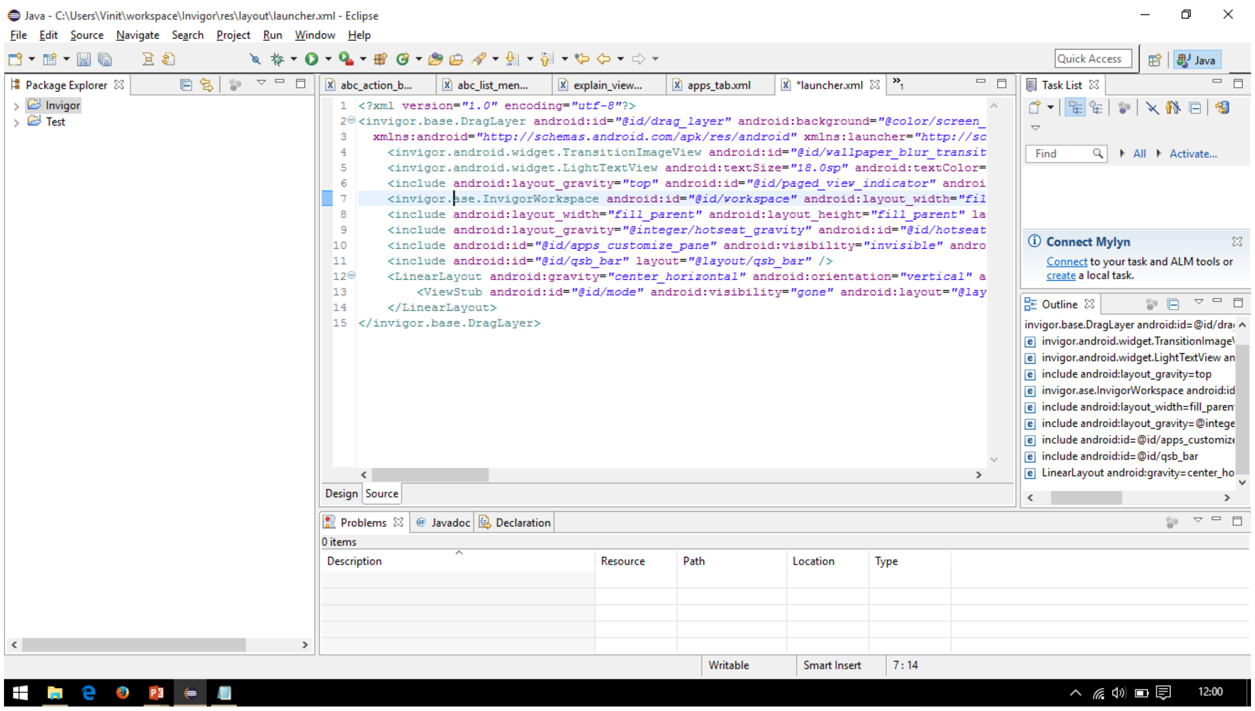
Java - C:\Users\Vinit\workspace\Invigor\res\layout\activity_wallpaper.xml - Eclipse

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:background="@color/wallpaper_bg" android:layout_width="fill_pa
  xmlns:android="http://schemas.android.com/apk/res/android">
    <ImageView android:id="@id/wallpaper_preview" android:visibility="gone" android:cl
    <ImageView android:id="@id/wallpaper_image" android:visibility="gone" android:clic
    <ProgressBar android:id="@id/preloader" android:layout_width="wrap_content" androi
    <TextView android:textColor="@android:color/black" android:gravity="center" androi
    <Button android:enabled="false" android:id="@id/set_wallpaper_btn" android:layout_
</RelativeLayout>
```



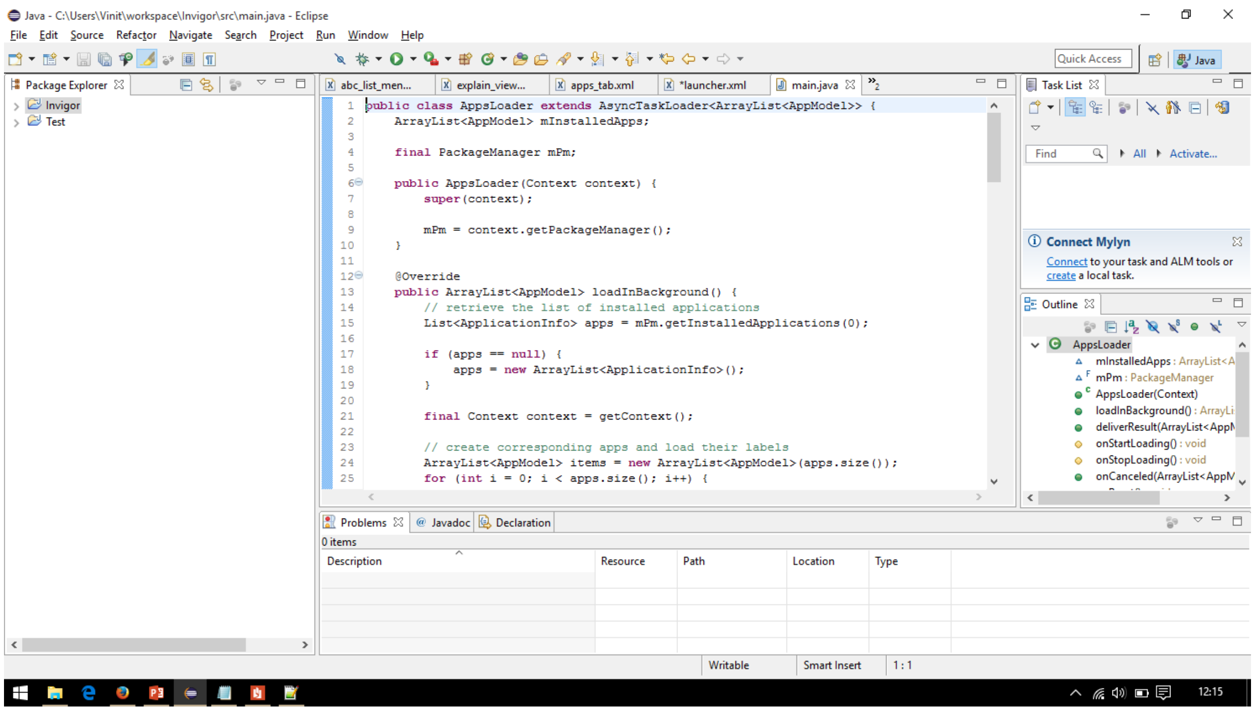
Java - C:\Users\Vinit\workspace\Invigor\res\menu\apps_tab.xml - Eclipse

```
<?xml version="1.0" encoding="utf-8"?>
<menu
  xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item android:id="@id/apps_sort_title" android:checked="true" android:title="@
        <item android:id="@id/apps_sort_install_date" android:title="@string/menu_apps
    </group>
</menu>
```



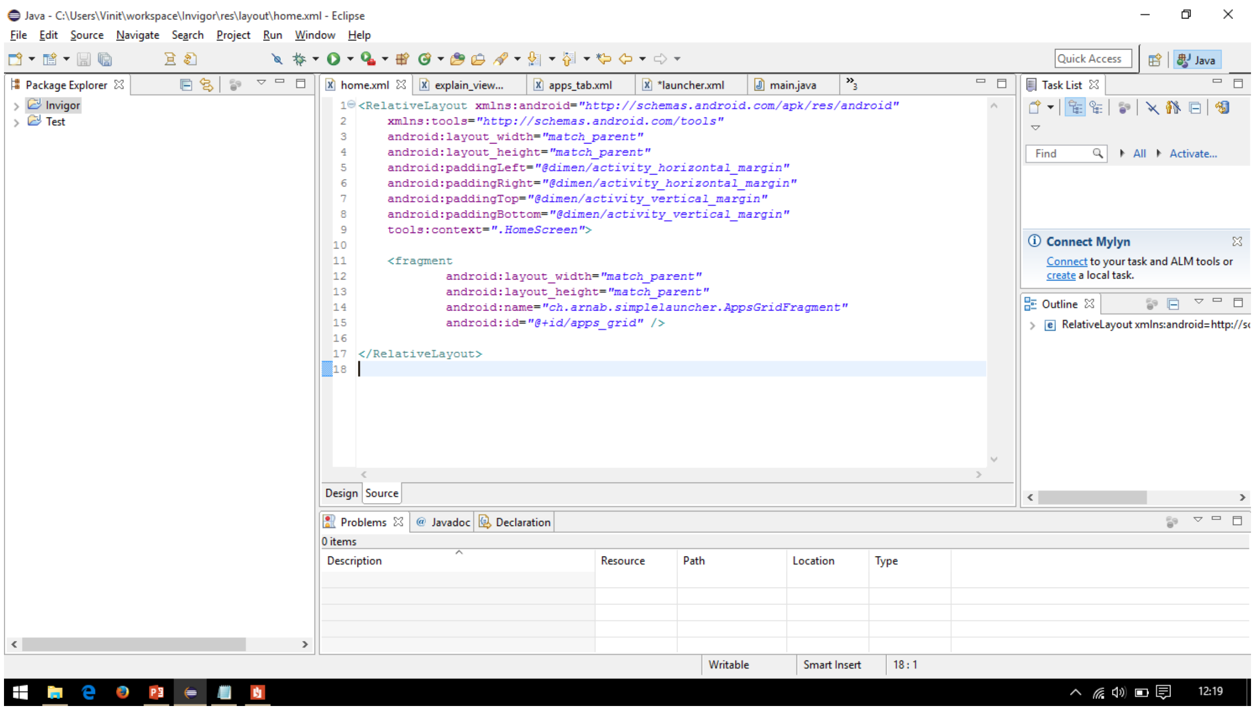
```
<?xml version="1.0" encoding="utf-8"?>
<invigor.base.DragLayer android:id="@id/drag_layer" android:background="@color/screen_
  xmlns:android="http://schemas.android.com/apk/res/android" xmlns:launcher="http://sc
    <invigor.android.widget.TransitionImageView android:id="@id/wallpaper_blur_transit
    <invigor.android.widget.LightTextView android:textSize="18.0sp" android:textColor=
    <include android:layout_gravity="top" android:id="@id/paged_view_indicator" androi
    <invigor.ase.InvigorWorkspace android:id="@id/workspace" android:layout_width="fil
    <include android:layout_width="fill_parent" android:layout_height="fill_parent" la
    <include android:layout_gravity="@integer/hotseat_gravity" android:id="@id/hotseat
    <include android:id="@id/apps_customize_pane" android:visibility="invisible" andro
    <include android:id="@id/qsb_bar" layout="@layout/qsb_bar" />
    <LinearLayout android:gravity="center_horizontal" android:orientation="vertical" a
        <ViewStub android:id="@id/mode" android:visibility="gone" android:layout="@lay
    </LinearLayout>
</invigor.base.DragLayer>
```

```
invigor.base.DragLayer android:id=@id/dra
invigor.android.widget.TransitionImage
invigor.android.widget.LightTextView an
include android:layout_gravity=top
invigor.ase.InvigorWorkspace android:id
include android:layout_width=fill_paren
include android:layout_gravity=@intege
include android:id=@id/apps_customiz
include android:id=@id/qsb_bar
LinearLayout android:gravity=center_ho
```

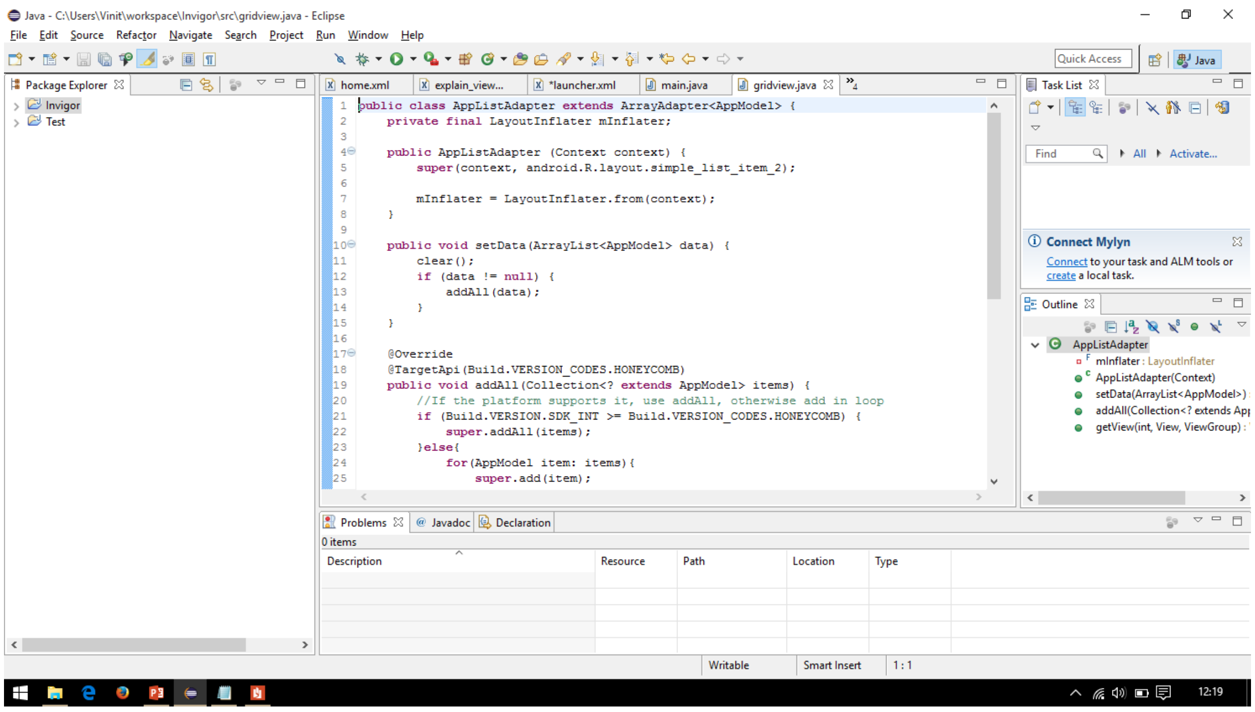


```java
public class AppsLoader extends AsyncTaskLoader<ArrayList<AppModel>> {
    ArrayList<AppModel> mInstalledApps;

    final PackageManager mPm;

    public AppsLoader(Context context) {
        super(context);

        mPm = context.getPackageManager();
    }

    @Override
    public ArrayList<AppModel> loadInBackground() {
        // retrieve the list of installed applications
        List<ApplicationInfo> apps = mPm.getInstalledApplications(0);

        if (apps == null) {
            apps = new ArrayList<ApplicationInfo>();
        }

        final Context context = getContext();

        // create corresponding apps and load their labels
        ArrayList<AppModel> items = new ArrayList<AppModel>(apps.size());
        for (int i = 0; i < apps.size(); i++) {
```

```
AppsLoader
  mInstalledApps : ArrayList<A
  mPm : PackageManager
  AppsLoader(Context)
  loadInBackground() : ArrayLi
  deliverResult(ArrayList<AppM
  onStartLoading() : void
  onStopLoading() : void
  onCanceled(ArrayList<AppM
```


Java - C:\Users\Vinit\workspace\Invigor\res\layout\home.xml - Eclipse

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".HomeScreen">

    <fragment
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:name="ch.arnab.simplelauncher.AppsGridFragment"
            android:id="@+id/apps_grid" />

</RelativeLayout>
```


Java - C:\Users\Vinit\workspace\Invigor\src\gridview.java - Eclipse

```
public class AppListAdapter extends ArrayAdapter<AppModel> {
    private final LayoutInflater mInflater;

    public AppListAdapter (Context context) {
        super(context, android.R.layout.simple_list_item_2);

        mInflater = LayoutInflater.from(context);
    }

    public void setData(ArrayList<AppModel> data) {
        clear();
        if (data != null) {
            addAll(data);
        }
    }

    @Override
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public void addAll(Collection<? extends AppModel> items) {
        //If the platform supports it, use addAll, otherwise add in loop
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            super.addAll(items);
        }else{
            for(AppModel item: items){
                super.add(item);
```

6. Basic Android Launcher Code Snippets

1. AndroidManifest.java

```
<activity
  android:name="ah.hathi.simplelauncher.HomeActivity"
  android:label="Simple Launcher Home"
  android:theme="@android:style/Theme.Wallpaper.NoTitleBar.Fullscreen"
  android:launchMode="singleTask"
  android:stateNotNeeded="true"
  >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
<activity
  android:name="ah.hathi.simplelauncher.AppsListActivity"
  android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
  >
</activity>
```

2. HomeActivity.java

```
package ah.hathi.simplelauncher;


import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;


public class HomeActivity extends Activity {


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_home);

    }


    public void showApps(View v){

        Intent i = new Intent(this, AppsListActivity.class);

        startActivity(i);

    }

}
```

3. AppsListActivity.java

```
package ah.hathi.simplelauncher;
```

```
import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;


public class AppsListActivity extends Activity {


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_apps_list);

    }


}


private PackageManager manager;

private List<AppDetail> apps;

private void loadApps(){

    manager = getPackageManager();

    apps = new ArrayList<AppDetail>();


    Intent i = new Intent(Intent.ACTION_MAIN, null);

    i.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> availableActivities = manager.queryIntentActivities(i, 0);
    for(ResolveInfo ri:availableActivities){
        AppDetail app = new AppDetail();
        app.label = ri.loadLabel(manager);
        app.name = ri.activityInfo.packageName;
        app.icon = ri.activityInfo.loadIcon(manager);
        apps.add(app);
    }
}
private ListView list;
private void loadListView(){
    list = (ListView)findViewById(R.id.apps_list);

    ArrayAdapter<AppDetail> adapter = new ArrayAdapter<AppDetail>(this,
            R.layout.list_item,
            apps) {
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            if(convertView == null){
                convertView = getLayoutInflater().inflate(R.layout.list_item, null);
            }

            ImageView appIcon = (ImageView)convertView.findViewById(R.id.item_app_icon);
            appIcon.setImageDrawable(apps.get(position).icon);
            TextView appLabel = (TextView)convertView.findViewById(R.id.item_app_label);
            appLabel.setText(apps.get(position).label);

            TextView appName = (TextView)convertView.findViewById(R.id.item_app_name);
            appName.setText(apps.get(position).name);

            return convertView;
        }
    };

    list.setAdapter(adapter);
}
private void addClickListener(){
    list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> av, View v, int pos,
                long id) {
            Intent i = manager.getLaunchIntentForPackage(apps.get(pos).name.toString());
            AppsListActivity.this.startActivity(i);
        }
    });
}
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_apps_list);

    loadApps();
    loadListView();
    addClickListener();
}
```

4. AppDetail.java

```
package ah.hathi.simplelauncher;

import android.graphics.drawable.Drawable;

public class AppDetail {
        CharSequence label;
        CharSequence name;
        Drawable icon;
}
```

5. activity_apps_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/apps_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >
    </ListView>

</LinearLayout>
```

6. activity_home.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeActivity" >

    <Button
        android:id="@+id/apps_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:text="Show Apps"
        android:onClick="showApps"
        />

</RelativeLayout>
```

7. list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    >

    <ImageView
        android:id="@+id/item_app_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        />

    <TextView
        android:id="@+id/item_app_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/item_app_icon"

        android:paddingLeft="10dp"

        />


    <TextView

        android:id="@+id/item_app_name"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_below="@+id/item_app_label"

        android:layout_toRightOf="@+id/item_app_icon"

        android:paddingLeft="10dp"

        />


</RelativeLayout>
```

8. list_menu_item_layout.java

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v7.internal.view.menu.ListMenuItemView  android:layout_width="fill_parent"
android:layout_height="?listPreferredItemHeightSmall"

  xmlns:android="http://schemas.android.com/apk/res/android">

    <RelativeLayout                    android:layout_gravity="center_vertical"
android:duplicateParentState="true"                  android:layout_width="0.0dip"
android:layout_height="wrap_content"
android:layout_marginLeft="?listPreferredItemPaddingLeft"
android:layout_marginRight="?listPreferredItemPaddingRight" android:layout_weight="1.0">

        <TextView              android:textAppearance="?textAppearanceListItemSmall"
android:ellipsize="marquee"     android:id="@id/title"     android:fadingEdge="horizontal"
android:duplicateParentState="true"                  android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content" android:singleLine="true" android:layout_alignParentLeft="true" android:layout_alignParentTop="true" />

        <TextView android:textAppearance="?android:textAppearanceSmall" android:id="@id/shortcut" android:duplicateParentState="true" android:layout_width="wrap_content" android:layout_height="wrap_content" android:singleLine="true" android:layout_below="@id/title" android:layout_alignParentLeft="true" />

    </RelativeLayout>

</android.support.v7.internal.view.menu.ListMenuItemView>
```

9. AppsGridFragment.java [Grid View]

```
public class AppsGridFragment extends GridFragment implements LoaderManager.LoaderCallbacks<ArrayList<AppModel>> {


    AppListAdapter mAdapter;


    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);


        setEmptyText("No Applications");


        mAdapter = new AppListAdapter(getActivity());
        setGridAdapter(mAdapter);


        // till the data is loaded display a spinner
        setGridShown(false);

        // create the loader to load the apps list in background
        getLoaderManager().initLoader(0, null, this);
    }

    @Override
    public Loader<ArrayList<AppModel>> onCreateLoader(int id, Bundle bundle) {
        return new AppsLoader(getActivity());
    }

    @Override
    public void onLoadFinished(Loader<ArrayList<AppModel>> loader, ArrayList<AppModel>
apps) {
        mAdapter.setData(apps);

        if (isResumed()) {
            setGridShown(true);
        } else {
            setGridShownNoAnimation(true);
        }
    }

    @Override
    public void onLoaderReset(Loader<ArrayList<AppModel>> loader) {
        mAdapter.setData(null);
    }


    @Override
    public void onGridItemClick(GridView g, View v, int position, long id) {
        AppModel app = (AppModel) getGridAdapter().getItem(position);
        if (app != null) {
            Intent                                          intent                                          =
getActivity().getPackageManager().getLaunchIntentForPackage(app.getApplicationPackageNa
me());


            if (intent != null) {
                startActivity(intent);
            }
        }
    }
}
```

10. homescreen.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".HomeScreen">
    <fragment
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:name="ch.arnab.simplelauncher.AppsGridFragment"
            android:id="@+id/apps_grid" />

</RelativeLayout>
```

11. prediction_bar.xml

```
<?xml version="1.0" encoding="utf-8"?>

<me.everything.android.fragments.PredictionBar                    android:layout_gravity="center"
android:layout_width="wrap_content"                               android:layout_height="fill_parent"
launcher:cellWidth="@dimen/hotseat_cell_width"
launcher:cellHeight="@dimen/hotseat_cell_height"                  launcher:widthGap="0.0dip"
launcher:heightGap="0.0px" launcher:maxGap="0.0dip"

  xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:launcher="http://schemas.android.com/apk/res-auto" />
```

12. search_bar.xml

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout android:layout_gravity="bottom|center" android:id="@id/qsb_search_bar"
android:layout_width="fill_parent" android:layout_height="@dimen/search_bar_height"
style="@style/SearchDropTargetBar"

  xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:launcher="http://schemas.android.com/apk/res-auto">

    <me.everything.base.HolographicLinearLayout android:layout_gravity="center_vertical"
android:id="@id/search_button_container" android:paddingLeft="8.0dip"
android:focusable="true" android:clickable="true" android:layout_width="fill_parent"
android:layout_height="fill_parent" android:layout_toLeftOf="@id/voice_button_container"
android:layout_alignParentLeft="true" android:layout_alignParentTop="true"
android:onClick="onClickSearchButton"
android:contentDescription="@string/accessibility_search_button_content_description"
launcher:sourceImageViewId="@id/search_button" style="@style/SearchButton">
```

```
        <ImageView android:id="@id/search_button" android:layout_width="wrap_content"
android:layout_height="fill_parent" android:src="@drawable/ic_home_search_normal_holo"
android:adjustViewBounds="true" />

    </me.everything.base.HolographicLinearLayout>

    <me.everything.base.HolographicLinearLayout android:gravity="end"
android:layout_gravity="center_vertical" android:id="@id/voice_button_container"
android:paddingRight="8.0dip" android:focusable="true" android:clickable="true"
android:layout_width="@dimen/search_bar_height" android:layout_height="fill_parent"
android:layout_alignParentTop="true" android:layout_alignParentRight="true"
android:onClick="onClickVoiceButton"
android:contentDescription="@string/accessibility_voice_search_button"
launcher:sourceImageViewId="@id/voice_button">

        <ImageView android:id="@id/voice_button" android:layout_width="wrap_content"
android:layout_height="fill_parent" android:src="@drawable/ic_home_voice_search_holo"
android:adjustViewBounds="true" />

    </me.everything.base.HolographicLinearLayout>

</RelativeLayout>
```

7. RESULTS AND DISCUSSIONS

After doing an exhaustive and comprehensive research on the existing system we have confronted all the issues plaguing it. Hence, our proposed system tackles all these issues and provides the measures to enhance and eradicate the flaws that exist. We have also conducted a local survey for determining application access times for different kind of users. We have also mentioned different self-learning algorithms that we are going to implement in Invigor.

Our exhaustive research has aided us in partially creating a meticulously advanced system.

We conducted a study on access times of five applications on a mobile device. An android mobile with five common applications installed on it was given to 5 students of the class and were asked to start these specific applications from different categories. We recorded the time it took to click the correct icon and invoke the application.

The applications the users were asked to find and invoke are Zomato, Google Maps, Whatsapp, Snapchat and Angry Birds in random order. We measured the time taken to access each of the applications with a stopwatch and the statistics thus obtained are mentioned in the table given below.

App Name/Users	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Whatsapp	**2.0**	**3.4**	**1.5**	**6.6**	**2.5**	**5.9**
Zomato	**3.9**	**3.6**	**5.3**	**4.2**	**2.2**	**3.84**
Snapchat	**2.8**	**4.4**	**6.1**	**1.8**	**5.2**	**4.06**
Angry Birds	**5.8**	**6.1**	**2.9**	**3.1**	**4.7**	**4.52**
Google Maps	**1.9**	**2.8**	**3.0**	**4.5**	**3.9**	**3.22**

Table 7.1: User Testing

8. CONCLUSION AND FUTURE WORK

Improving the previously created launcher will not happen overnight. It will take persistent, sedulous efforts. The future work is highly dependent on the below factors:

i. Understanding the user for whom we create the phone. This requires a comprehensive and elaborate collection of his traits, personality etc. Trial and error is one way forward.

ii. We need to minimize the resource usage. This involves optimizing battery usage.

iii. The confidence interval is one of the most critical areas that needs meticulous attention and detail. Maximum value and minimum range conditions need to be attained.

Our proposed system to develop an adaptive user interface is still at a developmental stage and we are trying to develop a prototype for the same. Once it is completed, in future we hope to implement this system on mobile devices and extend the system for other kinds of adaptations rather than limiting it to application icons only.

Addition of this feature of customizing the user interface or applications based on the user profile would reduce clutter on the screen and increase the user satisfaction with using the device and save the user time while accessing a specific application, leading to higher productivity.

9. REFERENCES

[1] Rahul Jain, Jay Bose, Tasleem Arif, “Contextual Adaptive User Interfaces for android devices”, WMG Group, Samsung R&D Department.

[2] William Burns, Liming Chen, Chris Nugent, Mark Donnelly, Kerry Louise Skillen, Ivar Solheim, “Mining usage data for adaptive personalization of smartphone based help-on-demand services”, May 2013

[3] Jacob Eisenstein, Jean Vanderdonckt, and Angel Puerta, “Adapting to Mobile Contexts with User-Interface Modeling”

[4] developer.android.com

[5] http://code.tutsplus.com/tutorials/build-a-custom-launcher-on-android--cms-2135

[6] http://schemas.android.com/apk/res/android

[7] http://en.wikipedia.org/wiki/Context sensitive_user_interface

[8] http://developer.android.com/training/index.html

[9] Nishkam Ravi, James Scott**, Lu Han* and Liviu Iftode*Intel Corporation, Santa Clara, USA Department of Computer Science, Rutgers University, USA,**Microsoft Research, Cambridge, UK, Context-aware Battery Management for Mobile Phones

[10] Melanie Hartmann, “Challenges in Developing User-Adaptive Intelligent User Interfaces”, Telecooperation Group

[11] http://thenextweb.com/insider/2012/05/16/nielsen-us-smartphones-havean-average-of-41-apps-installed-up-from-32-last-year