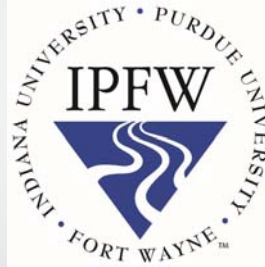


# Android Database Map App



5/1/2015

Advisor: Michelle Parker

CPET 491

Allan Burris

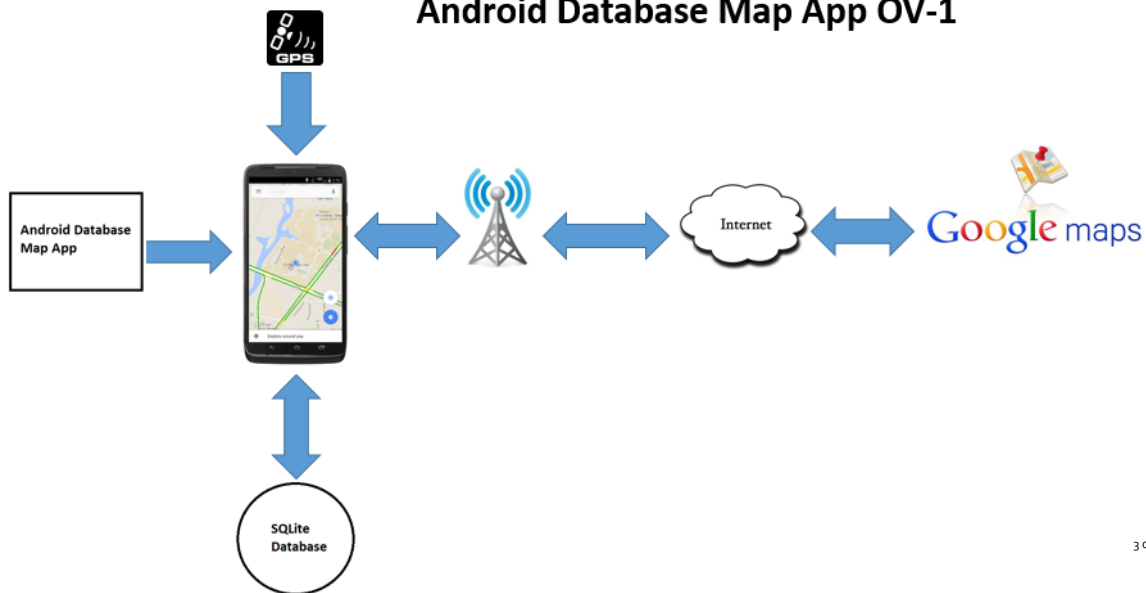
## Outline

- Executive Summary
- Introduction
- Problem Statement and Solution
- System and App Requirements
- Software Analysis
- Software Design
- Software Integration Testing and Validation
- Conclusion

2 of 26

## Executive Summary

### Android Database Map App OV-1



3 of 26

## Executive Summary

- Motivation
  - Haiti Trip
  - Nonprofit organization Onebody
- Deliverables
  - App
  - Presentation
  - Report
- Summary of Schedule
  - 16 weeks
- Summary of Cost
  - Materials: \$354.00, Labor: 144 hours

4 of 26

## Problem Statement and Solution

- Problem
  - Need a better way than pencil and paper or spreadsheets to organize contact information and locate addresses on a map with multiple markers.
- Solution
  - Create an Android App that will use information stored in a database, and locate addresses using Google Maps filtering by different categories. The information from the database will be shown in each pin placed on the map.
- Value
  - Making information more accessible and easier to manage. Increasing usability and impact of gathered information for a better outreach to people.

5 of 26

## App Requirements and Validation

- Physical and Environmental Requirements
  - Android KitKat 4.4.4
  - Android Studio
  - SQLite
  - Google Maps
- App Operational Requirements
  - Shall link database addresses to Google Maps
  - Shall be able to add SQLite database entries
  - Shall be able to edit SQLite database entries

6 of 26

## App Requirements

1. Operational: Android app shall link database addresses to a digital map
2. Operational: The Android app shall add new entries to database
3. Operational: The Android shall allow editing of database entries
4. Functional: The Android app shall use Google Maps API to access digital map
5. The Android app shall be able to select entries to display based on user-selectable categories

7 of 26

## App Requirements

6. Functional: Google Maps shall be used to locate addresses
7. Functional: The Android app shall be used to locate more than 10 addresses on a digital map
8. Performance: The Android app shall provide a minimum of TBD categories
9. Physical: The Android app will require less than 50MB of RAM
10. Physical: The Android app shall be less than 20 MB in size on phone
11. Environmental The Android app shall be created using Android Studio
12. Development Software

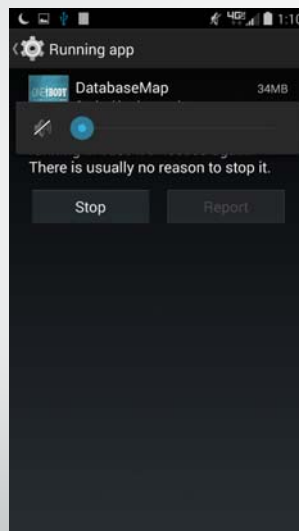
8 of 26

## App Requirements

- 13.Environmental: The shall run on Android KitKat operating system
- 14.Environmental: The Android app shall use SQLite database

9 of 26

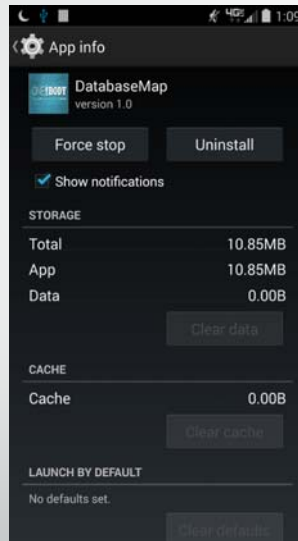
## App Requirements



The Android Database Map app is less than 50 MB

10 of 26

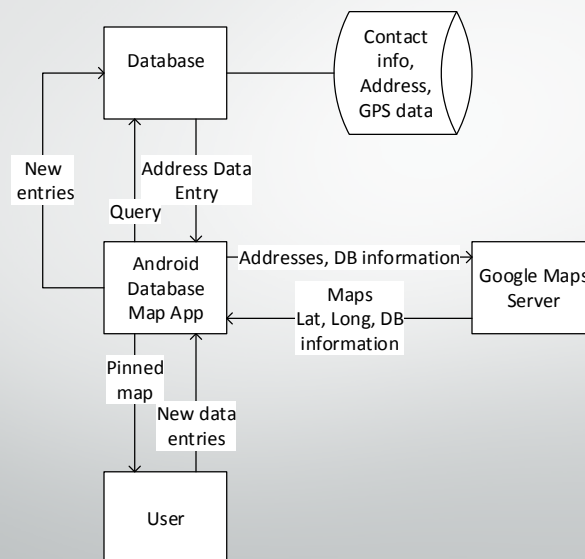
## App Requirements



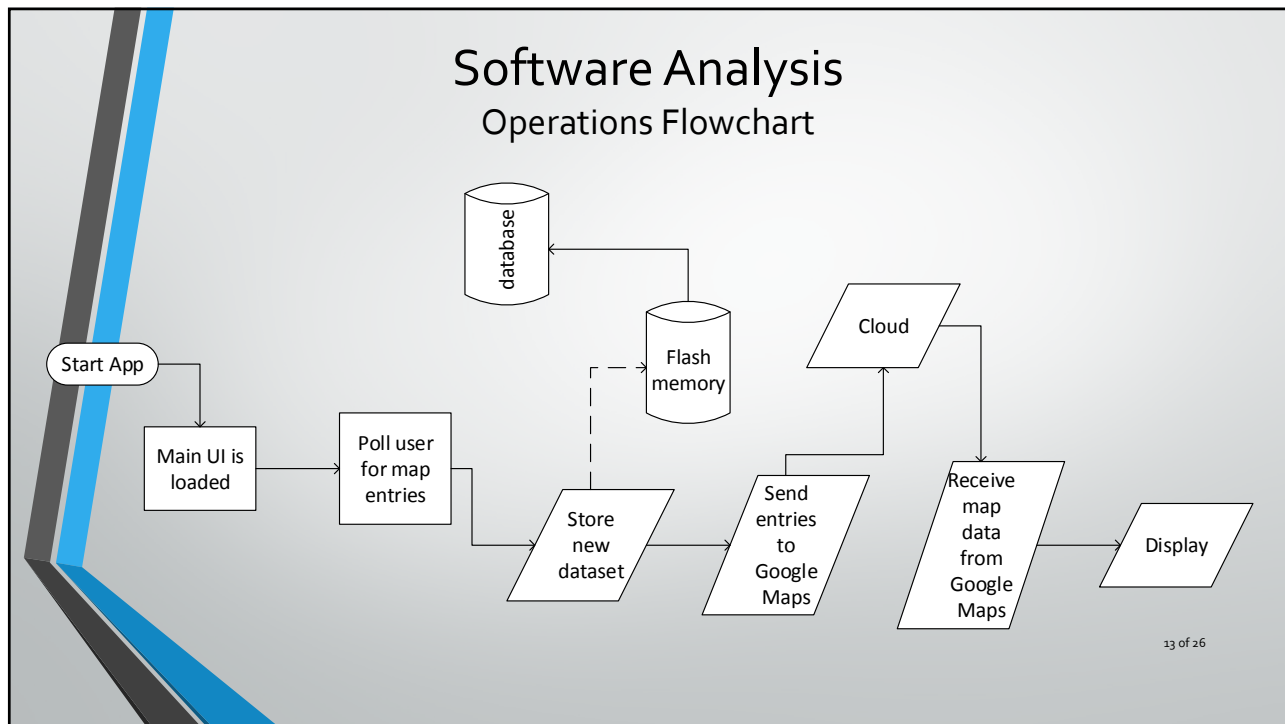
11 of 26

The Android Database Map app is less than 20 MB in size in flash memory

## Software Analysis



12 of 26



## Software Analysis and Design

Database entry screen to add contact

15 of 26

## Software Design

```
if(editName.getText().toString().trim().length()==0 ||
    editStreet.getText().toString().trim().length()==0 ||
    editCity.getText().toString().trim().length()==0 ||
    editState.getText().toString().trim().length()==0)
{
    showMessage("Oops", "Please enter all Required Fields");
    return;
}
```

Error checking for user input in database entry screen

16 of 26



## Software Design

```
public void onCreate(SQLiteDatabase db){
    String CREATE_CONTACT_TABLE = "CREATE TABLE " + TABLE_CONTACT + "(" + COLUMN_PERSONID
    + " INTEGER PRIMARY KEY," + COLUMN_NAME + " TEXT," + COLUMN_SPOUSE +
    " TEXT," + COLUMN_STREET + " TEXT," + COLUMN_CITY + " TEXT," + COLUMN_STATE +
    " TEXT," + COLUMN_ZIP + " INTEGER," + COLUMN_PHONE + " TEXT," + COLUMN_GENDER +
    " TEXT," + COLUMN_CONTACTDATE + " TEXT," + COLUMN_KIDSUNDER18 + " TEXT," +
    COLUMN_OVER18 + " TEXT," + COLUMN_LATITUDE + " REAL," + COLUMN_LONGITUDE +
    " REAL," + COLUMN_RELIGIONID + " INTEGER," + COLUMN_CHURCHID + " INTEGER," +
    " FOREIGN KEY " + "(" + COLUMN_RELIGIONID + ")" + " REFERENCES " +
    "" + TABLE_RELIGION + " " + "(" + COLUMN_RELIGIONID + ")" + " , FOREIGN KEY " +
    + "(" + COLUMN_CHURCHID + ")" + " REFERENCES " + "" + TABLE_CHURCH + " " +
    + "(" + COLUMN_CHURCHID + ")" + " )";

    String CREATE_RELIGION_TABLE = "CREATE TABLE " + TABLE_RELIGION + "(" + COLUMN_RELIGIONID
    + " INTEGER PRIMARY KEY," + COLUMN_RELIGIONNAME + " TEXT,"
    + COLUMN_DENOMINATION + " TEXT" + ")";

    String CREATE_CHURCH_TABLE = "CREATE TABLE " + TABLE_CHURCH + "(" + COLUMN_CHURCHID
    + " INTEGER PRIMARY KEY," + COLUMN_CHURCHNAME + " TEXT," + COLUMN_RELIGIONID +
    " INTEGER," + " FOREIGN KEY (" + COLUMN_RELIGIONID + ") REFERENCES " +
    "" + TABLE_RELIGION + " " + "(" + COLUMN_RELIGIONID + ")" + " )";

    db.execSQL(CREATE_RELIGION_TABLE);
    db.execSQL(CREATE_CHURCH_TABLE);
    db.execSQL(CREATE_CONTACT_TABLE);
}
```

Source code to create SQLite tables

17 of 26

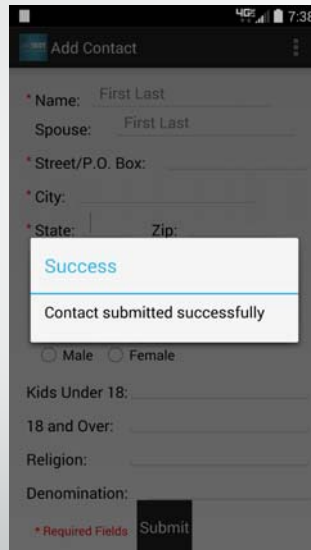
## Software Integration Design and Testing

The screenshot shows a mobile application interface with a form titled "Database". The form contains several input fields, some of which are marked as required with a red asterisk. The fields are: Name, Spouse, Street/P.O. Box, City, State, Zip, Phone, Church, Gender (with radio buttons for Male and Female), Kids Under 18, and Kids Over 18. The State field has an autocomplete dropdown menu open, showing a list of state abbreviations: ID, IL, IN, and IA. At the bottom of the form, there is a "Submit" button and a "New Text" label.

Autocomplete textview to display State abbreviations

18 of 26

## Software Integration Design and Testing

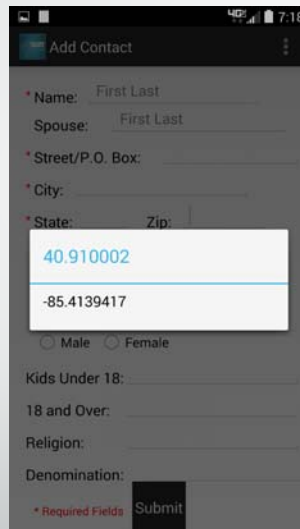


The screenshot shows a mobile application interface for adding a contact. A white dialog box with a blue border is centered on the screen, displaying the word "Success" in blue text, followed by a horizontal line and the message "Contact submitted successfully" in black text. Below the dialog box, the "Add Contact" form is visible, with fields for Name, Spouse, Street/P.O. Box, City, State, Zip, Gender (Male/Female), Kids Under 18, 18 and Over, Religion, and Denomination. A "Submit" button is at the bottom right of the form. The status bar at the top of the phone screen shows the time as 7:38.

19 of 26

Contact was added to database and completed successfully

## Software Integration Design and Testing

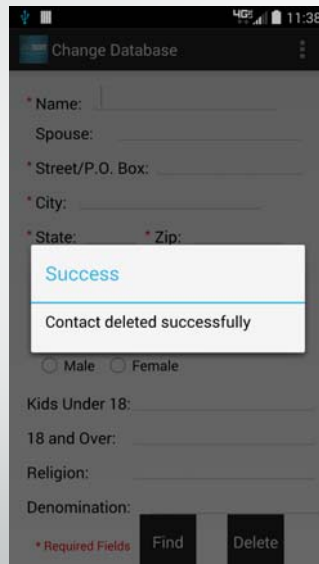


The screenshot shows the same mobile application interface as the previous slide. A white dialog box with a blue border is centered on the screen, displaying the coordinates "40.910002" in blue text, followed by a horizontal line and "-85.4139417" in black text. The "Add Contact" form is visible in the background, with the same fields and "Submit" button as in the previous slide. The status bar at the top of the phone screen shows the time as 7:18.

20 of 26

Geocoder Latitude/Longitude coordinates returned during testing

## Software Integration Design and Testing



Change Database

\* Name: \_\_\_\_\_

Spouse: \_\_\_\_\_

\* Street/P.O. Box: \_\_\_\_\_

\* City: \_\_\_\_\_

\* State: \_\_\_\_\_ \* Zip: \_\_\_\_\_

Success

Contact deleted successfully

☐ Male ☐ Female

Kids Under 18: \_\_\_\_\_

18 and Over: \_\_\_\_\_

Religion: \_\_\_\_\_

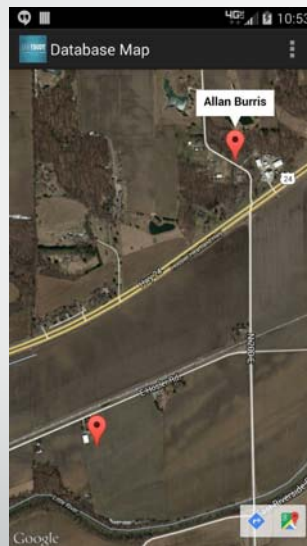
Denomination: \_\_\_\_\_

\* Required Fields Find Delete

21 of 26

Contact deleted from database

## Software Integration Design and Testing



22 of 26

Markers on Google Maps window with Contact Name

## Software Integration Design and Testing



23 of 26

Allan Burris contact removed from database and only showing other contact

## Conclusion

- Deliver the app to OneBody
- Thoughts on Android development
- Other Useful Applications
- Lessons Learned
  - **Time Management**

24 of 26

