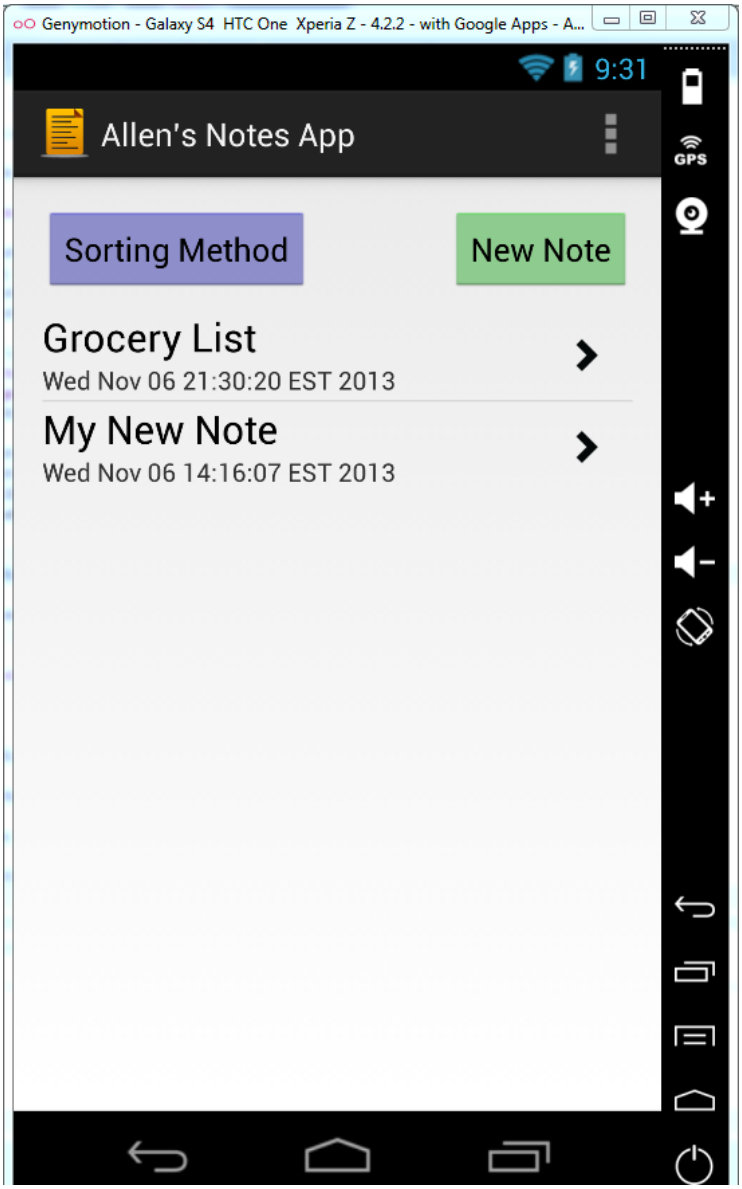
CSCI4100U – Mobile Devices Assignment 2

Assignment 2: Notepad Application using built in SQLite data storage for Android.

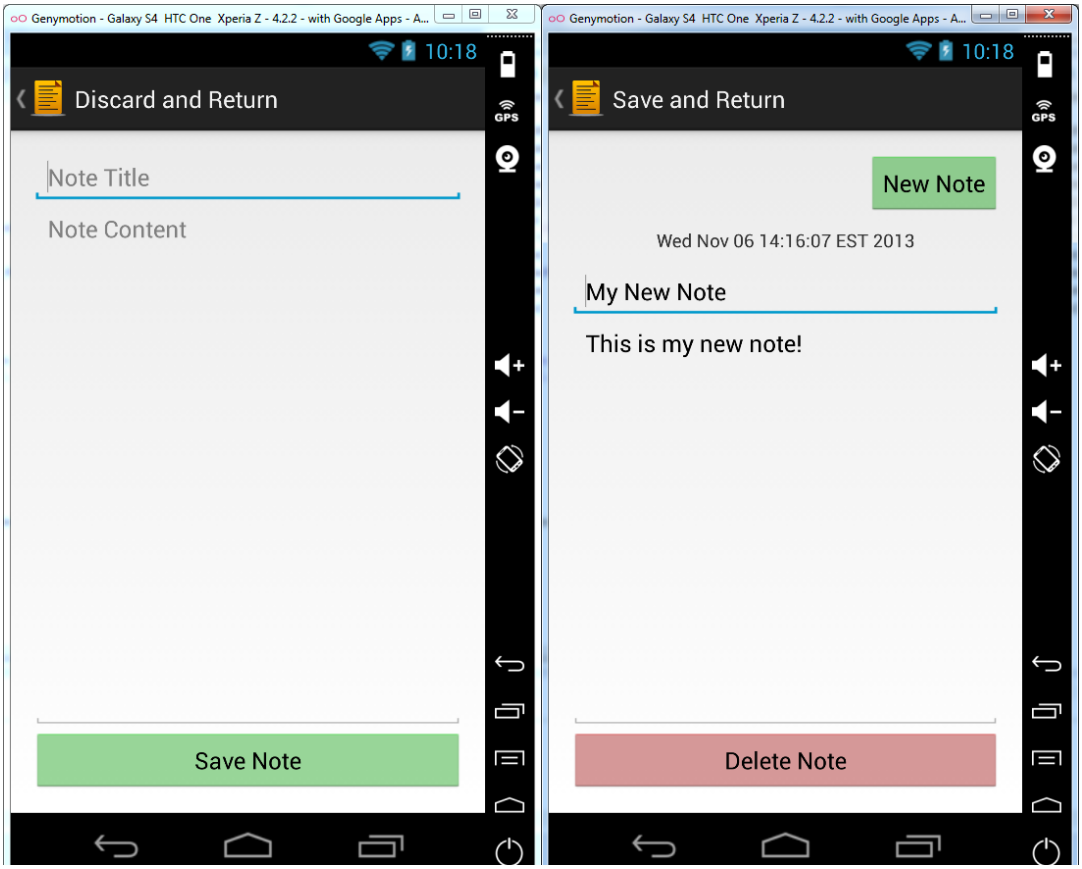
By: Allen Kaganovsky | 100389429



This assignment was to produce a simple note keeping application with the ability to create and store notes, recall and edit notes, as well as delete any previously stored notes. The first step I took in creating my note application was determine the basic interface and what data the application would need to keep track of for each note, as well as persistent storage of all the notes. Each note object contains a Title, Content of the note, Date of creation and an ID which is used for database manipulation and row element identification.

I created a Note class which contained the above 4 attributes. The constructor simply initialized the title, content and generated the date of the note through passed method parameters. Each of the fields also have a subsequent getter and setter method for access to the note attributes by methods.

Once my Note class was complete, I proceeded by completing my main activity layout, which includes a New Note button, a Sorting function for the list of notes, as well as the list of notes. I created another 2 activities, one for creating a new note, and the other for viewing/editing and deleting previous notes. These two activities look similar, with exception with a Delete and New Note buttons on the viewing/editing view.



With the interface of the application complete, I needed a way to store, retrieve, modify and delete notes – I also want the notes to be persistent and remain on the device even after powered down. I choose to use SQLite database which is included within the Android OS and I'm fairly familiar with MySQL already.

In order to use the SQLite database, I created a DBControl class that contains the database connection object as well as the appropriate methods to add, update, delete and query from the database. The DBControl class extends the SQLiteOpenHelper class which features helpful functions such as onCreate to create the database table, onUpdate to recreate the database table if needed, and a database object to allow us to send SQL commands or use a Cursor object to access the database.

When creating a new note, the save button simply gets the text from the title and content text fields, creates a new note object, and adds the note object data into the database using the SQLiteOpenHelper and included "insert" method to insert new rows into the database. Returning to the parent activity discards the creation of the new note.

When viewing/editing a pre-existing note, the main activity with the list has an onItemClick method that is able to determine which element in the list was selected, obtains the corresponding ID from the note object and passes that ID through a intent to the view/edit activity. When navigating away from the view activity back to the parent Main activity, both the current title text field and content text field contents are obtained and updated in the database by executing a DB query with a typical SQL UPDATE TABLE query.

The user may also choose to delete a note they are currently viewing/editing. When the delete button is pressed, the corresponding row is deleted according to the notes ID. The user is then automatically brought back to the parent Main activity and shown the updated list.

Finally, I created a custom NoteAdapter class by extending the ArrayAdapter, however I first created another new layout that would represent each cell of the table. I choose to use a large text for the title, a small text for the date as a subtitle and an arrow image to display a new view will open when pressed. The getView method was modified in order to set the Title text field and Date text field from each Note in the list. The resulting cell design can be seen on the title page. In the Main activity, I simply query all of the notes stored in the database and store them in a list of notes. This list of notes is then passed to the ArrayAdapter and displayed appropriately in the view.

I felt an important feature to have in such an application is a sort function. I would like to sort the notes by title, or by date of creation. The sorting of the list was easily done through database ORDER BY query modifications and refreshing the list of notes. The sorting method was stored on the device as a shared preference key value.