App Proposal: Energyflo

Idea:

Energyflo is a personal-productivity optimizer.

Our bodily energy levels are not static, but change throughout the day. Labeling ourselves as an "early bird" or "night owl" is of limited help in day-planning.

By observing and recording our energy levels at regular intervals over a number of days, we can begin to identify times of peak-energy. This in turn allows us to prioritize our most important work for our most productive hours of the day.

App features:

- Simple user-interface for data input
- Notifications reminding the user to record their current energy level
- Graph showing average energy levels for each hour of the day
- Recommendations for peak-hours of productivity
- Notifications notifying users that they are about to enter a peak energy period and should plan to do their most important tasks

Design Components:

- Input Activity
- Graph/Recommendation Activity
- Settings/Notifications Config Activity
- Database

Estimated Effort:

Expected areas of high effort are the database implementation (5-6 hrs), graphical design (5-8 hrs) and notification scheduling (5 hrs).

Areas of low anticipated effort are the basic app design (1-2 hrs) and data input mechanisms (1-2hrs).

Anticipated Hours: 17-23

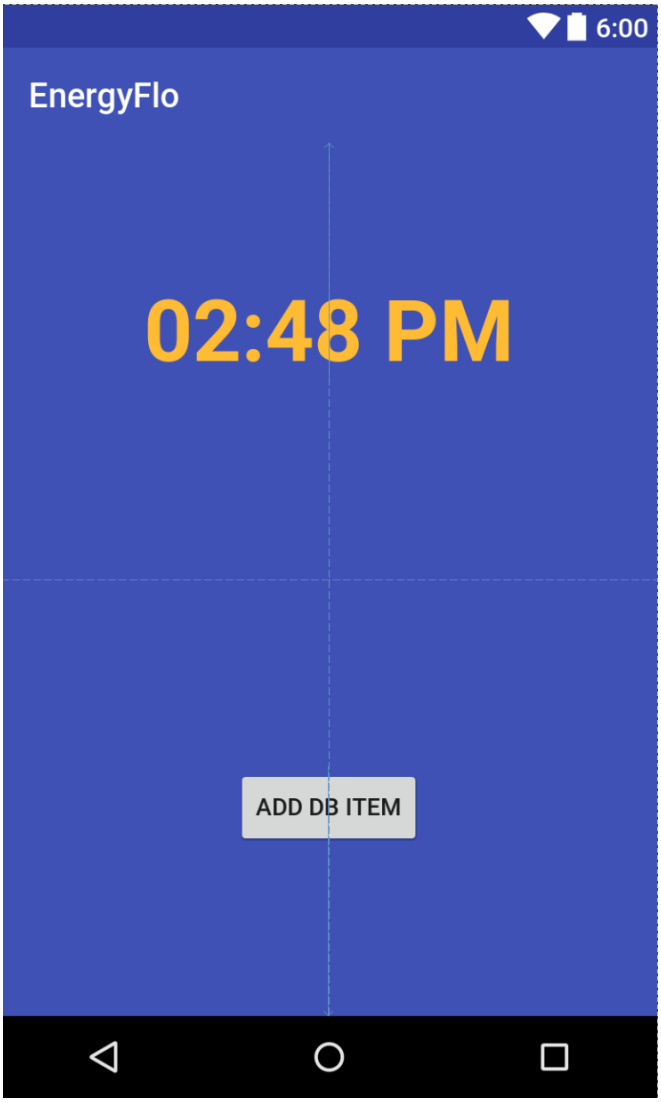
Work Distribution:

William
- Database implementation
- Graphics

Dillon
- Graphics
- Basic App Layout
- Notifications

Nam
- Notifications
- Database

Design Prototype:



Mock-up of main activity – to be used for entering new energy ratings and updating database.

Will add menu for access to:

- Graphical representation of past ratings
- Options/Settings Menu
    - Configure app notifications

Button is placeholder – will replace with rating mechanism.

- Scoring model will be 1 through 10
- Will allow only 1 entry per hour

Function for adding entries to database – currently uses hardcoded value for testing purposes. Will take a user-defined value, parse the current time and update the database entry for the current hour.

```
public void addItem(View view){
    //TODO query db, check if value exits, if yes update, if no create
    CharSequence now = time.getText();
    //TODO raise API to accomodate this function
    //time.getFormat24Hour();
    CharSequence amPm = now.subSequence(6,8);
    CharSequence current_time = now.subSequence(0,2);
    int number_time = Integer.parseInt(current_time.toString());
    int number_rating = 5;

    if((amPm.toString().equals("PM") && !current_time.toString().equals("12")) || ((amPm.toString().equals("AM") && current_time.toString()
.equals("12")))){

        //number = Integer.parseInt(current_time.toString());
        number_time += 12;
        //android.util.Log.v("TRUE", "IT IS NIGHTTIME add 12 = "+number_time);
    }


    try{
        Log current_log = mDbHelper.getLog(number_time);
        //TODO edit log to add new params and update
        current_log.updateLog(number_rating);
        mDbHelper.updateHour(current_log);
        android.util.Log.v("Success", "updated hour " + current_log.getHour() +"Average = "+ current_log.getAverage() +
                "Number of ratings = "+current_log.getNumber_of_ratings() +"Total = " + current_log.getTotal());

    }
    catch(Exception e){
        android.util.Log.v("Exception", "Adding value " + number_time);
        Log log = new Log(number_time,number_rating,1,number_rating);
        mDbHelper.addHour(log);
        return;
    }
    //android.util.Log.v("BAD", "Added value but didn't return");

}
```

Class representing a specific hour's information.

```
public Log(int hour,double average,int number_of_ratings, int total)
{
    this.hour=hour;
    this.average=average;
    this.number_of_ratings = number_of_ratings;
    this.total = total;
}
public void setHour(int hour) { this.hour = hour; }
public void setAverage(Float avg) { this.average = avg; }
public void setNumber_of_ratings(int num) { this.number_of_ratings = num; }
public void setTotal(int tot){ this.total = tot; };
public int getHour() { return this.hour; }
public double getAverage() { return this.average; }
public int getNumber_of_ratings() { return this.number_of_ratings; }
public int getTotal(){ return this.total; }
public void updateLog(int rating){
    //method updates the log that was grabbed from the database
    //NOTE this does not update the database!
    this.total += rating;
    this.number_of_ratings++;
    this.average = this.total/this.number_of_ratings;
}
```

Creating the database and table for log entries.

```
/**
 * Created by Billy on 2/15/17.
 */

public class DBHandler extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "energyflowdb";
    //tables
    private static final String TABLE_LOG = "Log";
    // Shops Table Columns names
    private static final String KEY_HR = "Hour";
    private static final String KEY_AVG = "Average";
    private static final String KEY_NUM_OF_RATINGS = "Number_Of_Ratings";
    private static final String KEY_TOTAL = "Total";
    private static final String CREATE_LOG_TABLE = "CREATE TABLE " + TABLE_LOG + "("
            + KEY_HR + " INTEGER PRIMARY KEY," + KEY_AVG + " DOUBLE,"
            + KEY_NUM_OF_RATINGS + " INTEGER," + KEY_TOTAL + " INTEGER)";
    //DBHandler mDbHelper = new DBHandler(getContext());
```