Group 15

Arian Caraballo, CpE

Daniela Zicavo, CpE

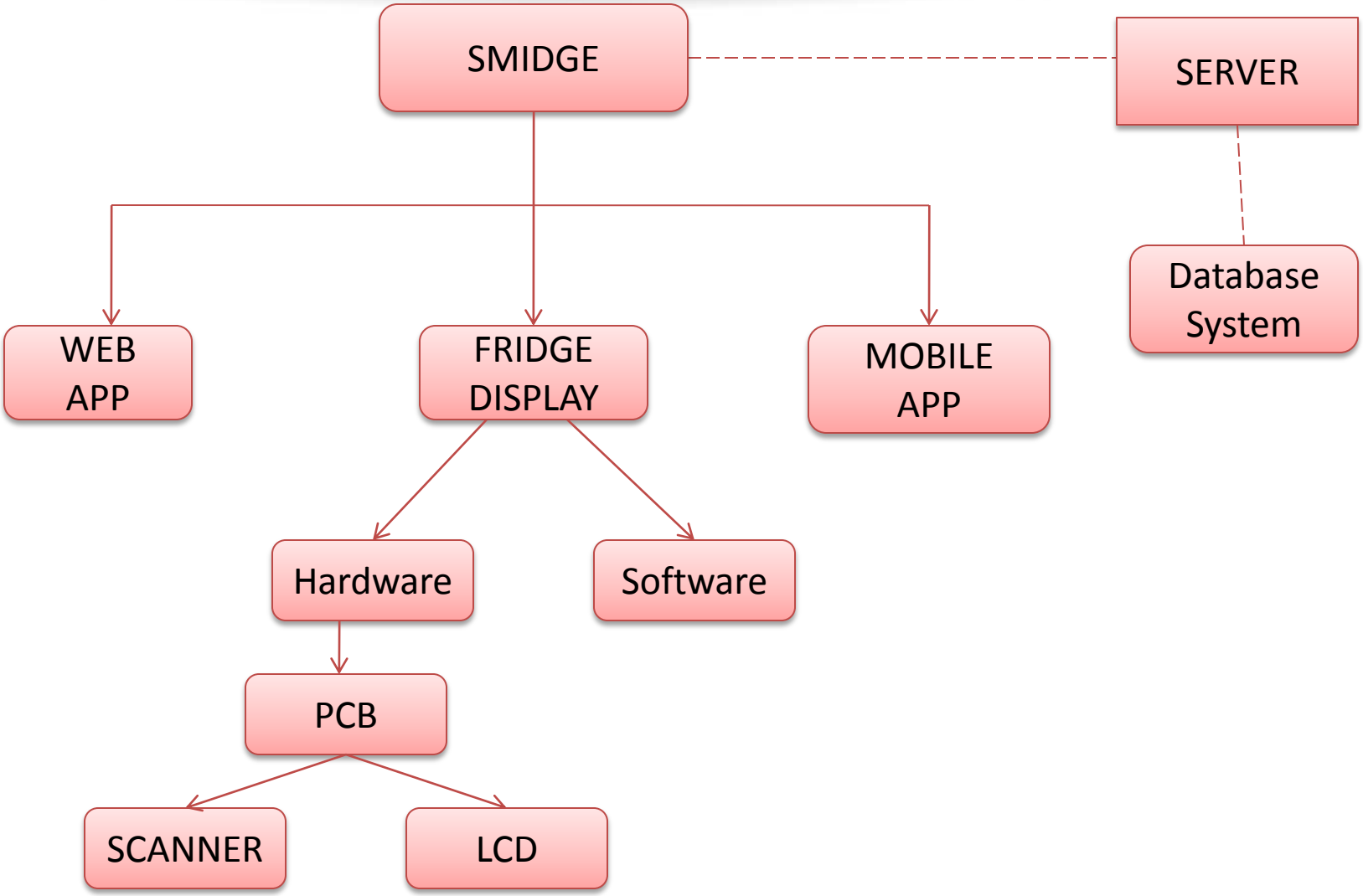
Felipe Bernal, CpE

Isabel Virag, CpE

What is it?

Smidge stands for Smart Fridge System. It will offer the user the ability to store their groceries inventory in order to improve their shopping experience through automated shopping lists and recipe lookup.

Project Block Diagram



Specifications

- Processor minimum speed: 667MHz
- Pop Memory:
  - -SDRAM > 1 GB
  - -NAND > 1 GB
- - PCB board < 4 inches^2
- - 5v power source
- Able to decode:
  - 12 digits for a Type-A UPC code
  - 8 digits for a Type-E UPC code

System Requirements

- Access via a website, a phone application or the fridge system.
- The fridge client shall have wireless internet connection capabilities
- Synchronize local databases periodically with the remote server.
- The fridge client shall be able to scan a UPC code and find a matching item with an implemented API.
- Handle multiple accounts, each with its own inventory and preferences.
- Ability to view, add, modify and delete items, shopping lists and recipes.

Fridge Client
- Hardware Components -

- PCB with processor powerful enough to run Android
- Wireless Internet connection
- Touchscreen LCD
- UPC scanner
- USB hub to handle all peripherals:

  All components powered by a single connector

Internet Connection

- Internet connection is needed to keep databases in sync.
- Wireless network connection allows for better system integration.
- Primary Concern:

  Communication to Android OS

  -Custom Drivers needed
- Belkin Wireless G USB Network Adapter F5D7050

TouchScreen LCD
- Imo Pivot Touch -

- USB powered
- Relatively low cost: $279
- 7” display
- Resistive touch-screen
- Resolution:

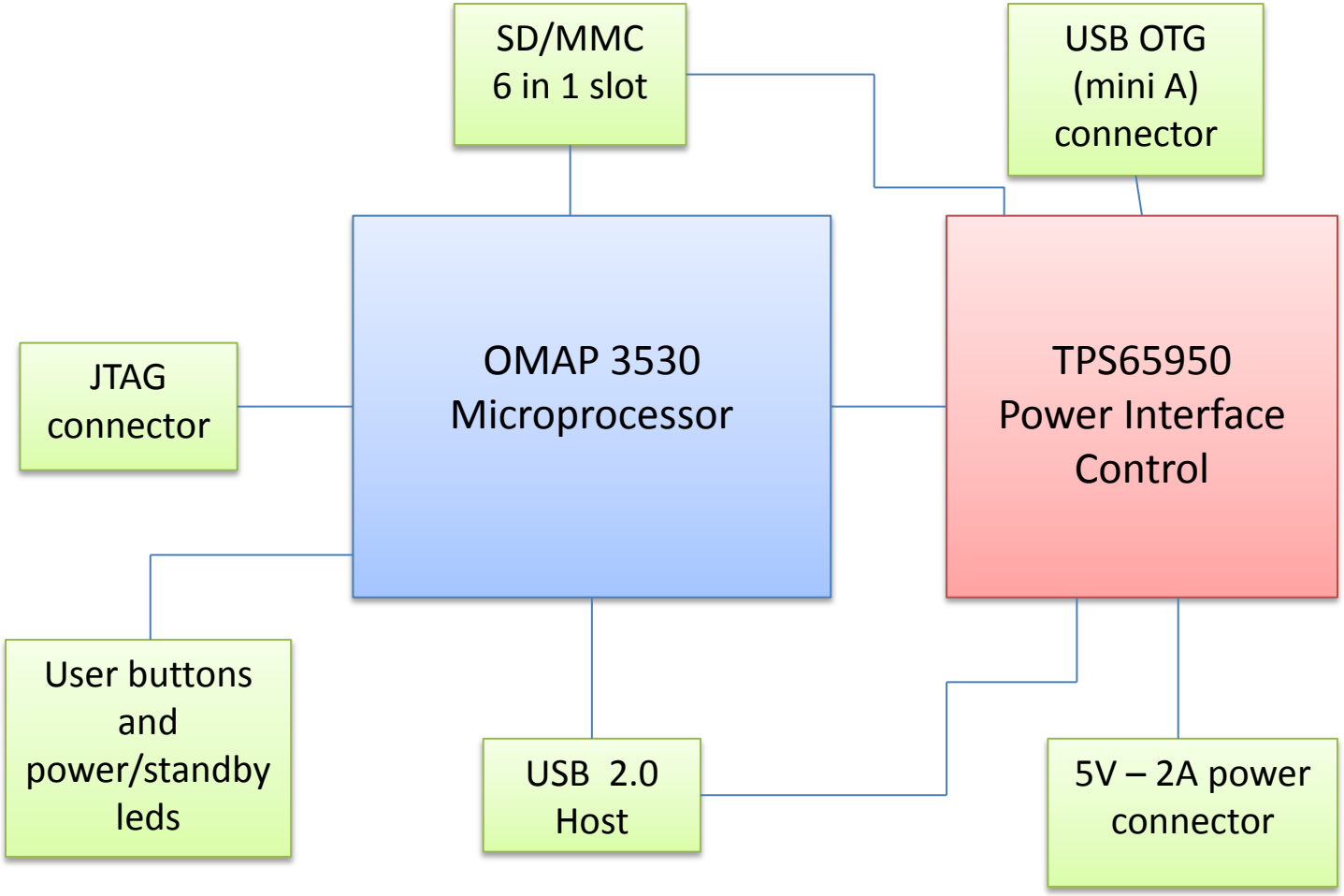
  800 x 480
- Current consumption:

  100-240 V
- Custom drivers built to interface with Android OS

UPC Scanner
- Metrologic ScanGlove IS4225 -

- Optimum shape and size
- Barcode is read as keyboard input → No drivers required
- USB powered
- UPC and full ASCII supported
- Infrared sensor for standby mode
- LED status indicator
- Low-cost: $35

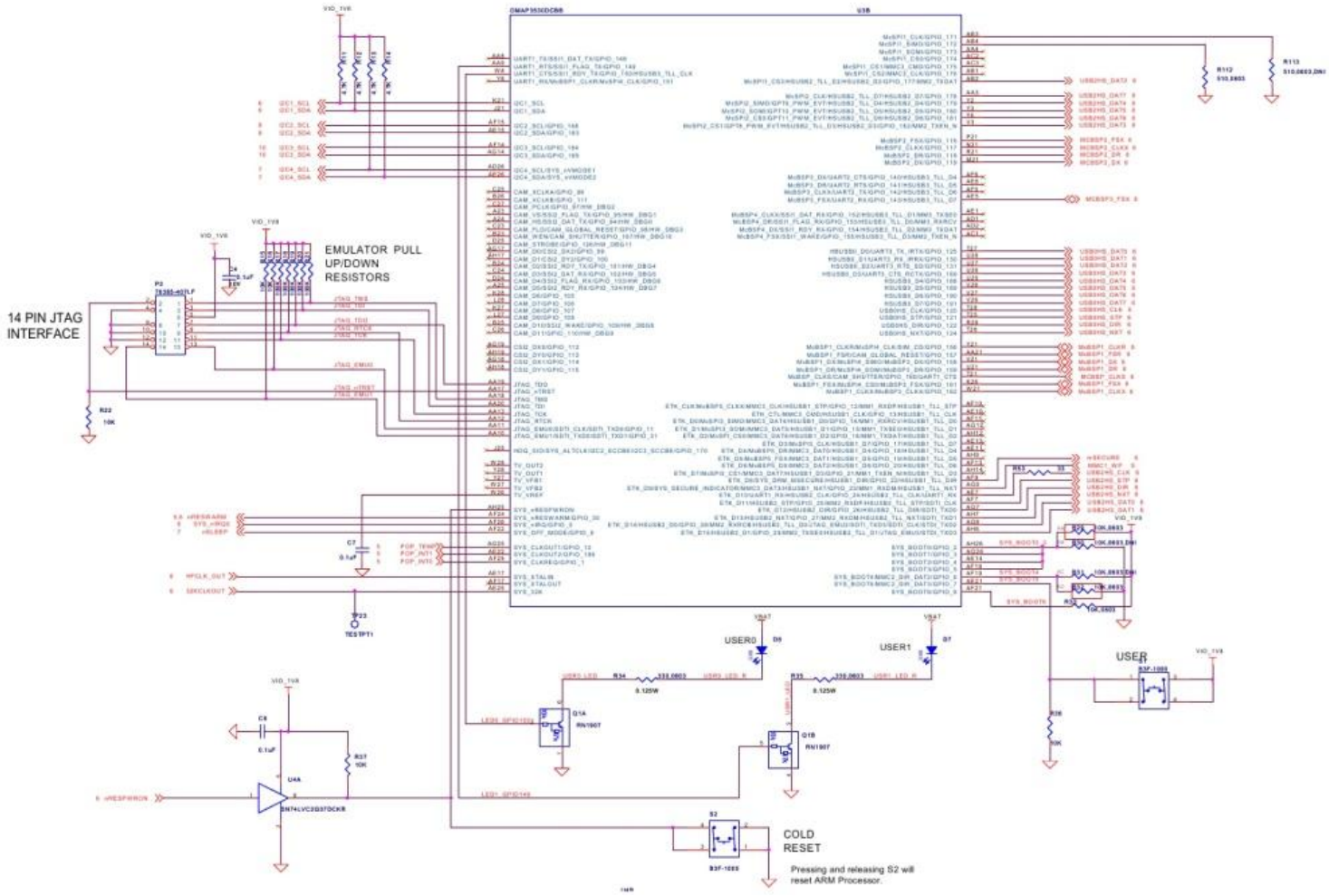
PCB block diagram



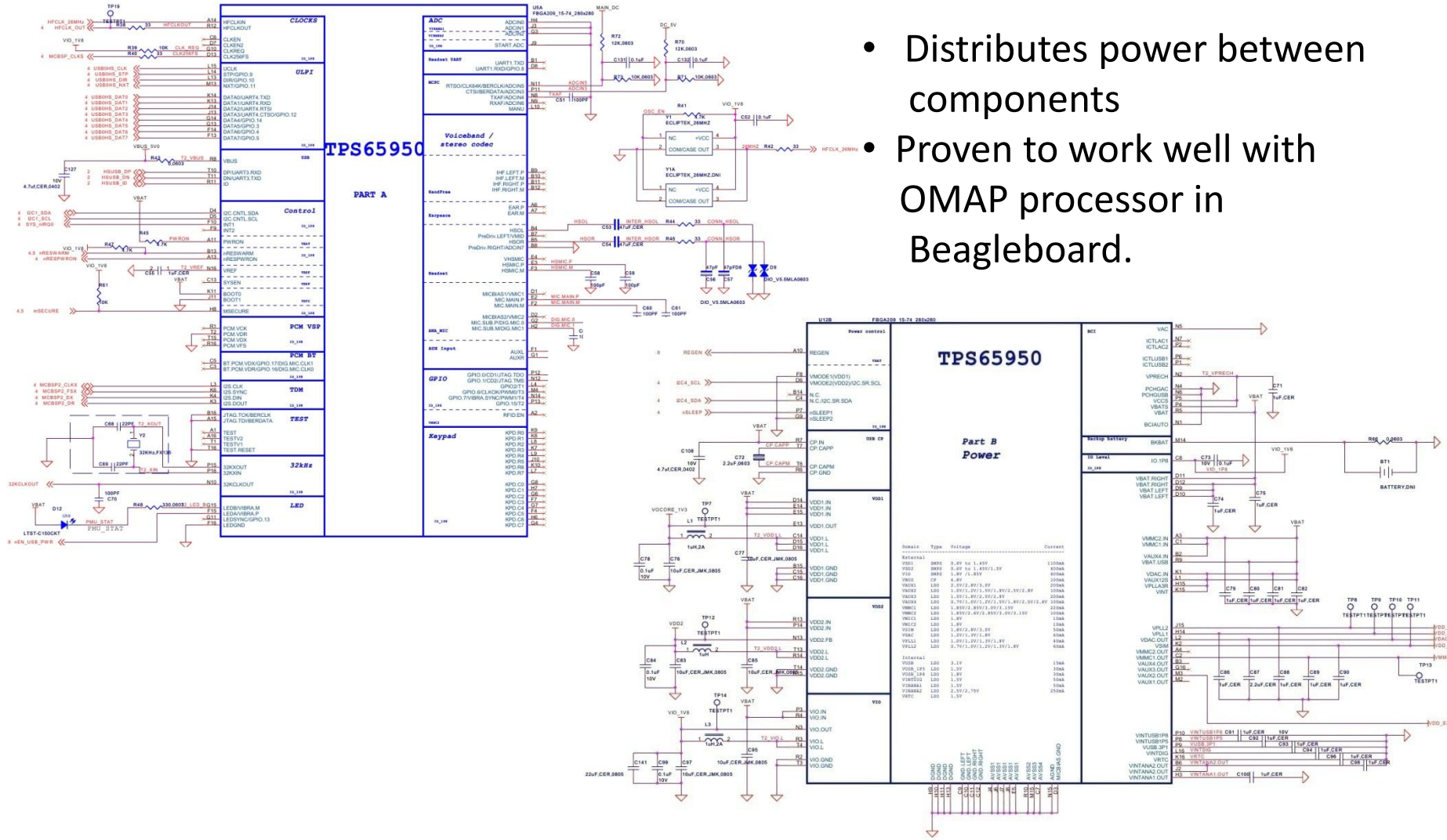
Processor: OMAP3530

- Low power
- ARM-based processor
- High speed GPU
- IC, POP Memory Flash/SDRAM
  - 2GB NAND x 16 (256MB) and 2GB MDDR SDRAM
- Proven capability of running Android OS
- Jtag connector straight to processor for development purposes
- Reset and User/boot button
- Four LEDs for booting and software testing.

OMAP3530 Microprocessor



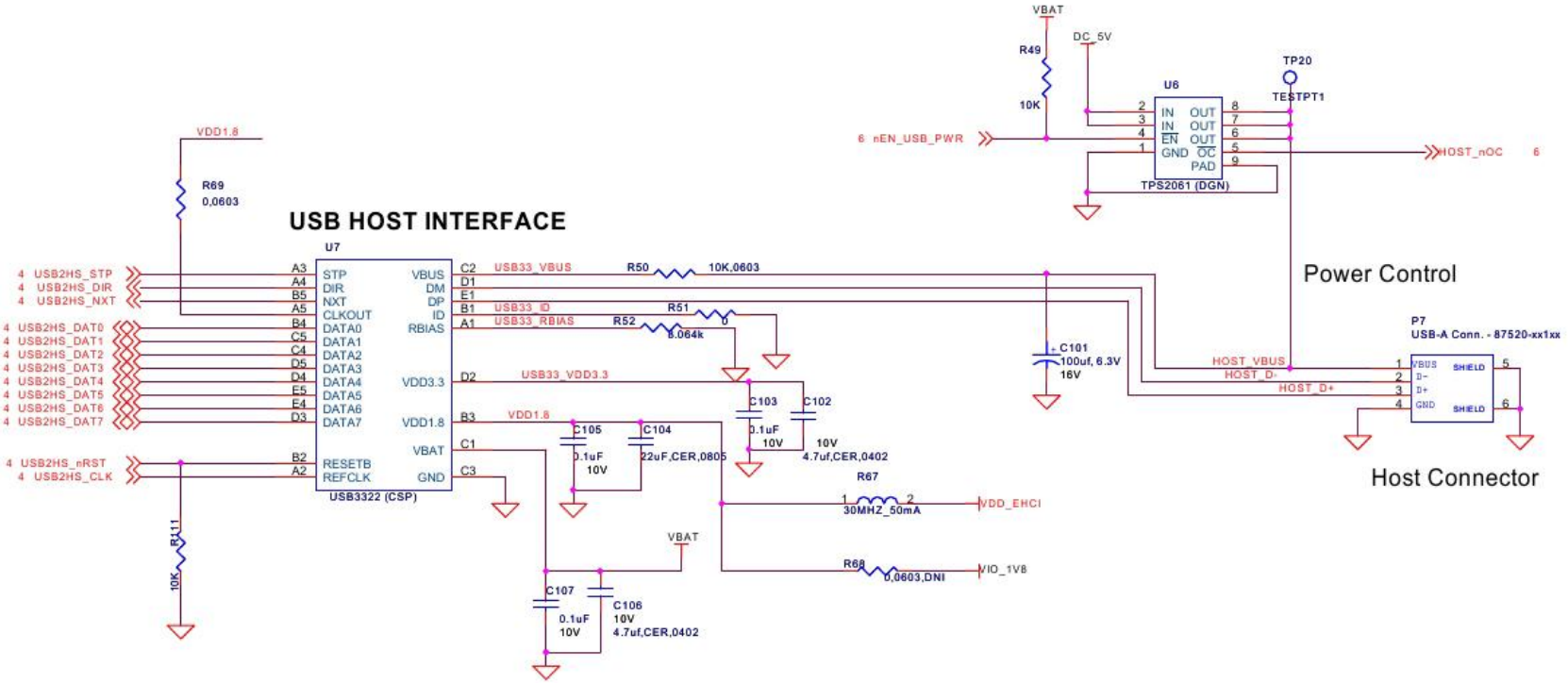
TPS65950 Power Control



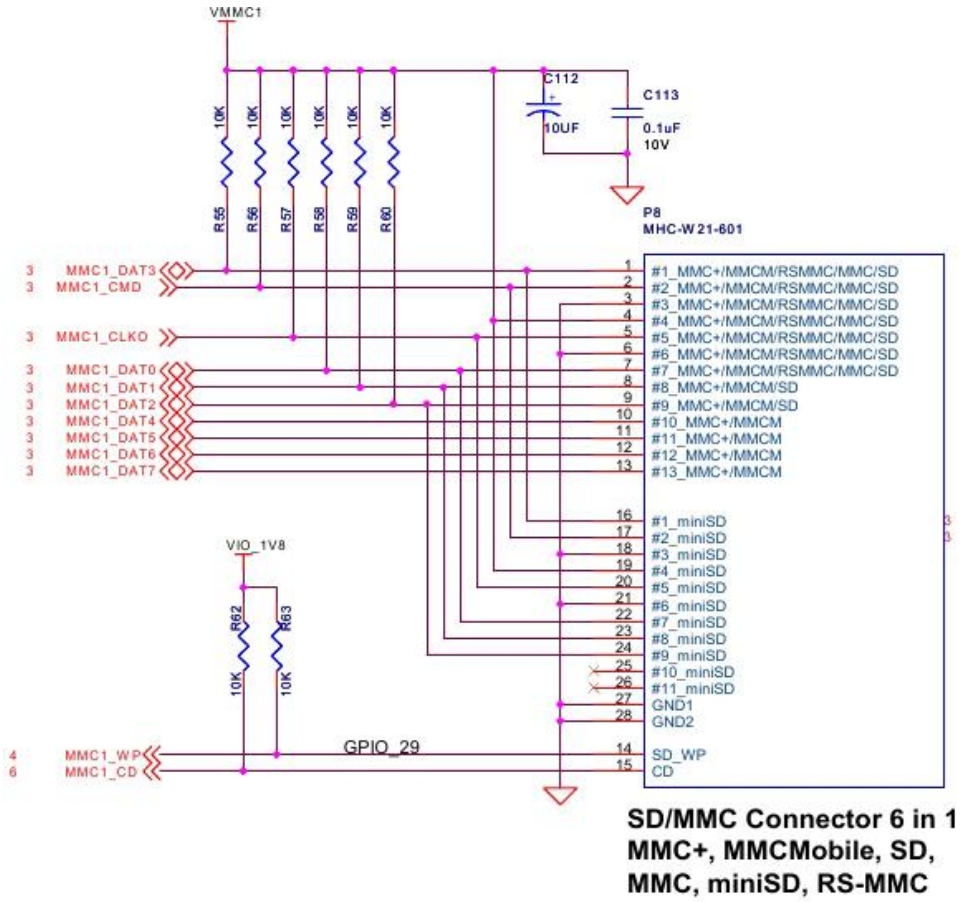
- Distributes power between components
- Proven to work well with OMAP processor in Beagleboard.

USB OTG port

USB Host will be used for developing purposes to mount the bootloader, OS and application onto the board.



MMC/SD Slot 6-in-1 (MHC-W21-601)



- An SD/MMC 6 in 1 connector is needed to store Android's file system
- Allows for memory expansion
- It supports:

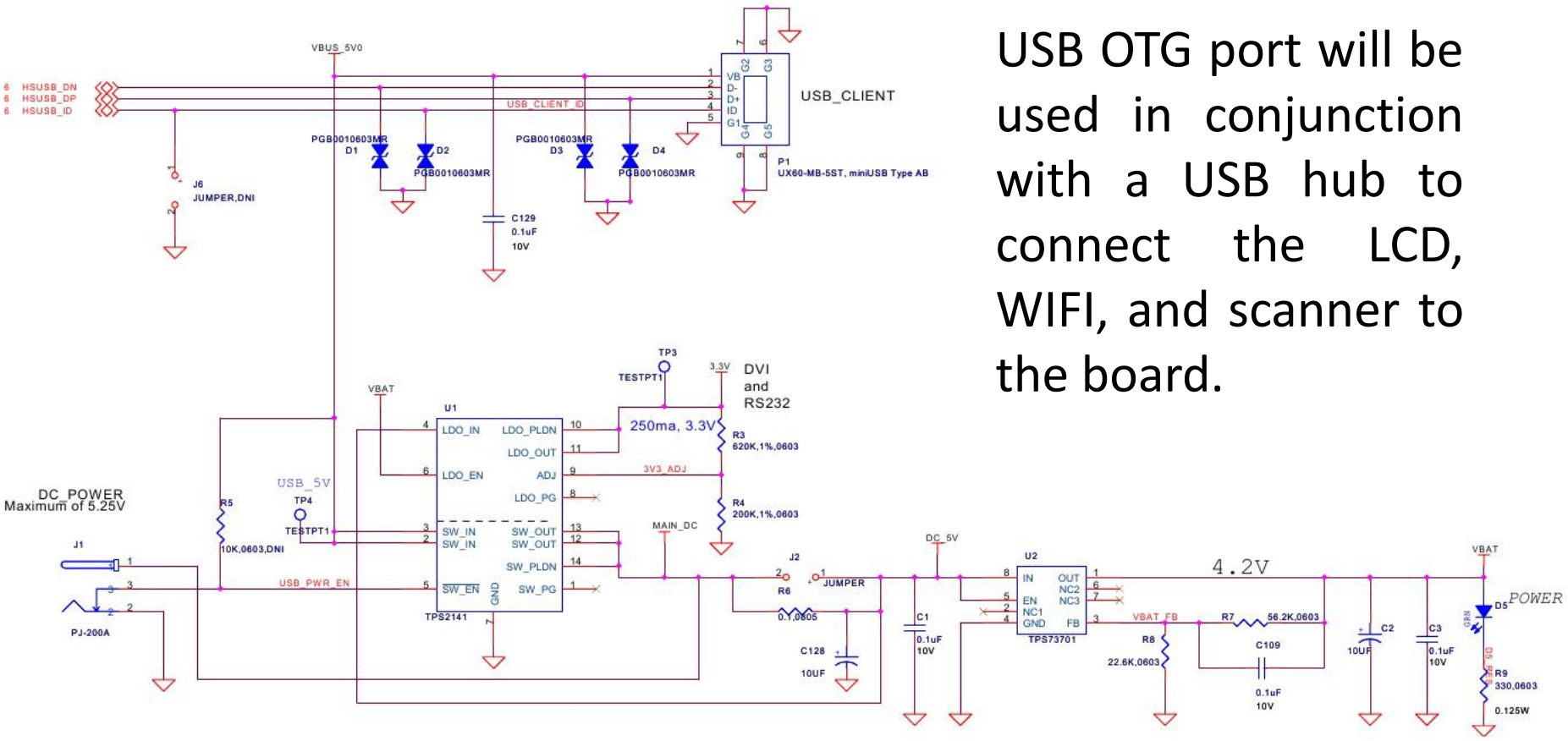
SD Memory Cards

MMC Memory Card SDIO Cards MMCMobile Cards

RS-MMC Cards

miniSD Cards

USB client and 5V power connector

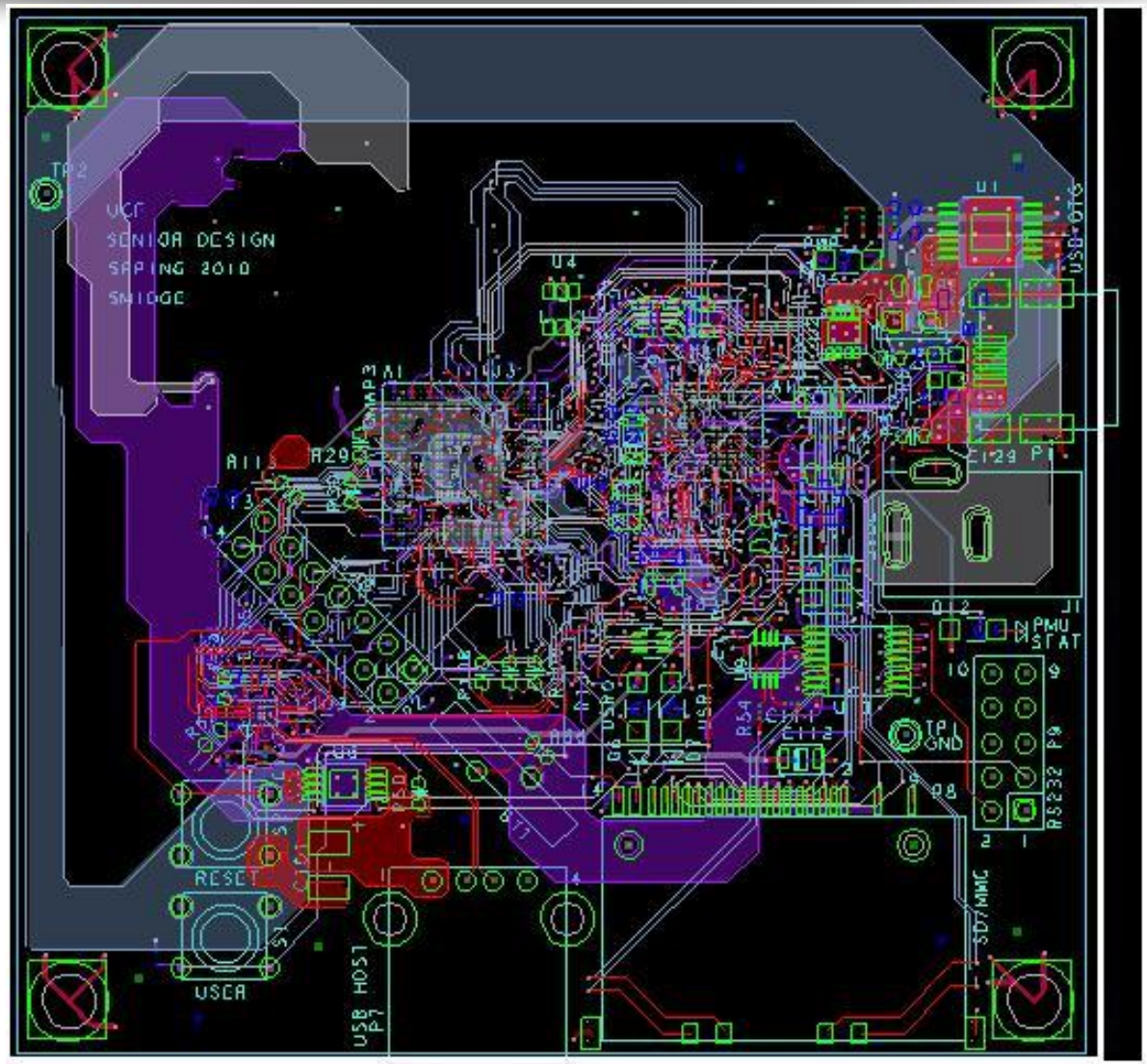


USB OTG port will be used in conjunction with a USB hub to connect the LCD, WIFI, and scanner to the board.

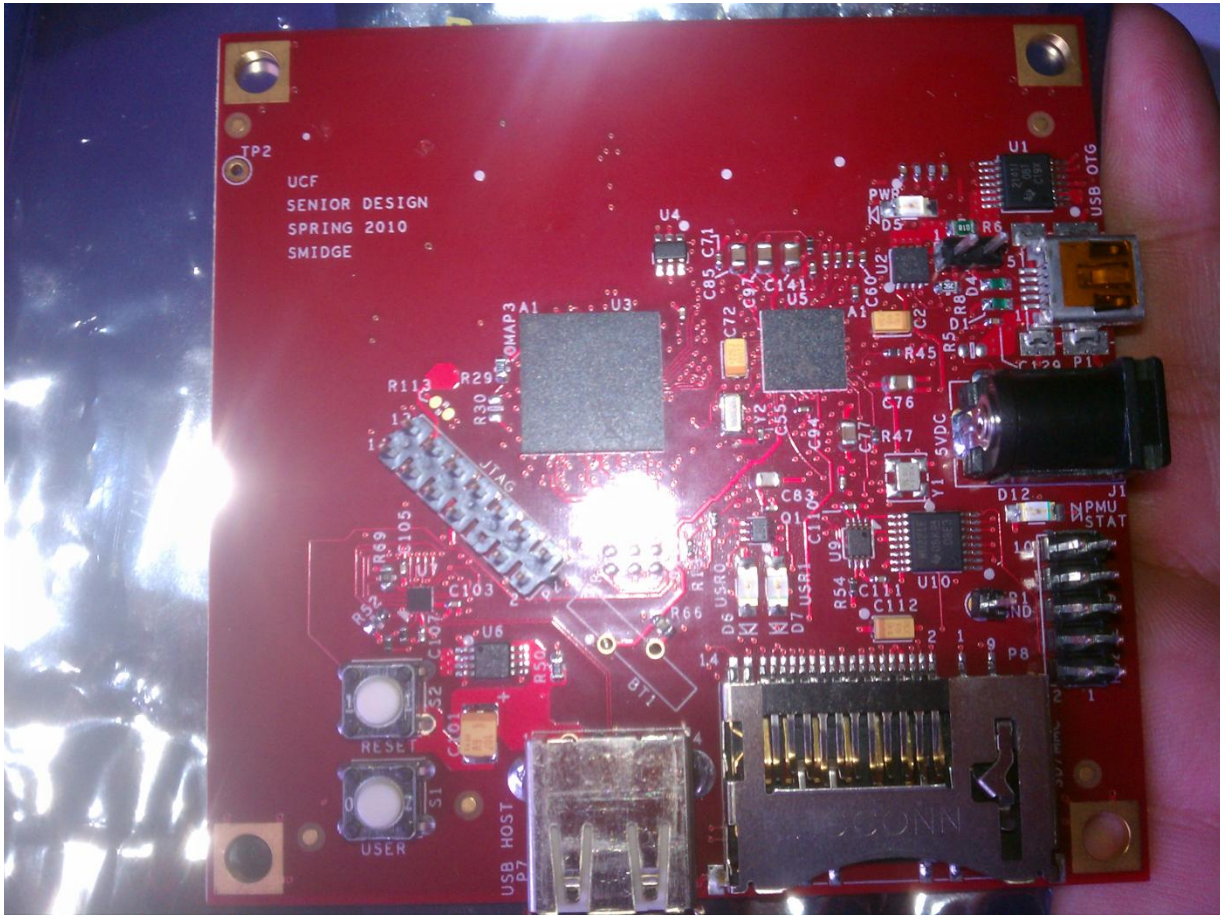
PCB implementation

- The PCB will serve the fridge client
- We generated schematics in OrCad and transferred them to Allegro PCB layout in order to manufacture our own custom PCBs with the following target specifications:
  - 6 layer board
  - 3”x3.1” or smaller outside dimensions
  - Top and bottom side component placement

Allegro PCB Design



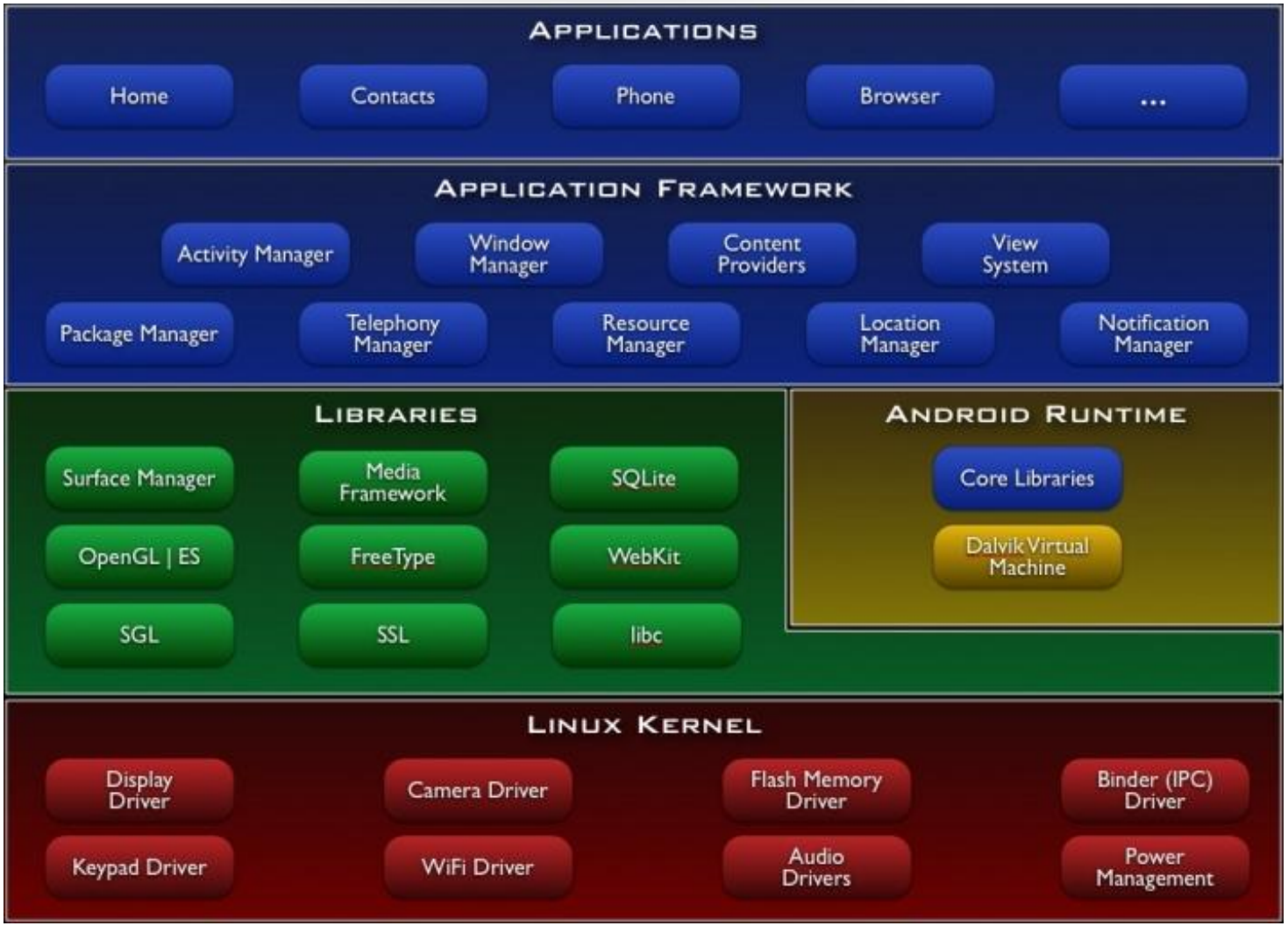
PCB


UCF
SENIOR DESIGN
SPRING 2010
SMIDGE
OMAP3
JTAG
RESET
USER
USB HOST
USB OTG
PWR
5VDC
PMU
STAT
USR0
USR1

PCB Manufacturer Choice

- CircuitCo Electronics
  - Experience with the complex processes involved in manufacturing boards with TI processors
    - Quick turn-around time
    - Small in size
    - large amount of components → high quality board
    - Very competitive price for both manufacturing the board and mounting the parts

Building Android



Testing board functionality

- 1st Step:
  - Test board functionality using prebuilt images for Kernel Android 2.2 from 'rowboat' project: Enables Android for Texas Instruments devices
  - Prepare SD Card:
    - Three Partitions:
      - Boot Loader: Uimage, and Linux Kernel 2.6.32
      - Android partition
      - Data partition
  - Plug Rs232 cable – To see kernel console
  - Plug USB host – To see android console (debugging)

All components in our PCB worked 100%

- successfully booted Linux and Android -

Custom Kernel

- Custom Kernel build was necessary to include:
  - LCD drivers →

    DisplayLink drivers: enables USB output for LCD

  - Egalax Touchscreen drivers
  - SGX driver →
    - Graphics Acceleration from Android
      - Build communication bridge with DisplayLink
  - Belkin Wireless drivers
    - Drivers available for Linux
      - Build communication bridge for Android

Challenges

- Last Step:
  - Touchscreen calibration:
    - Modify absolute X and Y values to fit the touchscreen in the kernel drivers built.
- Challenges:
  - DisplayLink drivers had **never** been built for Android 2.2 - successful
  - SGX drivers communication with DisplayLink - successful
  - Belkin Wireless drivers available for Linux distributions but not for Android on OMAP processor. - unsuccessful
    - Proved to be the most challenging task
      - Complete custom Firmware needed to communicate with Android

Linux Kernel

```
mmc1 is available
reading uImage

2803444 bytes read
## Booting kernel from Legacy Image at 84000000 ...
   Image Name:   Linux-2.6.32
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2803380 Bytes = 2.7 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux.....................................................................
..................................................... done, booting the kernel.
Linux version 2.6.32 (velazcod@velazcod-desktop) (gcc version 4.4.0 (GCC) ) #74 Wed Apr 20 06:00:22 EDT 2011
CPU: ARMv7 Processor [411fc083] revision 3 (ARMv7), cr=10c53c7f
CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
```

USB Touchscreen Driver

```
mice: PS/2 mouse device common for all mice
input: gpio-keys as /devices/platform/gpio-keys/input/input0
usbcore: registered new interface driver usbtouchscreen
i2c /dev entries driver
Linux video capture interface: v2.00
omap-iommu omap-iommu.0: isp: version 1.1
OMAP Watchdog Timer Rev 0x31: initial timeout 60 sec
mmci-omap-hs mmci-omap-hs.1: err -16 configuring card detect
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
logger: created 64K log 'log_main'
logger: created 256K log 'log_events'
logger: created 64K log 'log_radio'
logger: created 64K log 'log_system'
usbcore: registered new interface driver udlfb
```

Device Detection, Wireless

```
Waiting for root device /dev/mmcblk0p2...
hub 1-2:1.0: USB hub found
hub 1-2:1.0: 4 ports detected
mmc0: new high speed SDHC card at address 0007
usb 1-2.1: new low speed USB device using ehci-omap and address 3
mmcblk0: mmc0:0007 SD04G 3.68 GiB
 mmcblk0: p1 p2 p3
input: Plus More Enterprise LTD. USB-compliant keyboard as /devices/platform/ehci-omap.0/usb1/1-2/1-2.1/1-2.1:1.0/input/input1
generic-usb 0003:0518:0001.0001: input: USB HID v1.10 Keyboard [Plus More Enterprise LTD. USB-compliant keyboard] on usb-ehci-omap.0-2.1/i
nput0
input: Plus More Enterprise LTD. USB-compliant keyboard as /devices/platform/ehci-omap.0/usb1/1-2/1-2.1/1-2.1:1.1/input/input2
generic-usb 0003:0518:0001.0002: input: USB HID v1.10 Mouse [Plus More Enterprise LTD. USB-compliant keyboard] on usb-ehci-omap.0-2.1/inpu
t1
usb 1-2.3: new high speed USB device using ehci-omap and address 4
hub 1-2.3:1.0: USB hub found
hub 1-2.3:1.0: 4 ports detected
usb 1-2.3.2: new high speed USB device using ehci-omap and address 5
phy0 -> rt2x00_print_chip: Info - Chipset detected - rt: 1300, rf: 0002, rev: 0002573a.
usb 1-2.3.3: new high speed USB device using ehci-omap and address 6
hub 1-2.3.3:1.0: USB hub found
hub 1-2.3.3:1.0: 4 ports detected
usb 1-2.3.3.2: new high speed USB device using ehci-omap and address 7
DisplayLink device attached
ret control msg 0: 4 1400f0
EDID XRES 800 YRES 480
INIT VIDEO 0 800 480
ret control msg 1 (STD_CHANNEL): 16
ret bulk 2: 156 156
ret bulk 3: 0
found valid mode...25000
screen base allocated !!!
colormap allocated
Console: switching to colour frame buffer device 100x30
usb 1-2.3.3.3: new full speed USB device using ehci-omap and address 8
input: eGalax Inc. USB TouchController as /devices/platform/ehci-omap.0/usb1/1-2/1-2.3/1-2.3.3/1-2.3.3.3/1-2.3.3.3:1.0/input/input3
```

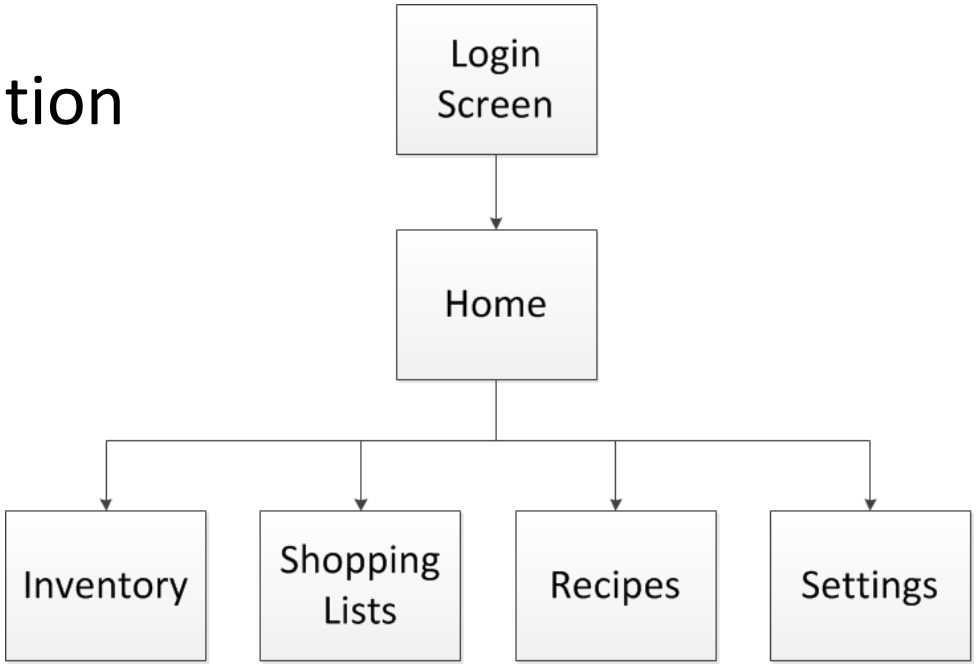
Website

- PHP
  - dynamic and interactive Web pages.
- AJAX
  - exchange data with a server.
- JavaScript (jQuery)
  - validate forms, communicate with the server.

Website

User sets an account if it does not have one

User logs in the application



Application Menu:

- Inventory
- Shopping Lists
- Recipes
- Users (this option will only show if the user is an administrator)
- Settings

User information

- Inventory
  - User items
- Settings
  - General
  - Your profile
- Users (only available for the administrator)
  - Users status

Shopping List

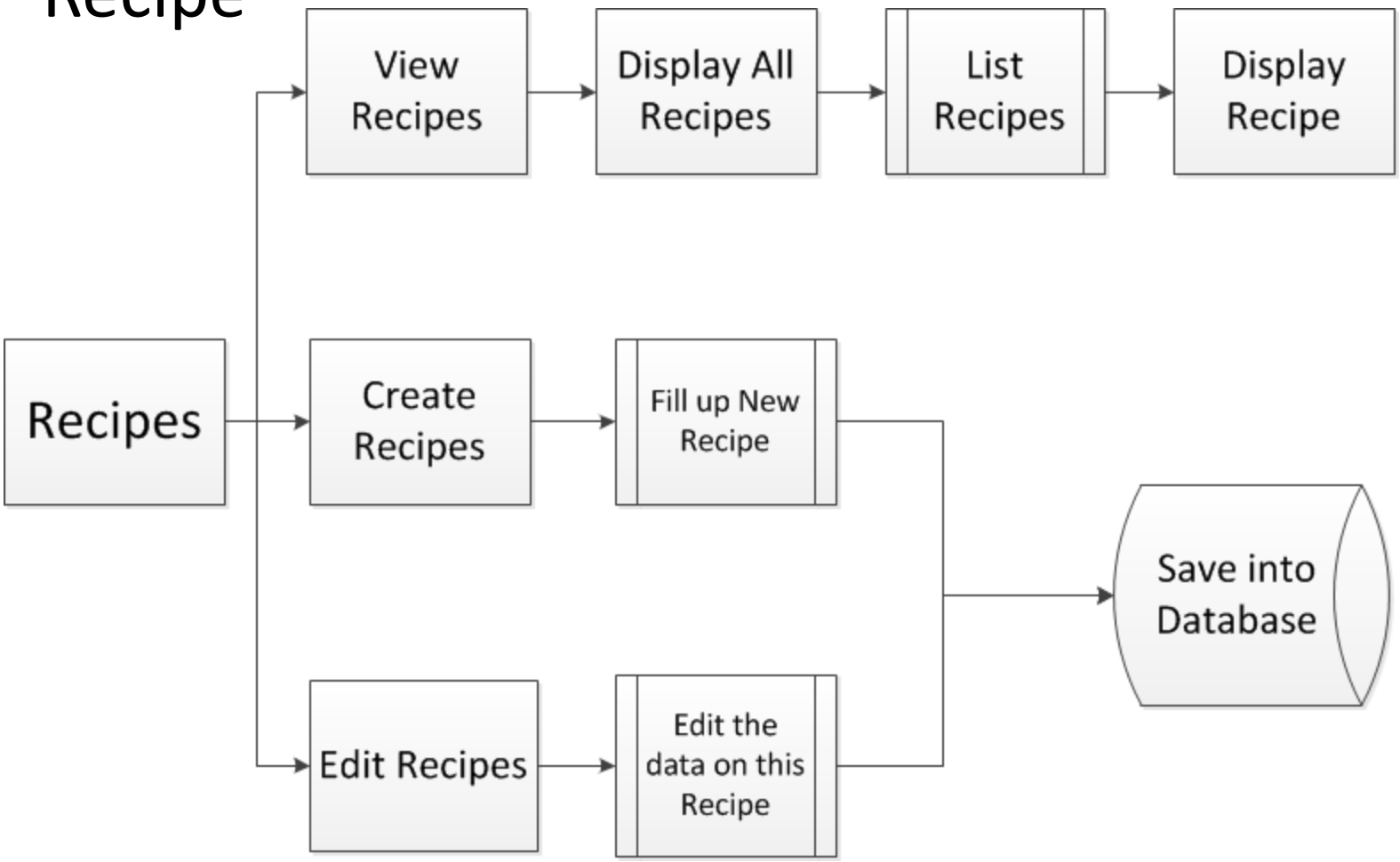
- Search for existing shopping lists
- Create new shopping lists
- Modified existing shopping lists

Recipes

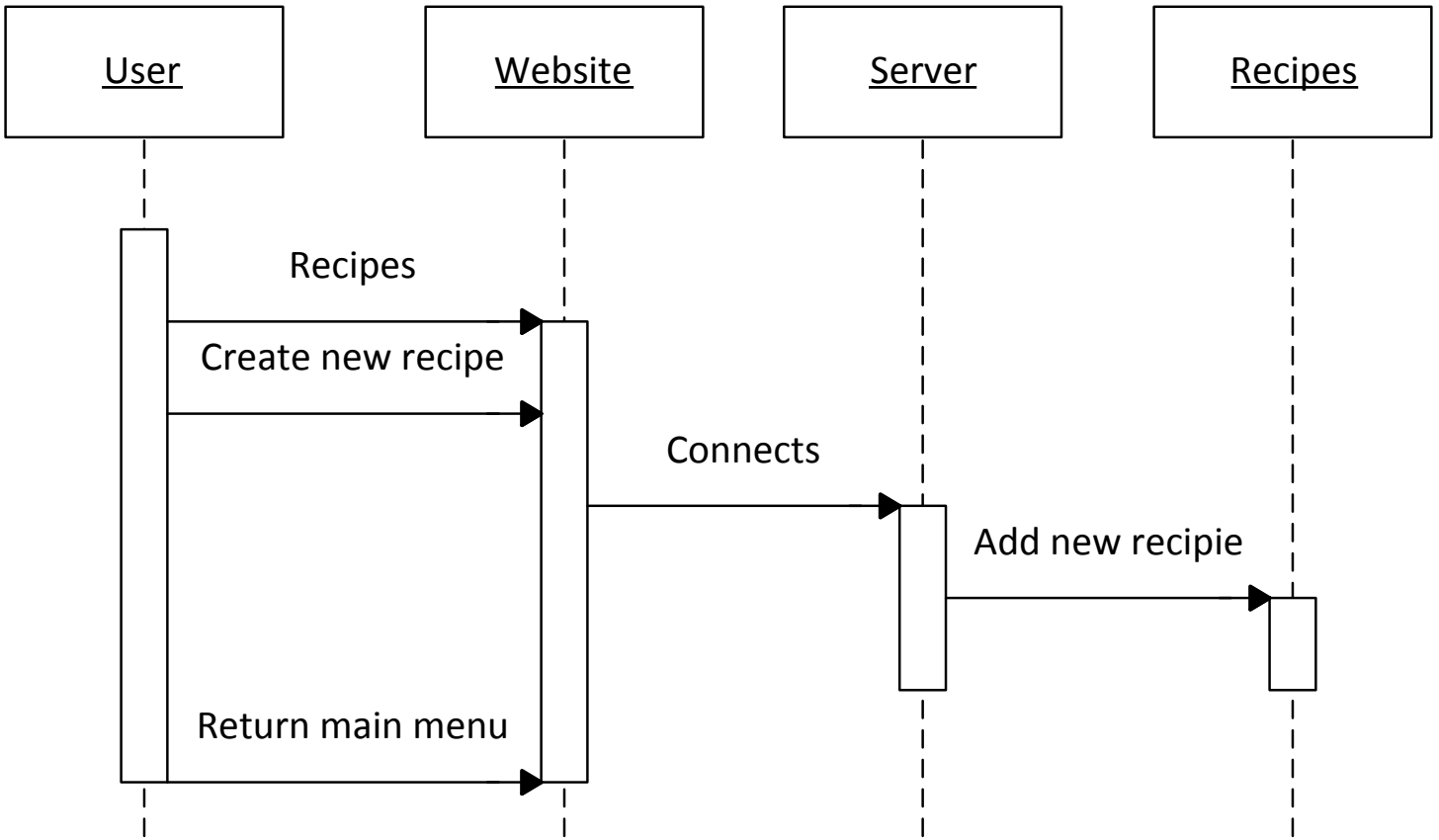
- Search Recipe
  - Database of created recipes.
  - Basic Search
  - Advance Search
- Recipe
  - Database of user created recipes.
  - Option to edit saved recipes.

Website
-Block Diagram-

- Recipe



Website
-Sequence Diagram-



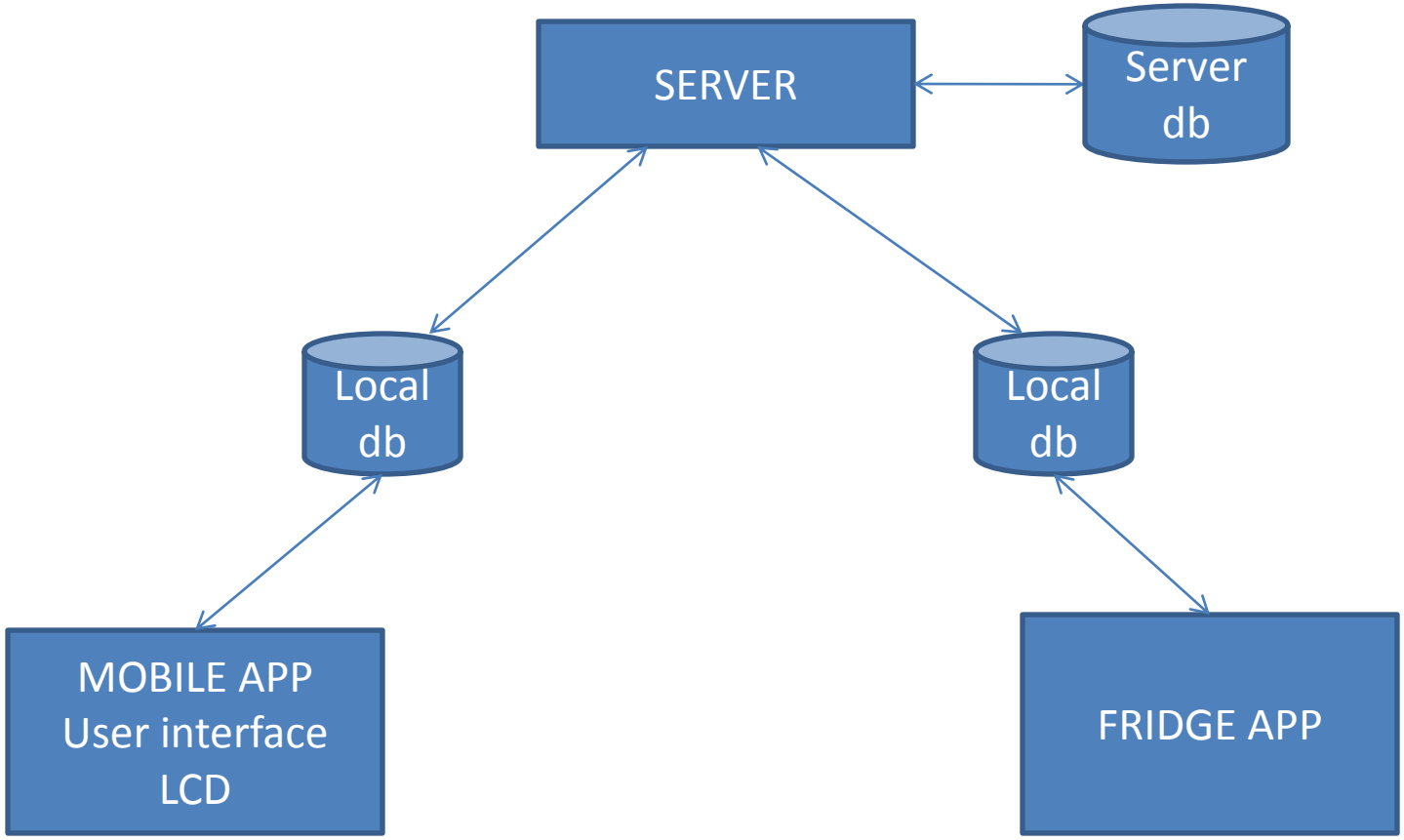
Server

- Apache HTTP Server
- MYSQL
- PHPMyAdmin

PHP Classes

- class.recipe.php
- class.inventory.php
- class.items.php
- class.login.php
- class.recipe_ingredients.php
- class.recipex.php
- class.shoppinglist.php
- class.user.php
- class.mysql.php

Server
-Structure-

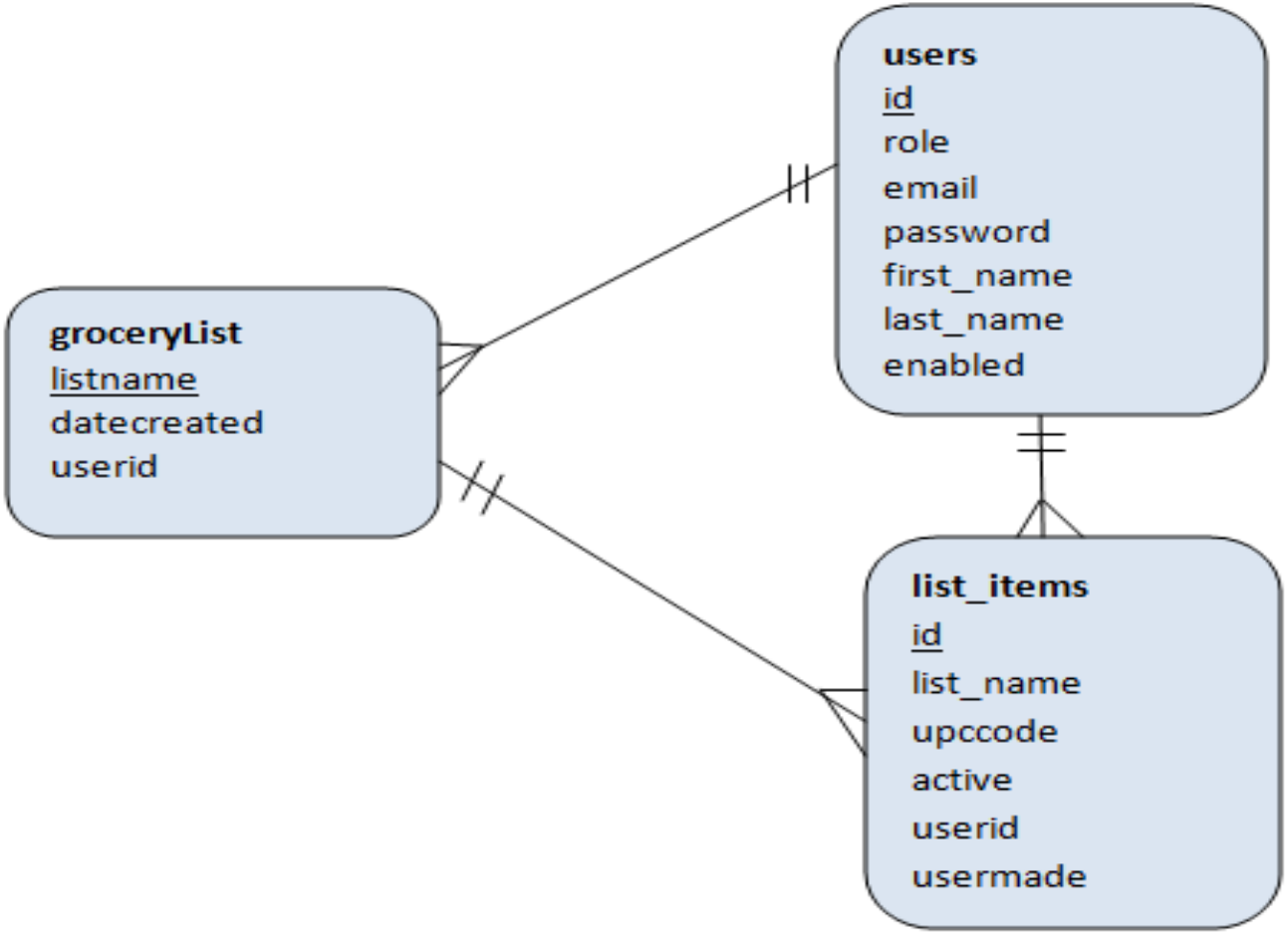


Server
-Database Structure-

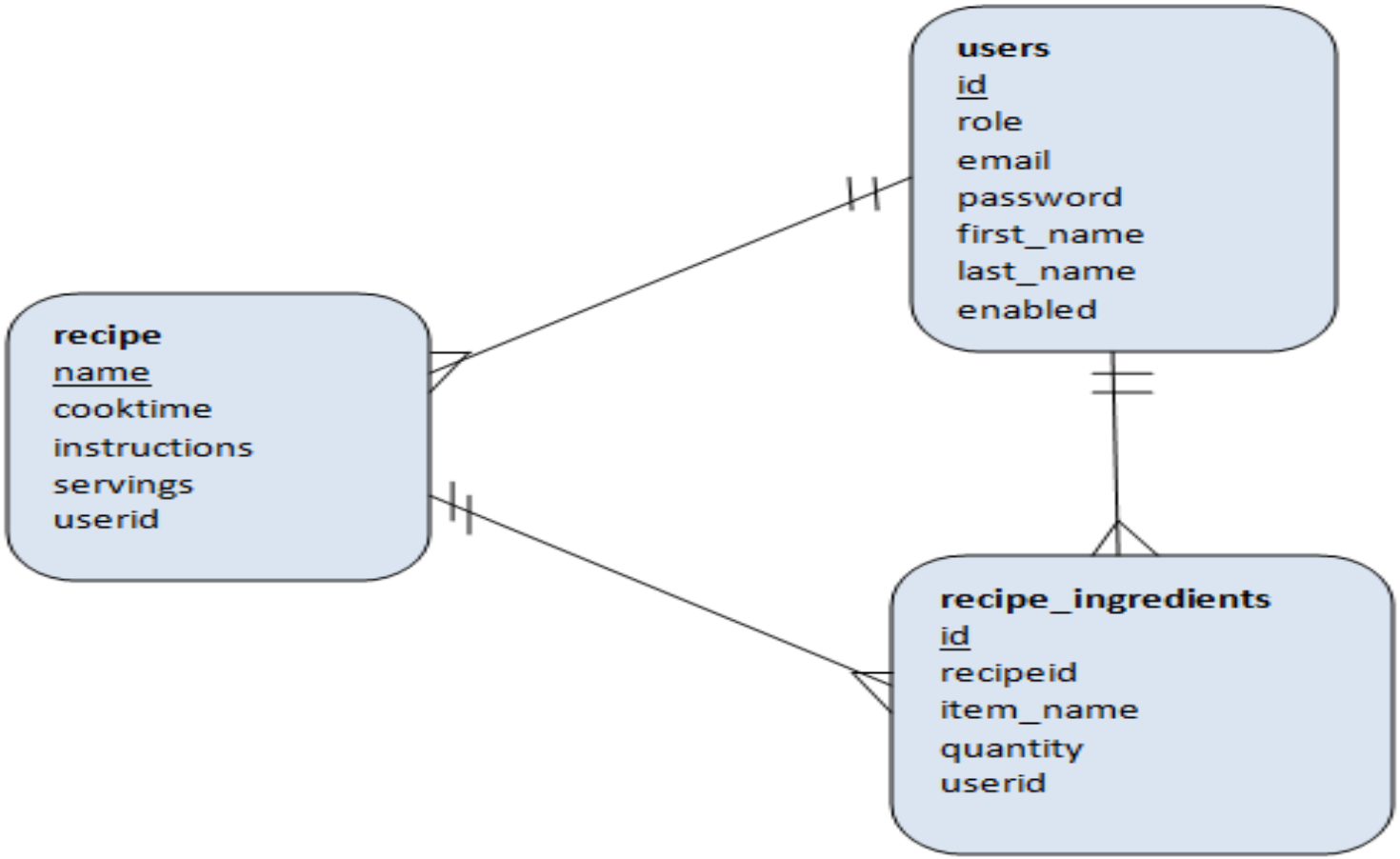
TABLES

- GroceryList
- HealthyTips
- Inventory
- Items
- ListItems
- Recipe
- RecipeX
- RecipeIngredients
- Users
- UsersLog
- UsersRoles

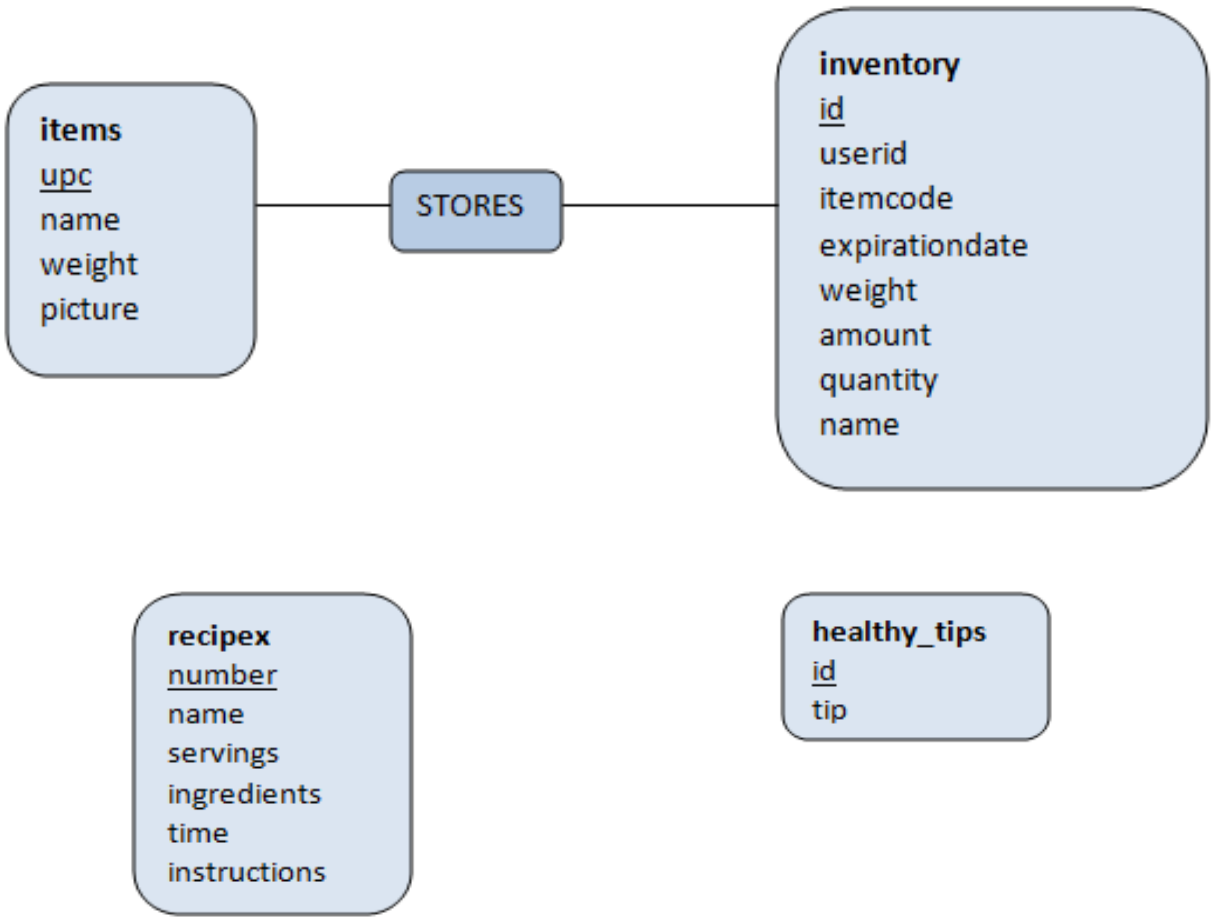
Database
-Tables-



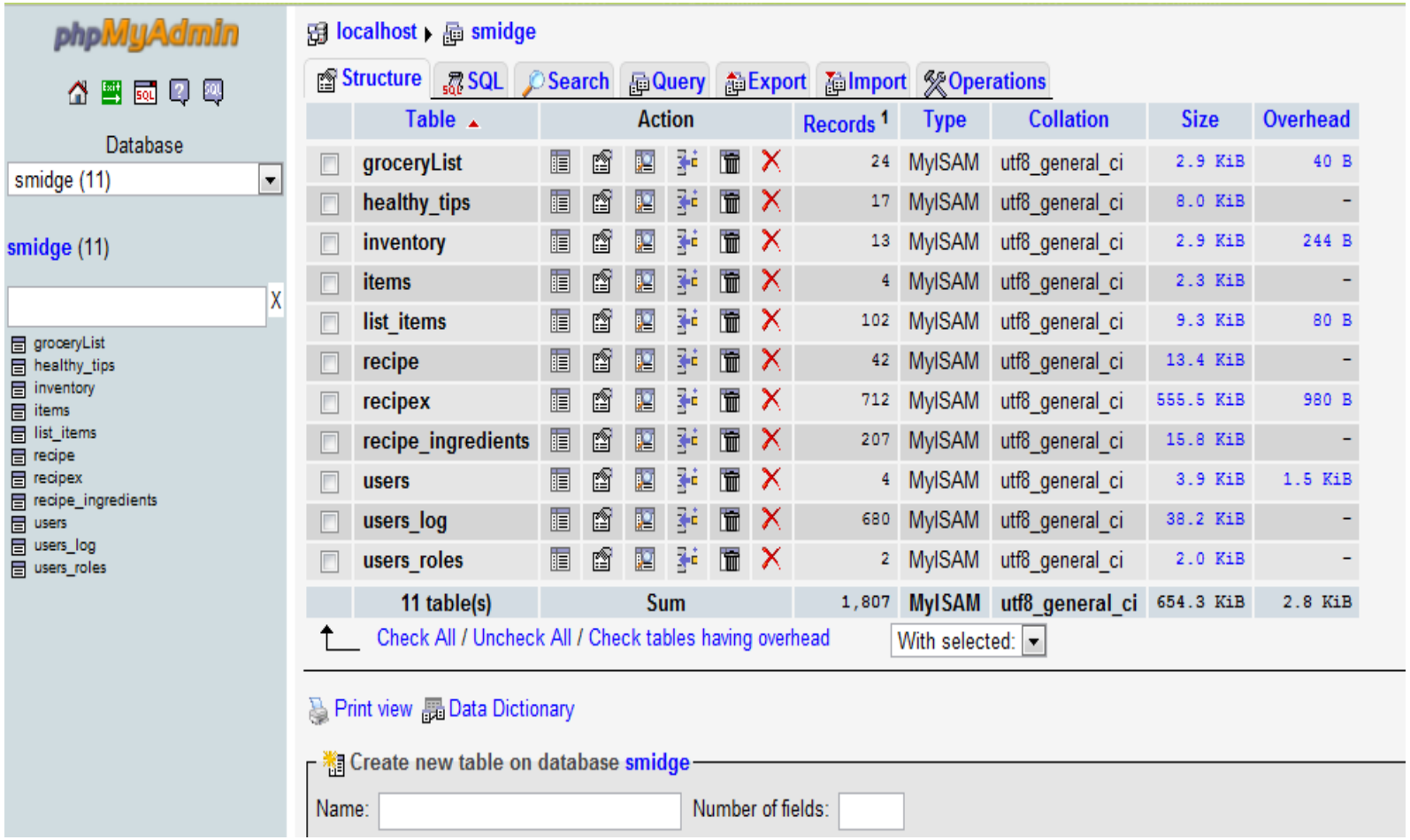
Database
-Tables-



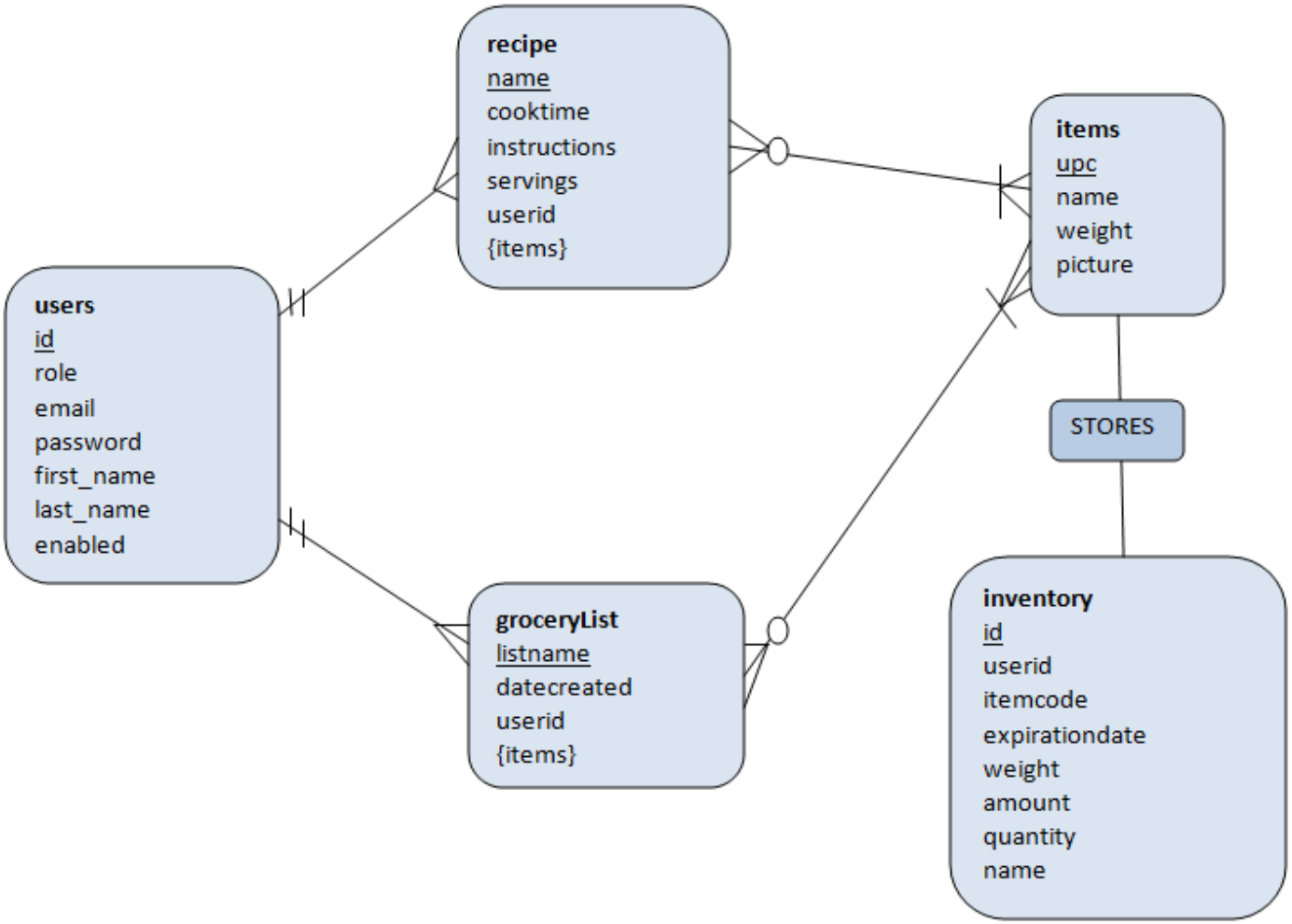
Database
-Tables-



Database
-Tables-



Database
-ER Diagram-



Database-Android

Use of SQLite Database Helpers:

InventoryDBHelper
ItemsDBHelper
ProduceDBHelper
RecipeDBHelper
SettingsDBHelper
ShoppingListDBHelper

API

- Use of built in $_POST and $_GET PHP functions in order to obtain and send information to the mobile app and the fridge client.
- Information sent from a form is invisible to others and has no limit on the amount of information to send or receive.

API

Information received:

- JSON message is received decoded and the information is saved to the database.

Information to send:

- SQL query → JSON encoded message

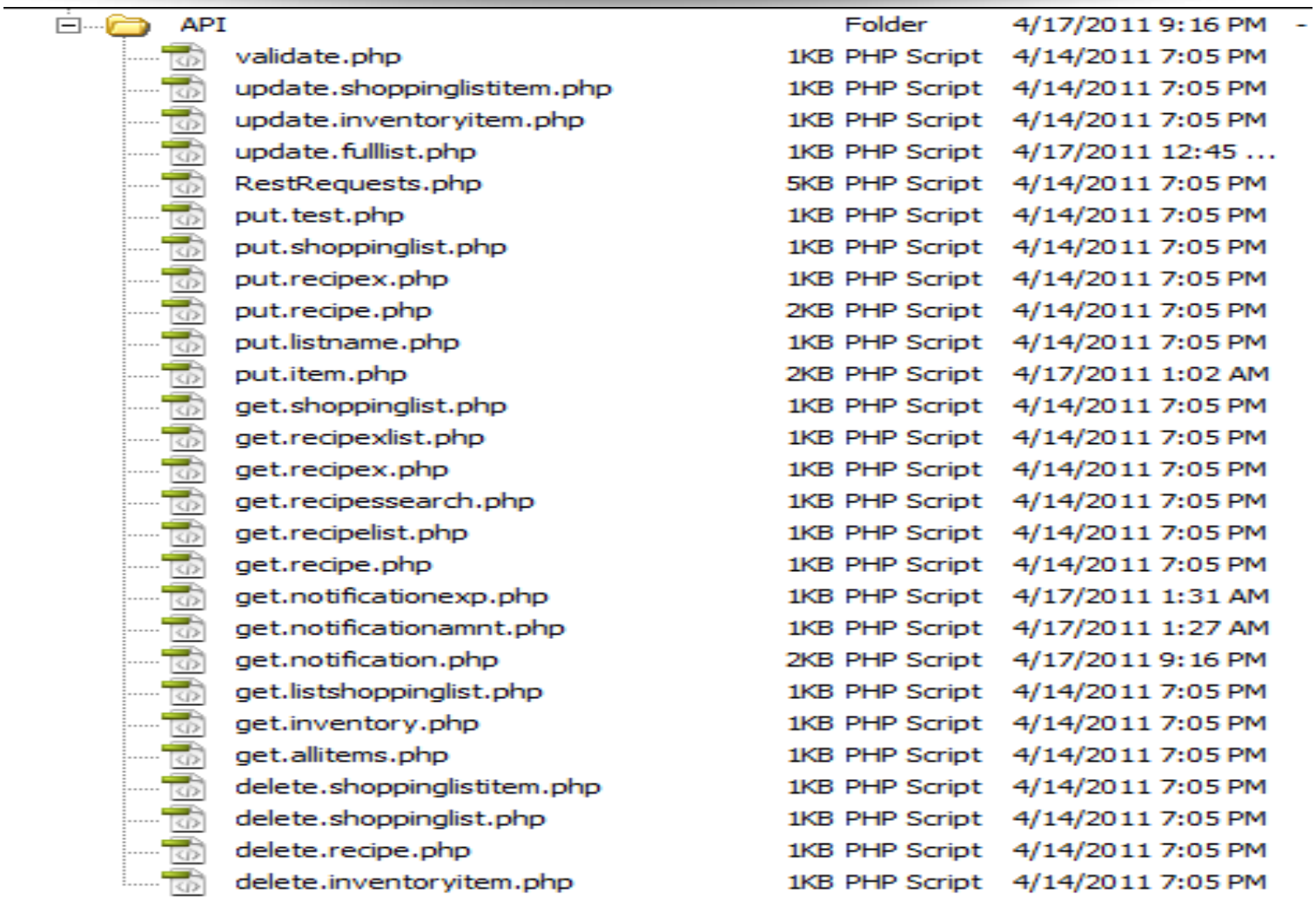
API-ANDROID SIDE

- Accessing from Android:

- HttpPost: sends a json encoded message with the information that will be added to the database.

- HttpGet: receives a json encoded message and saves the information from it in the local database.

MANAGE-API



API	Folder	4/17/2011 9:16 PM	-
validate.php	1KB PHP Script	4/14/2011 7:05 PM	
update.shoppinglistitem.php	1KB PHP Script	4/14/2011 7:05 PM	
update.inventoryitem.php	1KB PHP Script	4/14/2011 7:05 PM	
update.fulllist.php	1KB PHP Script	4/17/2011 12:45 ...	
RestRequests.php	5KB PHP Script	4/14/2011 7:05 PM	
put.test.php	1KB PHP Script	4/14/2011 7:05 PM	
put.shoppinglist.php	1KB PHP Script	4/14/2011 7:05 PM	
put.recipex.php	1KB PHP Script	4/14/2011 7:05 PM	
put.recipe.php	2KB PHP Script	4/14/2011 7:05 PM	
put.listname.php	1KB PHP Script	4/14/2011 7:05 PM	
put.item.php	2KB PHP Script	4/17/2011 1:02 AM	
get.shoppinglist.php	1KB PHP Script	4/14/2011 7:05 PM	
get.recipexlist.php	1KB PHP Script	4/14/2011 7:05 PM	
get.recipex.php	1KB PHP Script	4/14/2011 7:05 PM	
get.recipessearch.php	1KB PHP Script	4/14/2011 7:05 PM	
get.recipelist.php	1KB PHP Script	4/14/2011 7:05 PM	
get.recipe.php	1KB PHP Script	4/14/2011 7:05 PM	
get.notificationexp.php	1KB PHP Script	4/17/2011 1:31 AM	
get.notificationamnt.php	1KB PHP Script	4/17/2011 1:27 AM	
get.notification.php	2KB PHP Script	4/17/2011 9:16 PM	
get.listshoppinglist.php	1KB PHP Script	4/14/2011 7:05 PM	
get.inventory.php	1KB PHP Script	4/14/2011 7:05 PM	
get.allitems.php	1KB PHP Script	4/14/2011 7:05 PM	
delete.shoppinglistitem.php	1KB PHP Script	4/14/2011 7:05 PM	
delete.shoppinglist.php	1KB PHP Script	4/14/2011 7:05 PM	
delete.recipe.php	1KB PHP Script	4/14/2011 7:05 PM	
delete.inventoryitem.php	1KB PHP Script	4/14/2011 7:05 PM	

Website

- PHP
  - dynamic and interactive Web pages.
- AJAX
  - exchange data with a server.
- JavaScript (*THE* scripting language of the Web)
  - jQuery
  - validate forms, communicate with the server.

Website
-How does it work?-

- The website was developed using html, PHP and Javascript programming languages.
- Javascripts request from php forms that connect to the database through the “class.table.php” classes.

Website

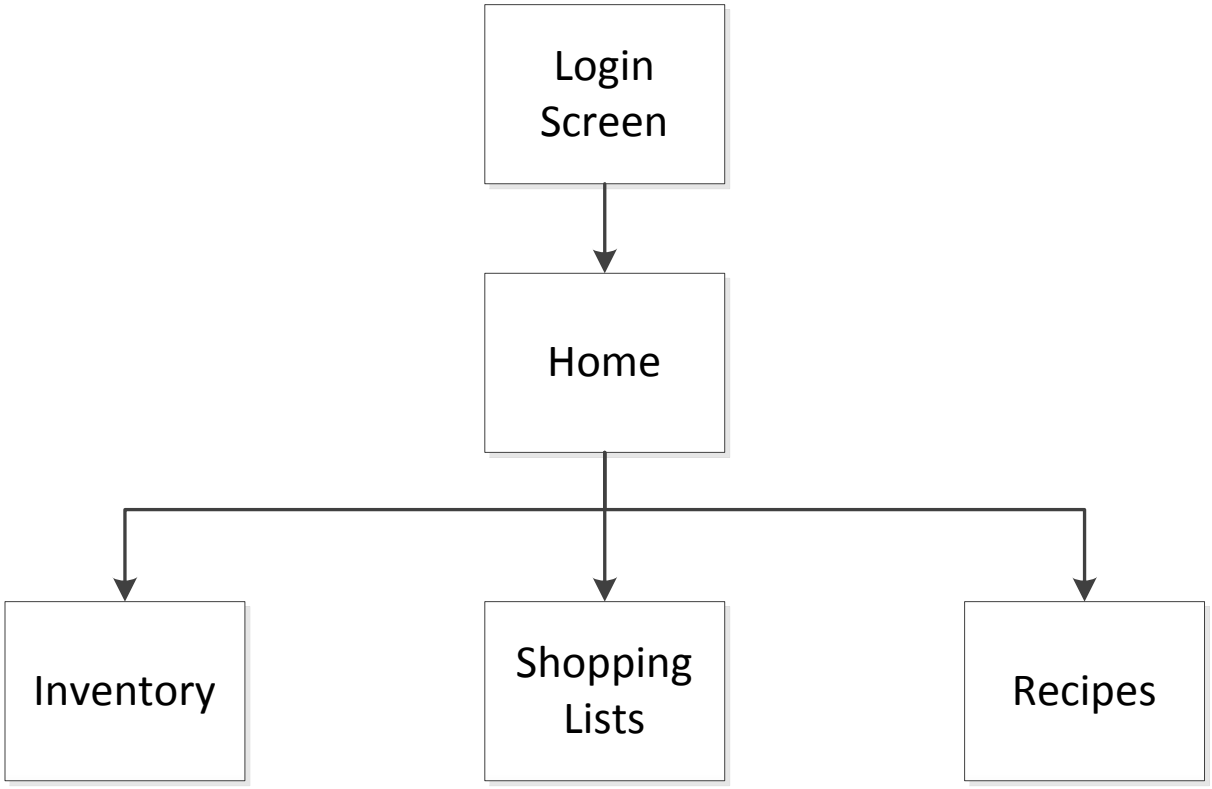
User sets an account if it does not have one
User logs in the application

Application Menu:

- Dashboard
- Inventory
- Shopping Lists
- Recipes
- Users (this option will only show if the user is an administrator)
- Settings

Website
-Block Diagram-

- Home



User information

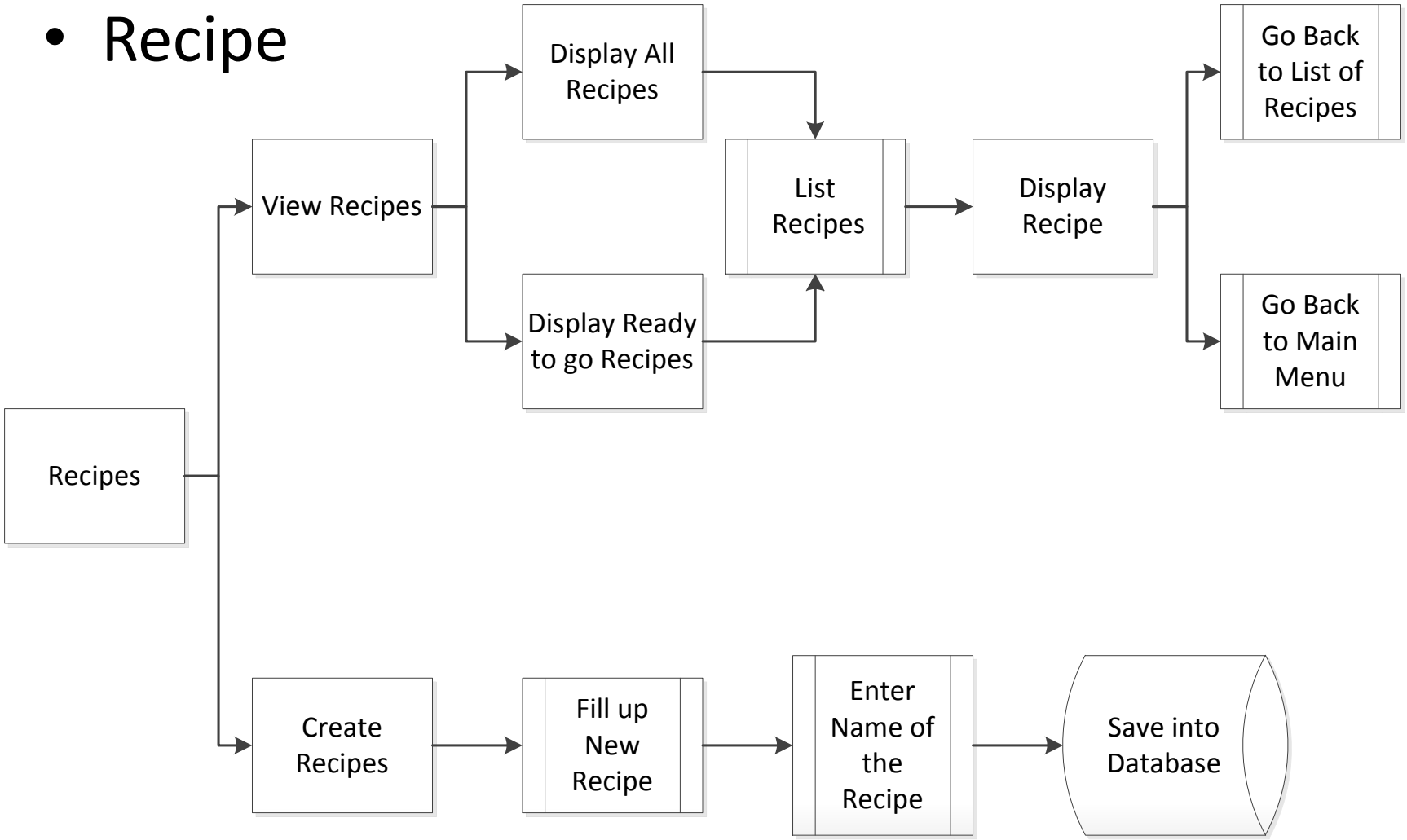
- Inventory
  - User items
- Settings
  - General
  - Your profile
- Users (only available for the administrator)
  - Users status

Shopping List

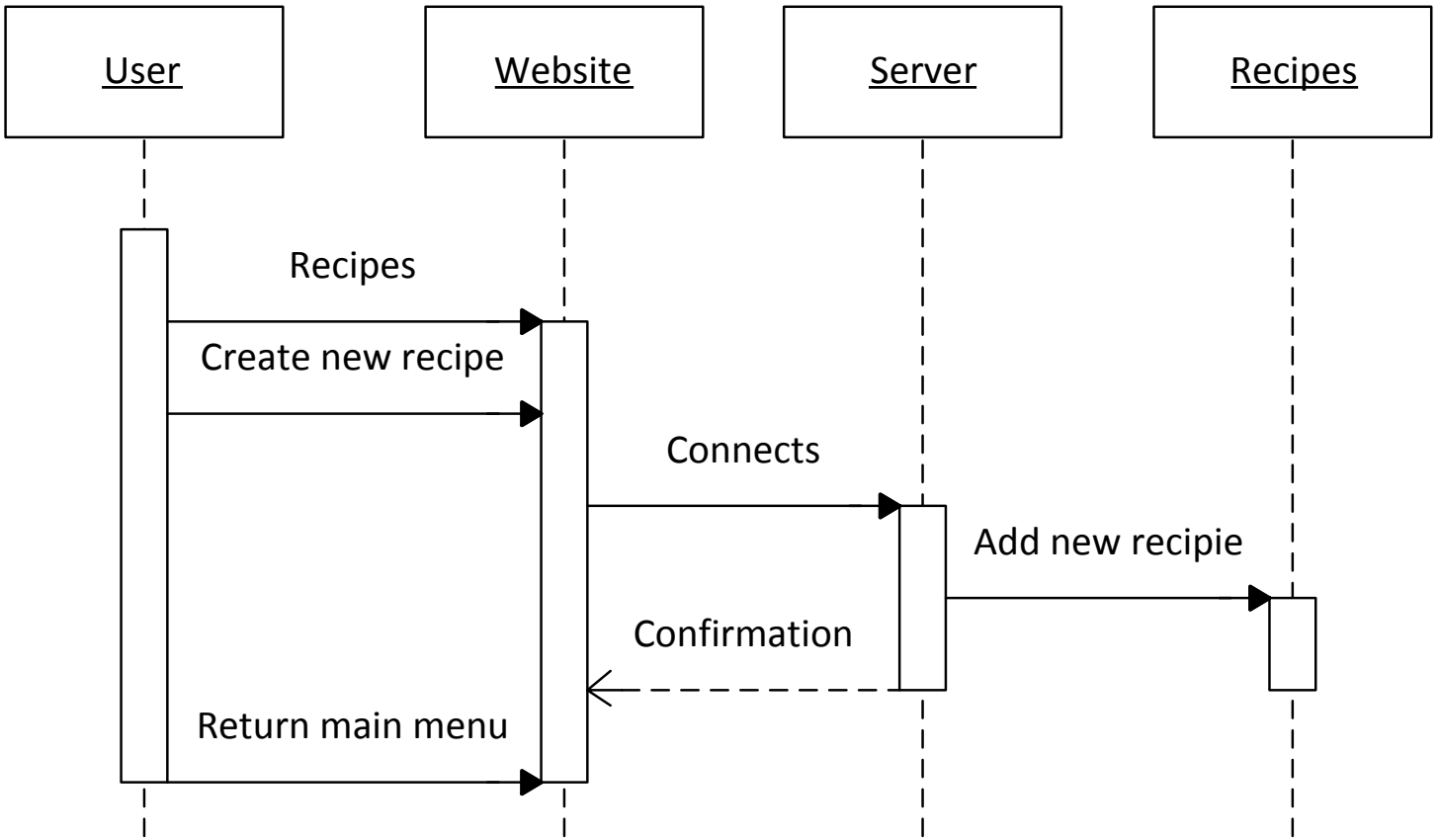
- Search for existing shopping lists
- Create new shopping lists
- Modified existing shopping lists

Website
-Block Diagram-

- Recipe



Website
-Sequence Diagram-

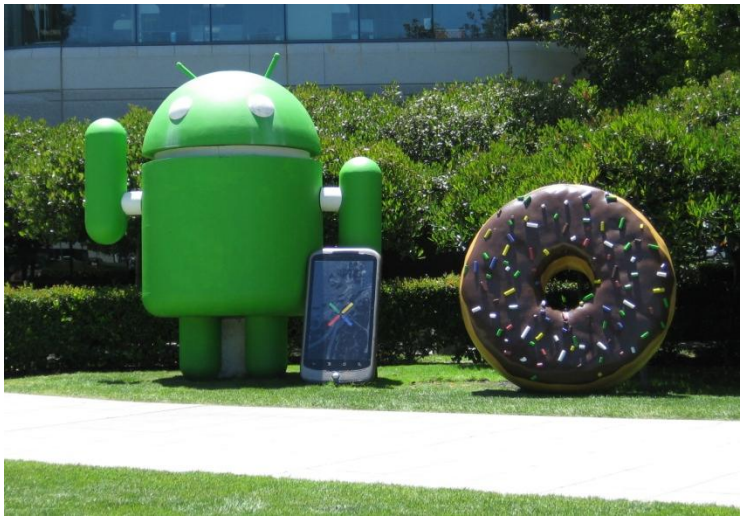


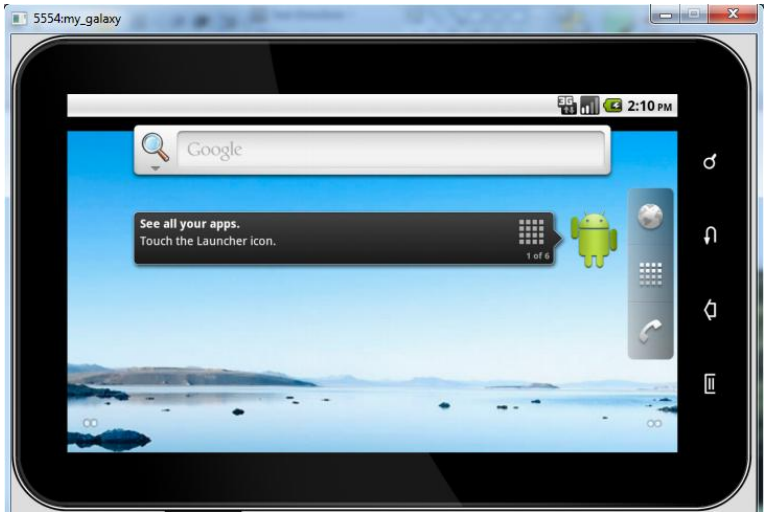
Recipes

- Search Recipe
  - Database of created recipes.
  - Basic Search
  - Advance Search
- Recipe
  - Database of user created recipes.
  - Option to edit saved recipes.

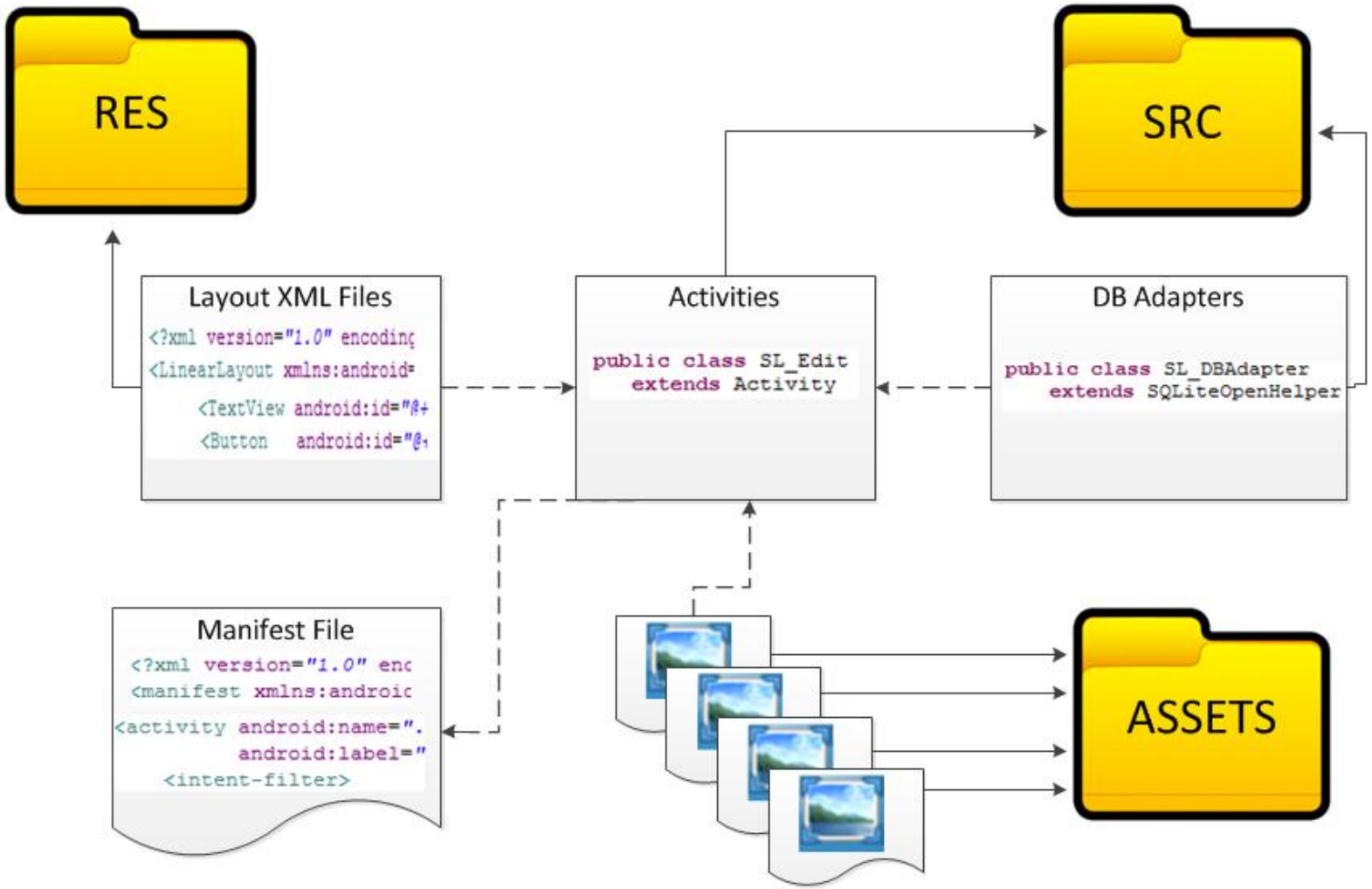
Development Environment

- Eclipse IDE v.3.6 with ADT plugin
- SDK Platform Android 2.2
- Emulators:
  - Avd vs. Galaxy Tab

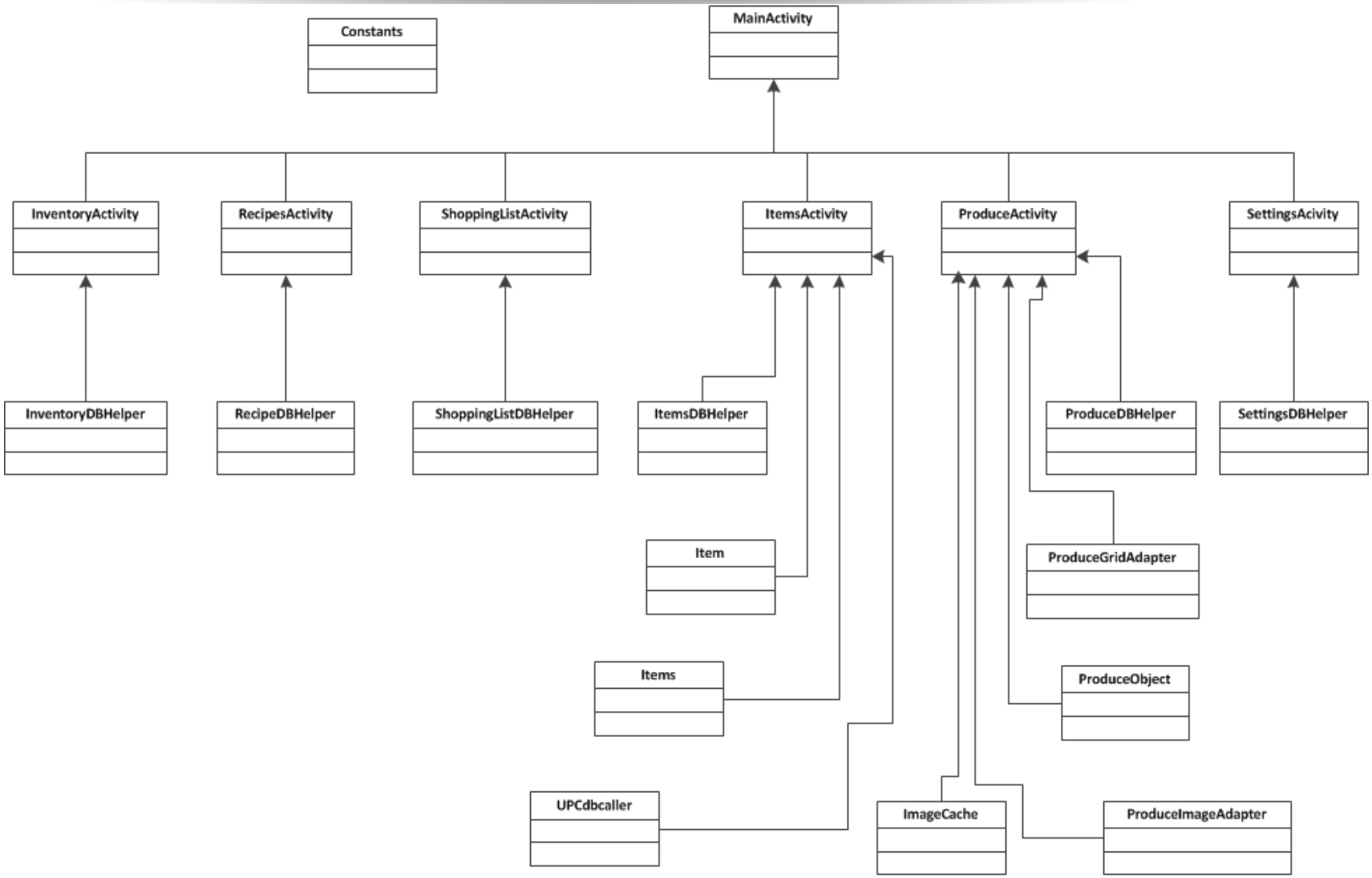




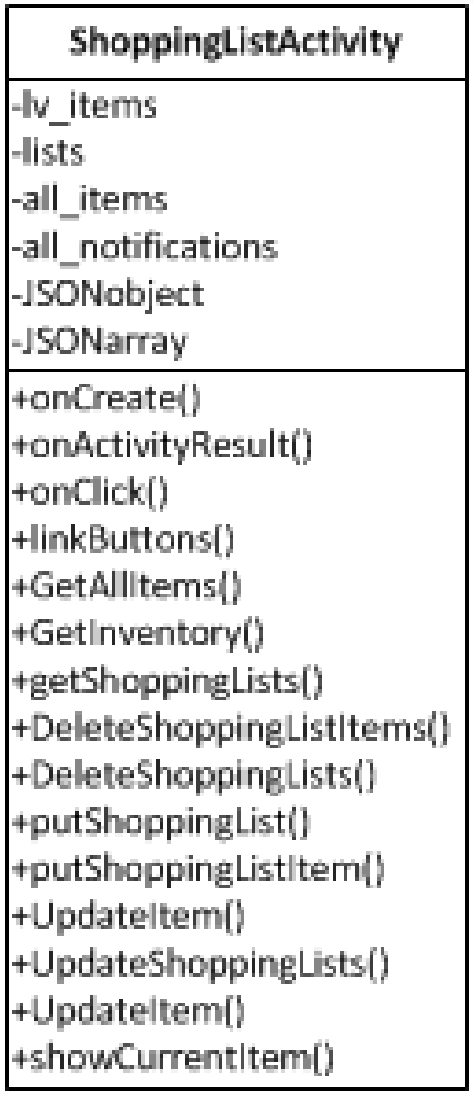
Android framework

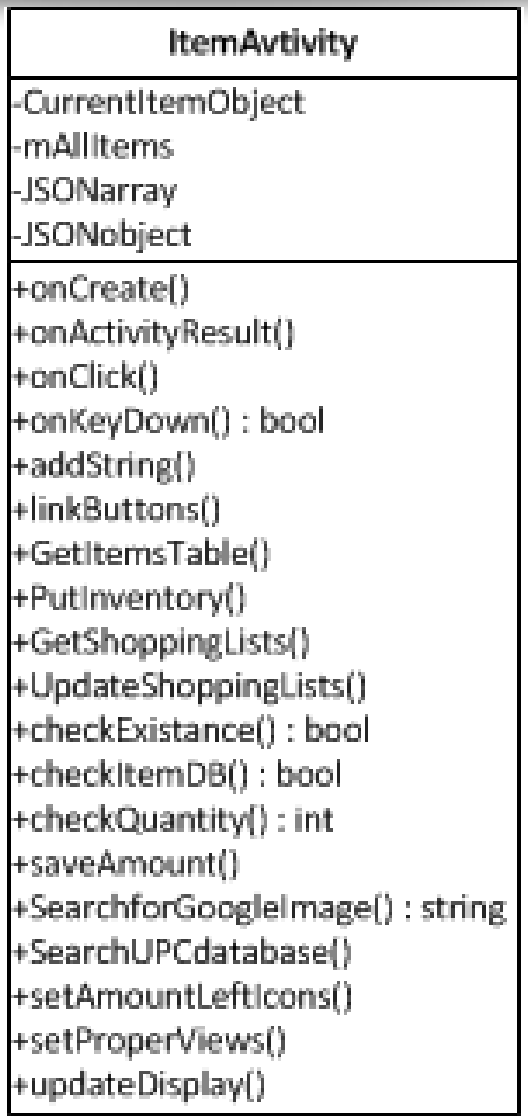


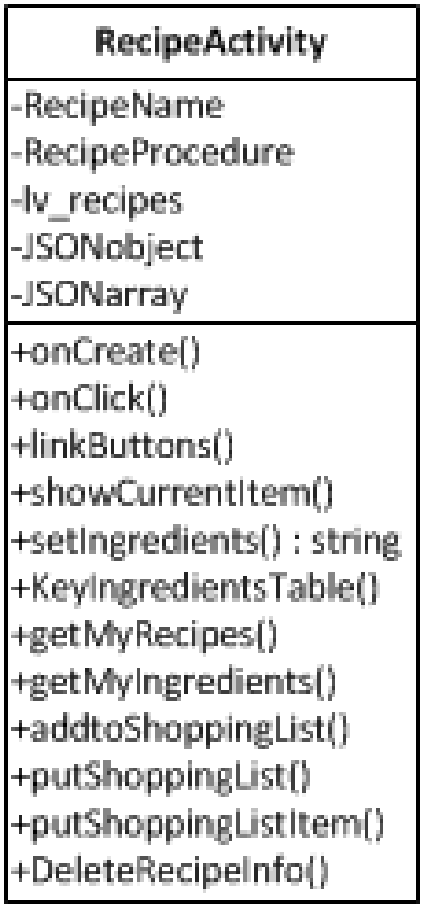
Class Diagram



Example Activity Classes



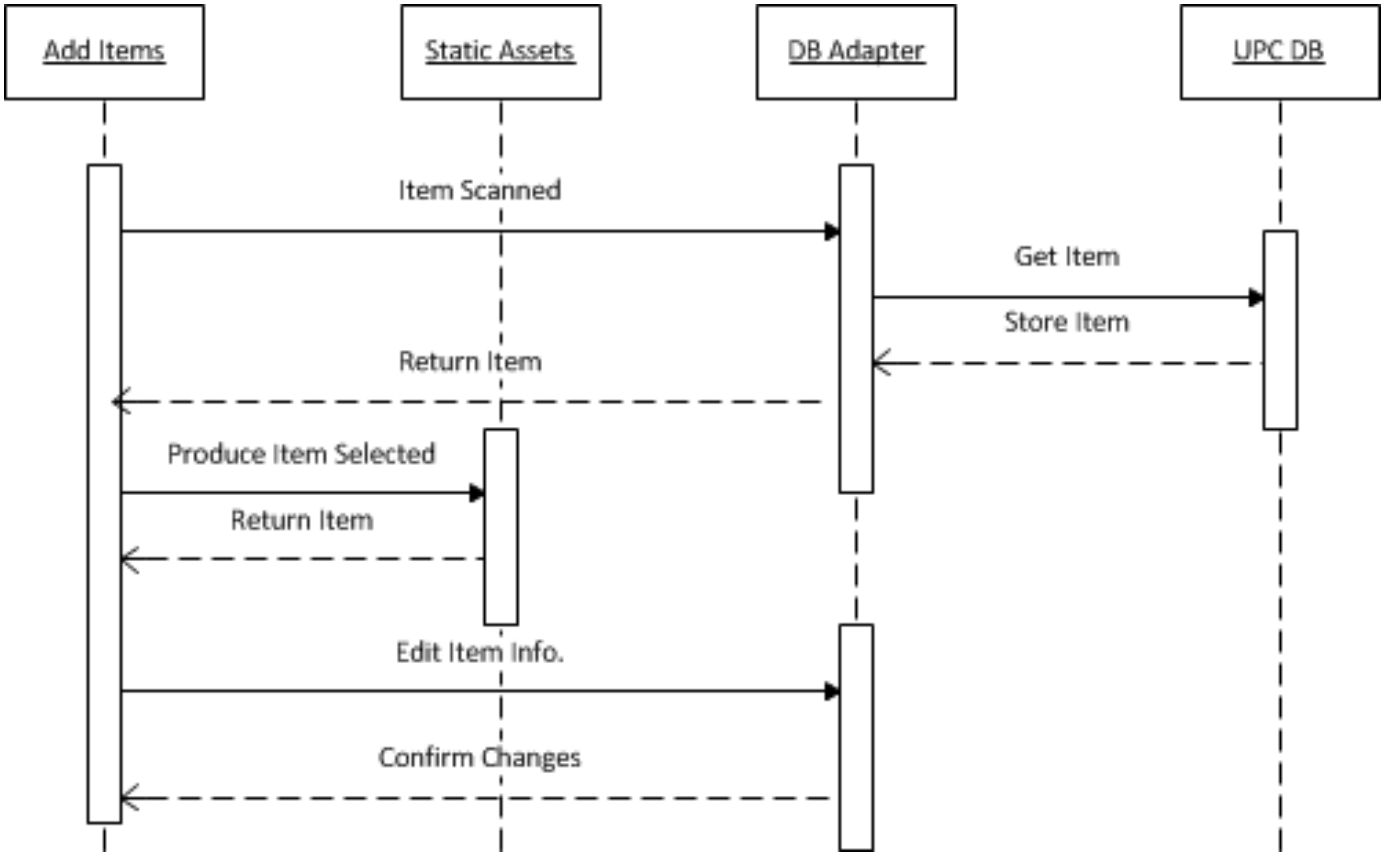




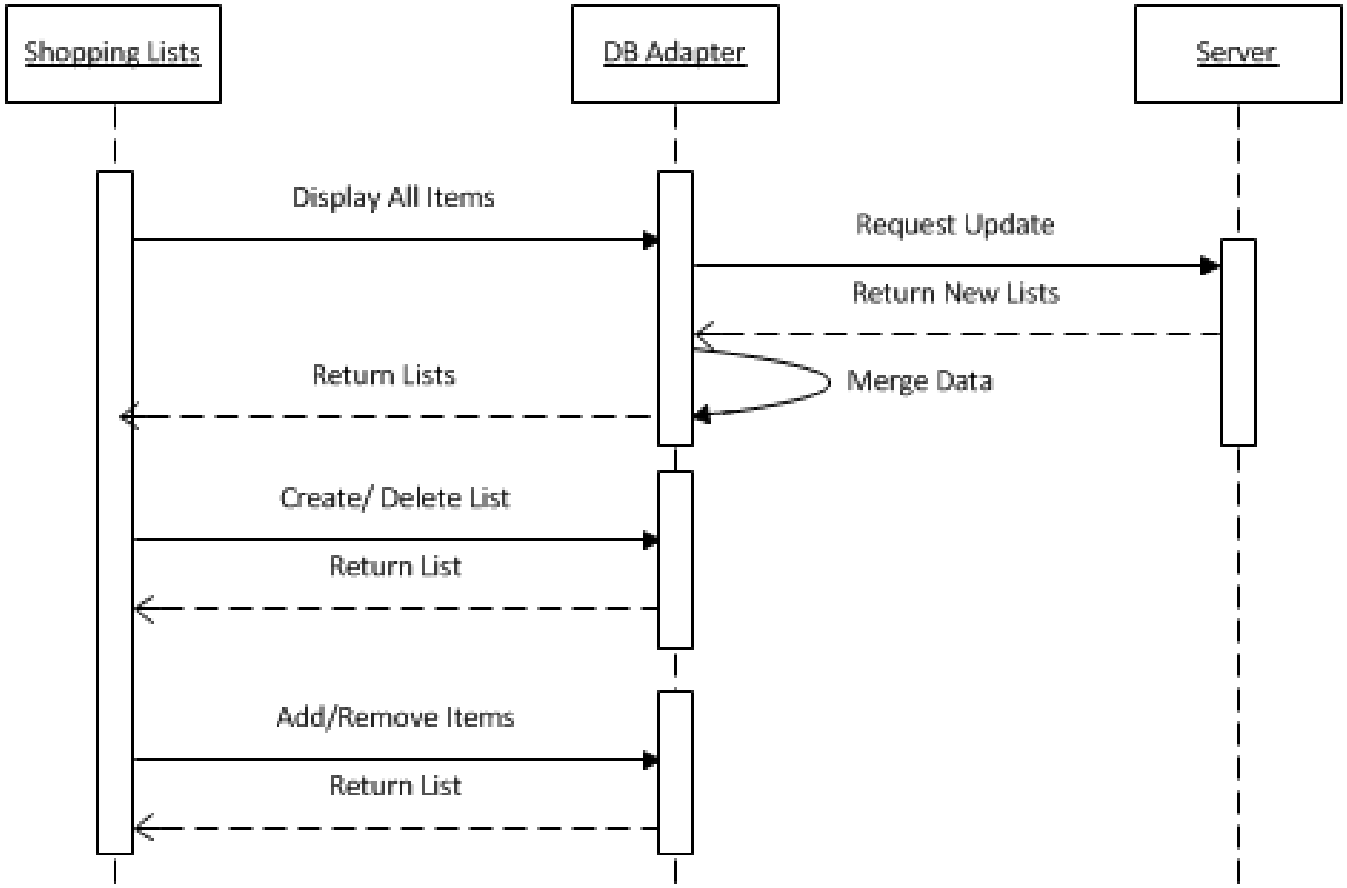
Example DB Helper Classes


```
private static final String DB_TABLE_RECIPES = "recipes";
private static final String DB_TABLE_ITEMS = "recipeitems";
private static final String DB_TABLE_KEYING = "recipekeys";


private static final int DB_VERSION = 1;

public final static String _ID = "_id";
public final static String NAME = "name";
public final static String PROCEDURE = "procedure";
public final static String RECIPENAME = "recipe_name";
public final static String DESCRIPTION = "description";

private static final String DATABASE_CREATE1 =
    "create table " + DB_TABLE_RECIPES + " (" +
    _ID + " integer primary key autoincrement, " +
    NAME + " TEXT, " +
    PROCEDURE + " BLOB);";

private static final String DATABASE_CREATE2 =
    "create table " + DB_TABLE_ITEMS + " (" +
    _ID + " integer primary key autoincrement, " +
    RECIPENAME + " integer, " +
    DESCRIPTION + " integer);";

private static final String DATABASE_CREATE3 =
    "create table " + DB_TABLE_KEYING + " (" +
    _ID + " integer primary key autoincrement, " +
    NAME + " integer);";

private SQLiteDatabase DB = null;
private DBHelper DBHelper = null;
private final Context mContext;
```

Add Items Sequence Diagram



Shopping List Sequence Diagram



Activity Summaries

Shopping List

- Create List
- Delete List
- Add Items
- Remove Items

Inventory

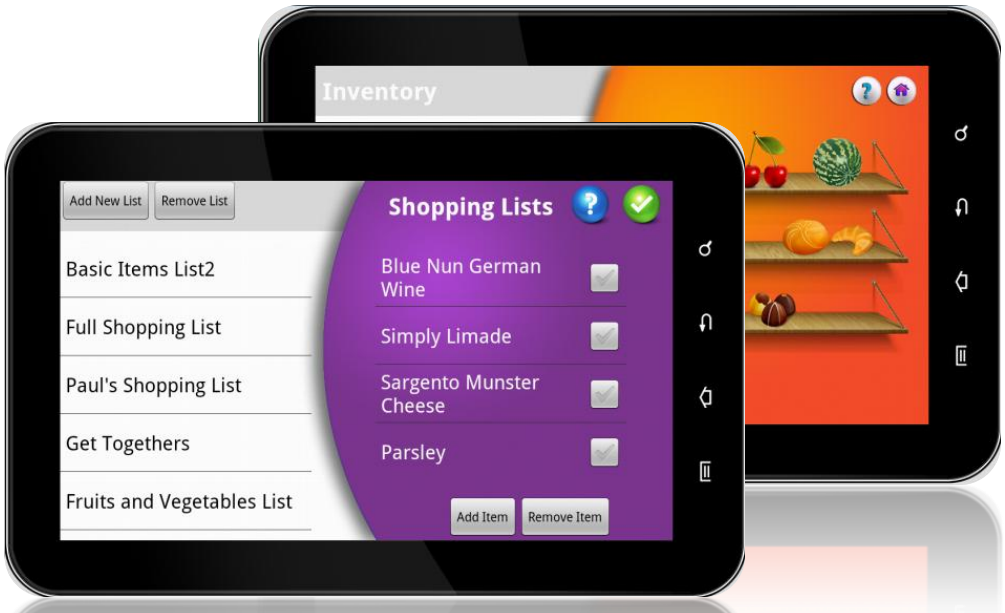
- Display Items
- Remove Items
- Edit Exp. Date
- Edit Amt. Left

Recipes

- List Recipes
- Display Recipe
- Add Items to shopping list

Settings

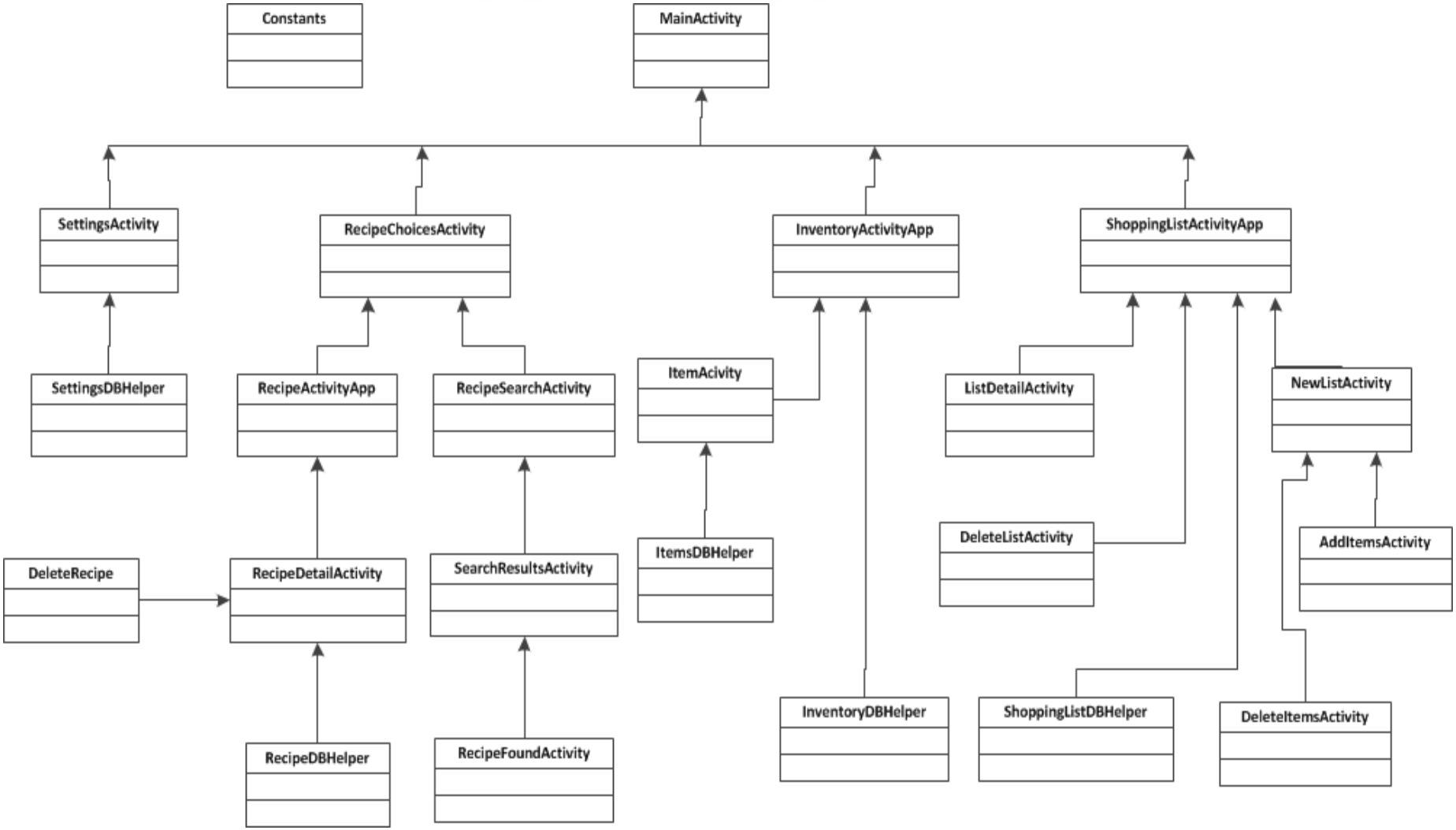
- Set username
- Set password



Mobile App
-Goals-

- Access the inventory at any time, any place.
- Access, create, and modify shopping lists while at the grocery store
- Manage Recipes
- Alert notifications:
  - Items about to expire
  - Items about to run out

Mobile App
-Class Diagram-



Project Timeline