Software Design Document

**Software Design Document**

**for**

**NoteIt**

**Prepared by CubeX**

**Software Systems, BITS Pilani 20-10-2015**

|  |  |  |
| --- | --- | --- |
|  |  | Software Design Document |
| **Table of contents** | |  |
| 1. | **Introduction** | 3 |
| 1.1 | Purpose | 3 |
| 1.2 | Scope | 3 |
| 1.3 | Overview | 3 |
| 2. | **System Overview** | 4 |
| 3. | **System Architecture** | 4 |
| 3.1 | Architectural Design | 4 |
| 3.2 | Decomposition Description | 8 |
| 3.3 | Design Rationale | 12 |
| 4. | **Data Design** | 14 |
| 4.1 | Data Description | 14 |
| 4.2 | Data Dictionary | 14 |
| 5. | **Component Design** | 15 |
| 6. | **Human Interface Design** | 16 |
| 6.1 | Overview of User Interface | 16 |
| 6.2 | Screen Images | 17 |
| 6.3 | Screen Objects and Actions | 20 |
| 7. | **Requirements Matrix** | 20 |

Software Design Document

**1. Introduction**

This section gives a description and overview of the design included in this design document. It will explain the software design in which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. It also shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it contains all the information required by a programmer to write code. The overall system architecture and data architecture is defined and also the detailed design models are developed for the defined architecture.

**1.1. Purpose**

The purpose of this document is to serve as a guide to developers and testers who are responsible for the engineering of ‘NoteIt’. It will illustrate the purpose and complete declaration for the development of system. This document contains a complete description of the design of the product.

It consists of various UML Diagrams - which, taken together form a complete design description of the software.

**1.2. Product Scope**

NoteIt is an android application for creating tasks and keeping track of activities the user is involved in. The user can sign in using google+ account, he can add and prioritize tasks, set reminders and track progress of tasks and also share those tasks with other users.

The idea for developing such an app came after realizing the hectic schedule in BITS for students. Students have to juggle between so many tasks including hobbies, organizing various events in the campus as well as academic deadlines. Although, our app is generic, and would be available to everyone on the google play store, but would mainly focus on students’ needs.

Its useful for keeping track of your day-to-day activities like appointments, hobby classes, meetings etc. It is also useful for maintaining a task list for personal or collaborative projects. For using this app the user must register through their google+ account. After logging in the user can add different types of tasks or update existing tasks.

**1.3. Overview**

This document contains various UML Diagrams that are created based on the requirements specification and also traceability of the requirements and the design is specified.

The following models which are useful in the process of development of the model:

* Data Flow Diagrams
* Activity Diagrams
* Detailed Class Diagram
* Collaboration Diagrams

Software Design Document

* + State Charts
  + Package Diagram

1. **System Overview**

This section gives an overview of the whole system. The system will be explained to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

Although the use of such task managing apps is very common to name a few, Keep which is provided by Google, Evernote provided by Microsoft and Trello is also one such popular app. This app will be more beneficial for students to prioritize tasks on deadline basis or otherwise. Our app provides three kinds of tasks namely General task, List task and Project task. In general, a task would caontain task name, description, due date and status. In General tasks the user can add subtasks, in List task the user can add subtasks in list format which include checkbox to indicate its completion and in Project task the user can add specific subtasks for projects and also the deadline. With this application, the user would also be able to set reminders and share specific tasks with their friends.

The various functions provided by the application to the user are as follows:

* + Authenticates the user from google+
  + Adding different types of tasks.
  + Modify tasks

D

* + Delete tasks
  + Prioritize tasks
  + Share task with friends
  + Set reminders for deadlines or meetings

1. **System Architecture**

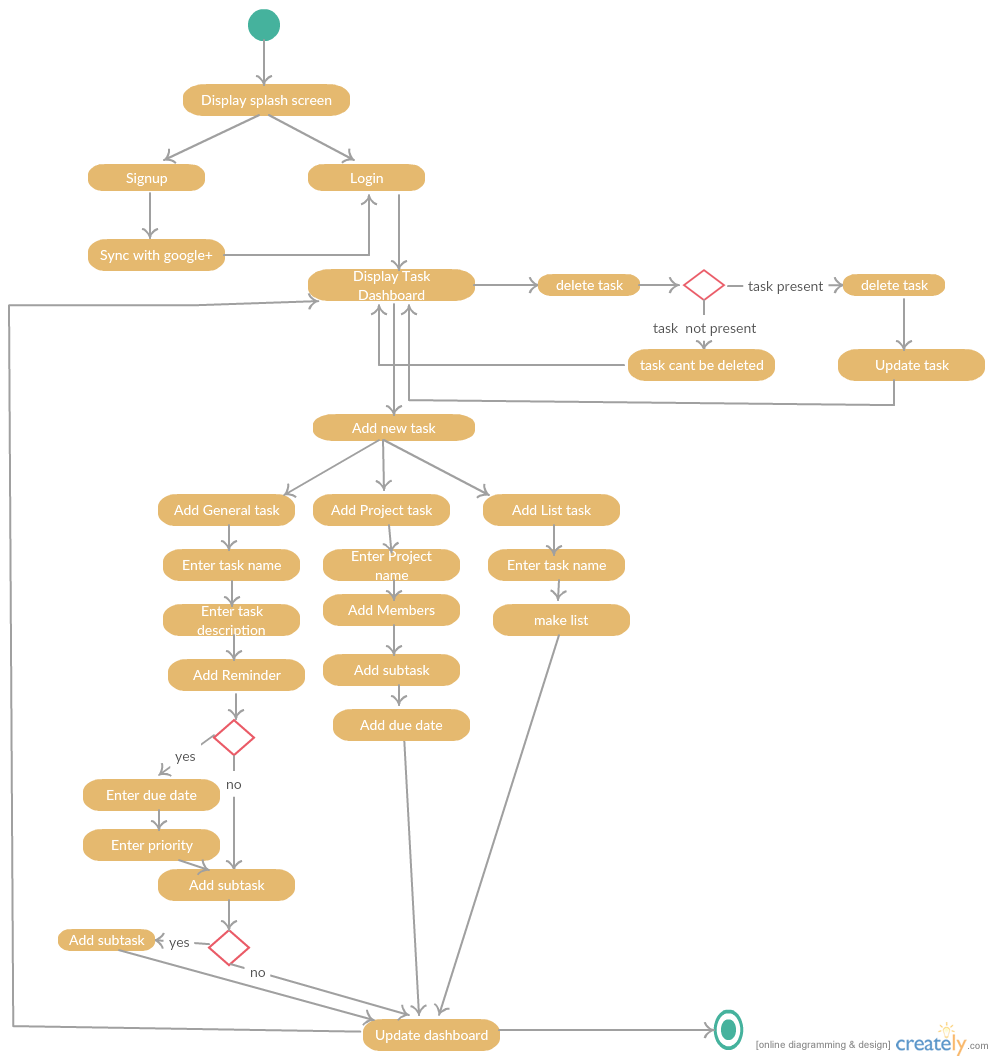
**3.1. Architectural Design**

This is a high level overview of how responsibilities of the system are partitioned and then assigned to subsystems. Each high level subsystem is identified and the roles or responsibilities are assigned to it. It also describes how these subsystems collaborate with each other in order to achieve the desired functionality. It is presented to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

The diagrams showing the major subsystems and data repositories and their interconnections are provided below:

Software Design Document

**Activity Diagram:**



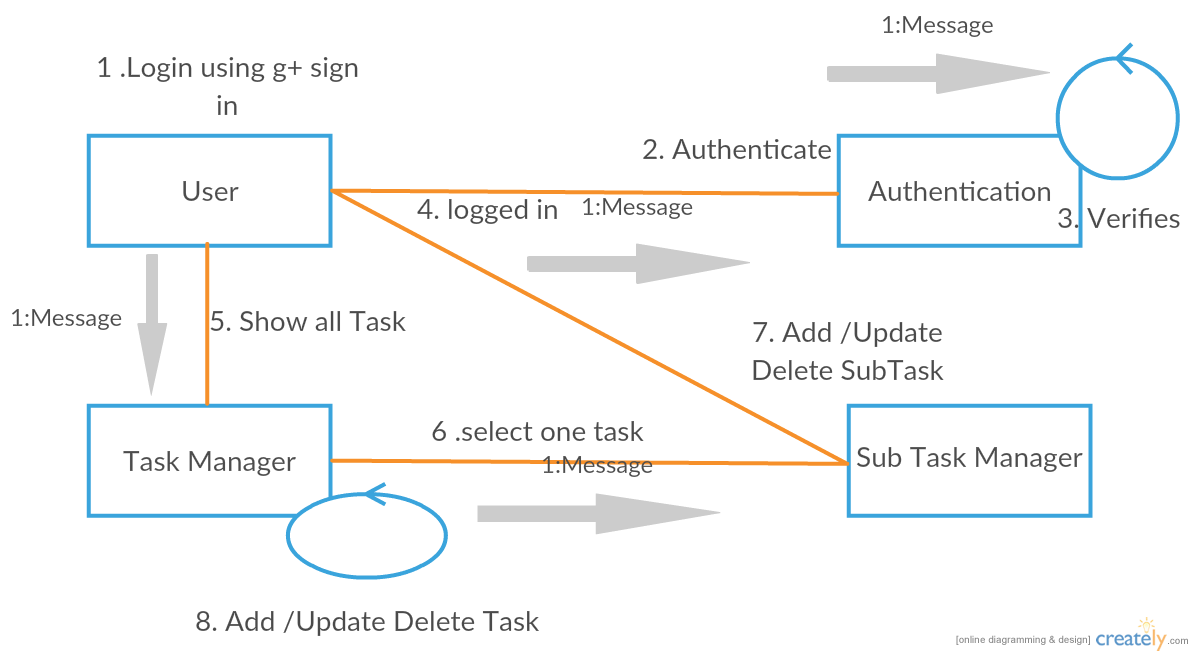
The above shown activity diagram covers most of the requirements specified and the various success scenarios along with additional paths for each activity have been described in this activity diagram.

**Collaboration Diagrams:**

The basic and the most important scenarios have been represented in the form of collaboration diagrams as follows. The success case has been considered in order to model these. Alert messages are generated and displayed to the user in case of any errors.

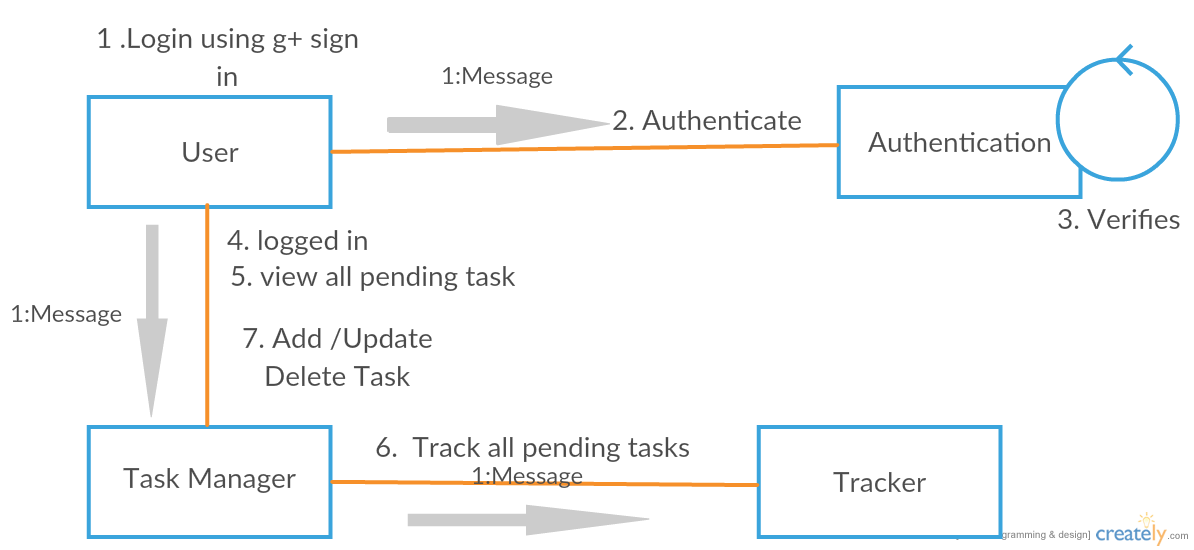
The major requirements can be observed in these scenarios considered like the need for authentication in order to view the seller details, ad-verification and also to requirement to be logged in to sell any product.

(1)



Software Design Document

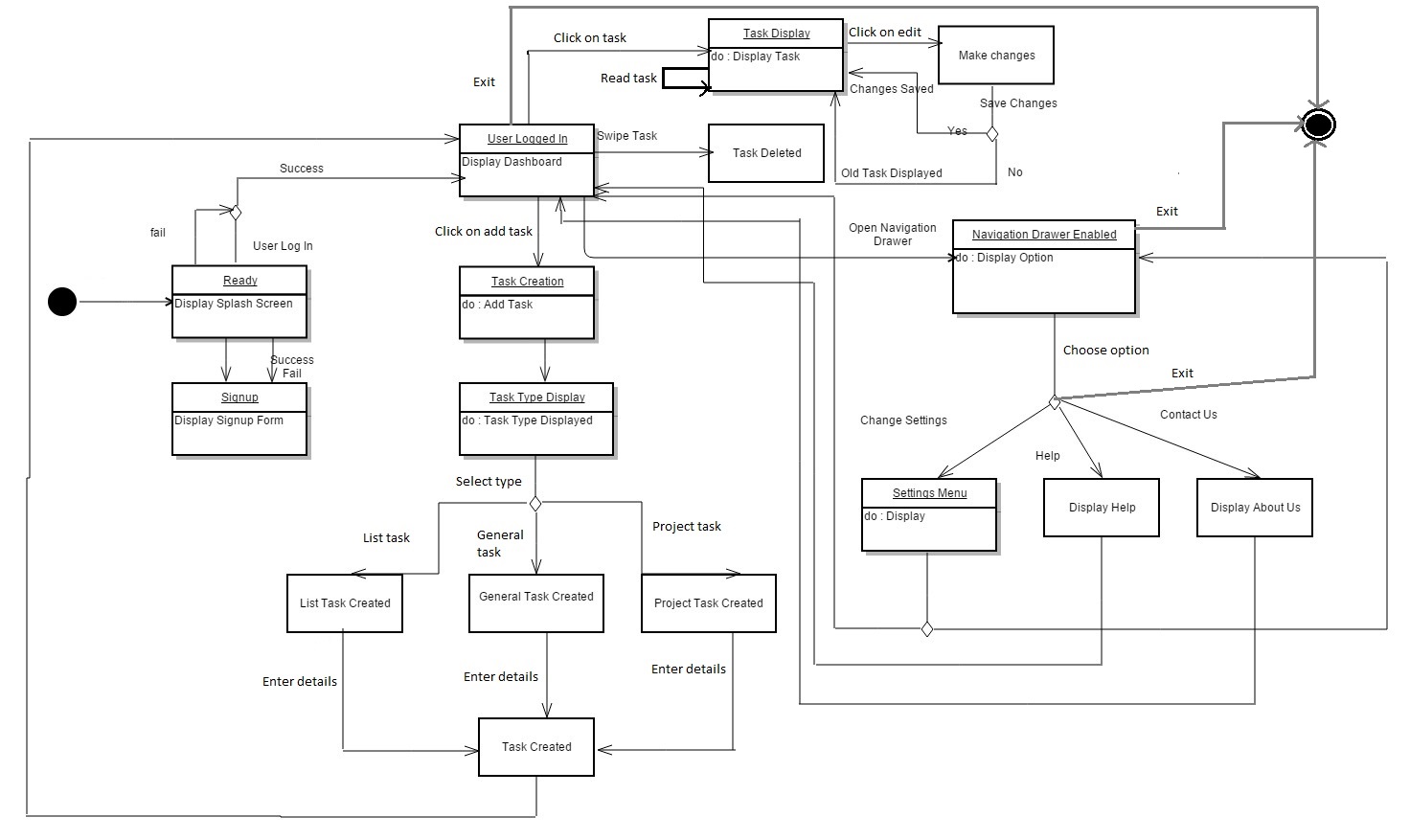
(2)



Software Design Document

**State Diagram:**

This diagram describes the different states of a component in a system. All the states are specific to a component or object of a system.

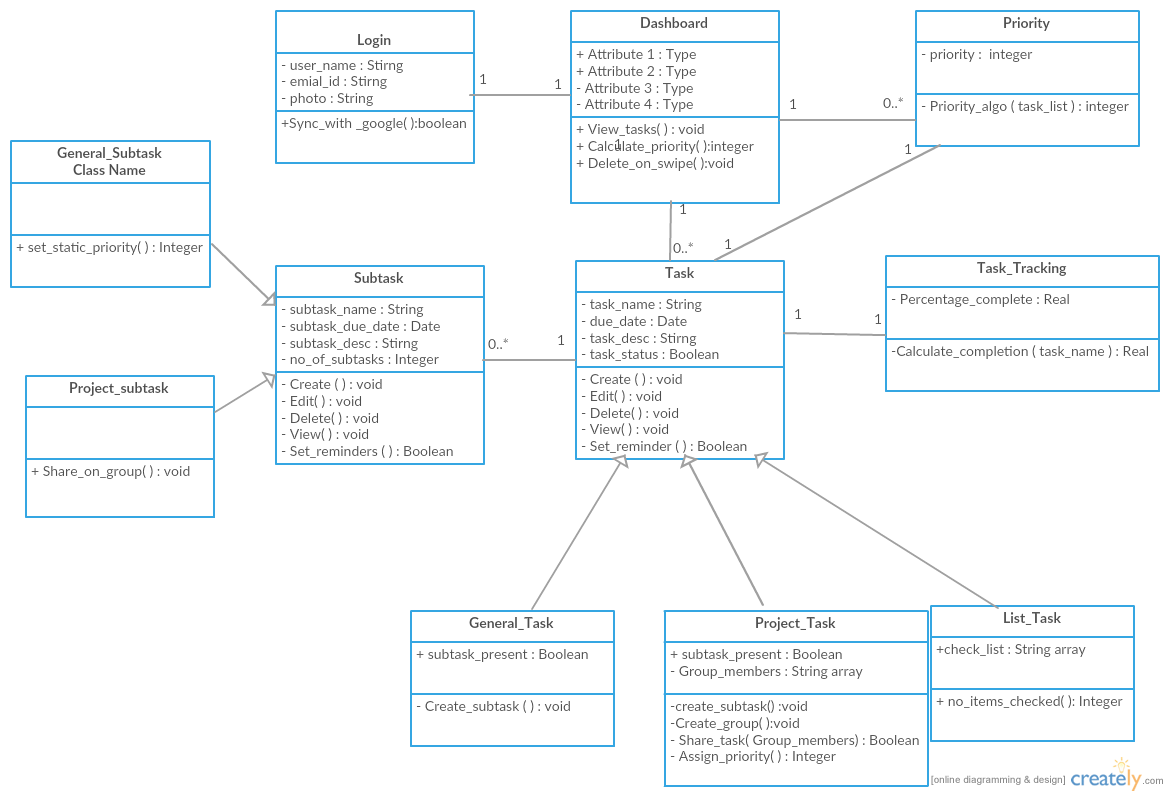


Software Design Document

**3.2. Decomposition Description**

This provides a decomposition of the subsystems in the architectural design. The decomposition can be better understood by the class diagram and data flow diagrams as shown below. The Spring MVC Architecture has been chosen for this product which enables the developers to separate the front end and the back-end modules distinctly.

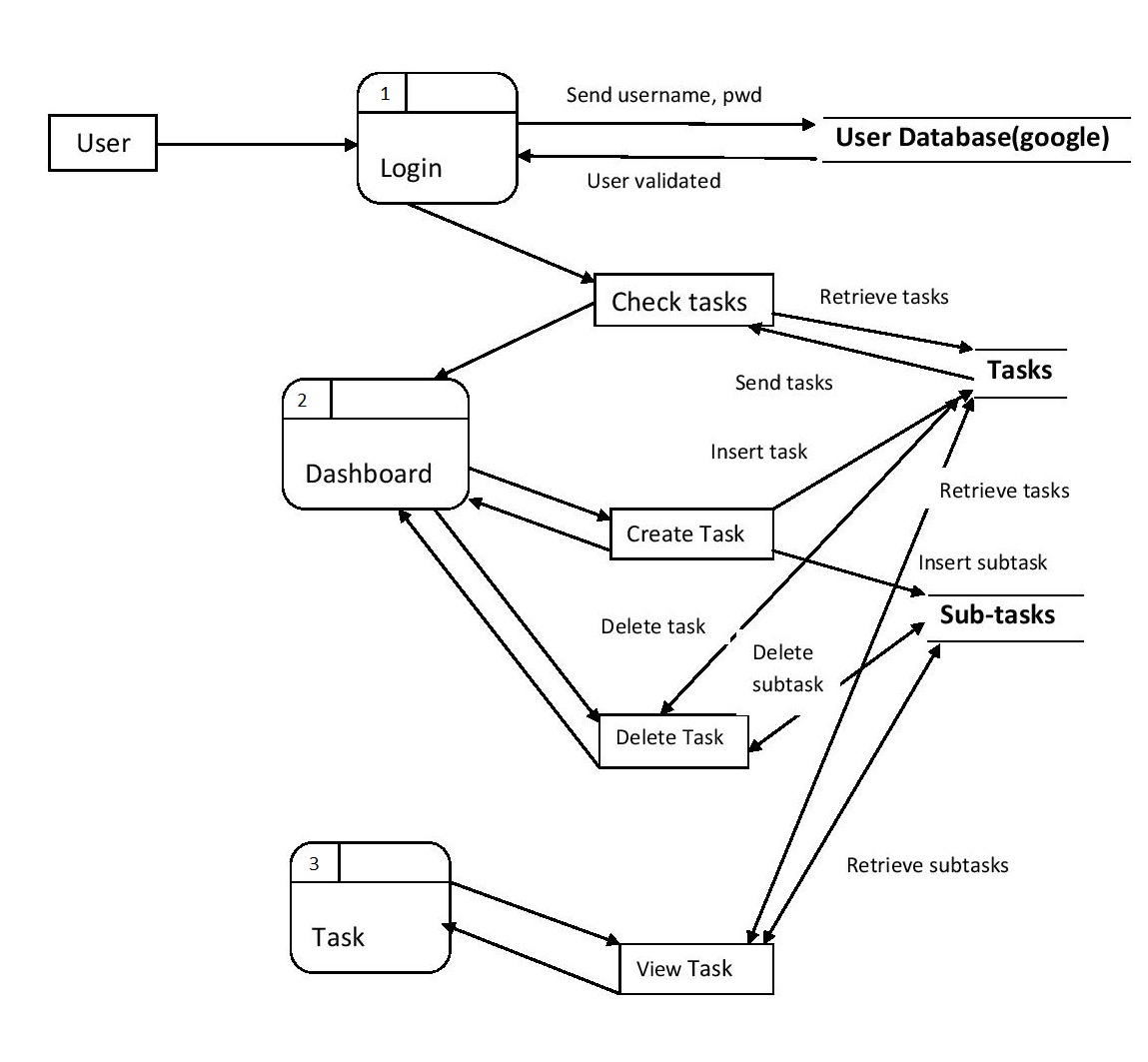
**Class Diagram:**

****

Software Design Document

**Data Flow Diagrams:**

Although the class diagram described the decomposition, the data flow diagram provides a view of how the data flows between the various objects in order to perform the various functions required.

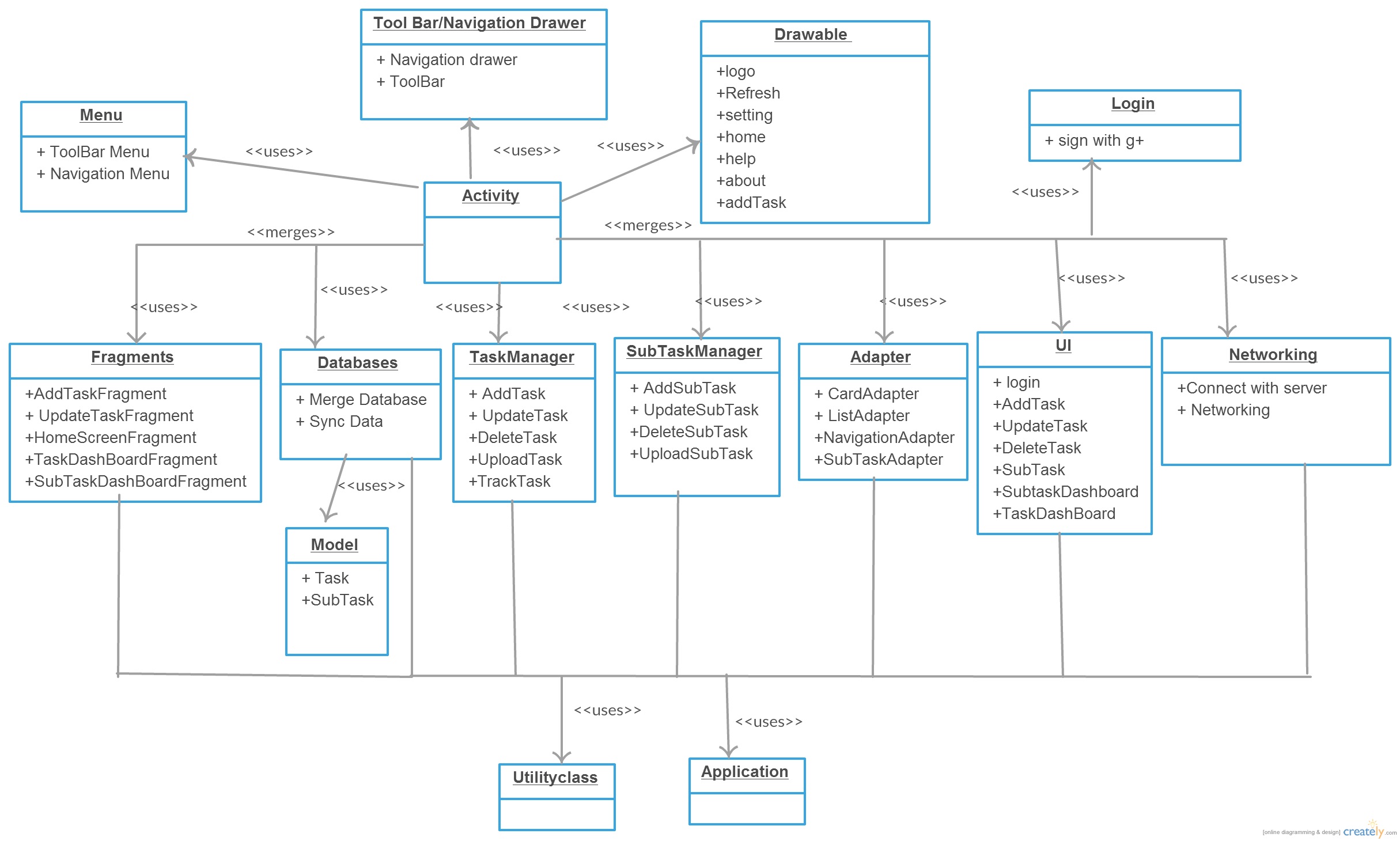


Software Design Document

**3.3. Design Rational****e**

**Package Diagram:**

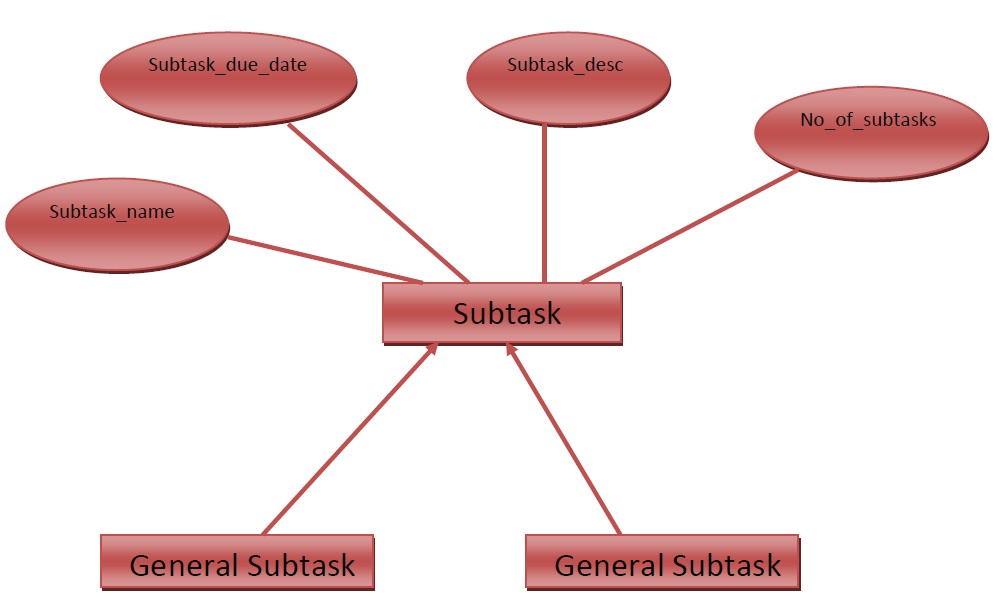
For the purpose of a more abstract view of the entire architecture and design, the package diagram is included in this document. This diagram depicts the dependencies between the various packages that make up the complete model.

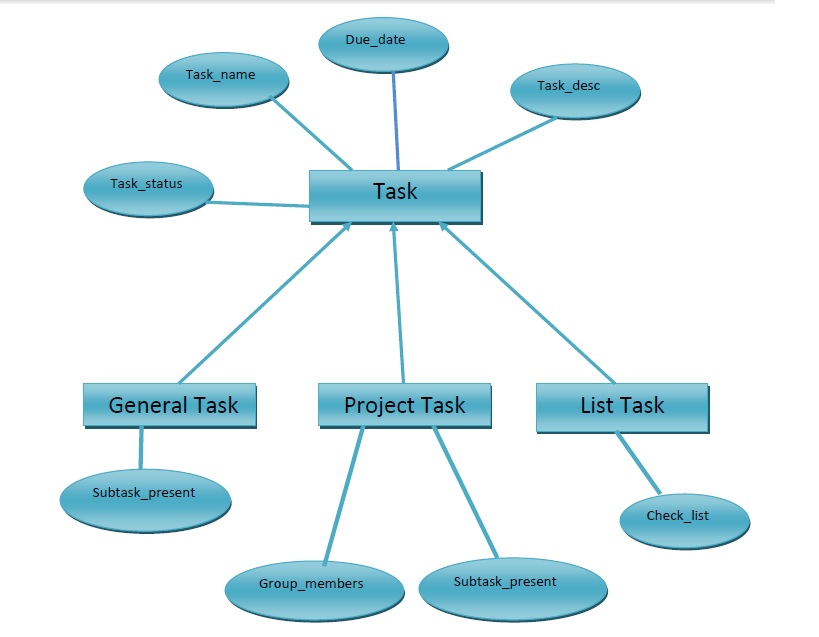


**4. Data Design**

**4.1. Data Description**

This describes how the major data i.e. system entities are stored, processed and organized. The database can be illustrated as follows:





**4.2. Data Dictionary**

The system entities and the major data are expressed along with their types and descriptions in this section of the document:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | S.No. | Item | Type |  | Explanation |
|  | **1** | Login | Class |  | The class Category stores the categories of |
|  |  |  |  |  | the products |
|  | **2** | DashBoard | Class |  | Each category stores subcategories |
|  | **3** | Priority | Class |  | The base Controller for all operations |
|  | **4** | Task | Class |  | Class stores one image which can be a part of |
|  |  |  |  |  | Product, Category or Sub Category |
|  | **5** | Subtask | Class |  | Each product is stored in this class |
|  |  |  |  | 14 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Software Design Document |
|  |  |  |  |  |
|  | **6** | User | Class | The data which stores a User Object |
|  | **7** | Task Tracking | Class | Manager class which interacts with the User |
|  |  |  |  | Dao |
|  | **8** | General\_Task | Class | Manager class which interacts with the |
|  |  |  |  | Image Dao |
|  | **9** | Project\_Task | Class | Responsible for the actual database interactions |
|  |  |  |  | using Hibernate |
|  | **10** | List\_Task | Class | Manager class which interacts with the Product |
|  |  |  |  | Dao |
|  | **11** | General\_Subtask | Class | Controller class for all Buy operations. |
|  | **12** | Project\_Subtask | Class | Controller class for all Sell operations. |
|  | **13** | createListing | Function | Creates a Listing for all the products. |
|  | **14** | Sync\_with\_google | Function | Uploads the given product/image/user details. |
|  |  |  |  | Present in Dao and in all manager classes. |
|  | **15** | Create | Function | Lists all the products. |
|  | **16** | Edit | Function | Gets the photo |
|  | **17** | Delete | Function | Gets the id |
|  | **18** | View | Function | Gets the parented of the given id |
|  | **19** | Set\_reminder | Function | Sets the parented of the given id |
|  | **20** | Calculate\_priority | Function | Gets the name of the given id |
|  | **21** | Delete\_on\_swipe | Function | Sets the name of the given id |
|  | **22** | Calculate\_completion(task) | Function | All getter and setter methods (eg |
|  |  |  |  | 16,17,18,19,20,21) have the same format. |
|  |  |  |  | They are used to access private variables. |
|  | **23** | Create\_subtask | Function | Data Access Object |
|  | **24** | Create\_Group | Function | Manager |
|  | **25** | Assign Priority | Function |  |

**5. Component Design**

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

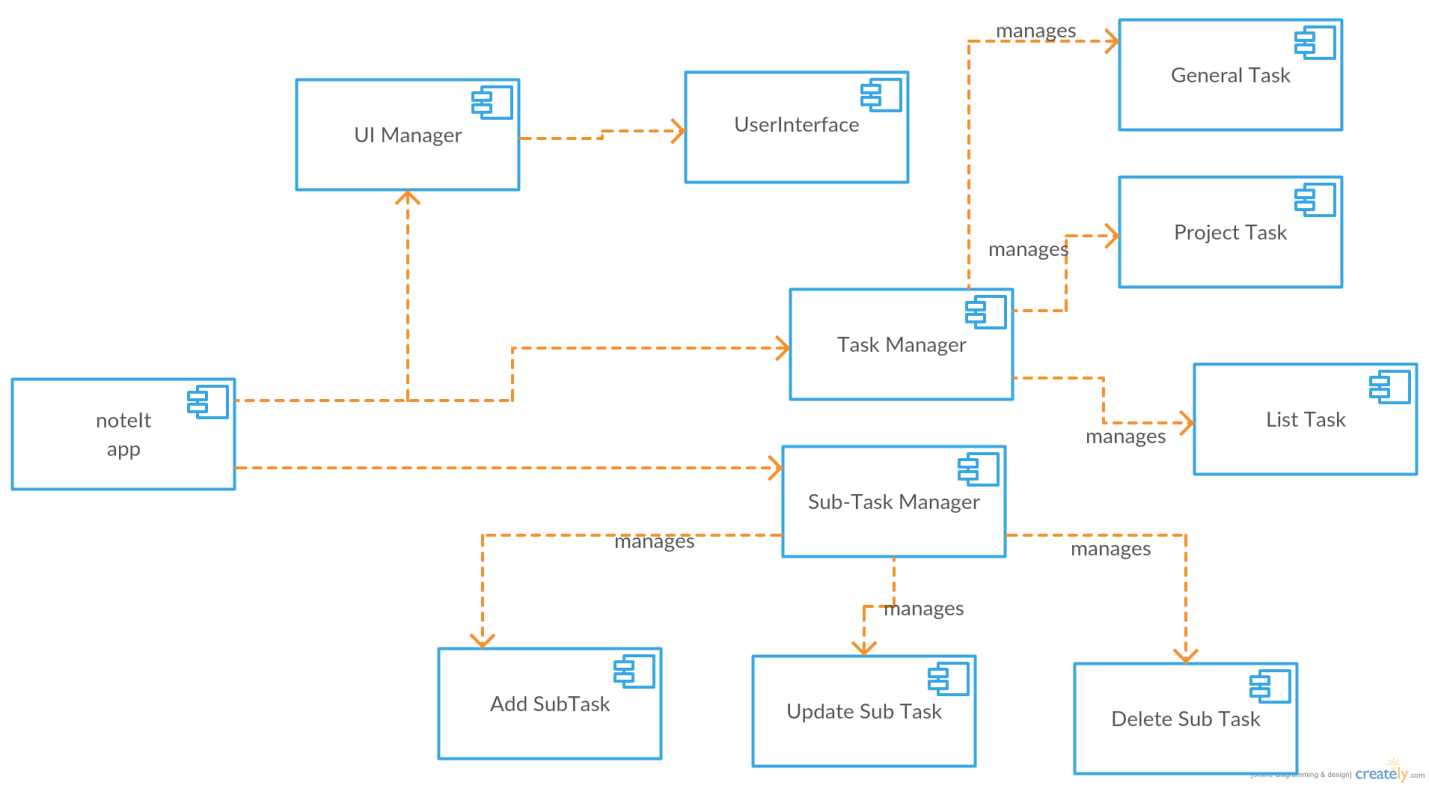
So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

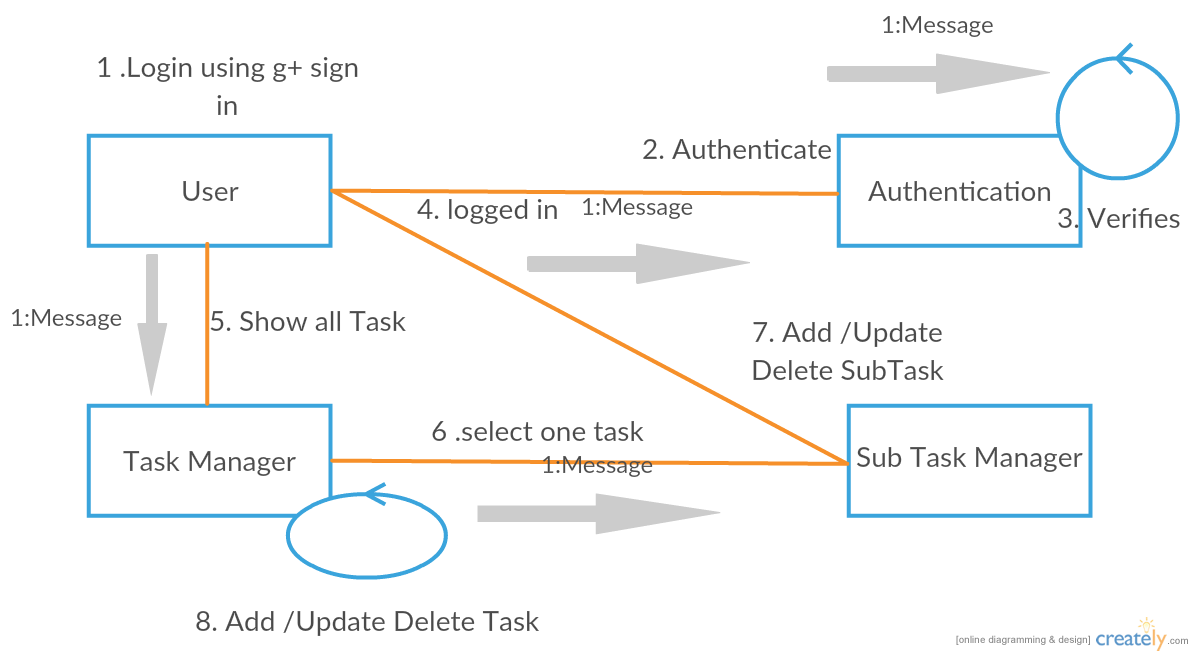
A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

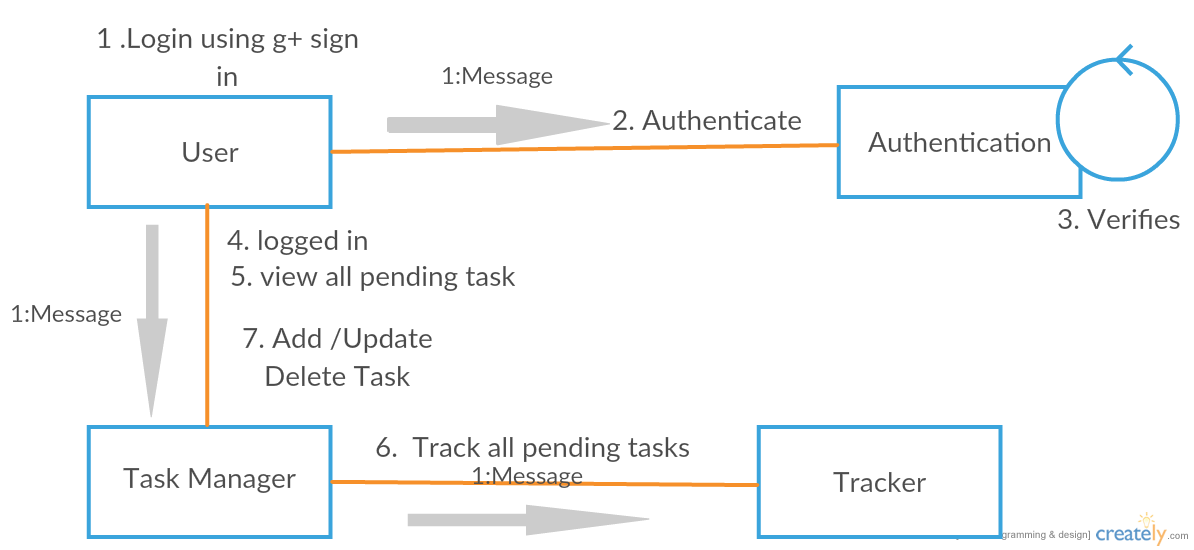
So the purpose of the component diagram can be summarized as:

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.

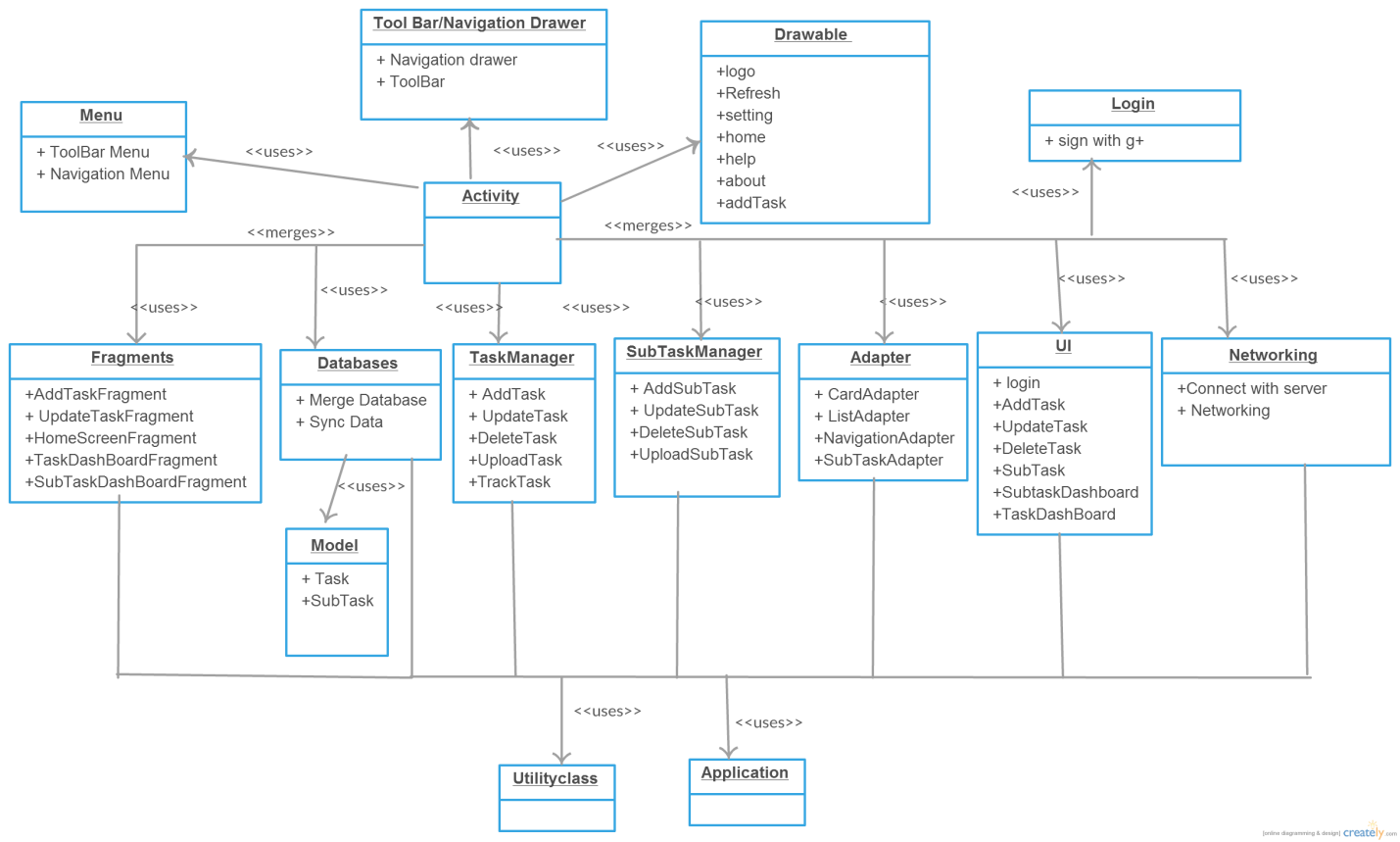


**Collaboration Diagram**

****

****

**Package Diagram**

****

**6. Human Interface Design**

**6.1. Overview of User Interface**

This section describes the functionality of the system from the user’s perspective explaining how the user will be able to use the system to complete all the features and the information that will be displayed for the user. It also gives a description of the interface provided for the users.

A first-time user of the web application sees the Home page when he/she opens the application. If the user has not registered, he/she is able to do so by clicking the Sign Up button on the Home page. In the Sign Up page, the user creates a new account by providing details such as Name, Email ID, and Password. An existing user can simply sign into the system through the Sign In page as shown. The user provides the email id and password which is used to authenticate the user. In case the password is forgotten, the user can recover the account using the mail ID provided at the time of account creation.

Once the user opens the application, he sees his own home page where two buttons are provided. One for buy and another for sell. The user can click on any of the buttons depending on his needs. If he clicks on buy button he will be taken to the category page.

In the category page, user will see a list of categories. The user can click on any of the categories to buy a product. After choosing a category the user will be directed to the sub-category page where the user will have to pick a sub- category. After this, he will see a list of products corresponding to that sub-category. From the available products the user will select a product and then the details about that product will be presented to the user. If the user wants to buy the product, he/she will have to log in into his account to view seller details.

16

Software Design Document

In the sell page, the user needs to first sign in. After signing in, he/she will fill the product details that they want to sell. After providing the details the ad will be first approved by the admin. Only after the approval of the admin the product will be added to the product list.

On every page header and footer have been added which includes the links to the pages: About, Contact, Log in, Sign-up for easy navigation between the various pages. Since the web portal does have any designated hardware, it does not have any direct hardware interfaces. The hardware connection to the database is managed by the underlying operating system on the web client and the web server.

The application connects to the Apache Web Server for retrieving and adding data to the database. This is handled by creating and maintaining database connections from the local application to the online Web Server.

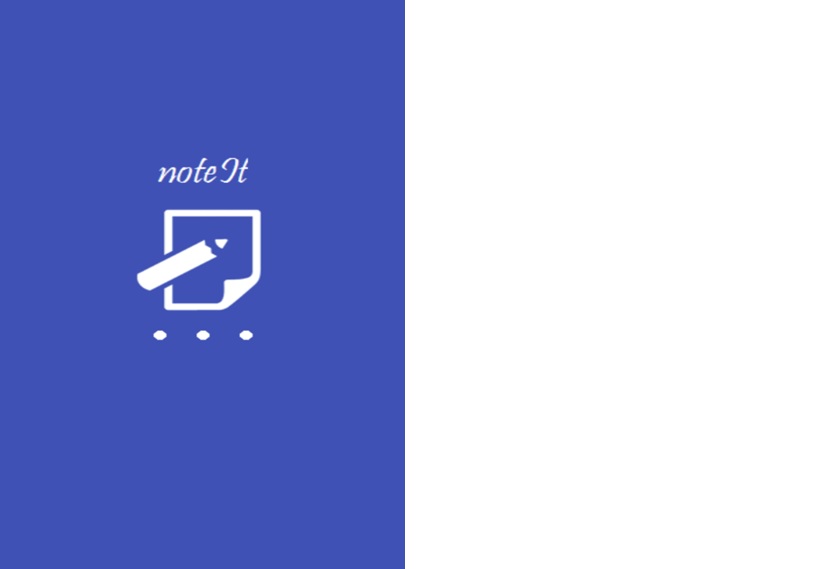
The three golden rules of GUI –

* Place Users in Control
* Reduce Users’ Memory Load
* Make the Interface Consistent

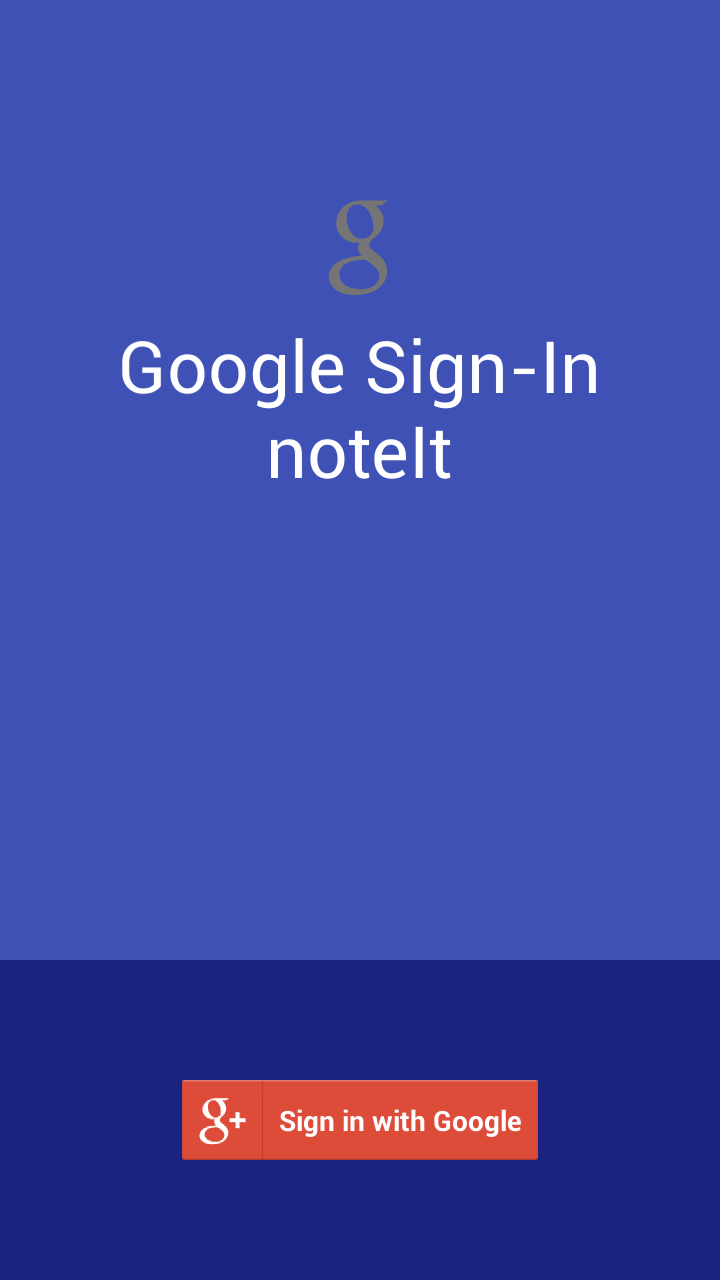
These rules have been followed in the development of the User Interface of this product so as to ensure usability and also enhance the user experience.

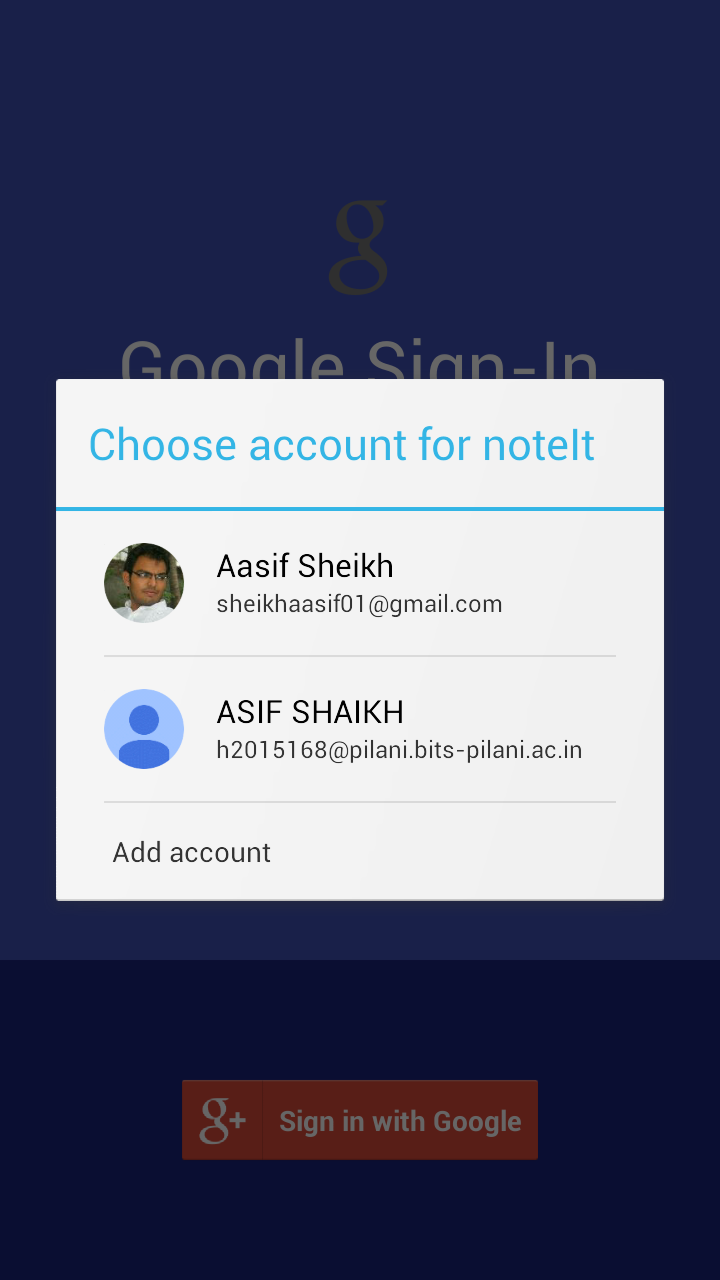
**6.2.Screen Images**

1 .Splash Screen

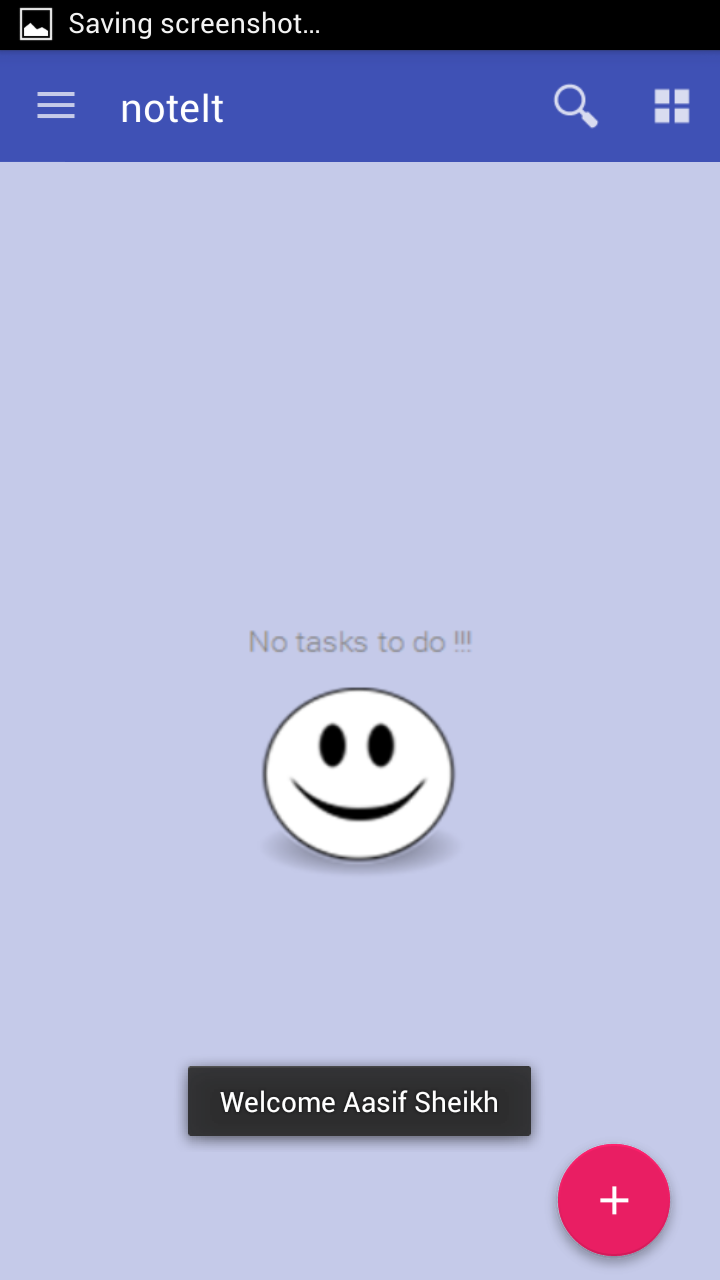


2 .Google Sign In

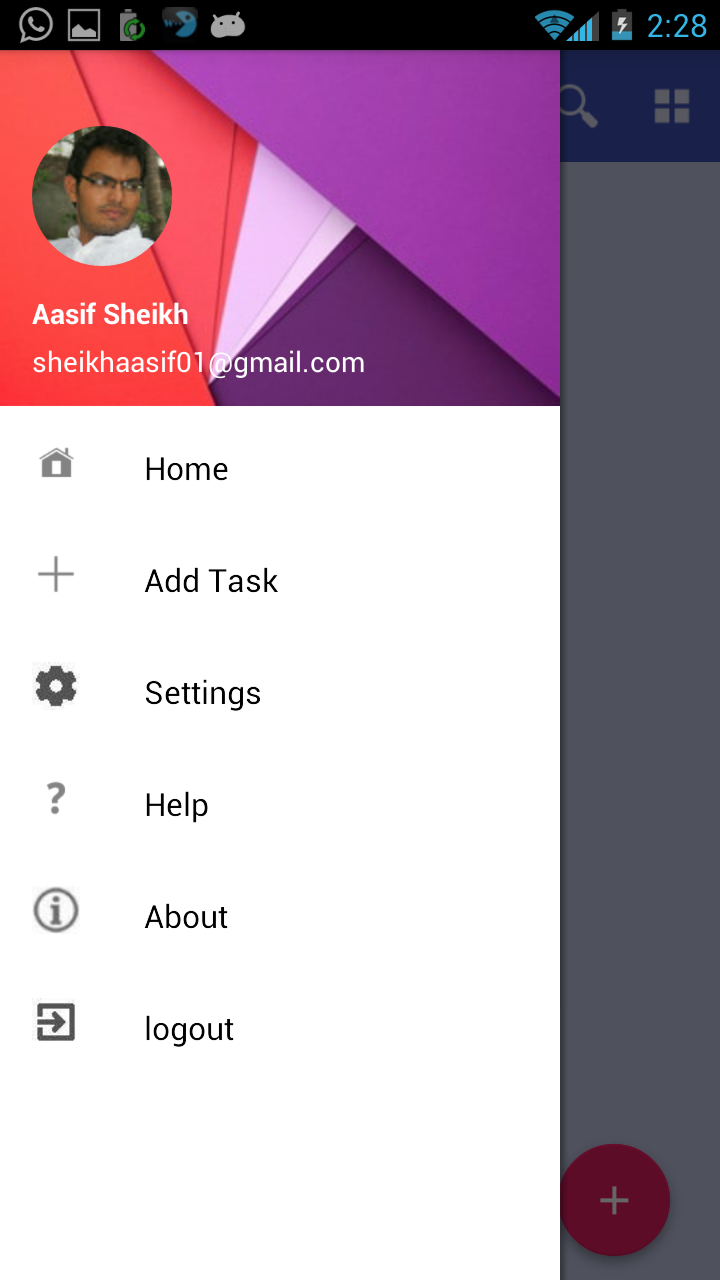


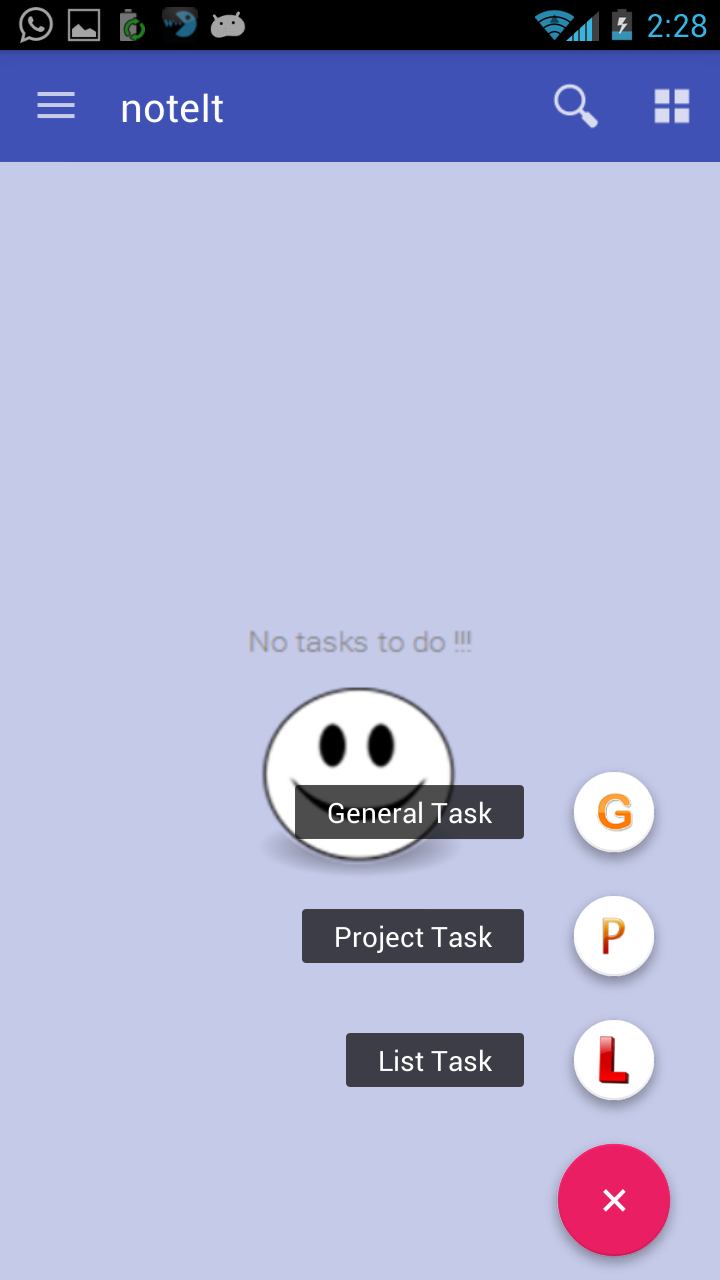


3 . User Authenticated

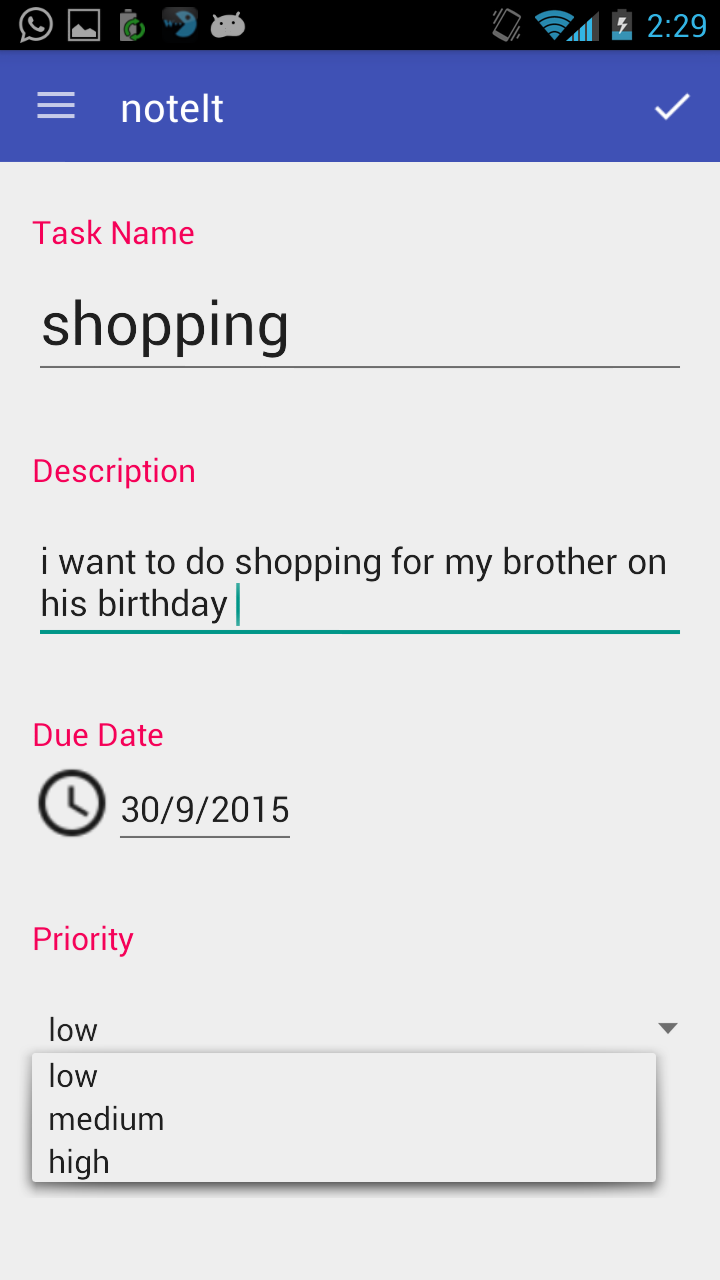


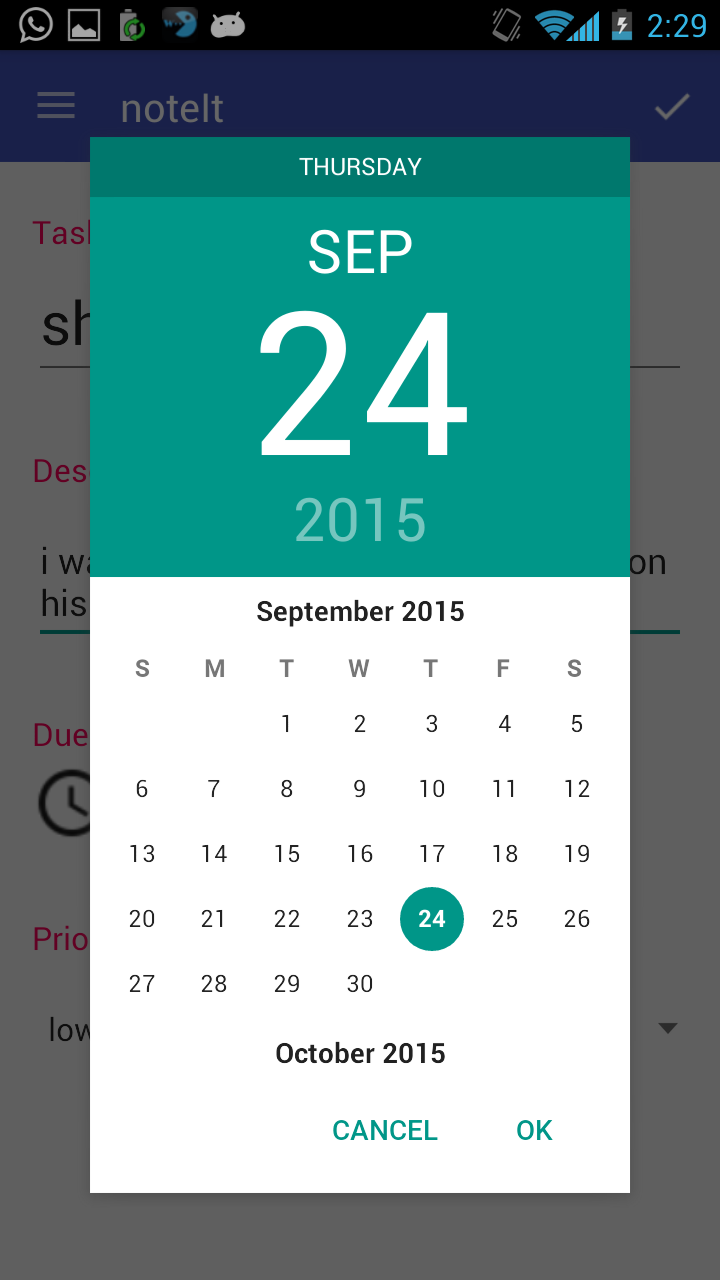
4 . Navigation Drawer



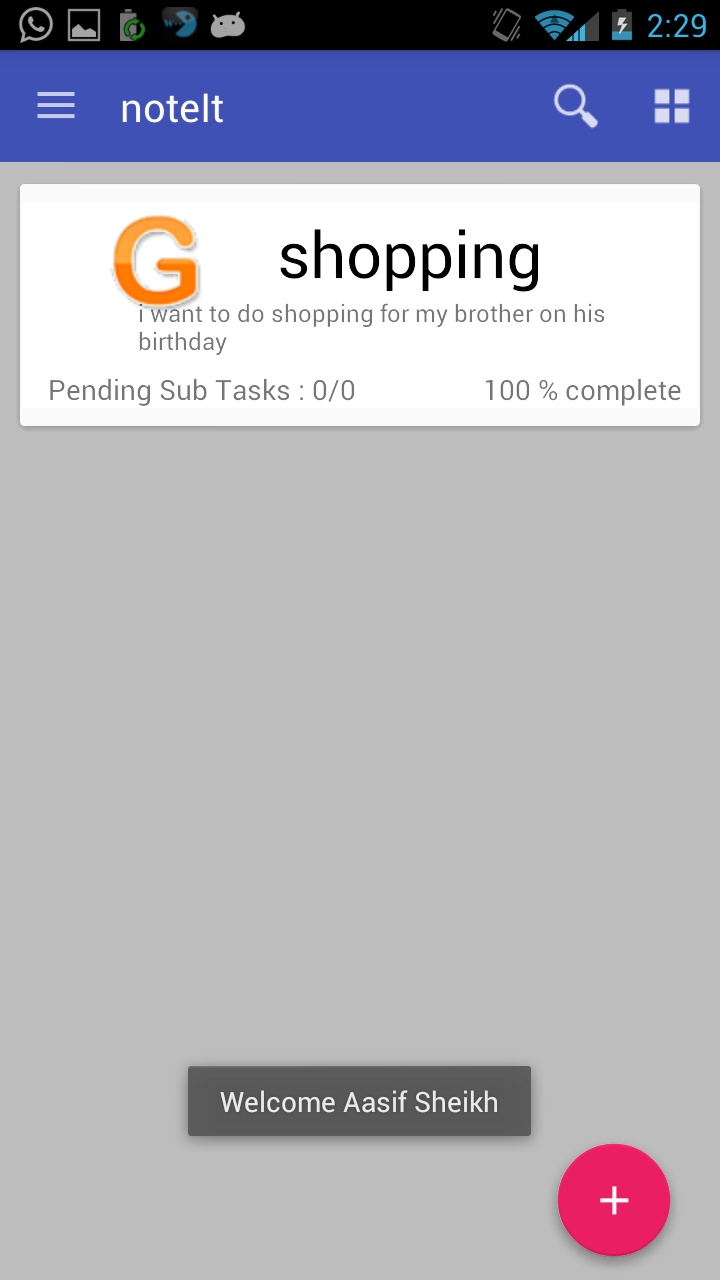


5 . Task Creation





6.Display All Tasks



19

Software Design Document

**6.3. Screen Objects and Action**

**Other Objects and Actions:** The various objects present on the screen are self explanatory and hence theusability of the product is very high – given the fact that the human interface is understandable to any new user.

**7. Requirements Matrix**

This section provides a crossreference that traces components and data structures to the requirements in the SRS document.

The following tabular format shows which system components satisfy each of the requirements:

|  |  |
| --- | --- |
| Requirements | Components satisfying the requirements |
| User Authentication | UI Manager,User Interface |
| Account Creation | UI Manager,User Interface |
| Adding new Task | User Interface,Task Manager,Add Sub Task |
| Editing existing task | Task Manager,Delete Sub Task |
| Deleting task | Task Manager,Update Sub Task |
| Setting priority of task | Task Manager |
| Modifying priority of task | Task Manager |
| Sharing task | Task Manager |
| Listing all tasks | Task Manager |
| Performance | All |
| Authenticated Users | UI Manager |
| Reliability | All |
| Availability | All |
| Usability | All |
| Testability | All |
| Correctness | All |

20