**­­**

**Department of Computer Engineering**

**Faculty of Technology, Dharmsinh Desai University**

**College Road,**

**Nadiad-387001**

**B. Tech CE, Semester VI**

**A**

**Project Report**

**On**

**Movie Recommender System**

*Prepared At*

Department Of Computer Engineering

Faculty of Technology, Dharmsinh Desai University

College Road, Nadiad-387001

*Prepared By*

Shail Shah (CE-123)

Hitarthi Tewani (CE-128)

*Guided By*

Prof. Jatayu H. Baxi

Dept. of Computer Engineering,

Faculty of Technology,

Dharmsinh Desai University



**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**

**CERTIFICATE**

This is to certify that the Project Work carried out in the subject of System Design Practice on the topic of “[Movie Recommender System](http://github.com/shailshah9/RecSysJava)” is bonafide work of Mr. Shail P. Shah (CE-123) and Ms. Hitarthi P. Tewani (CE-128) of B.Tech. Semester VI in the branch of Computer Engineering during the Academic year 2014-2015. To the best of my knowledge and belief, the matter presented by them is original in nature and have not been copied.

Prof. Jatayu H. Baxi

(Project Guide)

Department of Computer Engineering,

Faculty of Technology,

Dharmsinh Desai University, Nadiad

Date:

Prof. C.K. Bhensdadia

Head,

Department of Computer Engineering,

Faculty of Technology,

Dharmsinh Desai University, Nadiad

Date:

**Acknowledgement**

Every project big or small is successful largely due to the effort of a number of people who have always given their valuable advice or lent a helping hand. We sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success.

We, Shail Shah and Hitarthi Tewani, the students of DHARMSINH DESAI UNIVERSITY, are grateful to “The Department of Computer Engineering” for the confidence bestowed in us entrusting us with the project entitled “Movie Recommender System”.

At this juncture we feel deeply honoured in expressing my sincere thanks to our guide and mentor, Prof. Jatayu H. Baxi for providing critical advice, guidance, innovative ideas and valuable insights without which this project would not have been possible.

We sincerely express our gratitude to the Head of Computer Engineering Department, Prof. C .K. Bhensdadia for providing suitable resources, environment and support which lead to the completion of this project.

With Sincere Regards,

Shail Shah

Hitarthi Tewani

**Contents**

1 Abstract 1

2 Intoduction 2

2.1 Project Details 2

2.2 Technology Overview 2

2.3 Iterative Waterfall Model 2

3 Software Requirement Specifications 4

3.1 Usage Scenario 4

3.2 Functional Requirements 5

3.2.1 Main Menu Selection 5

3.4 Non-Functional Requirements 6

3.4.1 Device and Software Perspective 6

3.4.2 Reliability 6

3.4.2 Maintainability 6

3.4.3 Portability 6

4 System Design 7

4.1 Use Case Diagram 7

4.2 Class Diagram 8

4.3 Sequence Diagram (Ask Password) 9

4.3 Sequence Diagram (Select Blacklist) 10

4.4 Activity Diagram 11

4.5 State Chart Diagram 12

5 Implementation 13

5.1 Module Specifications 13

6 Testing 14

6.1 Testing Plan 14

6.2 Testing Strategy 15

6.3 Testing Methods 16

7 Screenshots 17

8 Conclusion 26

9 Future Extension 26

10 Bibliography 26

1. **Abstract**

As the World Wide Web continues to grow at an exponential rate, the size and complexity of many web sites grow along with it. For the users of these web sites it becomes increasingly difficult and time consuming to find the information they are looking for. User interfaces could help users find the information that is in accordance with their interests by personalizing a web site.

Some web sites present users with personalized information by letting them choose from a set of predefined topics of interest. Users however do not always know what they are interested in beforehand and their interests may change overtime which would require them to change their selection frequently. Recommender systems provide personalized information by learning the user’s interests from traces of interaction with that user.

1. **Introduction**
   1. **Project Details:**

* Application Locker is an Android based application developed in Java Programming Language using Eclipse Integrated Development Environment.

Objectives of the application:

* To enable the user to select the apps to lock
* To provide pin based password for locking the apps
* To provide a running service for handling the protected apps
* To enable the user to select the relock policy

A salient feature of this app which distinguishes it from other ‘App Lockers’ is the relock policy which allows the user to set a time period for which not to lock the protected apps again in case those apps are involved in some important and long task which requires moving back and forth between various apps.

* 1. **Technology Overview:**
* Platform and Tools used:
* Eclipse Integrated Programming Language (IDE)
* Android Software Development Kit (SDK) which includes Android SDK Tools
* Android SDK Platform for API 9 and above
* Android Virtual Device Manager for creating emulators to test run the project
  1. **Software Life Cycle Model Implemented**

We have used the waterfall model for development. In the Iterative waterfall model, if any difficulty or problem is encountered in any phase then it may be required to go back to the previous phase and performing the required modifications and then proceed sequentially. This backtracking allows modifying any corrections or modifications required in the previous phase.

As illustrated in fig, this model divides the software development life-cycle into the phases mentioned below:

1. Feasibility study

2. Requirements analysis and specification

3. Design

4. Coding and unit testing

5. Integration and system Testing

6. Maintenance



* + 1. **Advantages of using Iterative waterfall model:**

You are provided the changes to see the potential outcomes of every stage and make changes to area of concern if necessary. This is one of the reasons that make iterative model useful.

Iterative development is more adjustable to changes as it considers each stage like a vital portion of the development.

The time spent on each successive interval may be less and depends on how the last stage went and what knowledge was gained through adding new functionalities in the development part of all iterations.

* + 1. **Disadvantages of using Iterative waterfall model:**

When using the iterative model, people working on the project can get stuck in a loop. Informal request for improvement after each phase may lead to confusion and may also create scope creep, since user feedback following each phase may lead to increased demands. As users see the system develop, they may realize potential of other system capabilities which would enhance their work, this can be an advantage as much as in can be a disadvantage.

The first phase of the project was feasibility and requirement study and understanding the technological requirement of the application. The functionalities to be achieved were documented using various system level diagrams. The development was initialized after analyzing the features supported by various technologies.

1. **Software Requirement Specifications**
   1. **Usage scenario**

* **User profiles:**

Users of the system will be anyone who has a mobile device operating the Android OS with access to the Android Marketplace. The application will be accessible to everyone by these means and require only navigation of the menus through the touch-screen interface.

* 1. **Functional Requirements**

**R1 Account Management**

**R1.1 Add new account**

Input: User details

Output: Confirmation message

**R1.2 Update account**

Input: User id and new details

Output: Confirmation message

**R1.3 Delete account**

Input: User id

Output: Confirmation message

**R2 Search Movie**

**R2.1 Get movie by KLDivergence algorithm**

Input: Movie

Output: Similar movies

**R2.2 Get movie by other filters**

Input: Movie

Output: Similar movies

**R3 Login & Authentication**

**R3.1 Login**

Input: Username & password

Output: Confirmation message

**R3.2 Authenticate**

Input: Username & password

Output: Confirmation message

**R4 Manage Database**

**R4.1 Add details in database**

Input: User id & movie details

Output: Confirmation message

**R4.2 Update details in database**

Input: User id & movie details

Output: Confirmation message

**R4.3 Delete details in database**

Input: User id & movie id

Output: Confirmation message

**R5 Manage Past Search**

**R5.1 Manage watched/liked movies**

**R5.1.1 Add movies**

Input: Movies

Output: Confirmation message

**R5.1.2 Remove Movies**

Input: Movies

Output: Confirmation message

**R5.2 Add recommended movies**

Input: Movies

Output: Confirmation message

* 1. **Non-Functional Requirements** 
     1. **Device and Software Perspective**

The project is a simple Mobile application running on any Android device having following minimum specifications:-

**Software Requirements:**

Android SDK 9 (version 2.3) or above

**Hardware Requirements:**

1. RAM, 200 MB Device Storage.
   * 1. **Reliability**
   * System shall be available for maximum time.
   * System shall be robust enough to have a high degree of fault tolerance. The system should not crash in case of high CPU usage. 
     1. **Maintainability**

* Since the system has a modular design, it is easy to develop and add new modules for any new requirement in future version.
  + 1. **Portability**
* This software application is packaged into ‘.apk’ file, which is a standard format that runs on any device running Android OS. So any new user who wants the application has to get a single file.

**4 System Design**

* 1. **Use Case Diagram:**



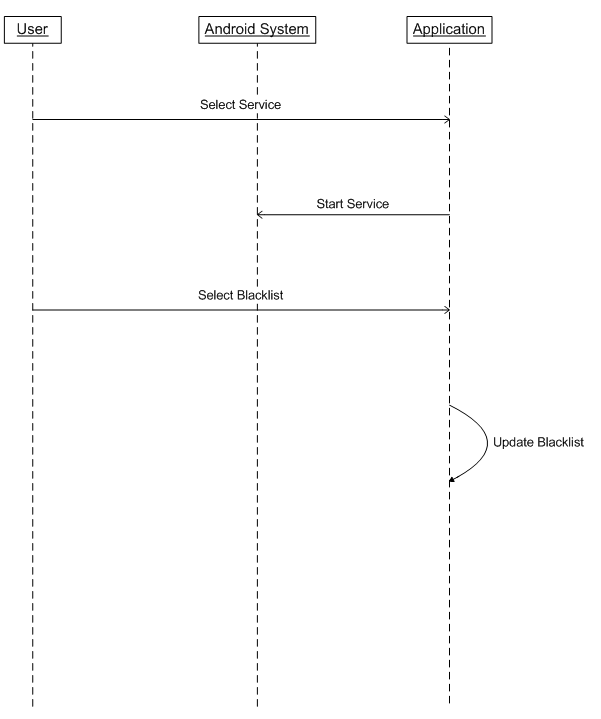
* 1. **Class Diagram**:



* 1. **Sequence Diagram (Ask Password):**



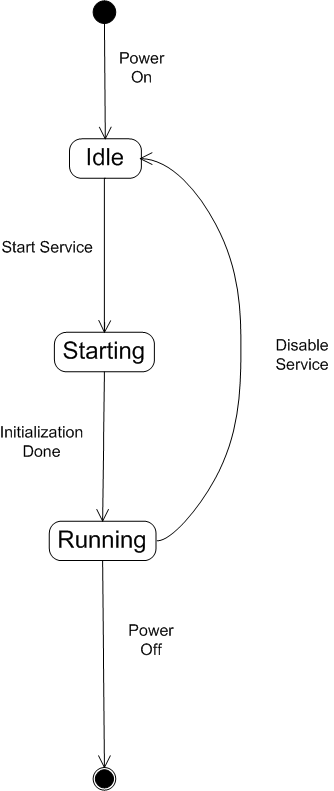
* 1. **Sequence Diagram (Select Blacklist):**

****

**4.4 Activity Diagram:**



**4.5 State Diagram**:



**Working**





1. **Implementation**
   1. **Module Specification**

This project consists of one module – the user application module:

* **User Module**: This module contains subroutines for functionality of the application. Modules include subroutines which implement actions as well as populate the User Interface of the application.

**Sub-Routines Details:**

Users can access the whole application and according to user’s call the system will function. The module contains subroutines like:

* editPassword(): To change the lock password.
* selectBlacklist(): To select the apps to lock.
* relockPolicy(): To decide whether to require password every time the locked app is launched.

1. **Testing**

Testing is the process carried out on software to detect the differences between its behavior and the desired behavior as stipulated by the requirements specifications. Testing is advantageous in several ways. Firstly, the defects found help in the process of making the software reliable. Secondly, even if the defects found are not corrected, testing gives an idea as to how reliable the software is. Thirdly, over time, the record of defects found reveals the most common kinds of defects, which can be used for developing appropriate preventive measures such as training, proper design and reviewing.

* 1. **TESTING PLAN**

The testing sub-process includes the following activities in a phase dependent manner:

a) Create Test Plans.

b) Create Test Specifications.

c) Review Test Plans and Test Specifications.

d) Conduct tests according to the Test Specifications, and log the defects.

e) Fix defects, if any.

f) When defects are fixed continue from activity.

* 1. **TESTING STRATEGY**

The development process repeats this testing sub-process a number of times for the following phases.

a) Unit Testing.

b) Integration Testing

Unit Testing tests a unit of code (module or program) after coding of that unit is completed. Integration Testing tests whether the various programs that make up a system, interface with each other as desired, fit together and whether the interfaces between the programs are correct. System Testing ensures that the system meets its stated design specifications. Acceptance Testing is testing by the users to ascertain whether the system developed is a correct implementation of the Software Requirements Specification.

Testing is carried out in such a hierarchical manner to ensure that each component is correct and the assembly/combination of components is correct. Merely testing a whole system at the end would most likely throw up errors in components that would be very costly to trace and fix.

We have performed both Unit Testing and System Testing to detect and fix errors. A brief description of both is given below.

* Unit Testing

**Objective**

The objective of Unit Testing is to test a unit of code (program or set of programs) using the Unit Test Specifications, after coding is completed. Since the testing will depend on the completeness and correctness of test specifications, it is important to subject these to quality and verification reviews.

**Input:** Unit Test Specifications

Testing Process

* Checking for availability of Code Walk-thru reports which have documented the existence of and conformance to coding standards.
* Review of Unit Test Specifications

Verify the Unit Test Specifications conform to the program specifications.

Verify that all boundary and null data conditions are included.

* 1. **TESTING METHODS**

Black-box and White-box Testing:

In black-box testing a software item is viewed as a black box, without knowledge of its internal structure or behavior. Possible input conditions, based on the specifications (and possible sequences of input conditions), are presented as test cases.

In white-box testing knowledge of internal structure and logic is exploited. Test cases are presented such that possible paths of control flow through the software item are traced. Hence more defects than black-box testing are likely to be found.

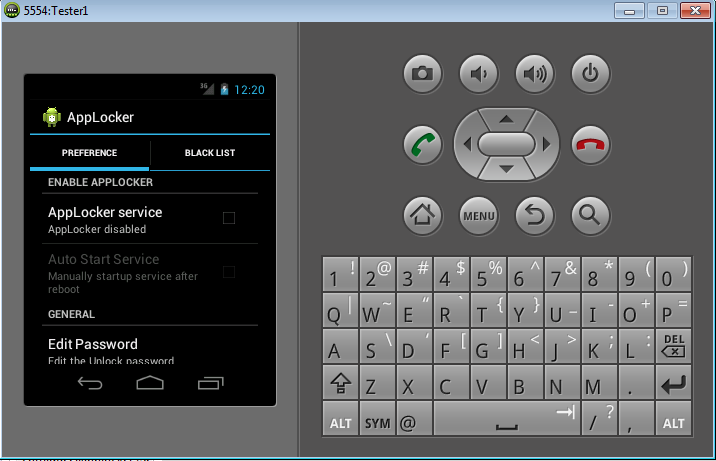
The disadvantages are that exhaustive path testing is infeasible and the logic might not conform to specification. Instrumentation techniques can be used to determine the structural system coverage in white box testing. For this purpose tools or compilers that can insert test probes into the programs can be used.

Code Coverage

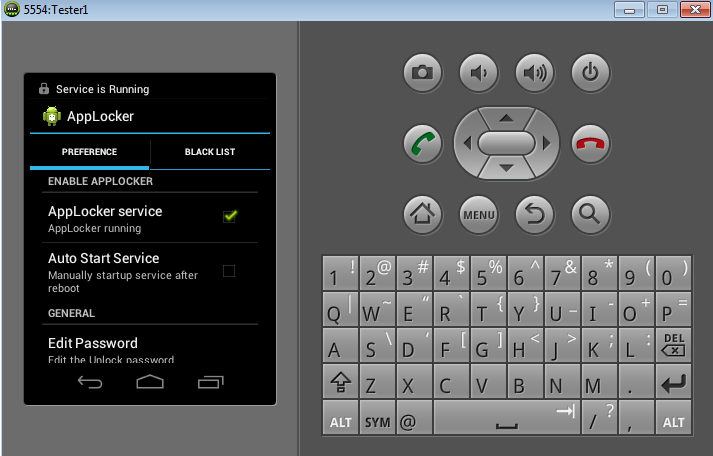
The way to make sure that you have got all the control flow covered is to cover all the paths in the program during the testing (via white-box testing). This implies that both branches are exercised for an ‘if’ statement, all branches are exercised for a case statement, the loop is taken once or multiple times as well as ignored for a while statement, and all components of complicated logical expressions are exercised. This is called Path Testing. Branch Testing reports whether entire Boolean expression tested in control structures evaluated to both true and false.

Additionally it includes coverage of switch statement cases, exception handlers and interrupts handlers. Path testing includes branch testing as it considers all possible combination of individual branch conditions. A simpler version is Statement Testing which determines if each statement in the program has been executed at least once. The coverage via Path Testing includes the coverage via Statement Testing.

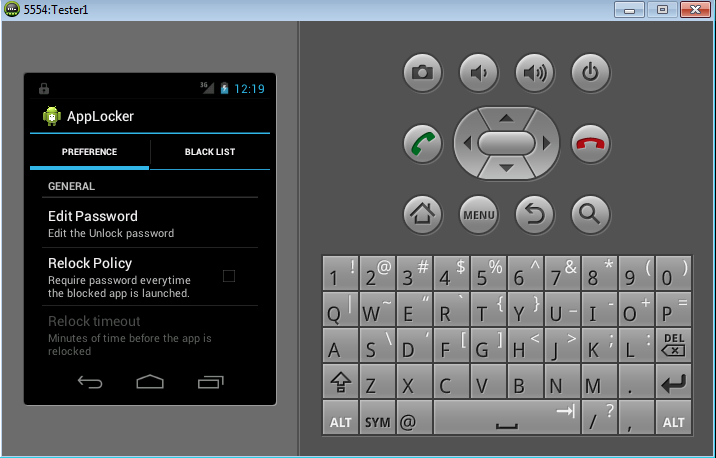
1. **Screenshots**



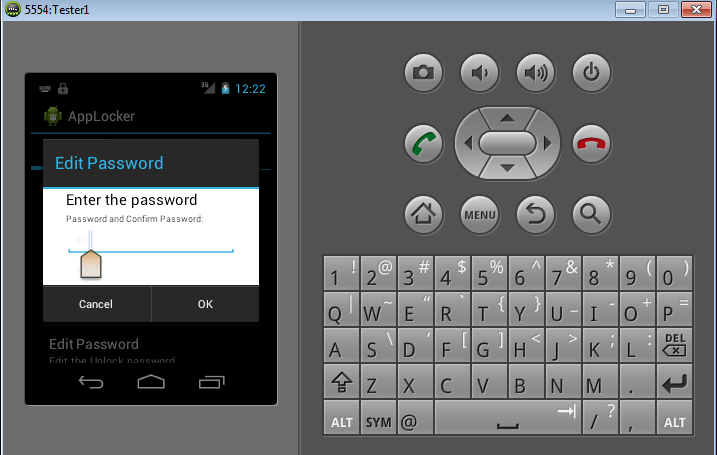
**Fig 7.1:** The application screen on starting up for the first time



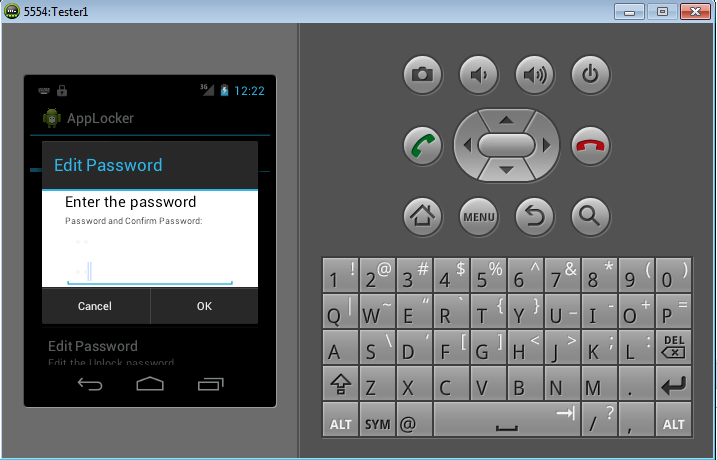
**Fig 7.2:** Starting the AppLocker Service



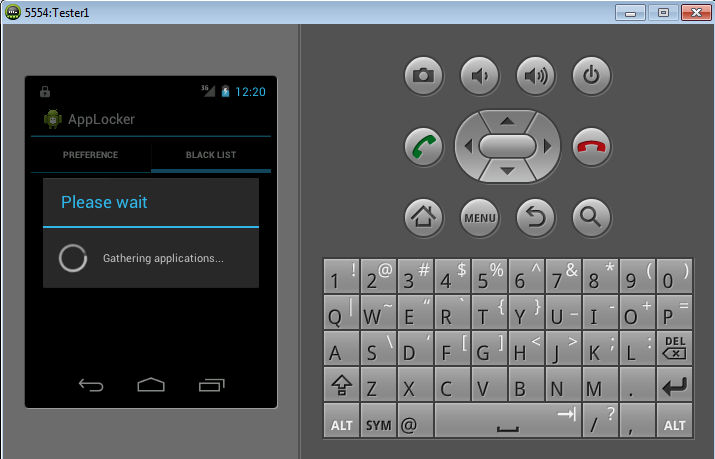
**Fig 7.3** Edit Password and Relock Policy facilities



**Fig 7.4** On clicking the Edit password button



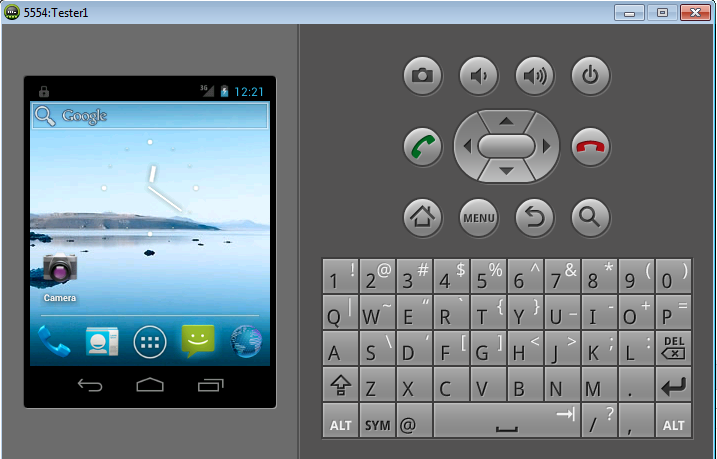
**Fig 7.5** Confirming the password



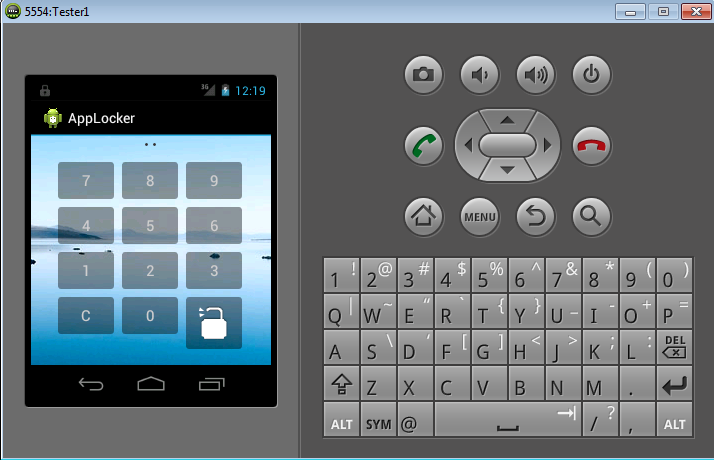
**Fig 7.6** Clicking on the BLACKLIST tab first gathers the applications for blacklisting



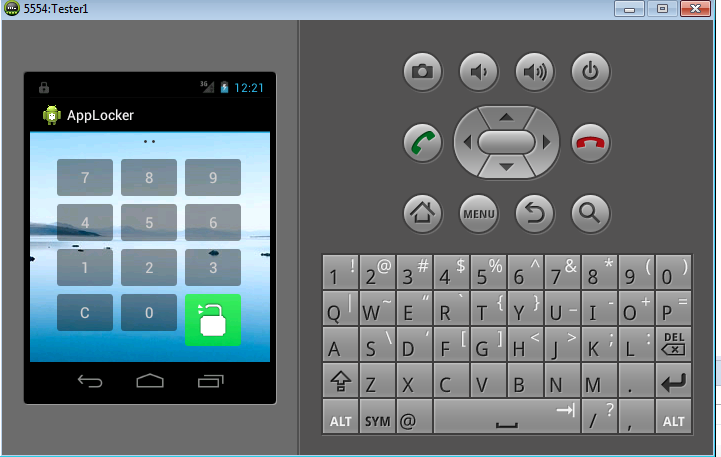
**Fig 7.7** The fetched apps with the options of blocking all – the Block All button, unblocking all – the Allow All button and blocking selectively by clicking on the individual app thumbnails.



**Fig 7.8** The lock sign on the top bar shows that AppLocker service is running



**Fig 7.9** On clicking a locked app, the password pin requiring screen will be displayed. The entered password can be seen as dots on the top.



**Fig 7.10** Once the password is entered, clicking on the unlock button will unlock the app and the intent will be changed to the locked app’s activity.

1. **Conclusion**

After developing our project in entirety we believe that the success of any project doesn’t depend only on better software development skills but also on the zeal to listen and help the target users. After having accomplished the AppLocker System, we believe two virtues “*Convenience*” is the crux of the application from the following perspective:

*Convenience*: Getting something that you lack and you exactly need to fill the void would be termed as convenience or a facility by a layman.

Using the AppLocker application, the user can enable/disable the locker service, edit the password, select the apps to protect and even decide on the relock policy. Then on opening any of the protected apps, the service takes control and loads the intent corresponding to the unlock pin acceptor.

1. **Future Extension**

Extensions planned to be implanted:

* Ringing up an alarm on entering incorrect password for three times
* Updating the relock policy to provide child lock mode which will require unlocking the children category apps only once

1. **Bibliography**

www.developer.android.com