BSC(Hon.) 4

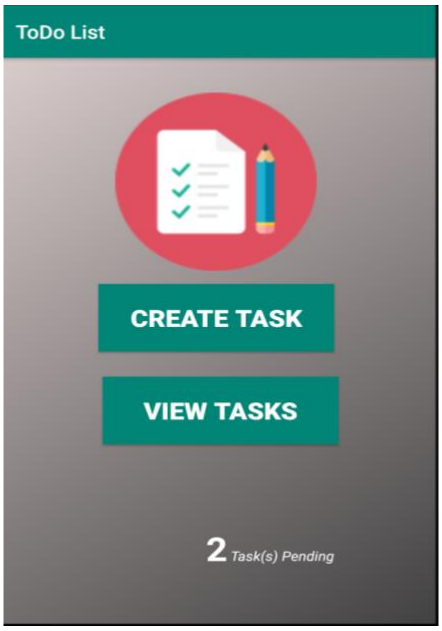
ToDoList Breif

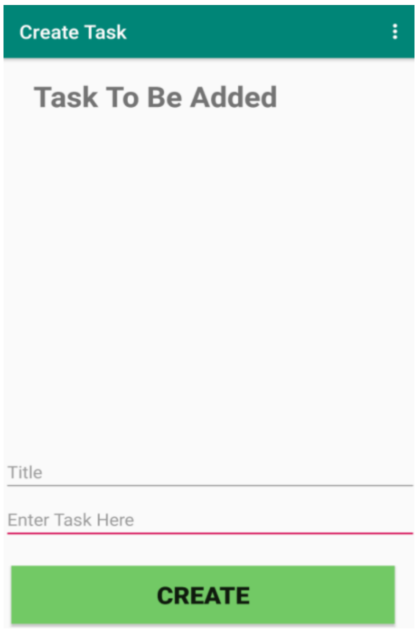
ASSIGNMENT 01

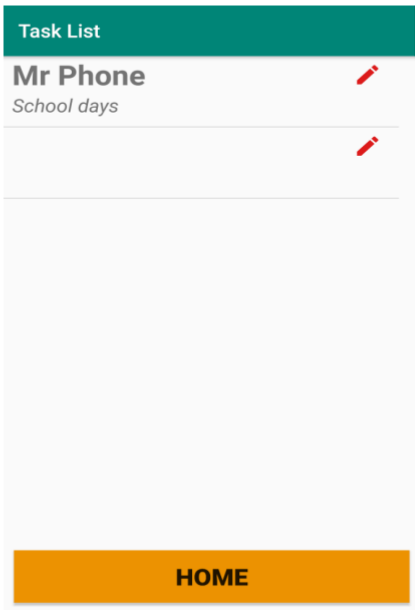
DANISH ALI 2875170

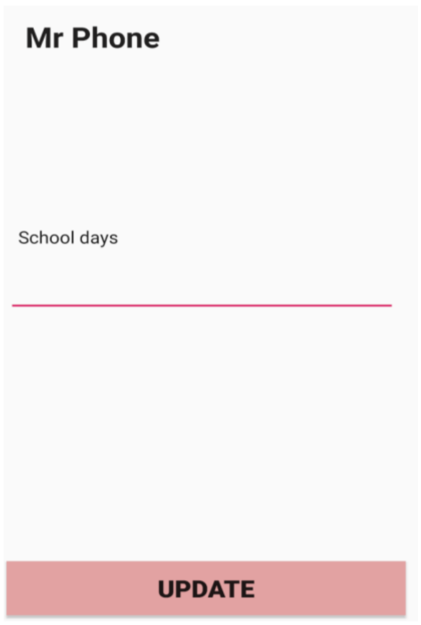
Table of Contents

My UI Design


ToDo List
CREATE TASK
VIEW TASKS
2 Task(s) Pending


Create Task
Task To Be Added
Title
Enter Task Here
CREATE


Task List
Mr Phone
School days
HOME


Mr Phone
School days
UPDATE

UI Design

The android application under discussion has following four activities:

1. Main Activity (Home Page)
2. Second Activity (Create Task)
3. Third Activity (Task List + Delete Task)
4. Edit Activity (Update Task)

While designing UI of Main activity minimal approach was kept in mind. *ImageView* in the centre of layout to stimulate visual memory of user. As to write-down tasks without any gadget humans use piece of paper and pen. Keeping described logic in mind, main central image is placed at welcome home activity.
Two traditional button views are used to navigate to Second and Third activity. As a user, to navigate through application, it is more convenient and efficient to have reasonable surface area which could take your touch stimulus. In contrary, fancy views (Floating button, Stylish texts etc.) do not meet users expected response.
TextView to show user number of pending tasks in its list now is simply to track modifications of tasks. Colour scheme for Main activity is primary colour of android application. However, background gradient colour is used to differentiate Main home activity from remaining activities.

Second activity is adding tasks to list. UI design of this activity consists of a *TextView* describing which part of application user is in, two edit fields (task title + task description) and a button to trigger all processing. *ListView* is showing tasks added. Reason to make it that is to make it look like register with multiple entries. The button is placed at the bottom of the screen as it is a frequent action that the user's thumb must reach.

UI of third activity is primarily based on touch events. On touch *edit* and on long touch *delete*. Reason of keeping UI like this is to make it convenient for user to operate, in comparison to button where user must touch on task first then eventually go to delete button to delete task. Consistency is out of focus a bit, while keeping create/update task on button and task modification/deleting on touch handler. However, as a user sometimes convenience is preferred over consistency of design.

ADDITIONAL FEATURES

1. ***AMINATIONS ON EDIT FIELDS UPON INVALID ENTRIES, PENDING TASKS COUNT. DUPLICATE TITLE NOT ALLOWED.***
2. ***MODIFY REQUEST OF EXISTING TASK UNDER SAME TITLE.***
3. ***RESETING OF TASKLIST***
4. ***RANDOM POPULATING TASK LIST***

Data Structure and methods

Reason I used arrayList is it supports dynamic arrays that can grow as needed. Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

void initialize() in each activity this method is initializing all private fields declared in the class. For example, Group View components, variables or DBHelper class. Thus, all major private fields are initialized in this method using respective *ids* associated with them. Reason to bind them all together is to make it look tidy and avoid congestion. By single call of *initialize()* in *Oncreate* initialize all declared variables and override to set content layout and show main activity when app is launched.
onCreateOptionsMenu() function override to inflate and put new options in menu.
onOptionsItemSelected() function to listen click when a option is selected/clicked from menu retreiveTaskList() – function to retrieve tasks data from database *buttonClickHandler()* function to handle click listener of items in the list *addTaskButtonHandler()* function to jump to add task activity when click happens
createDummyList() function to create 3 dummy tasks in to do list
AddTaskActivity.java
addNewTaskButtonHandler() it will add new task to the database *goBackButtonHandler()* function to go back to main activity
CustomArrayAdapter.java
CustomArrayAdapter() constructor of class
getView() – returns view of items in list
getCount() – returns size of list
getItemId() – returns position of item in list
getItem() – returns Item Object on a particular position.
EditActivity.java
update_btn.setOnClickListener(View) get task entered by user from editfield using *.matches()* *EditTask()* and *EditTitle()* prefills data of item clicked in list so that user can view info and edit easily. *UpdateTaskButtonHandler()* updates the task data in db when user clicks on this button
EditTask() and *EditTitle()* are updated using *updateData()* returns to main activity.
TaskDbOpenHelper.java
addData() insert task data in db *getListContents()* gets all tasks data from db *getItem()* return task data of a particular string task, *deleteData()* delete task data in db.

CustomItem.java
CustomItem() constructor of class. *getName()* returns id of task item. *getTitle()* returns name of task item. *getPlace()* returns place of task item. *getMessage()* returns message of task item. *setName()* sets name of task Item. *setTitle()* sets title of task item. *setData()* sets date of task item
TestDBOpenHepler.java onCreate() creates table upon call. *onUpgrade()* drops table and create new db. *addData(String item)* adds item (task title + task Despriction). *updateData(String existing)* update on particular column. *removeData(String item)* removes entry on that particular column. *getListContents()* returns *cursor* containing all entries in db. *String getitem(String where)* taking some clause in and spiting up item(task). *totalrecords()* part of additional feature to outline number of tasks in db. *clearDatabase(TABLE_NAME)* clears out all values from db, hence resets tasklist.

SecondActivity.java lv_mainlist.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() it figures out which position click is happening and gets db value associated with that entry and calls on toaste details of task

add_btn.setOnClickListener(new View.OnClickListener()) added task to db via addData() in tdb class and updates arraylist.

ThirdActivity.java onCreate() arrange all tasks in arraylist via *getListContent()* and represent it listview. *third_lv_task_db.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() it figures out which position click is happening and gets db value associated with that entry and calls on tdb.removeData() and update CustomArrayAdapter().third_lv_task_db.setOnItemClickListener(new AdapterView.OnItemLongClickListener() takes to new activity that creates update field for exisiting task.*