Fall 1-15-2016

Building and Safety Department Android Mobile Application

Nary Simms
La Salle University, tepn1@student.lasalle.edu

Building and Safety Department Android Mobile Application

LA SALLE UNIVERSITY

Nary Simms

Table of Contents

Introduction

Research has found that Americans spend 4.5 hours watching television, 1.5 hours listening to the radio, about half an hour reading print and spend a whooping five plus hours per day in digital media (online, mobile, other). Out of these five hours, two hours and twenty minutes are spent on a mobile device (phone or tablet), which is a massive increase of about 575 percent from the twenty-four minutes that was reported in 2010. Flurry, an analytic app company, released data about their tracking of more than 300,000 apps in 2013, and they found the average time spent per day on mobile devices is two hours and thirty-eight minutes. They further found that 80 percent of the time is spent on apps and only 20 percent on the actual browser itself. Based off these findings, it can be concluded that Americans are spending more time online than on any other media, that digital time is on mobile devices, and that mobile time is spent mostly on apps (Samson, 2015).

Businesses are catching on to the fact that people are spending more time on their mobile devices doing activities that they once did with computers such as shopping, maintaining their calendars, and playing games. They are creating apps to accommodate these activities on mobile devices. In order to keep current customers and attract new ones businesses realize that portability is the future in how a business interacts with customers. Companies realize that to have an effective mobile presence requires more than just creating a website that is mobile friendly. So, small to medium-sized companies are creating their own dedicated apps to try an even the playing field against the big boys like Walmart and Starbucks.

The project proposal is to develop of a simple mobile application for a medium to small company that does not have a mobile application. It typically takes a minimum of four people to design and build an application, a product manager (owner of the application), the UI/UX designer (the artist responsible for the look and feel), the App Developer (the person who builds the front end: what appears on the app) and the Backend Developer (the person who builds the back end: what makes the app work). With my entry-level knowledge of java, python, HTML, PHP and MySQL and various book reference and tutorials online (websites and YouTube) and guidance from an advisor, I will take on all four roles and attempt to create a simple Android app within a 16 – 18 week time frame.

Mobile App Scope and Overview

The mobile app will be used by the customers of the Building and Safety Department in Lincoln Nebraska. The app will allow customers to check the approval status of a requested permit or view previously approved permits and any relevant information pertaining thereto. Users will be able to search three different ways. They can search by the actual permit number, license number of the contractor associated with a permit, or by entering an address.

The department averages about 400 permit application submissions per week. These submissions are for Electrical, Mechanical, Building and Plumbing permits only. There are many other types of permits (Planning, Public Works, Flood Plan, Public Works and Health Departments); these permit types will not be included in this mobile search app project. Eventually, users will be able to apply for building permits, but that will be made available through the web platform first because of all the reviews and fees involved in that process.

The scope of this project will focus on three main search type features that already exist on the web platform known as the Accela Citizen Access: (https://my.lincoln.ne.gov/CitizenAccess/) , this web application allows contractors to apply for permits, to request inspections and to access contractor accounts. It also allows homeowners to look-up permits by property, to request inspections and to search applications.

The 3 Main Search Features

Search by Permit Number

This feature allows the user to search for various permits that they or a contractor might have. The user must know the permit number in order for him/her to use this feature. The types of permits that can be searched for are Building, Mechanical, Plumbing and Electrical.

Search by Licensed Professional (Contractors):

This feature allows a licensed professional to search for permits that have their licensed number associated with it. The search will display the permit number, the address of the permit, contractors associated with permit and its status.

Search by Address:

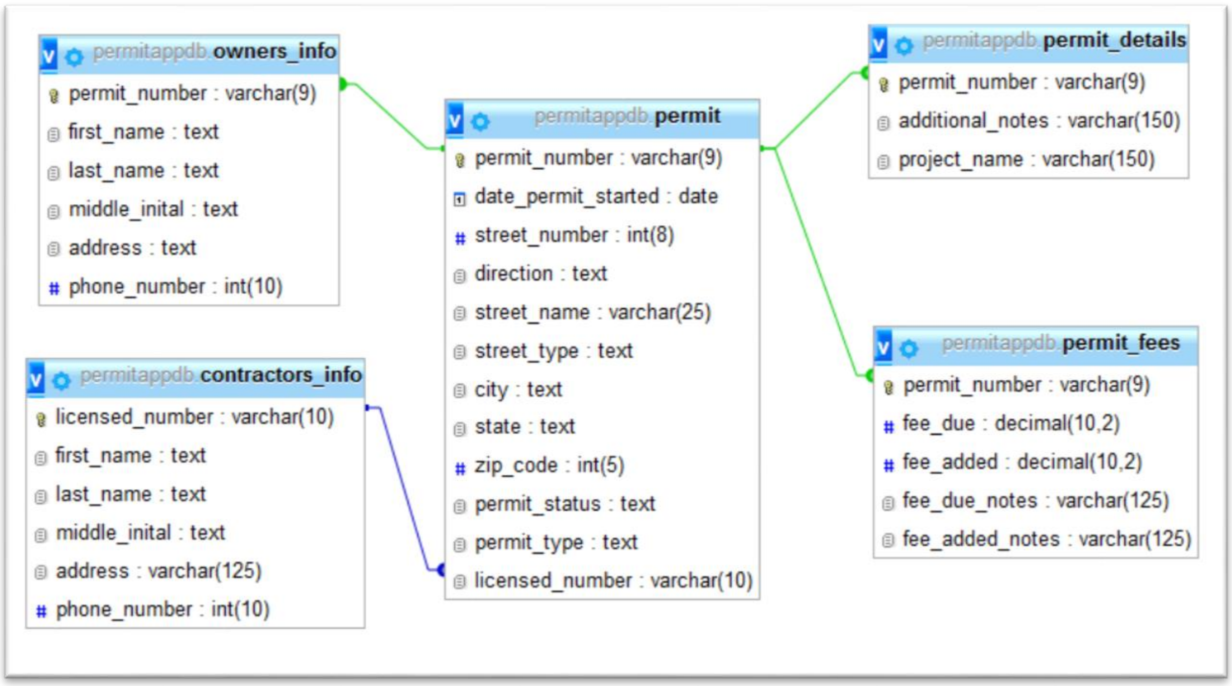
This feature will allow users to search by address. The user will only need the street number, street direction, street name and street type in order to find possible permits taken out on that property. The search will display the permit number, the address of the permit, contractors associated with permit and its status.

Database Design

Summary

The mobile app obtains its data from the database. I will be using a local server database: PHP/phpMyAdmin (MySQL Database) to house the data that will be used in the permit search application.

ER Diagram



permitappdb.owners_info
permit_number : varchar(9)
first_name : text
last_name : text
middle_inital : text
address : text
phone_number : int(10)
permitappdb.contractors_info
licensed_number : varchar(10)
first_name : text
last_name : text
middle_inital : text
address : varchar(125)
phone_number : int(10)
permitappdb.permit
permit_number : varchar(9)
date_permit_started : date
street_number : int(8)
direction : text
street_name : varchar(25)
street_type : text
city : text
state : text
zip_code : int(5)
permit_status : text
permit_type : text
licensed_number : varchar(10)
permitappdb.permit_details
permit_number : varchar(9)
additional_notes : varchar(150)
project_name : varchar(150)
permitappdb.permit_fees
permit_number : varchar(9)
fee_due : decimal(10,2)
fee_added : decimal(10,2)
fee_due_notes : varchar(125)
fee_added_notes : varchar(125)

The database consists of 5 tables: permit, permit_details, permit_fees, owners_info and contractors_info.

- The permit table houses the following fields: permit number [permit_number], address [street_number, direction, street_name, street_type, city, state, zip_code] and the licensed number [licensed_number]. This is the main table in the database where most of the information will be drawn from. The permit_number is a primary key and licensed_number is a foreign key.

- The Permit_Details houses the following fields: permit number [permit_number], additional notes [additional notes] and project name [project_name]. This table contains information such as the project name (if it exists) and any other additional notes pertaining to the permit. The permit_number is a primary key and it connects to the permit table using the permit_number.
- The Permit_Fees houses the following fields: permit number [permit_number], fees due [fee_due], additional fees added [fee_added], due fee notes [fee_due_notes] and additional added fees notes [fee_added_notes]. This table contains any fees and any related details that are associated with the permit. The permit_number is a primary key and it connects to the permit table using the permit_number
- The Owners_info table houses the following fields: permit number [permit_number], name of owner [first_name, last_name, and middle_inital], address of owner [address] and owner's phone number [phone_number]. The table contains information about owner of the permit or property. The permit_number is a primary key and it connects to the permit table using the permit_number
- Contractors_info houses the following fields: licensed number [licensed_number], contractor's name [first_name, last_name, and middle_inital], address [address] and phone number [phone_number]. The table contains information about the licensed contractors associated with the permit. The licensed_number is a primary key and it connects to the permit table using the licensed_number

For further details about database design please refer to Appendix A: Database Design

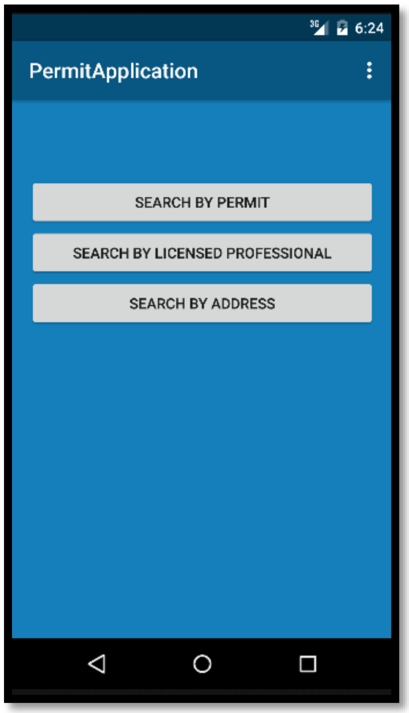
End User Documentation

Main Screen

Below is displayed the main screen for the application that users will see once they enter the app. A user has 3 option buttons to choose from and each option button will take them to a corresponding search screen.

3 Button Options:

- Search by permit: requires a permit number for search to work
- Search by licensed professional: requires a license number for search to work
- Search by address: requires the permit address (street number, street name, street type)



Search by Permit Screen:

The Search by Permit Activity allows the user to enter the permit number and get the corresponding information pertaining to that permit number.

Figure 1: Search by Permit Activity Screen

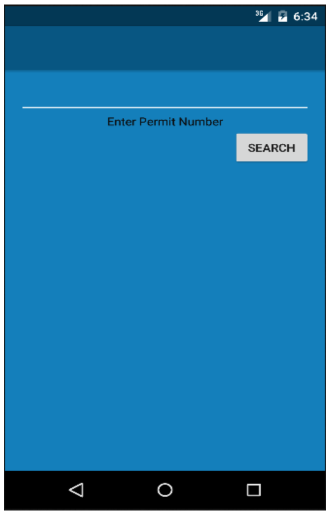


Figure 1. 2: If the user does not enter a permit number and simply clicks on the search button, then an alert will appear requesting the user to enter a permit number.

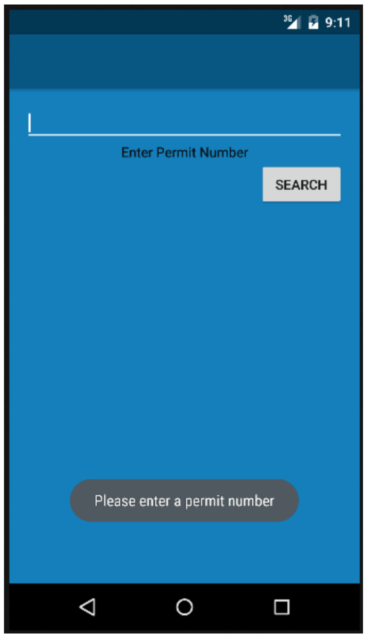
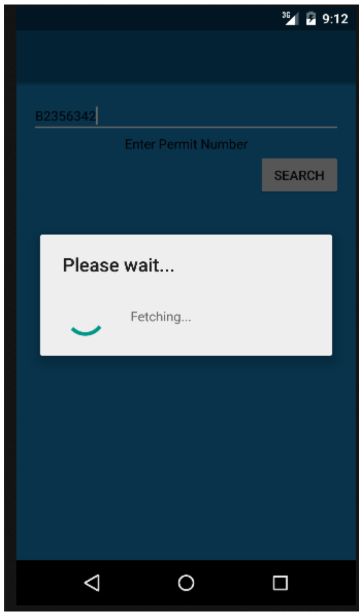
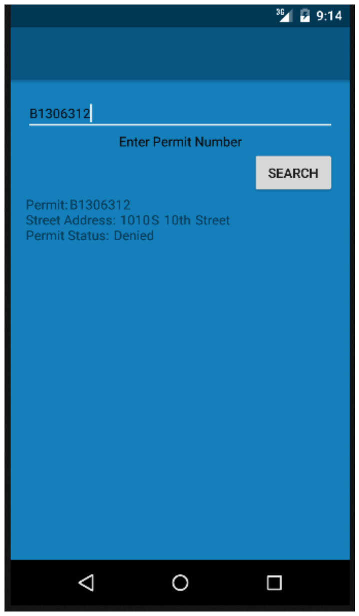


Figure 1. 3: If the user enters a permit number that exist then the results will appear under the search button.



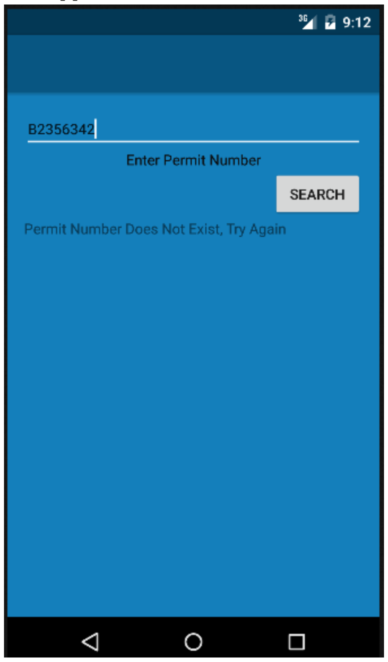
Please wait...

Fetching...



Permit: B1306312
Street Address: 1010S 10th Street
Permit Status: Denied

Figure 1. 4: If the user enters an incorrect permit number a message [permit number does not exist, try again] will appear below the search button.



Permit Number Does Not Exist, Try Again

Permit Search Use Case

Use Case Name: Permit Search	ID: 1	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: User – wants to search for the permit they applied for to determine status and information pertaining to it		
Brief Description: this use case describes the process for searching permit numbers that a particular user is looking for		
Trigger: user performs this task when they want to search for a permit Type: internal		
Relationships: Association: Include: Extend: Generalization:		
Normal Flow of Events: 1. The user selects the Search by Permit Button 2. The user enters the 9-10 digit permit number and clicks on search button 3. The system returns the results of the search in the same screen		
SubFlows:		
Alternate/Exceptional Flows:		

Search by License Screen:

The Search by License Activity allows the user to enter the license number and retrieve the corresponding information pertaining to that permit number they have entered.

Figure 2: Search by License Activity Screen

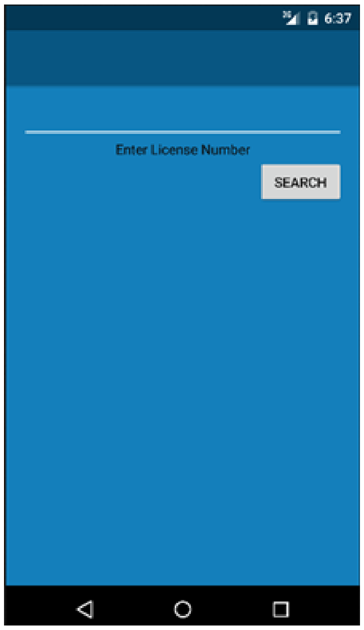
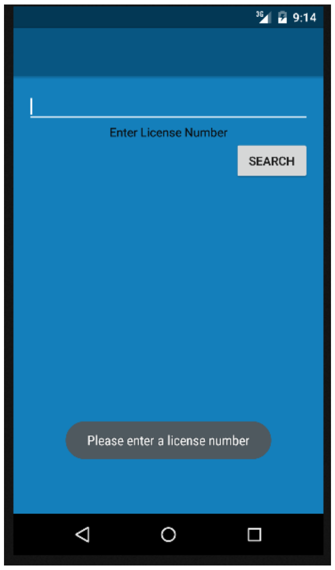
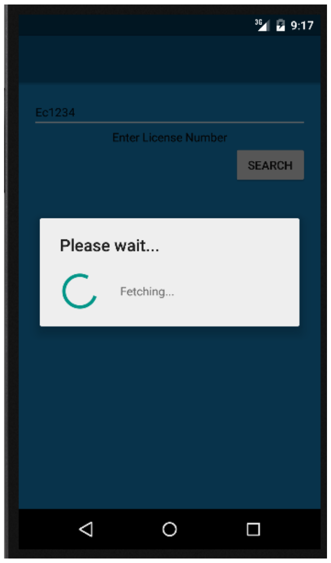


Figure 2.1: If the user does not enter a license number and simply clicks on the search button, then an alert will appear requesting the user to enter a license number.



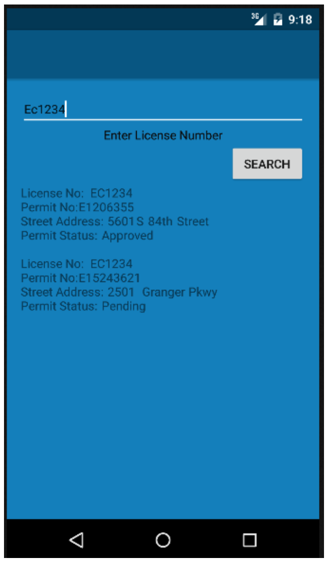
Please enter a license number

Figure 2. 3: If the user enters a license number that exists then the results will appear under the search button.



Please wait...

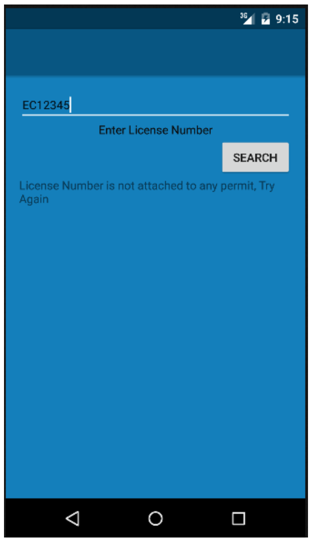
Fetching...



License No: EC1234
Permit No:E1206355
Street Address: 5601 S 84th Street
Permit Status: Approved

License No: EC1234
Permit No:E15243621
Street Address: 2501 Granger Pkwy
Permit Status: Pending

Figure 2. 4: If the user enters an incorrect license number a message [license number does not exist, try again] will appear below the search.



License Number is not attached to any permit, Try Again

License Search Use Case

Use Case Name: License Search	ID: 2	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: User – wants to search for the permits that associated with a particular license contractor		
Brief Description: this use case describes the process for searching license numbers that a particular user is looking for		
Trigger: user performs this task when they want to search for a permit Type: internal		
Relationships: Association: Include: Extend: Generalization:		
Normal Flow of Events: 1. The user selects the Search by License Button 2. The user enters the 3-6 digit license number and clicks on search button 3. The system returns the results of the search in the same screen		
SubFlows:		
Alternate/Exceptional Flows:		

Search by License Screen:

The Search by Address Activity allows the user to enter the street number, street direction, street name and street type. The street direction is optional since not all address has a street direction associated with it.

Figure 3: Search by Address Activity Screen

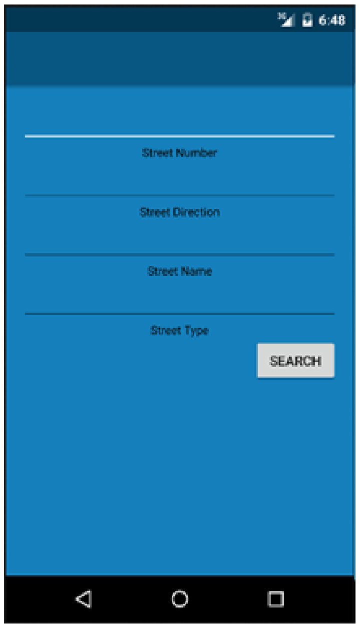


Figure 3.1: If the user does not enter any information in any of the fields and simply clicks on the search button, then an alert will appear requesting the user to enter a street number, street name and street type.

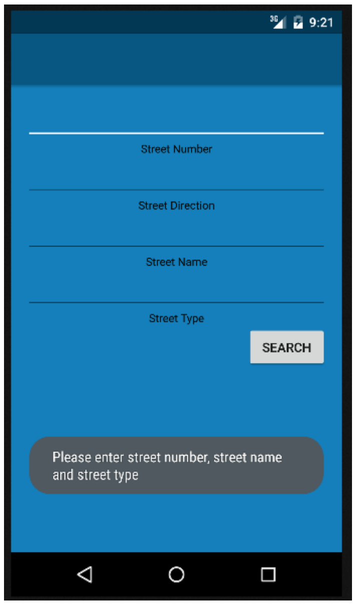


Figure 3.2: If the user does not enter the required fields for the address search a corresponding message will appear informing the user to enter the missing field.

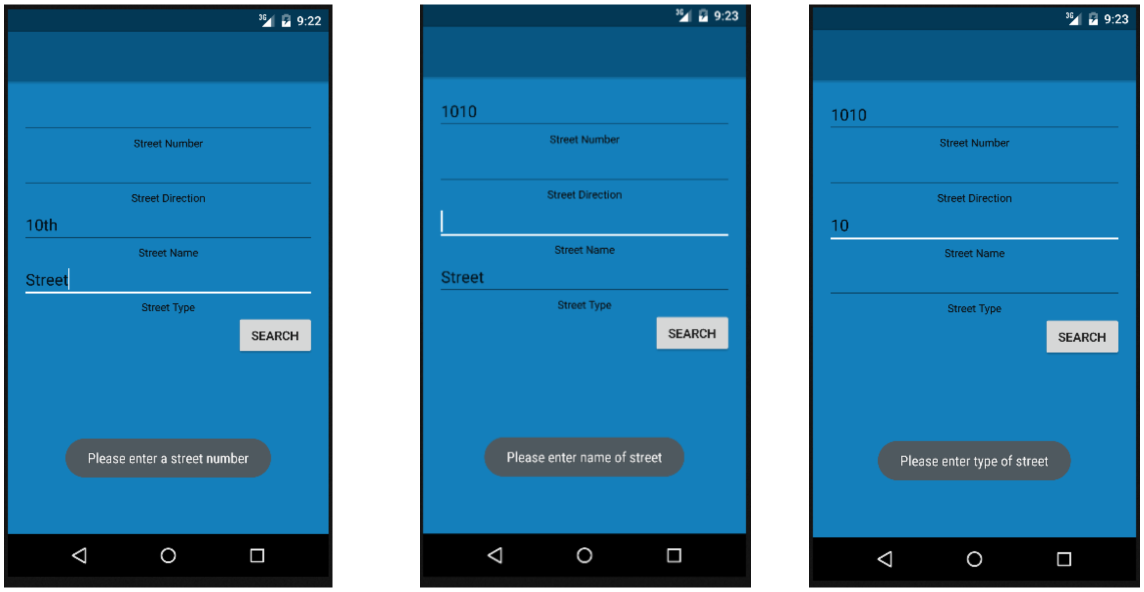
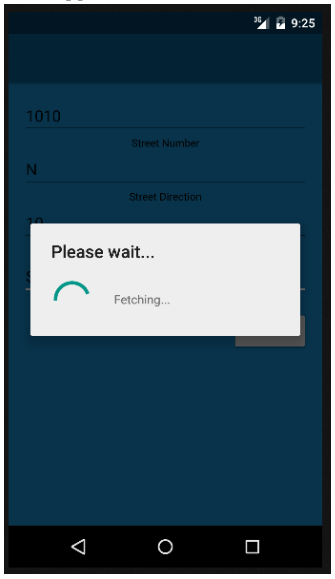
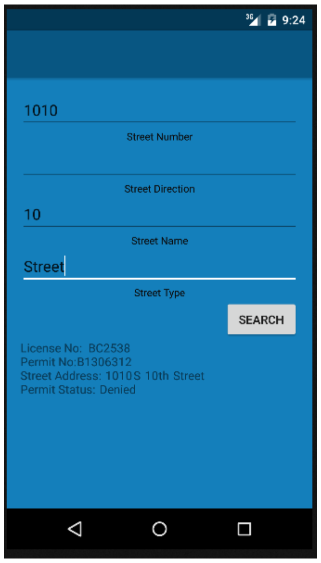


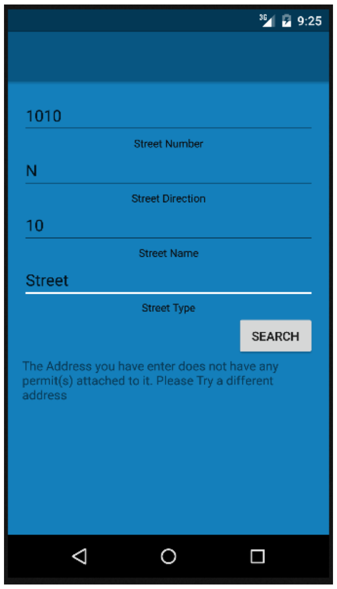
Figure 3.3: If the user enters an address that has permits associated with it then the results will appear under the search button.





License No: BC2538
Permit No:B1306312
Street Address: 1010S 10th Street
Permit Status: Denied

Figure 3. 4: If the user enters an address that does not have any permits associate with it [The Address you have enter does not have any permit(s) attached to it. Please try a different address] will appear below the search button.



The Address you have enter does not have any permit(s) attached to it. Please Try a different address

Address Search Use Case

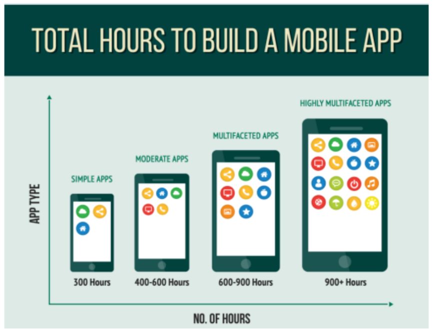
Use Case Name: Address Search	ID: 3	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: User – wants to search for the permits that are associated with a particular address		
Brief Description: this use case describes the process for searching permit numbers that a particular user is looking for		
Trigger: user performs this task when they want to search for a permit Type: internal		
Relationships: Association: Include: Extend: Generalization:		
Normal Flow of Events: 1. The user selects the Search by Permit Button 2. The user enters the street number, street name and street type/suffix and clicks on the search button 3. The system returns the results of the search on the same screen below the search button		
SubFlows:		
Alternate/Exceptional Flows:		

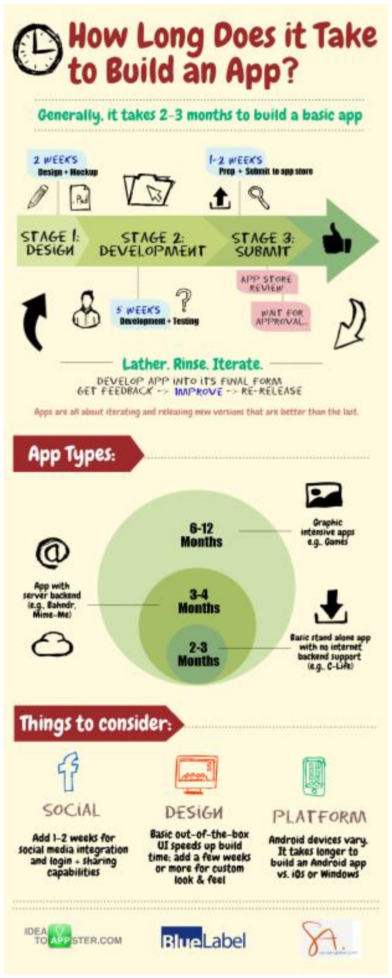
For more information about how the design came to be please refer to Appendix B: Mobile App

Story Board and to look at the Coding please refer to Appendix C: Mobile App Code.

Conclusion

According to Statista, (“Number of apps”, 2015) as of July 2015, there are roughly about four million apps out in the market, and those numbers are increasing daily. 1.6 million Apps are Android based (Google Play), and 1.5 million are iOS based (Apple App), while rest reside with Windows phone store or Blackberry. Apps are popping up left and right but many of them turn out to be “copycat” apps and get lost in the crowd. The question arise: who are creating all these game apps, social network apps, utility and productive apps that are popping up monthly? How much time and effort is going on in the background into building them? Kinvey conducted a survey of 100 native mobile developers, and they found that on average the time to design, build and test an app is about eighteen weeks, which is a little over more four months. Some claim that this is too much time, while others say that is not enough (Rowinski, 2013). Nine Hertz, an offshore development company, agrees with this number of eighteen weeks to develop and publish a mobile app, specifying further that it will take roughly ten weeks for back-end coding and eight weeks to construct the front end. They also said that it takes about three hundred hours to create a simple app and that a complex, multifaceted app would typically take more than 900 or more hours to produce (“Insight of Mobile App Development Process,2015”).




How Long Does it Take to Build an App?
Generally, it takes 2-3 months to build a basic app
2 WEEKS Design + Mockup
1-2 WEEKS Prep + Submit to app store
STAGE 1: DESIGN
STAGE 2: DEVELOPMENT
STAGE 3: SUBMIT
APP STORE REVIEW
5 WEEKS Development + Testing
WAIT FOR APPROVAL
Lather. Rinse. Iterate.
DEVELOP APP INTO ITS FINAL FORM
GET FEEDBACK -> IMPROVE -> RE-RELEASE
Apps are all about iterating and releasing new versions that are better than the last.
App Types:
6-12 Months
3-4 Months
2-3 Months
Graphic intensive apps e.g. Games
App with server backend (e.g. Bahndr, Mime-Me)
Basic stand alone app with no internet backend support (e.g. C-Life)
Things to consider:
SOCIAL
Add 1-2 weeks for social media integration and login + sharing capabilities
DESIGN
Basic out-of-the-box UI speeds up build time; add a few weeks or more for custom look & feel
PLATFORM
Android devices vary. It takes longer to build an Android app vs. iOS or Windows
IDEA TO APPSTER.COM
BlueLabel

Source: Nine Hertz **Source: Idea to Appster**

As stated in the introduction “the project is to create an android mobile application within 16-18 weeks” which according to a survey of 100 mobile developers this can be accomplished. According to Idea to Appster “How long does it take to build an App”, it should take 2 weeks to design and create the mockup. Fourteen weeks for actual development and testing for a mobile app with a backend server. For this project I estimated about 3 weeks for design, mockup, researching, 11 weeks for development and 2 weeks for documentation.

During the first 3 weeks I was able to accomplish many things: gathered requirements from Rita Cox, the Building and Safety Department stakeholder, created a story board and wireframe on how the app should be connected, figured out what tools and programs needed to installed to create the application and gathered instructional resources (YouTube videos, books and sample coding) to help guide me during the development phase.

There were 2 major issues that I had to face during these first two weeks. One issue was communication with the stakeholder. Our communication was mainly through email so it would take about 2-3 days for responses to questions that occured during the project. The second issue was downloading and installing the programs for my development environment. It was a struggle due to multiple failed downloading and changing development environments. The additional program that helped support the development platform required reconfiguration to function and to do this, additional research was done in order to get the right configuration.

I had initially allotted a week to learn to navigate the development environment and to do some practice coding. However, during this learning week, my two year old decided to use my laptop as a ramp for his toy cars, and drop the computer down a flight of stairs. Needless to say, I had to reinstall the development platform on an old laptop. This misfortune pushed my learning milestone back a couple of days. The allotted time for of 3 weeks turned into 4 weeks. I still wanted to spend the next 11 weeks for the development so documentation only now gets 1 week instead of the previous 2 weeks.

The next 11 weeks were development and testing of the app, to work on the front end, set up the backend, the testing of the backend as well as learning along the way. Five weeks for the front end development, five weeks on the backend development and one week to do additional final testing.

The next five weeks consisted of learning and programming the front end of the Android project. During that time I realized that the front end development gradually blended in with the back end development, so they took place at the same time. Many days were spent learning code and testing and breaking down code, and re-doing the mobile application multiple times. There were many problems along the way. Files become unusable and hours were spent trying to figure out the missing connection. If the error could not be fixed, it meant that the project had to be redone. Files became lost or corrupted because of unforeseen circumstances and could not be recovered. Besides the re-doing of files, there were also the issues of constant Android Studio updates every 2-3 days which would delay the work on the mobile app.

The final week was spent on trying to clean up the coding. Preparing documentation and bringing everything together. Through all the frustration and hours of sleepless nights the project managed to come together. From the standpoint of an IT project, this project failed because the requirements were not fully met even though the prototype successfully runs the 3 main features, it is not ready for release. From an educational standpoint the project was a success because of the valuable lessons learned from this project.

While learning about and creating an Android mobile app I realized I was using various concepts that I had learned from several of my graduate classes that I had taken at La Salle University. These concepts included client interface design, database design, object-oriented design, project management and many others. I learned a valuable lesson on the amount of planning involved in project because without a concise and clear plan of action before the actual development the initial timeline of what needed to be done changes drastically. I also realized value of working in a team with members who specialize in different areas of creating an IT project. It was a struggle trying to do every one of these tasks on my own and trying to balance work, life and this project. I realize how a project in the "real world" can easily fail due to unforeseen reasons, and I now realize the complexity of the project that was not projected at the start. It is critical to plan everything effectively and having the right resources in order to have a successful finish. During the project, scope creep became very real to me and it actually sneaked up on me.

The Android application development is very intricate and has many components that blend together so have the proper experience and expertise in these areas comes in handy when creating an Android application. The project requires a level of expertise in different areas so one person cannot do it alone, it requires team input.

Developing an app requires patience and time and a lot of tries because of how many ways to approach the development. Frustration can happen easily with the lack of resources and expertise. It is possible to create a simple Android application in 18 weeks or 300 hours. Yes, it is possible if you have the knowledge, experience and a team to divide the work because no one person can handle all the things that are required to build a mobile app that can be release to the public.

Appendix A:
Database Design

Database Design

The Building and Safety Department uses software with a backend database. While the Building and Safety Department uses the software interface to insert, update and delete permit data, Accela Citizens Access, a separate entity from the Building and Safety Department uses the backend database to retrieve relevant information. Both the Building and Safety Department and Accela have created their own database but the information for their database is retrieve from the larger backend database. It only seems natural and beneficial to try and mimic one of these pre-existing SQL databases (direct access was not supplied for proprietary reasons).

Accela Citizens Access has created an online permit related search application where users register to gain access to its various search features. I obtained a copy, really more of a snapshot of the database. After a week of trying to navigate through the 45 pages of the Accela Citizen ERD Diagram trying to determine the relationship of each table, it was determined that design will be inefficient for the mobile application database. The database had too many relationships and an excessive number of joins were required for even relatively simply queries. Furthermore trying to optimize by stripping down the non-essential information would take too much time. Hence this database design will not be used to base the android application database on.

The Building and Safety Department has a more simplified database with relevant information pull form the larger database. The snapshot of the database was on 1 page and consisted of about 13 tables. Figuring out the relationships between the two tables was much simpler as well. So, after navigating through both the databases, it was determined that the design for the backend of the Android App will be based off the second simplified database design. However, it will be even more stripped down with non-essential information taken out and simplified in order for the wait time for results to be minimal. Making smaller tables will hopefully cut down on query times.

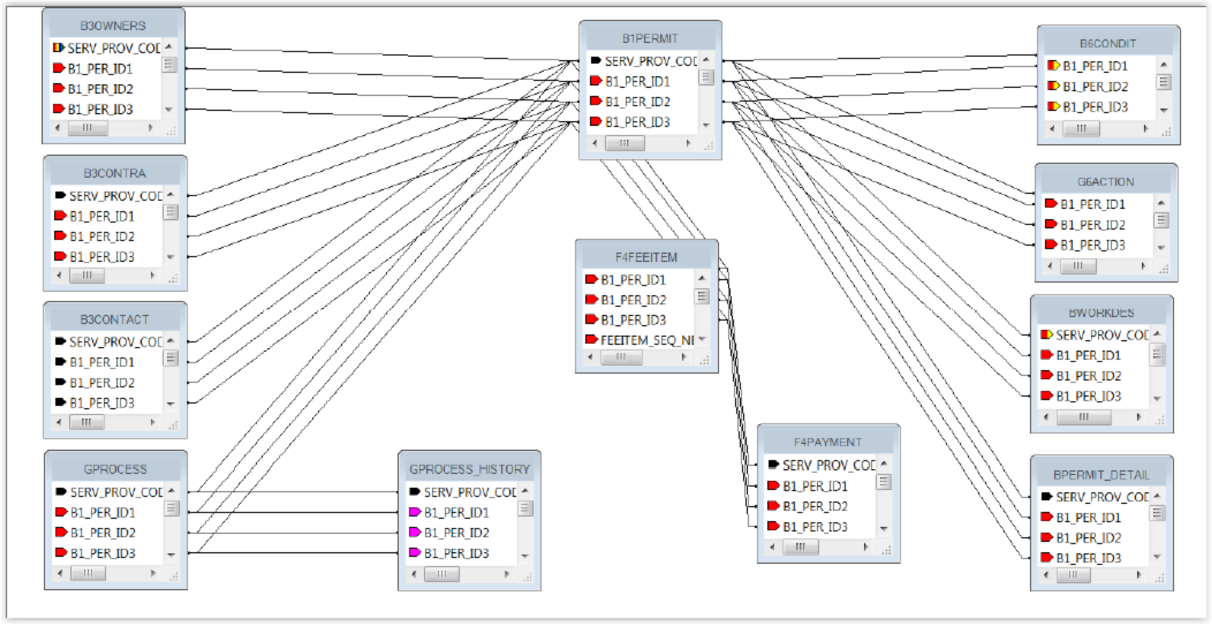


Figure 1: Building and Safety ERD (Snapshot)

Building and Safety Database Summary

The simplified system database consists of about 13 tables with only five main tables being used on a regular basis. The database itself is more of an out of the box product, so that it is similar in all jurisdictions, and it can be implemented and modified to meet the different needs of different jurisdictions.

- The system assigns more of a general number to the permit number and the department defines what they want to see in the system. For instance: “B1500525” that is actually stored in Alt_ID but behind the scenes the part would be (B1_Per_ID1 = 15BLD, B1_Per_ID2 = 00000, B1_Per_ID3 = 003OQ). The field ALT_ID is an ID that all the tables use.

- The Serv_Prov_Code: is the jurisdiction so it is automatically set to “Lincoln”
- Permit Number has between 8-9 characters but typically it will only have 8 characters, the extra character will determine if it is a county permit or regular permit.
  - The first character describes what type of permit it is (E for electrical, M for Mechanical, P for Plumbing, EC for Electrical County etc.).
  - Next two character are the year (e.g. E15*****)
  - Last five characters are just number in sequence. These numbers will start over each year on January 1st.
- Licensed Professional Number consists of a code between 3 and 6 characters long.
  - EC*** [Electrical], MC***[mechanical], PC***[plumbing]
  - After the first 2 characters the next characters are numbers in sequence which is the reason they vary in range in number of characters.
  - There are more Plumbing and Electrical codes than there are Mechanical codes.
- There are fields in the Database for Group, Type, Sub-Type and Category. Example: Electrical would be: Group-Building, Type-Electrical, subtype – City or County, Category – Licensed or Non-License

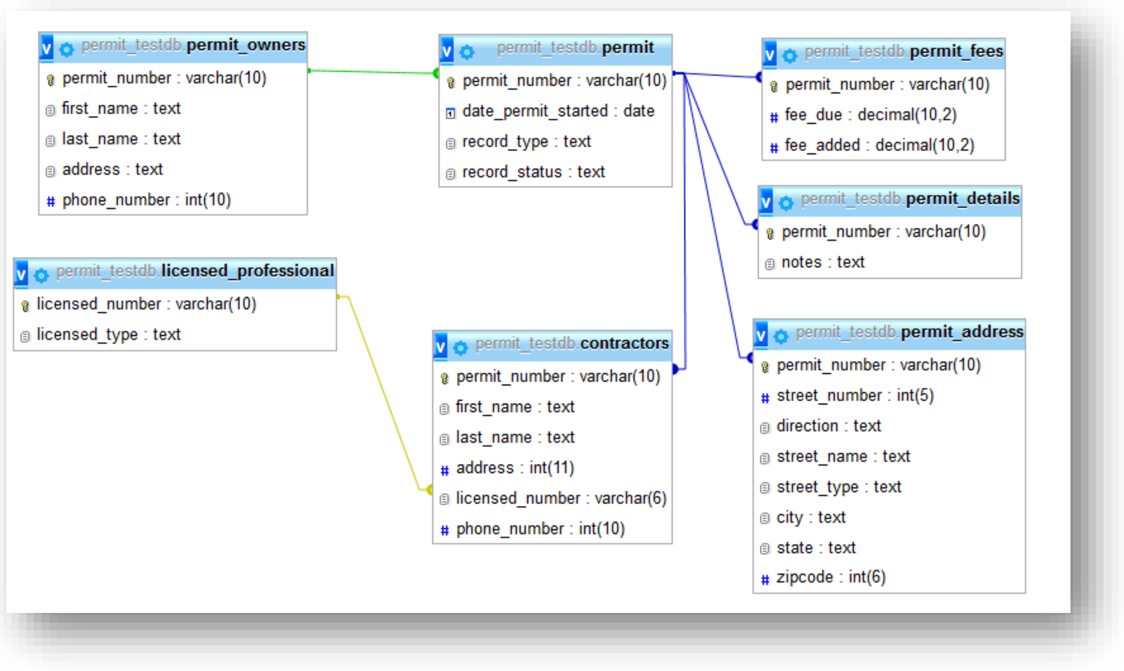
Building and Safety (simplified) Database Table Descriptions

B1Permit	Main table that includes permit basics like: date permit started, permit number, record status, record type
B3Owners	Owners associated to the permit and their address or phone number information
B3Contra	Contractors associated to the permit and their address or phone number information
B3Contact	Contacts such as applicant, architect or tenant plus their address or phone number information
GProcess	Current workflow step with due dates
GProcess_History	All of the history associated to the workflow with who did each step, what date and status of that step
F4FeeItem	Fees that have been added to the permit
F4Payment	Payments that have happened on the permit
B6Condit	Conditions that would hold or stop a permit
G6Action	Inspections associated with the permit
BWorkDes	Work Description for the permit
BPermit_Detail	More details associated with the permit
B3Address	Address information associated with the permit

Mobile App Database

Trying to stay as close as possible to create a simple database for the Android database that is quick and efficient in the mobile environment can become a difficult task because you want to create a database that will not only support the current features you want to design but also support possible new features in the future. After much design trial and errors on creating a feasible working database, the ER diagram shown below was deemed a simple but workable database. and it was created. The database will retain information about building, mechanic, plumbing and electrical permits for search purposes. The mobile app will utilize the permit information stored in the database and display it in an organized way for the user. The users will be able to search by permit, license number and pertinent data will be displayed; including the permit number, license number (if applicable), address and status of the permit. The system is mainly targeting citizens of Lincoln who have applied for a permit or just simply want to find publicly available information. Since this project is a mobile application, we are going to restrict the time frame of the permits to the last 3 years and the accessible data will be updated on a daily basis because permits are taken out daily.

First Draft ERD


permit_testdb permit_owners
permit_number : varchar(10)
first_name : text
last_name : text
address : text
phone_number : int(10)
permit_testdb permit
permit_number : varchar(10)
date_permit_started : date
record_type : text
record_status : text
permit_testdb permit_fees
permit_number : varchar(10)
fee_due : decimal(10,2)
fee_added : decimal(10,2)
permit_testdb permit_details
permit_number : varchar(10)
notes : text
permit_testdb licensed_professional
licensed_number : varchar(10)
licensed_type : text
permit_testdb contractors
permit_number : varchar(10)
first_name : text
last_name : text
address : int(11)
licensed_number : varchar(6)
phone_number : int(10)
permit_testdb permit_address
permit_number : varchar(10)
street_number : int(5)
direction : text
street_name : text
street_type : text
city : text
state : text
zipcode : int(6)

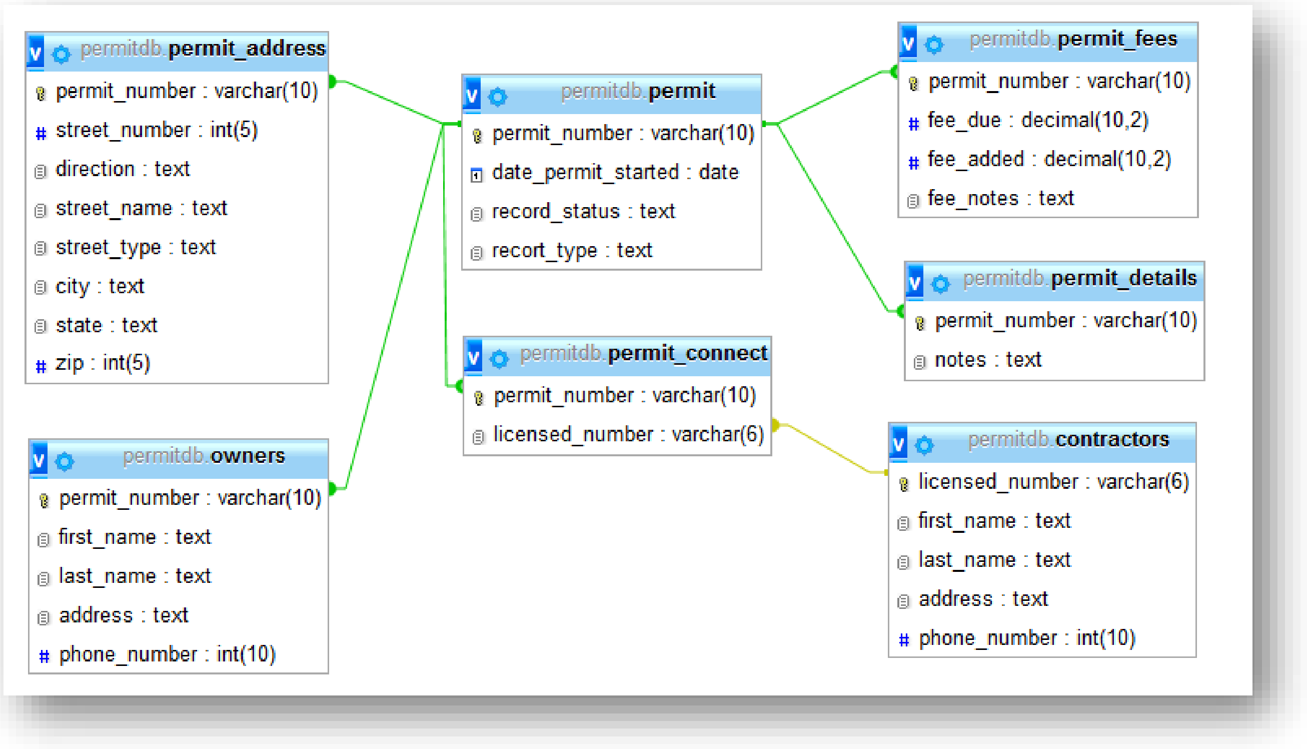
In my first attempt to consolidate the database I decided to have seven tables:

1. Permit table: contains the permit number, the date it started, the record type and the record status.
2. Permit_fees table: contains the permit number, the fees due and fees added to that particular permit
3. Permit_address: contains the permit number and address of the permit.
4. Permit_details table: contains the permit number and any comments or notes that go with that table.

5. Contractors table: contains the permit number, the licensed number of the contractor, contractors name, address and phone number
6. Licensed_professional table: contains the license number and type of license
7. Permit_owners table: contains the permit number, name, address and phone number owner.

After looking at the tables and trying to make a connection, the realization came that this table is not efficient and that some of the tables can be combine to make the queries quicker.

Second Draft ERD



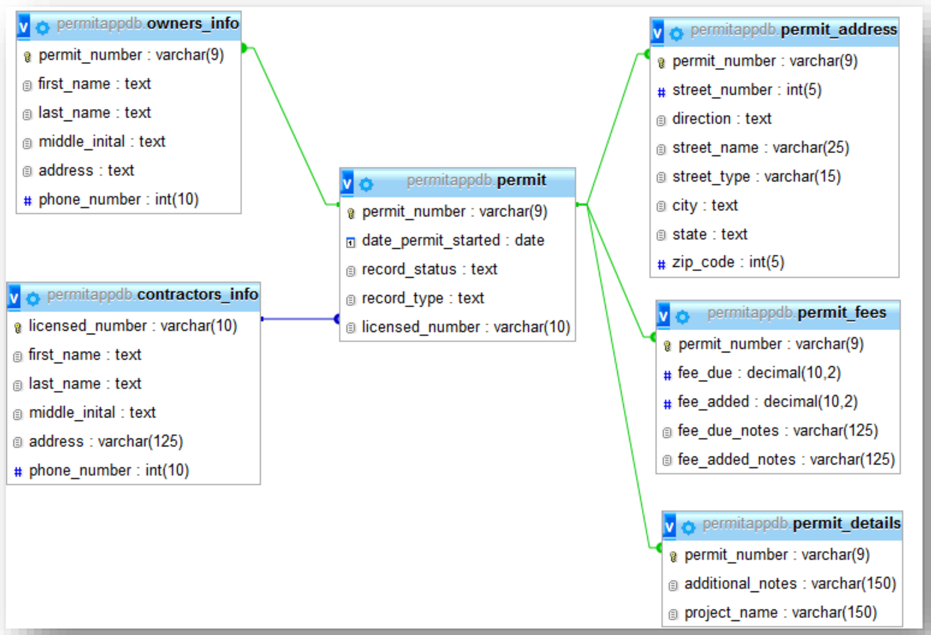
In my second attempt to consolidate the database resulted in 7 tables:

1. Permit table: contains the permit number, the date it started, the record type and the record status.

2. Permit_fees table: contains the permit number, the fees due and fees added to that particular permit
3. Permit_address: contains the permit number and address of the permit.
4. Permit_details table: contains the permit number and any comments or notes that go with that table.
5. Contractors table: contains the licensed number of the contractor, contractors name, address and phone number
6. Permit_connect table: contains the permit number and license number (which replaced Licensed_professional table)
7. Permit_owners table: contains the permit number, name, address and phone number owner.

My second attempt had replacing licensed_professional table with permit_connect. After looking at the tables and trying to make the connections between the tables, the realization came that this table is still not efficient and that some of the tables can still be combine to make the queries quicker.

Third ERD


permitappdb owners_info
permit_number : varchar(9)
first_name : text
last_name : text
middle_inital : text
address : text
phone_number : int(10)
permitappdb contractors_info
licensed_number : varchar(10)
first_name : text
last_name : text
middle_inital : text
address : varchar(125)
phone_number : int(10)
permitappdb permit
permit_number : varchar(9)
date_permit_started : date
record_status : text
record_type : text
licensed_number : varchar(10)
permitappdb permit_address
permit_number : varchar(9)
street_number : int(5)
direction : text
street_name : varchar(25)
street_type : varchar(15)
city : text
state : text
zip_code : int(5)
permitappdb permit_fees
permit_number : varchar(9)
fee_due : decimal(10,2)
fee_added : decimal(10,2)
fee_due_notes : varchar(125)
fee_added_notes : varchar(125)
permitappdb permit_details
permit_number : varchar(9)
additional_notes : varchar(150)
project_name : varchar(150)

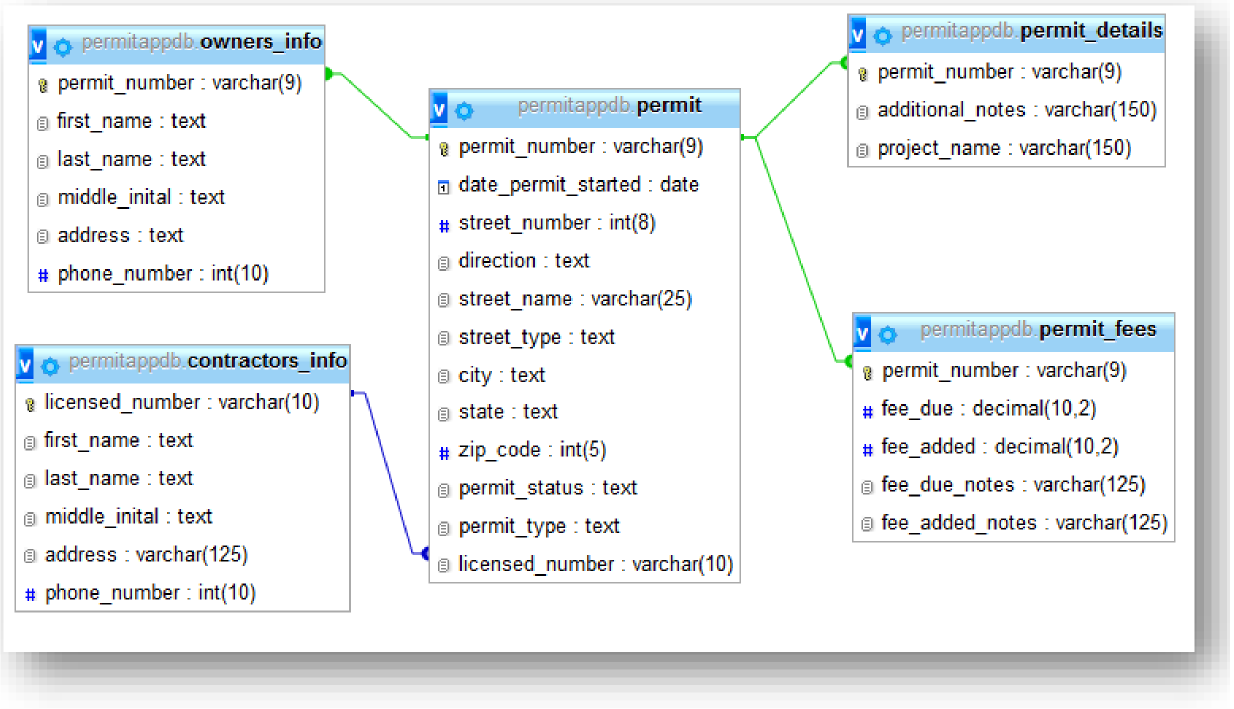
In my third attempt to consolidate the database led me to have 6 tables:

1. Permit table: contains the permit number, the date it started, the record type and the record status and licensed number.
2. Permit_fees table: contains the permit number, the fees due and fees added to that particular permit
3. Permit_address: contains the permit number and address of the permit.
4. Permit_details table: contains the permit number and any additional notes and project name if the permit has one.
5. Contractors table: contains the licensed number of the contractor, contractors name, address and phone number

6. Permit_owners table: contains the permit number, name, address and phone number owner.

My third attempt at the database got rid permit_connect all together and instead added licensed_number table to connect the licensed number to the main permit table. However, again, after looking at the tables and trying to make a connection, the realization came that this table structure is not efficient and that some of the tables could be combined to make the queries even quicker.

Final ERD



permitappdb.owners_info
- permit_number : varchar(9)
- first_name : text
- last_name : text
- middle_inital : text
- address : text
- phone_number : int(10)

permitappdb.contractors_info
- licensed_number : varchar(10)
- first_name : text
- last_name : text
- middle_inital : text
- address : varchar(125)
- phone_number : int(10)

permitappdb.permit
- permit_number : varchar(9)
- date_permit_started : date
- street_number : int(8)
- direction : text
- street_name : varchar(25)
- street_type : text
- city : text
- state : text
- zip_code : int(5)
- permit_status : text
- permit_type : text
- licensed_number : varchar(10)

permitappdb.permit_details
- permit_number : varchar(9)
- additional_notes : varchar(150)
- project_name : varchar(150)

permitappdb.permit_fees
- permit_number : varchar(9)
- fee_due : decimal(10,2)
- fee_added : decimal(10,2)
- fee_due_notes : varchar(125)
- fee_added_notes : varchar(125)

Table Descriptions (4th ERD)

Permit	
permit_number	primary key
date_permit_started	Date the permit was started (applied for)
street_number	Street number of where permit is taking place
Direction	Direction of street where permit is taking place
Street_name	Street Name of where the permit is taking place
Street_type	Street Type of where the permit is taking place
City	Needs to be set to Lincoln
State	Needs to be set to Nebraska
Zip_Code	5 digit code
Permit_status	3 possible options: Denied, Approved, Pending
Permit_type	Type of Permit: City Licensed Mechanical, City Licensed Electrical, City Licensed Plumbing, City Non-Licensed Mechanical, City Non-Licensed Electrical, City Non-Licensed Plumbing.
Licensed_number	Foreign Key: this relates Contractors information table, this field can be NULL because not all permits will require a contractor
Permit_Details	
Permit_number	primary key
Additional_notes	Additional notes about the project itself, for instance what kind of work is being done or the parcel number of the l and on which the building is located
Project_name	Name of the project if there is one
Permit_Fees	
permit_number	primary key
fee_due	The fee amount
Fee_added	Any additional fee that has been added
Fee_due_notes	Description of the fee due
Fee_added_notes	Detail/Description of the additional fee added
Owners_info	
Permit_number	primary key
First_name	First name of property owner
Last_name	Last name of property owner
middle_inital	Middle initial of property owner (can be null)
Address	Address of property owner: the owner can own many properties but reside somewhere else
Phone_number	Phone number of property owner (can be null)
Contractors_info	

Licensed_number	primary key: this connects the contractors info to the main permit table (cannot be null)
first_name	First name of contractor
last_name	Last name of contractor
middle_inital	Middle initial of contractor (can be null)
Address	Address of contractor (doesn't have to be home address, could be address of work building)
phone_number	Contractors phone number (can be null)

Database Design Conclusion

Creating a database is not as simple as seems because you have to figure out how each table connects to each other, and since this database will be used to run the back-end of a mobile app you want it to be not only efficient but also that the search query does not take forever to load in the app. Customers do not like to be kept waiting they want the search time kept to a minimum. It can take several times to figure out what will produce the best results and even when you figure out one that works, changes can continue to be made to make it more efficient. It took me about 2 weeks to figure and try out different types of database and how to make the query work with the search. Currently the mobile app does not use all the tables but like most projects, you have to figure out what might be needed in the future enhancements.

Appendix B:
Storyboard and Wireframe

Wireframe

This appendix provided a rough sketch of how I wanted the mobile app to function. When you start on a project it is a good idea to create a wireframe and storyboard because it helps build a roadmap that you can refer to throughout the project and helps give the customer a general idea of what the app would look like.

There will be a main page where the users select three options and each option will take them to a different search that will ultimately display the results of the search.

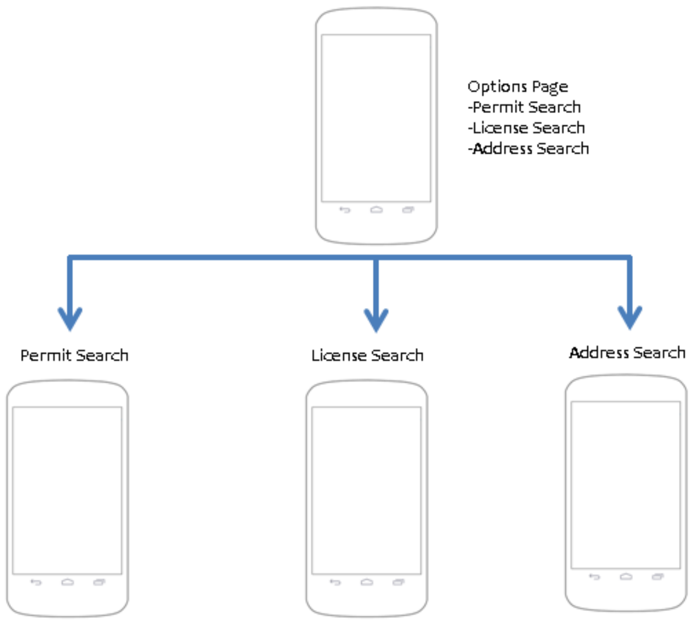


Figure 1: Mobile App Wire Frame

Back End Diagram

The mobile app's main function is a search feature, so there needs to be a database and a way for the app to connect to that database to retrieve the desired information. Since this is only a prototype and will be used for demonstration purpose there needs to be way to retrieve information from a development environment.

The diagram shows the front end will connect to the backend and how the device will send and receive information. Using PHP to connect the app to the MySQL database will be the method used in this development. Below is a simple wire frame for that connection.

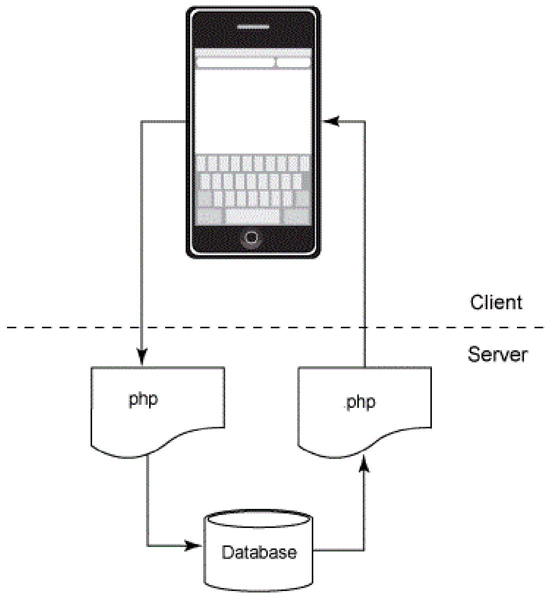


Figure 2: Back End Diagram

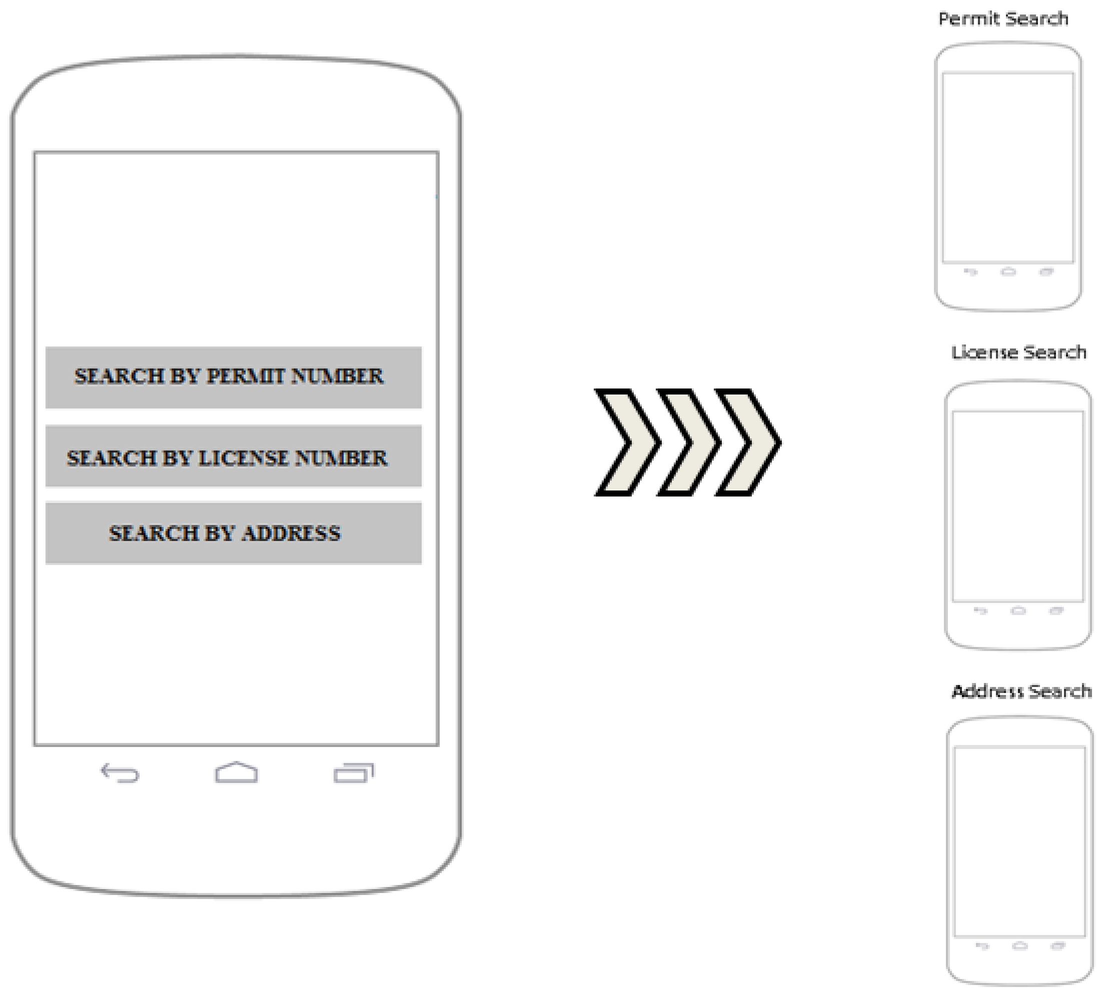
Story Board

Sketches will be used as a visual foundation and reference for the interface design. Here is where the visualization of the main features and layout of the structure of the application takes place.

Main Screen Activity (Search Options):

The diagram below represents the search option screen where users can select how they want to search for the permit information. They are given 3 options. The first option is “search by permit number”; this option will take the user to the permit search screen where they can search for information pertaining to that permit number. The second option is “search by license number”; this option will take the user to the permit search screen, where if the user knows a license number they pull all the permits that relate to the license number. The third and final option is “search by address”; this option will take the user to the address search screen where users can search for permit information tied to a particular address.

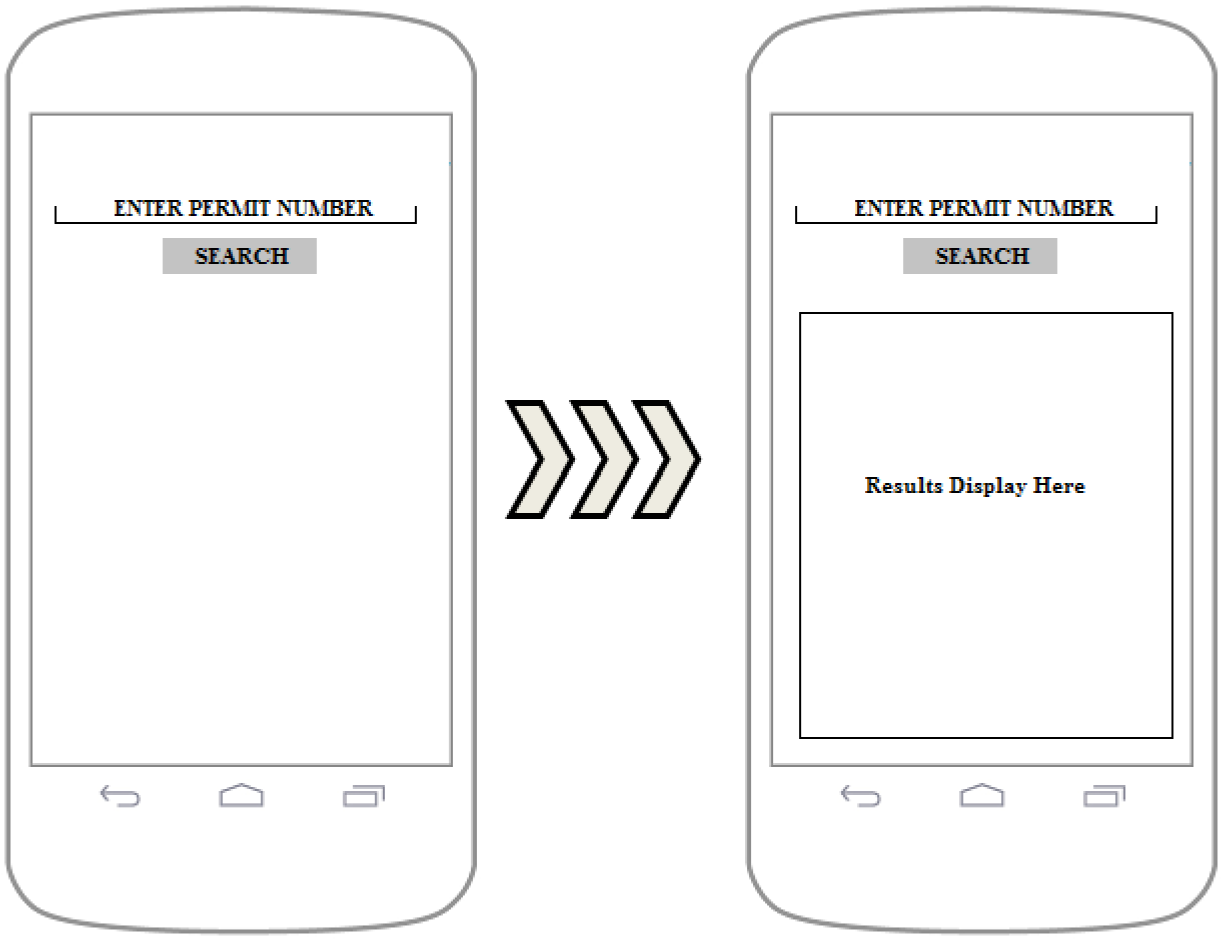
Figure 3: Main Screen Activity (Search Options) Diagram



Permit Search Activity Screen

The diagram below shows the search by permit number screen. The user must know the exact permit number to actually use this screen. If the user enters the correct permit number, then results of that search will be displayed below the search button. The search will display the permit number, the address of the permit that it corresponds to, and any contractors (or any other people associated with permit).

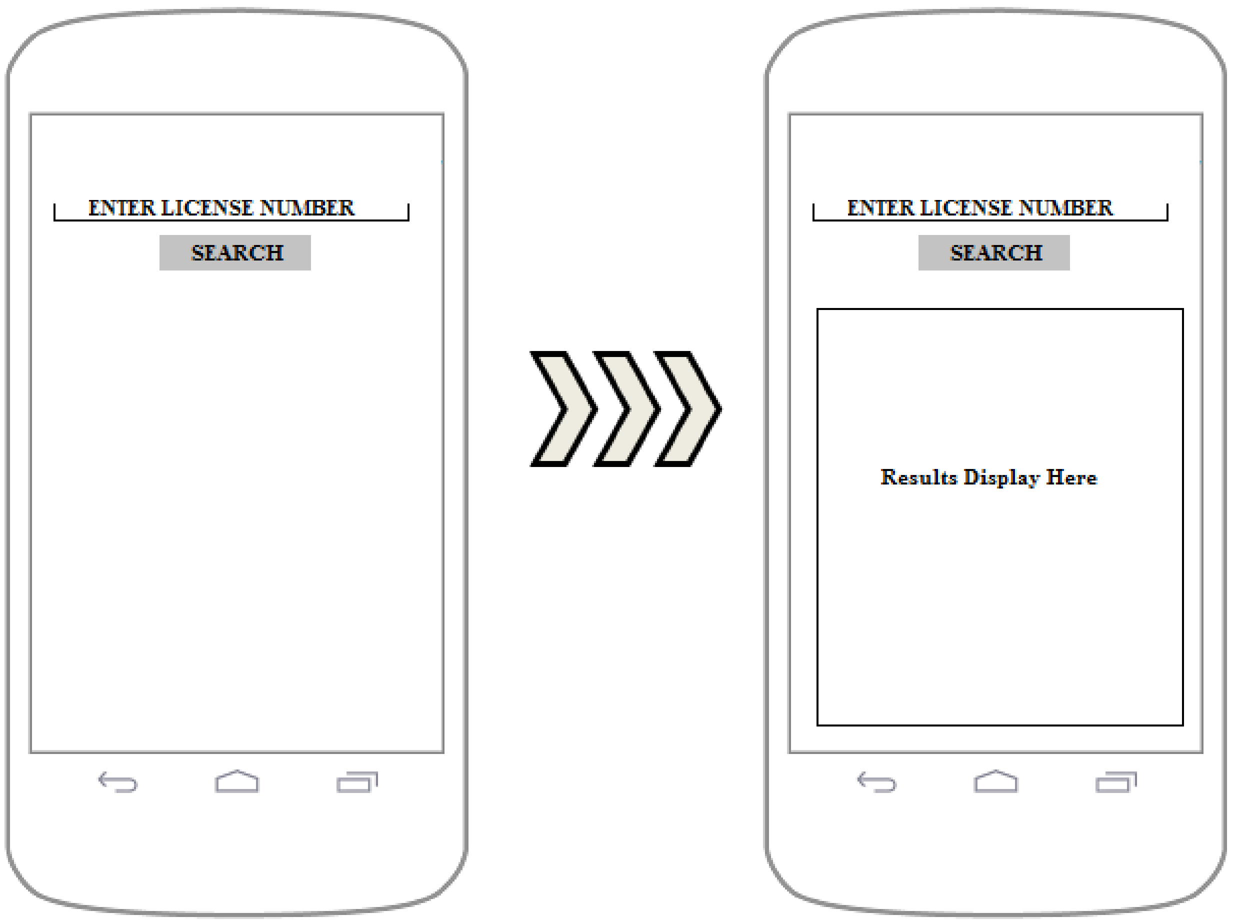
Figure 4: Permit Search Activity Diagram



License Search Screen

The diagram below shows the search license number screen. The user must know the exact license number to actually use this screen. If the user enters the correct license number, then results of that search will be displayed below the search button. The search will display the permit number, the address of the permit that it corresponds to, the contractors (or any other people associated with permit).

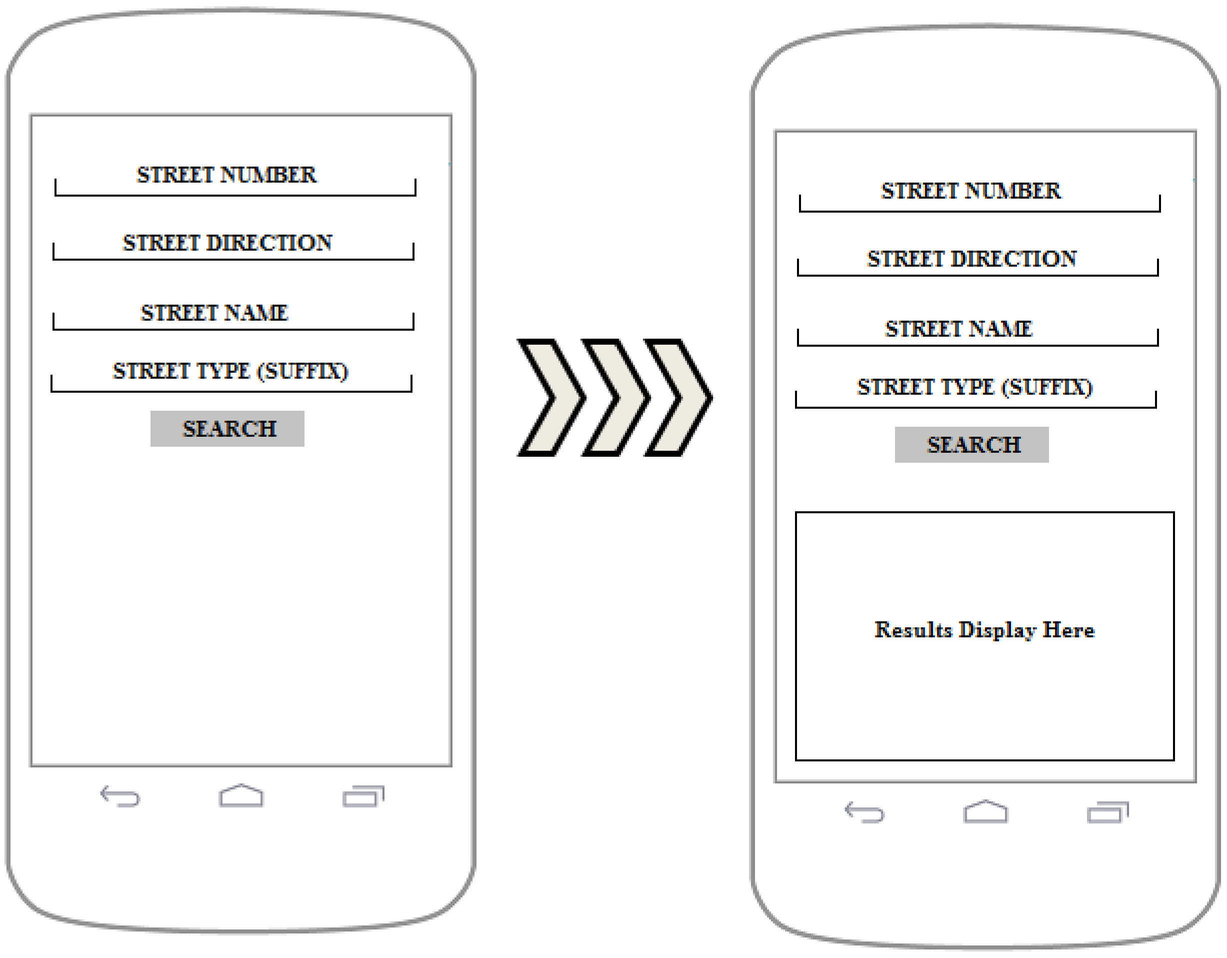
Figure 5: License Search Activity Diagram



Address Search Screen

The diagram below shows the search by address screen. Here the user must enter the street number, street direction, street name and street type. If the user enters correctly the fields correctly, the results of that search will be displayed below the search button. The search will display the permit number, the address of the permit that it corresponds to, the contractors (or any other people associated with permit).

Figure 6: Address Search Activity Diagram

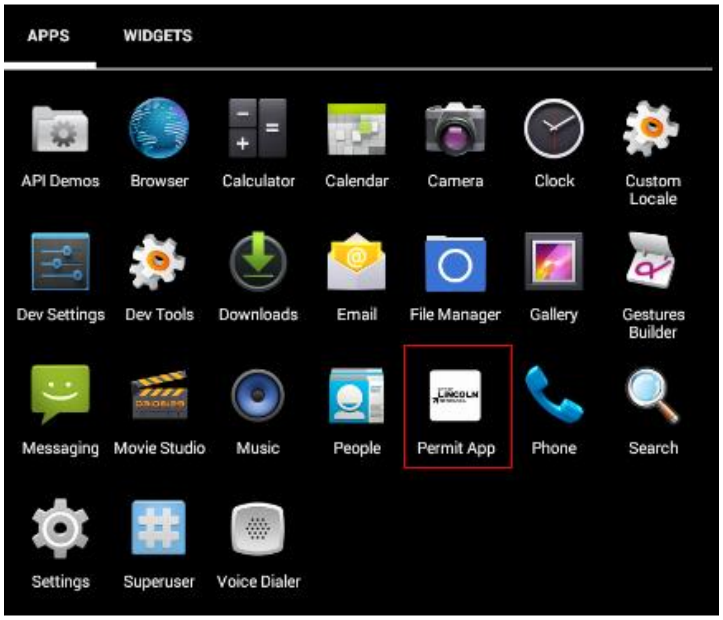


Conclusion

Before starting any project it is a good idea to provide a general sketch of what the app will look like and do to the stakeholders. By providing this, it will allow the stakeholders to change or add additional things that they want for the app. Most projects start off with the initial design so the developers have a general idea of how the app will look and how it should function. Throughout the project the initial design will change in minor or drastic ways. Once the development actually starts and the app is being tested by the stakeholders, new ideas or additional features that weren't initial thought of will come to light during that time. So, hashing out a good initial design that meets the stakeholder's requirements is vital to any project.

Appendix C:
Mobile App Code

Mobile App Icon



Below are files that make up the App Icon

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--This is the root directory. The manifest file presents essential information
about the app to the Android system, information the system must have before it can run any of the
app's code.-->

<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="permit.com.permit">

<!-- this permission statement is required, if not the app won't be able to connect to the network-->
   <uses-permission android:name="android.permission.INTERNET"/>"

<!-- allowBackup: allows the app to be backup and have infrastructure to restore
Contains attributes like icon, label
icon: reference the app_logo that is located in the mipmap folder
label: label of app
supportsRt1: app willing to support right-to-left layouts-->
   <application
      android:allowBackup="true"
      android:icon="@mipmap/app_logo"
      android:label="@string/app_name"
      android:supportsRtl="true">

<!--Activities must be declared and if it is not then they will not be seen by the system and will never
be run. -->
```

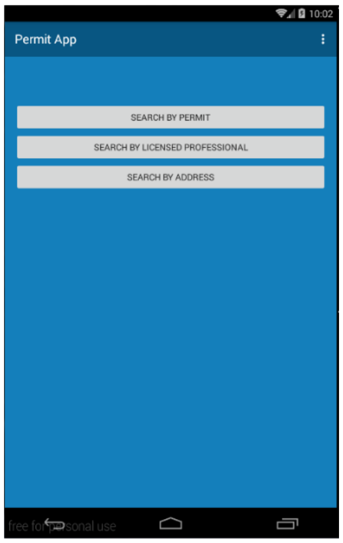
```
    <!-- Declares the Main Activity -->
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <!-- Declares the Permit Activity -->
    <activity
      android:name=".PermitActivity"
      android:label="@string/title_activity_permit"
      android:theme="@style/AppTheme.NoActionBar" />

    <!-- Declares the License Activity -->
    <activity
      android:name=".LicenseActivity"
      android:label="@string/title_activity_license"
      android:theme="@style/AppTheme.NoActionBar" />

    <!-- Declares the Address Activity -->
    <activity
      android:name=".AddressActivity"
      android:label="@string/title_activity_address"
      android:theme="@style/AppTheme.NoActionBar">

    </activity>
  </application>
</manifest>
```

Strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--All string resources with other simple resources in the one XML file, and is reference using the
value provide in the name attribute-->

<resources>
  <string name="app_name">Permit App</string>
  <string name="action_settings">Settings</string>
  <string name="title_activity_permit">Permit Activity</string>
  <string name="title_activity_license">License Activity</string>
  <string name="title_activity_address">Address Activity</string>

  //string resource used in content_main.xml
  <string name="SearchPermit">Search by Permit</string>
```

```
    <string name="SearchLicense">Search by Licensed Professional</string>
    <string name="SearchAddress">Search by Address</string>

    //string resource used in content_permit.xml
    <string name="askPermit">Enter Permit Number</string>
    <string name="askPermitButton">Search</string>
    <string name="permitMessage">Enter Permit Number</string>

    //string resource used in content_license.xml
    <string name="askLicense">Enter Licensed Number</string>
    <string name="askLicenseButton">Search</string>
    <string name="licenseMessage">Enter License Number</string>

    //string resource used in content_address.xml
    <string name="streetNum">Street Number</string>
    <string name="streetDir">Street Direction</string>
    <string name="streetName">Street Name</string>
    <string name="streetType">Street Type</string>
    <string name="askAddressButton">Search</string>
</resources>
```

Main Screen Activity



Below are files that make up the Main Activity Screen

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This xml determines how the look of the main activity should be -->

<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:fitsSystemWindows="true"
   tools:context="permit.com.permit.MainActivity">

   <android.support.design.widget.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:theme="@style/AppTheme.AppBarOverlay">

      <android.support.v7.widget.Toolbar
         android:id="@+id/toolbar"
         android:layout_width="match_parent"
         android:layout_height="?attr/actionBarSize"
         android:background="?attr/colorPrimary"
         app:popupTheme="@style/AppTheme.PopupOverlay" />
      </android.support.design.widget.AppBarLayout>

   <!-- reference the content found in content_main.xml -->
   <include layout="@layout/content_main" />
</android.support.design.widget.CoordinatorLayout>
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--
This xml determines the contents in the activity_main: contents that will user sees
 android:background="@color/colorBackground" : reference color from colors.xml
 tools:showIn="@layout/activity_main": will display contents in activity_main.xml
-->
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:background="@color/colorBackground"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context="permit.com.permit.MainActivity"
   tools:showIn="@layout/activity_main">
<!--
Permit Search Button:
android:onClick="PermitSearch": method name that will call when user clicks on button
```

```
android:text="@string/SearchPermit" : reference string value from strings.xml
-->
  <android.support.v7.widget.AppCompatButton
      android:layout_marginTop="57dp"
      android:id="@+id/permitSearchButton"
      android:text="@string/SearchPermit"
      android:onClick="PermitSearch"
      android:padding="12dp"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textSize="14sp" />

<!--
License Search Button:
android:onClick="LicenseSearch": method name that will call when user clicks on button
android:text="@string/SearchLicense" : reference string value from strings.xml
-->
   <android.support.v7.widget.AppCompatButton
       android:id="@+id/licenseSearchButton"
       android:text="@string/SearchLicense"
       android:onClick="LicenseSearch"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_below="@+id/permitSearchButton"
       android:padding="12dp"
       android:layout_alignParentLeft="true"
       android:layout_alignParentStart="true"
       android:textSize="14sp" />

<!--
Address Search Button:
android:onClick="AddressSearch" method name that will call when user clicks on button
android:text="@string/SearchAddress" : reference string value from strings.xml
-->

   <android.support.v7.widget.AppCompatButton
       android:id="@+id/addressSearchButton"
       android:text="@string/SearchAddress"
       android:onClick="AddressSearch"
       android:padding="12dp"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_below="@+id/licenseSearchButton"
       android:layout_centerVertical="true"
       android:layout_alignParentLeft="true"
       android:layout_alignParentStart="true"
       android:textSize="14sp" />
</RelativeLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--All string resources with other simple resources in the one XML file, and is reference using the
value provide in the name attribute-->
<resources>
  <string name="app_name">Permit App</string>
  <string name="action_settings">Settings</string>
  <string name="title_activity_permit">Permit Activity</string>
  <string name="title_activity_license">License Activity</string>
  <string name="title_activity_address">Address Activity</string>

  //string resource used in content_main.xml
  <string name="SearchPermit">Search by Permit</string>
  <string name="SearchLicense">Search by Licensed Professional</string>
  <string name="SearchAddress">Search by Address</string>

  //string resource used in content_permit.xml
  <string name="askPermit">Enter Permit Number</string>
  <string name="askPermitButton">Search</string>
  <string name="permitMessage">Enter Permit Number</string>

  //string resource used in content_license.xml
  <string name="askLicense">Enter Licensed Number</string>
  <string name="askLicenseButton">Search</string>
  <string name="licenseMessage">Enter License Number</string>

  //string resource used in content_address.xml
  <string name="streetNum">Street Number</string>
  <string name="streetDir">Street Direction</string>
  <string name="streetName">Street Name</string>
  <string name="streetType">Street Type</string>
  <string name="askAddressButton">Search</string>
</resources>
```

MainActivity.java

```
package permit.com.permit;

//imports that class that this java file uses
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
//end imports

public class MainActivity extends AppCompatActivity {
```

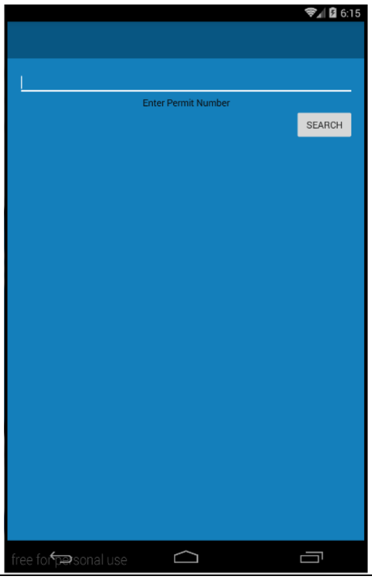
```
@Override
//initialize the activity
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  //call the activity_main layout resource to define the UI
  setContentView(R.layout.activity_main);
  //retrieve toolbar widget from activity_main.xml
  Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
  //method to designate the toolbar as the action bar for Activity
  setSupportActionBar(toolbar);
}

//Method called when user clicks on permit search button from content_main.xml
public void PermitSearch(View view){
  //start permit activity
  Intent intent = new Intent(this,PermitActivity.class);
  startActivity(intent);
}

//Method called when user clicks on license search button from content_main.xml
public void LicenseSearch(View view){
  //start license activity
  Intent intent = new Intent(this,LicenseActivity.class);
  startActivity(intent);
}

//Method called when user clicks on address search button from content_main.xml
public void AddressSearch(View view){
  //start address activity
  Intent intent = new Intent(this,AddressActivity.class);
  startActivity(intent);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu; this adds items to the action bar if it is present.
  getMenuInflater().inflate(R.menu.menu_main, menu);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle action bar item clicks here. The action bar will
  // automatically handle clicks on the Home/Up button, so long
  // as you specify a parent activity in AndroidManifest.xml.
  int id = item.getItemId();

  //noinspection SimplifiableIfStatement
```

```
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--This is the root directory. The manifest file presents essential information
about the app to the Android system, information the system must have before it can run any of the
app's code.-->

<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="permit.com.permit">

<!--this permission statement is required, if not the app won't be able to connect to the network-->
    <uses-permission android:name="android.permission.INTERNET"/>"

<!-- allowBackup: allows the app to be backup and have infrastructure to restore
Contains attributes like icon, label
icon: reference the app_logo that is located in the mipmap folder
label: label of app
supportsRt1: app willing to support right-to-left layouts-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/app_logo"
        android:label="@string/app_name"
        android:supportsRtl="true">

<!--Activities must be declared and if it is not then they will not be seen by the system and will never
be run. -->

        <!-- Declares the Main Activity -->
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    <!-- Declares the Permit Activity -->
    <activity
      android:name=".PermitActivity"
      android:label="@string/title_activity_permit"
      android:theme="@style/AppTheme.NoActionBar" />

    <!-- Declares the License Activity -->
    <activity
      android:name=".LicenseActivity"
      android:label="@string/title_activity_license"
      android:theme="@style/AppTheme.NoActionBar" />

    <!-- Declares the Address Activity -->
    <activity
      android:name=".AddressActivity"
      android:label="@string/title_activity_address"
      android:theme="@style/AppTheme.NoActionBar">

    </activity>
  </application>
</manifest>
```

Permit Search Activity



init.php

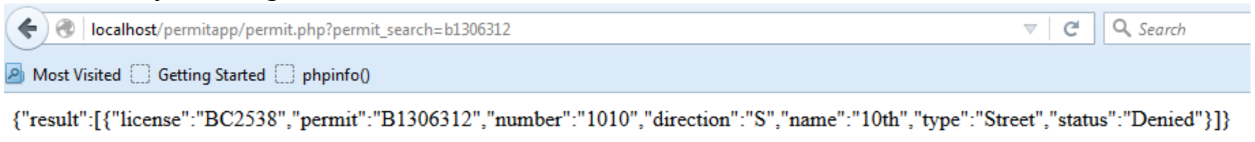
```
<?php

 /* Script to Connect to MySQL database
database name: permitappdb*/

 //Setup MySQL connection variables: defining Constants
 $db_name = "permitappdb";
 $mysql_user = "root";
 $mysql_pass = "root";
 $server_name = "localhost";

 //create connection  to database
 $con = mysqli_connect($server_name, $mysql_user, $mysql_pass, $db_name);

  //check connection
 if(!$con)
 {
	// echo "Connection Error...".mysqli_connect_error();
 }
 else
 {
	//echo "<h3>Database connection Success...</h3>";
 }
 ?>
```

permit.php

```
<?php

/**
	Script to handle permit search request.
	To get the permit search detail we use JSON.
	This script will get the permit number as an http request and will display the permit search
	details in json according to the given permit number
*/

//import init.php script:  db connection script
 require "init.php";

 //getting value
$permit_number = $_GET['permit_search'];

 // creating an sql query (setup search query for permit)
$sql_query = "SELECT *  from permit where permit_number = '$permit_number' ;";

  //run the query against the database
$r = mysqli_query($con,$sql_query);

//create a blank array
$result = array();

//looping through all the records
while ($row = mysqli_fetch_assoc($r)){

//The array_push() function inserts one or more elements to the end of an array
array_push($result,array(
/**specifies the value to add to the array:
 pushing license, permit, number, direction, name,type, status in the blank array created */
"license"=>$row['licensed_number'],
"permit"=>$row['permit_number'],
"number"=>$row['street_number'],
"direction"=>$row['direction'],
"name"=>$row['street_name'],
"type"=>$row['street_type'],
"status"=>$row['permit_status'],
 ));
}

//displaying the array in json format
 echo json_encode (array("result"=>$result));

 //closing the database
mysqli_close($con);

 ?>
```

We go to this url: localhost/permitapp/permit.php?permit_search=b1306312 to make sure we can pass the request [?permit_search=permit_number]. Wamp is the server so that is why it is working on a localhost.

Below is the json string



```
{"result":[{"license":"BC2538","permit":"B1306312","number":"1010","direction":"S","name":"10th","type":"Street","status":"Denied"}]}
```

activity_permit.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This xml determines how the look of the main activity should be -->
<android.support.design.widget.CoordinatorLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:fitsSystemWindows="true"
   tools:context="permit.com.permit.PermitActivity">

   <android.support.design.widget.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:theme="@style/AppTheme.AppBarOverlay">
```

```
     <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

  </android.support.design.widget.AppBarLayout>
  <!-- reference the content found in content_permit.xml -->
  <include layout="@layout/content_permit" />
</android.support.design.widget.CoordinatorLayout>
```

content_permit.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This xml determines the contents in the activity_permit contents that will user sees
 android:background="@color/colorBackground" : reference color from colors.xml
 tools:showIn="@layout/activity_permit": will display contents in activity_permit.xml
-->
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:background="@color/colorBackground"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context="permit.com.permit.PermitActivity"
  tools:showIn="@layout/activity_permit"
  android:weightSum="1">

  <LinearLayout
     android:orientation="horizontal"
     android:layout_width="match_parent"
     android:layout_height="wrap_content">

 <!-- Text Area where user enters permit number-->
  <EditText
     android:id="@+id/permitID"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:layout_weight="1"
     android:inputType="none"
     android:singleLine="true"
```

```
        android:textSize="14dp" />

    </LinearLayout>
    <!--
    Text displaying message
    android:text="@string/permitMessage": reference string value from strings.xml
    -->
    <TextView android:id="@+id/permitMessage"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/permitMessage"
        android:gravity="center"
        android:singleLine="false"
        android:textColor="#0b0a0a"
        android:textSize="14sp"
        android:textIsSelectable="true" />

    <!--
    Permit Search Button:
    android:text="@string/askPermitButton" : reference string value from strings.xml
     -->
    <android.support.v7.widget.AppCompatButton
        android:id="@+id/permitButton"
        android:text="@string/askPermitButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="12dp"
        android:layout_gravity="right"
        android:textSize="14sp" />

<!--Area where result will display-->
    <TextView
        android:id="@+id/permitResult"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="0.79"
        android:textSize="14dp"
        android:textIsSelectable="false" />

</LinearLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--All string resources with other simple resources in the one XML file, and is reference using the value
provide in the name attribute-->
<resources>
    <string name="app_name">Permit App</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_permit">Permit Activity</string>
```

```
   <string name="title_activity_license">License Activity</string>
   <string name="title_activity_address">Address Activity</string>

   //string resource used in content_main.xml
   <string name="SearchPermit">Search by Permit</string>
   <string name="SearchLicense">Search by Licensed Professional</string>
   <string name="SearchAddress">Search by Address</string>

   //string resource used in content_permit.xml
   <string name="askPermit">Enter Permit Number</string>
   <string name="askPermitButton">Search</string>
   <string name="permitMessage">Enter Permit Number</string>

   //string resource used in content_license.xml
   <string name="askLicense">Enter Licensed Number</string>
   <string name="askLicenseButton">Search</string>
   <string name="licenseMessage">Enter License Number</string>

   //string resource used in content_address.xml
   <string name="streetNum">Street Number</string>
   <string name="streetDir">Street Direction</string>
   <string name="streetName">Street Name</string>
   <string name="streetType">Street Type</string>
   <string name="askAddressButton">Search</string>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--This is the root directory. The manifest file presents essential information
about the app to the Android system, information the system must have before it can run any of the
app's code.-->

<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="permit.com.permit">

<!-- this permission statement is required, if not the app won't be able to connect to the network-->
   <uses-permission android:name="android.permission.INTERNET"/>"

<!-- allowBackup: allows the app to be backup and have infrastructure to restore
Contains attributes like icon, label
icon: reference the app_logo that is located in the mipmap folder
label: label of app
supportsRt1: app willing to support right-to-left layouts-->
   <application
      android:allowBackup="true"
      android:icon="@mipmap/app_logo"
```

```
        android:label="@string/app_name"
        android:supportsRtl="true">

<!--Activities must be declared and if it is not then they will not be seen by the system and will never be
run. -->

        <!-- Declares the Main Activity -->
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Declares the Permit Activity -->
        <activity
            android:name=".PermitActivity"
            android:label="@string/title_activity_permit"
            android:theme="@style/AppTheme.NoActionBar" />

        <!-- Declares the License Activity -->
        <activity
            android:name=".LicenseActivity"
            android:label="@string/title_activity_license"
            android:theme="@style/AppTheme.NoActionBar" />

        <!-- Declares the Address Activity -->
        <activity
            android:name=".AddressActivity"
            android:label="@string/title_activity_address"
            android:theme="@style/AppTheme.NoActionBar">

        </activity>
    </application>
</manifest>
```

ConfigPermit.php

```
package permit.com.permit;
/**Declare important constants*/
public class ConfigPermit {

    //Address of permit script
    public static final String PERMIT_URL = "http://192.168.56.1/permitapp/permit.php?permit_search=";

    //Keys that will be used to send this request to php scripts
    public static final String KEY_PERMIT = "permit";

    //Key values from inner array
    public static final String KEY_NUMBER = "number";
    public static final String KEY_DIRECTION = "direction";
    public static final String KEY_NAME = "name";
    public static final String KEY_TYPE = "type";
    public static final String KEY_STATUS = "status";

    //JSON string
    public static final String JSON_ARRAY = "result";
}
```

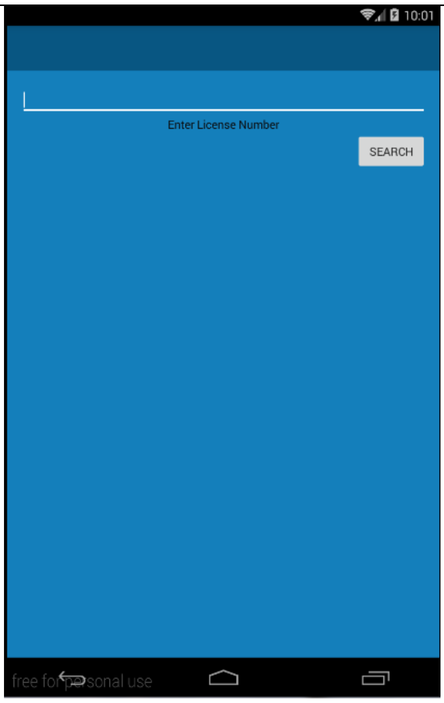
PermitActivity.java

```
package permit.com.permit;


//imports that class that this java file uses
import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

//end import

public class PermitActivity extends AppCompatActivity implements View.OnClickListener {

    //defining views
    private EditText permitID;
    private Button permitButton;
    private TextView permitResult;
    //define progressDialog class: displays dialog and spinner while search is going on in background
     private ProgressDialog loading;
```

```
@Override
//initialize the activity
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  //call the activity_permit layout resource to define the UI
  setContentView(R.layout.activity_permit);

  //initialize views
  permitID = (EditText)findViewById(R.id.permitID);
  permitButton = (Button)findViewById(R.id.permitButton);
  permitResult = (TextView)findViewById(R.id.permitResult);
  //set onclick listener to permit search button
  permitButton.setOnClickListener(this);
}

//Permit Search Function
public void getPermit(){
  //get the value the user enters
  String permit_search = permitID.getText().toString().trim();
  //if user does not enter any text
  if (permit_search.equals("")){
    //display this message
    Toast.makeText(this,"Please enter a permit number",Toast.LENGTH_LONG).show();
    return;
  }
  //start the dialog box
  loading = ProgressDialog.show(this,"Please wait...","Fetching...",false,false);

  //add search parameters to query URL that is called from ConfigPermit.java and create
  String url = ConfigPermit.PERMIT_URL+permitID.getText().toString().trim();

  /** Create String Request: Request string response from the provided URL* */
  StringRequest stringRequest = new StringRequest(url, new Response.Listener<String>(){
    @Override
    public void onResponse(String response){
      //make the dialog box close when search is complete
      loading.dismiss();
      //calls the JSON method
      showJSON(response);
    }
  },
      /**
       * callback method that an error has been occured with the provided error code
       * with message
       */
      new Response.ErrorListener(){
        @Override
        public void onErrorResponse(VolleyError error) {
              //display error message
              Toast.makeText(PermitActivity.this,"Not Working",Toast.LENGTH_LONG).show();
        }
      });
  /**
   *  Instantiate the RequestQueue
   *  request queue using Volley library: volley handles network connections
   *  Volley maps the JSON response in respective class objects, it allows you to add
   *  getter setter for the response model objects which allows you to access JSON values/
   *  parameters using .operator like normal JAVA object, which makes response handling simple
   */

  RequestQueue requestQueue = Volley.newRequestQueue(this);
  //adding the request to RequestQueue
  requestQueue.add(stringRequest);
}
//showJSON method: this is where the JSON is parse
private void showJSON(String response){
  //create constants
  String permit="";
  String number="";
  String name="";
  String direction="";
  String type="";
  String status="";
  String display="";

  //create a JSONObject
  JSONObject jsonObject = null;
  try {
    //get the JSON string
    jsonObject = new JSONObject(response);
  } catch (JSONException e) {
    Toast.makeText(this,"JSON object error",Toast.LENGTH_LONG).show();
    e.printStackTrace();
  }
  //create an JSONArray
  JSONArray result = null;
  try {
    //extract data array from json string: retrieve from ConfigPermit.java file
    result = jsonObject.getJSONArray(ConfigPermit.JSON_ARRAY);
  } catch (JSONException e) {
    //if the key is not found or if the value is not a JSONArray
    Toast.makeText(this,"JSON array error",Toast.LENGTH_LONG).show();
    e.printStackTrace();
  }

  //if there is nothing in the JSON array
```

```
      if (result.length()==0){
         //display message
         display="Permit Number Does Not Exist, Try Again";
      }

      //loop to get all json object from json array
      for (int i = 0; i < result.length();i++) {

         try {
            //retrieves the JSONObject with the associated index value
            JSONObject permitData = result.getJSONObject(i);
            //getting fields inside in the JSON array
            permit = permitData.getString(ConfigPermit.KEY_PERMIT);
            status = permitData.getString(ConfigPermit.KEY_STATUS);
            number = permitData.getString(ConfigPermit.KEY_NUMBER);
            name = permitData.getString(ConfigPermit.KEY_NAME);
            direction = permitData.getString(ConfigPermit.KEY_DIRECTION);
            type = permitData.getString(ConfigPermit.KEY_TYPE);

         } catch (JSONException e) {
            Toast.makeText(this,"error2",Toast.LENGTH_LONG).show();
            e.printStackTrace();
         }
         //add the found fields in display variable
         display+= "Permit:\t" + permit + "\nStreet Address:\t" + number + "\t" + direction + "\t" + name +
"\t" + type + "\t" + "\nPermit Status:\t" + status;
      }
   //display the result in the designated text area
   permitResult.setText(display);
   }

   @Override
   //When user clicks the button, call the getPermit Function
   public void onClick(View v) {
      getPermit();
   }
}
```

License Search Activity



init.php

```
<?php

 /* Script to Connect to MySQL database
database name: permitappdb*/

 //Setup MySQL connection variables: defining Constants
 $db_name = "permitappdb";
 $mysql_user = "root";
 $mysql_pass = "root";
 $server_name = "localhost";

 //create connection  to database
 $con = mysqli_connect($server_name, $mysql_user, $mysql_pass, $db_name);

  //check connection
 if(!$con)
 {
 	// echo "Connection Error...".mysqli_connect_error();
 }
 else
 {
 	//echo "<h3>Database connection Success...</h3>";
 }
 ?>
```

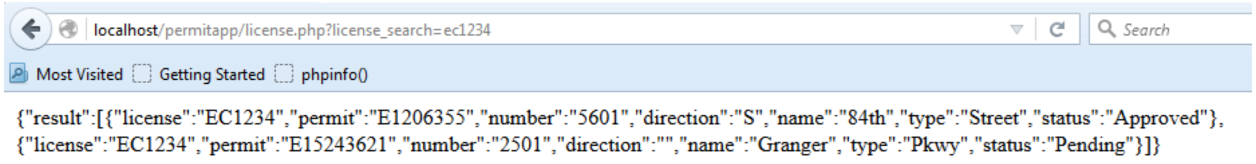
license.php

```
<?php
```

```
/**
		Script to handle license search request.
		To get the license search detail we use JSON.
		This script will get the license number as an http request and will display the license search
		details in json according to the given license number
*/

//import init.php script: db connection script
require "init.php";


 //getting value
$license_number = $_GET['license_search'];

 //creating sql query (setup search query for license)
$sql_query = "SELECT *  from permit where licensed_number = '$license_number' ;";


 //run the query against the database
$r = mysqli_query($con,$sql_query);

//create a blank array
$result = array();


//loop through the records
while ($row = mysqli_fetch_array($r)){
//The array_push() function inserts one or more elements to the end of an array
array_push($result,array(
/**specifies the value to add to the array:
 pushing license, permit, number, direction, name,type, status in the blank array created */

"license"=>$row['licensed_number'],
"permit"=>$row['permit_number'],
"number"=>$row['street_number'],
"direction"=>$row['direction'],
"name"=>$row['street_name'],
"type"=>$row['street_type'],
"status"=>$row['permit_status']
 ));

}
//display array in json format
echo json_encode (array("result"=>$result));

//close the connection
mysqli_close($con);
 ?>
```

We go to this url: localhost/permitapp/license.php?license_search=ec1234 to make sure we can pass the request [?license_search=license_number]. Wamp is the server so that is why it is working on a localhost.

Below is the json string



```
{"result":[{"license":"EC1234","permit":"E1206355","number":"5601","direction":"S","name":"84th","type":"Street","status":"Approved"},
{"license":"EC1234","permit":"E15243621","number":"2501","direction":"","name":"Granger","type":"Pkwy","status":"Pending"}]}
```

activity_license.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This xml determines how the look of the main activity should be -->
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="permit.com.permit.LicenseActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />
    </android.support.design.widget.AppBarLayout>
    <!-- reference the content found in content_permit.xml -->
    <include layout="@layout/content_license" />
</android.support.design.widget.CoordinatorLayout>
```

content_license.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This xml determines the contents in the activity_license contents that will user sees
 android:background="@color/colorBackground" : reference color from colors.xml
 tools:showIn="@layout/activity_license": will display contents in activity_license.xml
-->
```

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:background="@color/colorBackground"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context="permit.com.permit.LicenseActivity"
  tools:showIn="@layout/activity_license">

  <LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <!-- Text Area where user enters license number-->
    <EditText
      android:id="@+id/licenseID"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:inputType="text"
      android:textSize="14sp"
      android:singleLine="true" />

  </LinearLayout>
  <!--
 Text displaying message
 android:text="@string/licenseMessage": reference string value from strings.xml
 -->

  <TextView android:id="@+id/permitMessage"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/licenseMessage"
    android:gravity="center"
    android:singleLine="false"
    android:textColor="#0b0a0a"
    android:textSize="14sp"
    android:textIsSelectable="true" />

  <!--
  License Search Button:
  android:text="@string/askLicenseButton" : reference string value from strings.xml
   -->

   <android.support.v7.widget.AppCompatButton
       android:id="@+id/licenseButton"
       android:text="@string/askLicenseButton"
       android:padding="12dp"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="right" />

   <!--Area where result will display-->
   <TextView
       android:id="@+id/licenseResult"
       android:layout_width="fill_parent"
       android:layout_height="375dp"
       />

</LinearLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--All string resources with other simple resources in the one XML file, and is reference using the value
provide in the name attribute-->
<resources>
   <string name="app_name">Permit App</string>
   <string name="action_settings">Settings</string>
   <string name="title_activity_permit">Permit Activity</string>
   <string name="title_activity_license">License Activity</string>
   <string name="title_activity_address">Address Activity</string>

   //string resource used in content_main.xml
   <string name="SearchPermit">Search by Permit</string>
   <string name="SearchLicense">Search by Licensed Professional</string>
   <string name="SearchAddress">Search by Address</string>

   //string resource used in content_permit.xml
   <string name="askPermit">Enter Permit Number</string>
   <string name="askPermitButton">Search</string>
   <string name="permitMessage">Enter Permit Number</string>

   //string resource used in content_license.xml
   <string name="askLicense">Enter Licensed Number</string>
   <string name="askLicenseButton">Search</string>
   <string name="licenseMessage">Enter License Number</string>

   //string resource used in content_address.xml
```

```
  <string name="streetNum">Street Number</string>
  <string name="streetDir">Street Direction</string>
  <string name="streetName">Street Name</string>
  <string name="streetType">Street Type</string>
  <string name="askAddressButton">Search</string>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--This is the root directory. The manifest file presents essential information
about the app to the Android system, information the system must have before it can run any of the
app's code.-->

<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="permit.com.permit">

<!-- this permission statement is required, if not the app won't be able to connect to the network-->
  <uses-permission android:name="android.permission.INTERNET"/>"

<!-- allowBackup: allows the app to be backup and have infrastructure to restore
Contains attributes like icon, label
icon: reference the app_logo that is located in the mipmap folder
label: label of app
supportsRt1: app willing to support right-to-left layouts-->
  <application
    android:allowBackup="true"
    android:icon="@mipmap/app_logo"
    android:label="@string/app_name"
    android:supportsRtl="true">

<!--Activities must be declared and if it is not then they will not be seen by the system and will never be
run. -->

    <!-- Declares the Main Activity -->
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <!-- Declares the Permit Activity -->
    <activity
```

```
        android:name=".PermitActivity"
        android:label="@string/title_activity_permit"
        android:theme="@style/AppTheme.NoActionBar" />

      <!-- Declares the License Activity -->
      <activity
        android:name=".LicenseActivity"
        android:label="@string/title_activity_license"
        android:theme="@style/AppTheme.NoActionBar" />

      <!-- Declares the Address Activity -->
      <activity
        android:name=".AddressActivity"
        android:label="@string/title_activity_address"
        android:theme="@style/AppTheme.NoActionBar">

      </activity>
  </application>
</manifest>
```

ConfigLicense.php

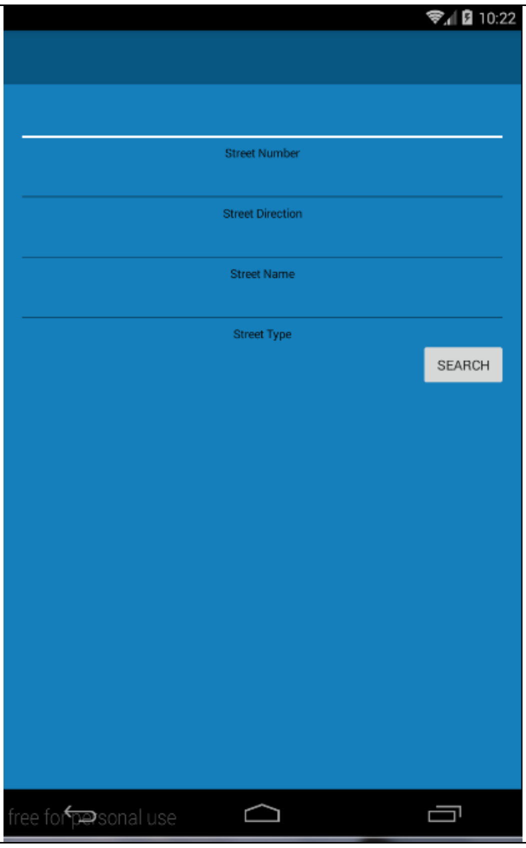
```
package permit.com.permit;

/**Declare important constant*/
public class ConfigLicense {
  //Address of permit script
  public static final String LICENSE_URL =
"http://192.168.56.1/permitapp/license.php?license_search=";

  //Keys that will be used to send this request to php script
  public static final String KEY_LICENSE = "license";
  //Keys values from inner array
  public static final String KEY_PERMIT = "permit";
  public static final String KEY_NUMBER = "number";
  public static final String KEY_DIRECTION = "direction";
  public static final String KEY_NAME = "name";
  public static final String KEY_TYPE = "type";
  public static final String KEY_STATUS = "status";
  //JSON string
  public static final String JSON_ARRAY = "result";

}
```

LicenseActivity.java

```
package permit.com.permit;

//imports that class that this java file uses
import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```java
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
//end import

public class LicenseActivity  extends AppCompatActivity implements View.OnClickListener {
    //defining views
    private EditText licenseID;
    private Button licenseButton;
    private TextView licenseResult;
    //define progressDialog class: displays dialog and spinner while search is going on in background
    private ProgressDialog loading;


    @Override
    //initialize the activity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //call the activity_license layout resource to define the UI
        setContentView(R.layout.activity_license);
        //initialize views
        licenseID = (EditText)findViewById(R.id.licenseID);
        licenseButton = (Button)findViewById(R.id.licenseButton);
        licenseResult = (TextView)findViewById(R.id.licenseResult);
        //set onclick listener to license search button
        licenseButton.setOnClickListener(this);
    }

  //License Search Function
  public void getLicense(){
    //get the value the user enters
    String license_search = licenseID.getText().toString().trim();
    //if user does not enter any text
    if (license_search.equals("")){
        //display this message
        Toast.makeText(this, "Please enter a license number", Toast.LENGTH_LONG).show();
        return;
    }
    //start the dialog box
    loading = ProgressDialog.show(this,"Please wait...","Fetching...",false,false);
    //add search parameters to query URL that is called from ConfigLicense.java and create
    String url = ConfigLicense.LICENSE_URL+licenseID.getText().toString().trim();
    /** Create String Request: Request string response from the provided URL* */
    StringRequest stringRequest = new StringRequest(url, new Response.Listener<String>(){
      @Override
      public void onResponse(String response){
        //make the dialog box close when search is complete
        loading.dismiss();
        //calls the JSON method
        showJSON(response);
      }
    },
        /**
         * callback method that an error has been occured with the provided error code
         * with message
         */
        new Response.ErrorListener(){
          @Override
          public void onErrorResponse(VolleyError error) {

Toast.makeText(LicenseActivity.this,error.getMessage().toString(),Toast.LENGTH_LONG).show();
          }
        });
    /**
     *  Instantiate the RequestQueue
     *  request queue using Volley library: volley handles network connections
     *  Volley maps the JSON response in respective class objects, it allows you to add
     *  getter setter for the response model objects which allows you to access JSON values/
     *  parameters using .operator like normal JAVA object, which makes response handling simple
     */
    RequestQueue requestQueue = Volley.newRequestQueue(this);
    //adding the request to RequestQueue
    requestQueue.add(stringRequest);
  }
  //showJSON method: this is where the JSON is parse
  private void showJSON(String response) {
    //create constants
    String license = "";
    String permit = "";
    String number = "";
    String name = "";
    String direction = "";
    String type = "";
    String status = "";
    String display="";
    //create a JSONObject
    JSONObject jsonObject = null;
    try {
      //get the JSON string
      jsonObject = new JSONObject(response);
    } catch (JSONException e) {
      Toast.makeText(this,"JSON object error",Toast.LENGTH_LONG).show();
      e.printStackTrace();
    }
    //create an JSONArray
    JSONArray result = null;
    try {
      //extract data array from json string: retrieve from ConfigLicense.java file
      result = jsonObject.getJSONArray(ConfigLicense.JSON_ARRAY);
    } catch (JSONException e) {
      Toast.makeText(this,"JSON array error",Toast.LENGTH_LONG).show();
      e.printStackTrace();
    }

    //if there is nothing in the JSON array
    if (result.length()==0){
      //Toast.makeText(this,"no result were found",Toast.LENGTH_LONG).show();
      display="License Number is not attached to any permit, Try Again";
    }
    //loop to get all json object from json array
    for (int i = 0; i < result.length(); i++) {
      try {
        //retrieves the JSONObject with the associated index value
        JSONObject licenseData = result.getJSONObject(i);
        //getting fields inside in the JSON array
        license = licenseData.getString(ConfigLicense.KEY_LICENSE);
        permit = licenseData.getString(ConfigLicense.KEY_PERMIT);
        status = licenseData.getString(ConfigLicense.KEY_STATUS);
        number = licenseData.getString(ConfigLicense.KEY_NUMBER);
        name = licenseData.getString(ConfigLicense.KEY_NAME);
        direction = licenseData.getString(ConfigLicense.KEY_DIRECTION);
        type = licenseData.getString(ConfigLicense.KEY_TYPE);

      } catch (JSONException e) {
        Toast.makeText(getBaseContext(),"not working",Toast.LENGTH_SHORT).show();
        e.printStackTrace();
      }
      //add the found fields in display variable
      display+= "License No: \t" + license + "\nPermit No:\t" + permit + "\nStreet Address:\t" + number
+ "\t" + direction + "\t" + name + "\t" + type + "\t" + "\nPermit Status:\t" + status + "\n\n";
    }
    //display the result in the designated text area
    licenseResult.setText(display);
    }

    @Override
    //When user clicks the button, call the getLicense Function
    public void onClick(View v) {
        getLicense();
    }
}
```

License Search Activity



init.php

```
<?php

 /* Script to Connect to MySQL database
database name: permitappdb*/

 //Setup MySQL connection variables: defining Constants
 $db_name = "permitappdb";
 $mysql_user = "root";
 $mysql_pass = "root";
 $server_name = "localhost";

 //create connection  to database
 $con = mysqli_connect($server_name, $mysql_user, $mysql_pass, $db_name);
 //check connection
 if(!$con)
 {
           // echo "Connection Error...".mysqli_connect_error();
 }
 else
 {
           //echo "<h3>Database connection Success...</h3>";
 }
 ?>
```

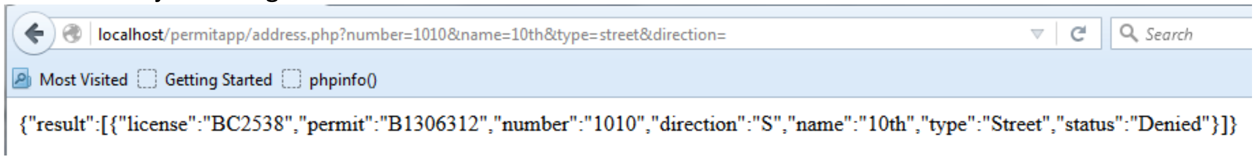
address.php

```
<?php
/**
Script to handle address search request.
To get the address search detail we use JSON.
This script will get the number, name, type, direction as an http request and will display the  address
search details in json according to the given address
*/

//import database script
require "init.php";


 //getting the values
$number = $_GET['number'];
$name = $_GET['name'];
$type = $_GET['type'];
$direction = $_GET['direction'];

 //creating sql query (setup search query for address)

$sql_query = "SELECT * FROM  permit  WHERE
(street_number like '$number' )
AND  (street_name like '$name%' )
AND (street_type like '$type%' )
AND (direction = null or direction like '$direction%')
 ;";

 //AND direction like '$direction'
 //AND street_name like '$name%'
 //AND street_type like '$type'

 //run the query against the database
$r = mysqli_query($con,$sql_query);

//create a blank array
```

```
$result = array();

//loop through the records
while ($row = mysqli_fetch_array($r)){
//The array_push() function inserts one or more elements to the end of an array
array_push($result,array(
/**specifies the value to add to the array:
 pushing license, permit, number, direction, name,type, status in the blank array created */
"license"=>$row['licensed_number'],
"permit"=>$row['permit_number'],
"number"=>$row['street_number'],
"direction"=>$row['direction'],
"name"=>$row['street_name'],
"type"=>$row['street_type'],
"status"=>$row['permit_status']
 ));
}
//display array in json format
echo json_encode (array("result"=>$result));
//close the connection
mysqli_close($con);

 ?>
```

We go to this url: localhost/permitapp/address.php?number=1010&name=10th&type=street&direction= to make sure we can pass the request [?number=street_number & ?name = street_name & &type = street_type&direction=street_direction]. Wamp is the server so that is why it is working on a localhost.

Below is the json string


localhost/permitapp/address.php?number=1010&name=10th&type=street&direction=
Search
Most Visited
Getting Started
phpinfo()

```
{"result":[{"license":"BC2538","permit":"B1306312","number":"1010","direction":"S","name":"10th","type":"Street","status":"Denied"}]}
```

activity_address.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This xml determines how the look of the main activity should be -->
<android.support.design.widget.CoordinatorLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:fitsSystemWindows="true"
   tools:context="permit.com.permit.AddressActivity">

   <android.support.design.widget.AppBarLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
      android:id="@+id/toolbar"
      android:layout_width="match_parent"
      android:layout_height="?attr/actionBarSize"
      android:background="?attr/colorPrimary"
      app:popupTheme="@style/AppTheme.PopupOverlay" />

  </android.support.design.widget.AppBarLayout>
  <!-- reference the content found in content_address.xml -->
  <include layout="@layout/content_address" />

</android.support.design.widget.CoordinatorLayout>
```

content_address.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--
This xml determines the contents in the activity_license contents that will user sees
 android:background="@color/colorBackground" : reference color from colors.xml
 tools:showIn="@layout/activity_address": will display contents in activity_address.xml
-->
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:background="@color/colorBackground"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context="permit.com.permit.AddressActivity"
  tools:showIn="@layout/activity_address"
  android:weightSum="1">

  <!-- Text Area where user enters street number-->
  <EditText
    android:id="@+id/streetNoID"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:singleLine="true" />
```

```
<!--
 Text displaying message
 android:text="@string/streetNum": reference string value from strings.xml
 -->

<TextView android:id="@+id/streetNoMessage"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/streetNum"
    android:gravity="center"
    android:singleLine="false"
    android:textColor="#0b0a0a"
    android:textSize="12sp"
    android:textIsSelectable="true" />

<!-- Text Area where user enters street direction-->
<EditText
    android:id="@+id/streetDirID"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="none"
    android:singleLine="true" />

<!--
Text displaying message
android:text="@string/streetDir": reference string value from strings.xml
-->

<TextView android:id="@+id/directionMessage"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/streetDir"
    android:gravity="center"
    android:singleLine="false"
    android:textColor="#0b0a0a"
    android:textSize="12sp"
    android:textIsSelectable="true" />

<!-- Text Area where user enters street name-->
<EditText
    android:id="@+id/streetNameID"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="none"
    android:singleLine="true" />
<!--
    Text displaying message
    android:text="@string/streetName": reference string value from strings.xml
    -->
  <TextView android:id="@+id/streetNameMessage"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/streetName"
    android:gravity="center"
    android:singleLine="false"
    android:textColor="#0b0a0a"
    android:textSize="12sp"
    android:textIsSelectable="true" />
  <!-- Text Area where user enters street type -->
  <EditText
    android:id="@+id/streetTypeID"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="none"
    android:singleLine="true" />
  <!--
  Text displaying message
  android:text="@string/streetType": reference string value from strings.xml
  -->

  <TextView android:id="@+id/streetTypeMessage"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/streetType"
    android:gravity="center"
    android:singleLine="false"
    android:textColor="#0b0a0a"
    android:textSize="12sp"
    android:textIsSelectable="true" />
  <!--
  Address Search Button:
  android:text="@string/askAddressButton" : reference string value from strings.xml
   -->
  <android.support.v7.widget.AppCompatButton
    android:id="@+id/addressButton"
    android:text="@string/askAddressButton"
    android:padding="12dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right" />
  <!--Area where result will display-->
  <TextView
    android:id="@+id/addressResult"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="0.71" />
```

```
</LinearLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--All string resources with other simple resources in the one XML file, and is reference using the value
provide in the name attribute-->
<resources>
   <string name="app_name">Permit App</string>
   <string name="action_settings">Settings</string>
   <string name="title_activity_permit">Permit Activity</string>
   <string name="title_activity_license">License Activity</string>
   <string name="title_activity_address">Address Activity</string>
   //string resource used in content_main.xml
   <string name="SearchPermit">Search by Permit</string>
   <string name="SearchLicense">Search by Licensed Professional</string>
   <string name="SearchAddress">Search by Address</string>
   //string resource used in content_permit.xml
   <string name="askPermit">Enter Permit Number</string>
   <string name="askPermitButton">Search</string>
   <string name="permitMessage">Enter Permit Number</string>
   //string resource used in content_license.xml
   <string name="askLicense">Enter Licensed Number</string>
   <string name="askLicenseButton">Search</string>
   <string name="licenseMessage">Enter License Number</string>

   //string resource used in content_address.xml
   <string name="streetNum">Street Number</string>
   <string name="streetDir">Street Direction</string>
   <string name="streetName">Street Name</string>
   <string name="streetType">Street Type</string>
   <string name="askAddressButton">Search</string>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--This is the root directory. The manifest file presents essential information
about the app to the Android system, information the system must have before it can run any of the
app's code.-->

<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="permit.com.permit">

<!-- this permission statement is required, if not the app won't be able to connect to the network-->
   <uses-permission android:name="android.permission.INTERNET"/>"

<!-- allowBackup: allows the app to be backup and have infrastructure to restore
```

```
Contains attributes like icon, label
icon: reference the app_logo that is located in the mipmap folder
label: label of app
supportsRt1: app willing to support right-to-left layouts-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/app_logo"
        android:label="@string/app_name"
        android:supportsRtl="true">

<!--Activities must be declared and if it is not then they will not be seen by the system and will never be
run. -->

        <!-- Declares the Main Activity -->
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Declares the Permit Activity -->
        <activity
            android:name=".PermitActivity"
            android:label="@string/title_activity_permit"
            android:theme="@style/AppTheme.NoActionBar" />

        <!-- Declares the License Activity -->
        <activity
            android:name=".LicenseActivity"
            android:label="@string/title_activity_license"
            android:theme="@style/AppTheme.NoActionBar" />

        <!-- Declares the Address Activity -->
        <activity
            android:name=".AddressActivity"
            android:label="@string/title_activity_address"
            android:theme="@style/AppTheme.NoActionBar">

        </activity>
    </application>
</manifest>
```

ConfigAddress.php

```
package permit.com.permit;
```

```
/**Declare important constant*/
public class ConfigAddress {

   //Address of permit script
   public static final String ADDRESS_URL = "http://192.168.56.1/permitapp/address.php?number=";

   //Keys that will be used to send this request to php script
   public static final String KEY_NUMBER = "number";
   public static final String KEY_DIRECTION = "direction";
   public static final String KEY_NAME = "name";
   public static final String KEY_TYPE = "type";

   //Keys values from inner array
   public static final String KEY_PERMIT = "permit";
   public static final String KEY_STATUS = "status";
   public static final String KEY_LICENSE = "license";

   //JSON string
   public static final String JSON_ARRAY = "result";
}
```

AdressActivity.java

```
package permit.com.permit;
//imports that class that this java file uses
import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

//end import

public class AddressActivity extends AppCompatActivity implements View.OnClickListener{
   //defining views
   private EditText streetNoID;
   private EditText streetNameID;
```

```
    private EditText streetTypeID;
    private EditText streetDirID;
    private Button addressButton;
    private TextView addressResult;
    //define progressDialog class: displays dialog and spinner while search is going on in background
    private ProgressDialog loading;

    @Override
    //initialize the activity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //call the activity_address layout resource to define the UI
        setContentView(R.layout.activity_address);
        //initialize views
        streetNoID = (EditText)findViewById(R.id.streetNoID);
        streetNameID = (EditText)findViewById(R.id.streetNameID);
        streetTypeID = (EditText)findViewById(R.id.streetTypeID);
        streetDirID = (EditText)findViewById(R.id.streetDirID);
        addressButton = (Button)findViewById(R.id.addressButton);
        addressResult = (TextView)findViewById(R.id.addressResult);
        //set onclick listener to license search button
        addressButton.setOnClickListener(this);
    }
    //Address Search Function
    private void getAddress(){
        //get the value the user enters
        String number = streetNoID.getText().toString().trim();
        String name = streetNameID.getText().toString().trim();
        String type = streetTypeID.getText().toString().trim();

        //if user does not enter street number, street name and street type
        if(number.equals("")& name.equals("")& type.equals("")){
            //display this message
            Toast.makeText(this,"Please enter street number, street name and street
type",Toast.LENGTH_LONG).show();
            return;
        }
        //if user does not enter street number
        if (number.equals("")){
            //display this message
            Toast.makeText(this, "Please enter a street number", Toast.LENGTH_LONG).show();
            return;
        }
        //if user does not enter street name
        if(name.equals("")){
            //display this message
            Toast.makeText(this,"Please enter name of street",Toast.LENGTH_LONG).show();
            return;
        }
        //if user does not enter street type
        if(type.equals("")){
            //display this message
            Toast.makeText(this,"Please enter type of street",Toast.LENGTH_LONG).show();
            return;
        }

        //start the dialog box
        loading = ProgressDialog.show(this,"Please wait...","Fetching...",false,false);

        //add search parameters to query URL that is called from ConfigAddress.java and create
        String url = ConfigAddress.ADDRESS_URL+streetNoID.getText().toString().trim() + "&name=" +
streetNameID.getText().toString().trim() + "&type=" + streetTypeID.getText().toString().trim() +
"&direction=" + streetDirID.getText().toString().trim();
        /** Create String Request: Request string response from the provided URL* */
        StringRequest stringRequest = new StringRequest(url, new Response.Listener<String>(){
            @Override
            public void onResponse(String response){
              //make the dialog box close when search is complete
              loading.dismiss();
              //calls the JSON method
              showJSON(response);
            }
        },
              /**
               * callback method that an error has been occured with the provided error code
               * with message
               */
              new Response.ErrorListener(){
                @Override
                public void onErrorResponse(VolleyError error) {
                  //display error message
                  Toast.makeText(AddressActivity.this,"Not Working",Toast.LENGTH_LONG).show();
                }
              });

        /**
         *  Instantiate the RequestQueue
         *  request queue using Volley library: volley handles network connections
         *  Volley maps the JSON response in respective class objects, it allows you to add
         *  getter setter for the response model objects which allows you to access JSON values/
         *  parameters using .operator like normal JAVA object, which makes response handling simple
         */
        RequestQueue requestQueue = Volley.newRequestQueue(this);
        //adding the request to RequestQueue
        requestQueue.add(stringRequest);
    }
```

```
   //showJSON method: this is where the JSON is parse
   private void showJSON(String response) {
      //create constants
      String license = "";
      String permit = "";
      String number = "";
      String name = "";
      String direction = "";
      String type = "";
      String status = "";
      String display = "";

      //create a JSONObject
      JSONObject jsonObject = null;
      try {
         //get the JSON string
         jsonObject = new JSONObject(response);
      } catch (JSONException e) {
         Toast.makeText(this,"JSON object error",Toast.LENGTH_LONG).show();
         e.printStackTrace();
      }
      //create an JSONArray
      JSONArray result = null;
      try {
         result = jsonObject.getJSONArray(ConfigAddress.JSON_ARRAY);
      } catch (JSONException e) {
         Toast.makeText(this,"JSON array error",Toast.LENGTH_LONG).show();
         e.printStackTrace();
      }


      //if there is nothing in the JSON array
      if (result.length()==0){
         //display message
         display="The Address you have enter does not have any permit(s) attached to it. Please Try a
different address";
      }

      for (int i = 0; i < result.length(); i++) {
         try {
            //retrieves the JSONObject with the associated index value
            JSONObject addressData = result.getJSONObject(i);
            //getting fields inside in the JSON array
            license = addressData.getString(ConfigAddress.KEY_LICENSE);
            permit = addressData.getString(ConfigAddress.KEY_PERMIT);
            status = addressData.getString(ConfigAddress.KEY_STATUS);
            number = addressData.getString(ConfigAddress.KEY_NUMBER);
            name = addressData.getString(ConfigAddress.KEY_NAME);
            direction = addressData.getString(ConfigAddress.KEY_DIRECTION);
            type = addressData.getString(ConfigAddress.KEY_TYPE);

         } catch (JSONException e) {
            e.printStackTrace();
         }
         //add the found fields in display variable
         display += "License No: \t" + license + "\nPermit No:\t" + permit + "\nStreet Address:\t" + number
+ "\t" + direction + "\t" + name + "\t" + type + "\t" + "\nPermit Status:\t" + status + "\n\n";

      }
      //display the result in the designated text area
      addressResult.setText(display);
   }
      @Override
      //When user clicks the button, call the getAddress Function
   public void onClick(View v){
      getAddress();
   }
}
```

Appendix D: Development Environment

Development Environment

There are two Android Development Environments to choose from, either Eclipse or Android Studio. Since I was familiar with Eclipse, I figured that was the best route at first. Then I had an issue with downloading the Eclipse and Android SDK that I needed to integrate with the Eclipse to help develop my Android prototype. I spent about 3 hours trying to download Eclipse but I never succeed with that download because it would download and when it was almost complete the download would fail, so my frustration won out and I decided to go with another application to help me develop my Android app.

After reading more about Android Studio, I decided to download it instead. It is the latest development environment for programming a mobile application, and I figured I can adjust and work with it from the various resources on how to navigate it. So I decided to download Android Studio and the SDK, and it took me about 2 hours do download both.

What I find frustrating with Android Studio after I successfully installed it using various YouTube videos explaining how to do so, is the constant updating of the tool. With all the updates and newer version of Android environment I would say I have spent over 10 plus hours on updates alone. Updates happen at least twice a week and can take anywhere from 45 minutes to 2 hours depending on the speed of my wireless connection. If I didn’t use my wireless and used a direct connection, it would typically take about 20 minutes depending on the update. What was particularly frustrating was when Android Studio updated to the latest version because it would change the structure of the application itself. What I mean by this statement is that the pages (XML and java files) would change. For example, the biggest change is the separation of layout of the xml files. So instead of one file to design the layout of the app, there are now two, one stores the look while the only stores the content of what is on the app.

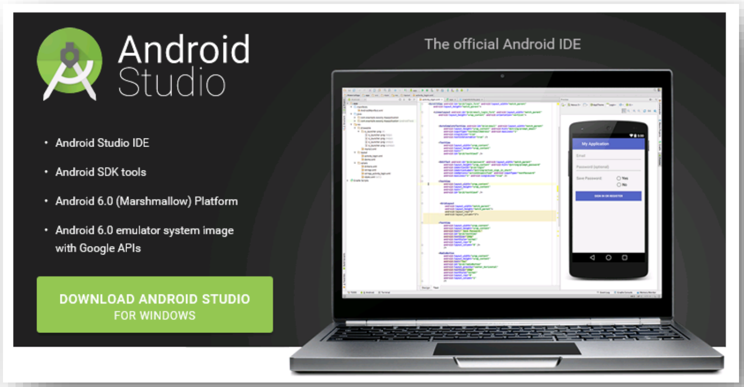
Another frustration I found was the using the AVD (android virtual device) Emulator that comes with the development platform. It did not work properly and after about 2 hours of research I was able to make it work but it takes time for it to run and at most times does not function properly. I had to change my BIOS to enable my Hardware Virtualization, then install the Intel x86 Emulator Accelerator (HAXM installer), which according to Android Studio it was already installed. Even though I am able to use it, it takes forever to load and a lot of times it still does not functioning properly.

After searching around I found another AVD to work with Android and that is called Genymotion. Genymotion was a free download; however that has recently changed in the last month where you have to buy it to use it. It only took me about 20 minutes between download and set up, and it works great. The problem I ran into is getting the AVD to display properly. Since I can no longer download various AVD simulators I am stuck to one which does not display my app in a decent screen. Still it allows me to test my features quicker than the standard AVD emulator.

For the back end of the mobile application I will need a database, a web server and a server-side scripting language that will connect the development environment to the server to grab the information off the database. I installed WAMPSERVER (**http://www.wampserver.com/en/)** to take care of all the back end tools I will need. Installing WAMP server, gets me a database (MySQL database and PHP/phpMyAdmin), a Web Server (Apache Web Server) and a Server Side Scripting Language (PHP). I am entry level knowledge of MySQL and PHP and the rest I can learn through books and searching along the way. It took me about 2 hours to install and test to make sure everything ran properly and another 2 hours of research and testing to figure out how to install the design features for phpMyAdmin in order for me to make my table connections. So overall it took me about 10 hours to set up everything I need for my development platform for my application and another 10+ hours of updates. So development setup took me roughly 2 weeks.

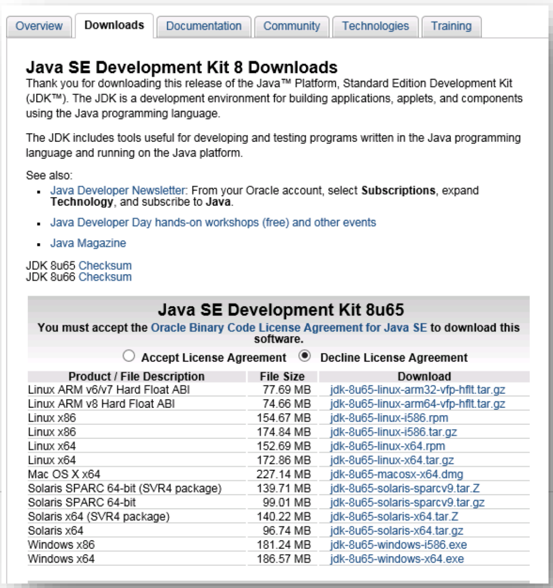
Download Sources:

1. **Android Development Environment**: https://developer.android.com/sdk/index.html


Android Studio
The official Android IDE
Android Studio IDE
Android SDK tools
Android 6.0 (Marshmallow) Platform
Android 6.0 emulator system image with Google APIs
DOWNLOAD ANDROID STUDIO FOR WINDOWS

2. Install the latest Android SDK to go with Android Environment:

   http://www.oracle.com/technetwork/java/javase/downloads/index.html



Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u65 Checksum
JDK 8u66 Checksum

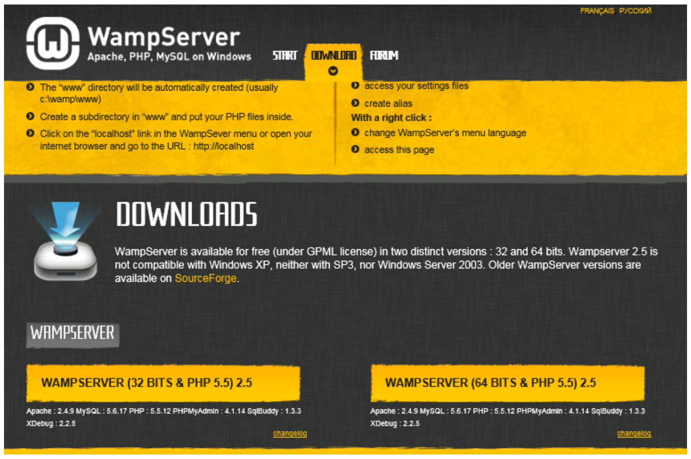
Java SE Development Kit 8u65

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.69 MB	jdk-8u65-linux-arm32-vfp-hflt.tar.gz
Linux ARM v8 Hard Float ABI	74.66 MB	jdk-8u65-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.67 MB	jdk-8u65-linux-i586.rpm
Linux x86	174.84 MB	jdk-8u65-linux-i586.tar.gz
Linux x64	152.69 MB	jdk-8u65-linux-x64.rpm
Linux x64	172.86 MB	jdk-8u65-linux-x64.tar.gz
Mac OS X x64	227.14 MB	jdk-8u65-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.71 MB	jdk-8u65-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.01 MB	jdk-8u65-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.22 MB	jdk-8u65-solaris-x64.tar.Z
Solaris x64	96.74 MB	jdk-8u65-solaris-x64.tar.gz
Windows x86	181.24 MB	jdk-8u65-windows-i586.exe
Windows x64	186.57 MB	jdk-8u65-windows-x64.exe

3. Download and install WAMPSERVER because the server installer contains the following components that you need: Apache Server Application, MySQL Database and PHP/phpMyAdmin. Website: **http://www.wampserver.com/en/**



FRANÇAIS РУССКИЙ

WampServer
Apache, PHP, MySQL on Windows

START DOWNLOAD FORUM

- The "www" directory will be automatically created (usually c:\wamp\www)
- Create a subdirectory in "www" and put your PHP files inside.
- Click on the "localhost" link in the WampSever menu or open your internet browser and go to the URL : http://localhost

- access your settings files
- create alias

With a right click :
- change WampServer's menu language
- access this page

DOWNLOADS

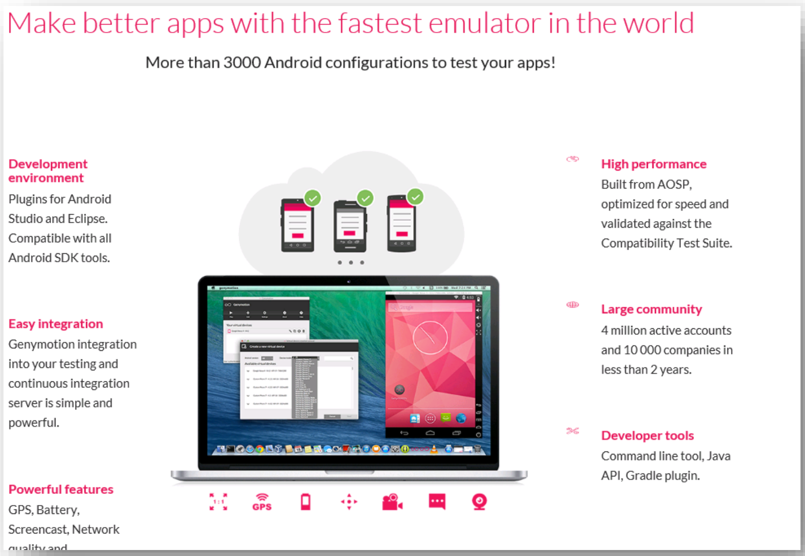
WampServer is available for free (under GPML license) in two distinct versions : 32 and 64 bits. Wampserver 2.5 is not compatible with Windows XP, neither with SP3, nor Windows Server 2003. Older WampServer versions are available on SourceForge.

WAMPSERVER

WAMPSERVER (32 BITS & PHP 5.5) 2.5
Apache : 2.4.9 MySQL : 5.6.17 PHP : 5.5.12 PHPMyAdmin : 4.1.14 SqlBuddy : 1.3.3 XDebug : 2.2.5
changelog

WAMPSERVER (64 BITS & PHP 5.5) 2.5
Apache : 2.4.9 MySQL : 5.6.17 PHP : 5.5.12 PHPMyAdmin : 4.1.14 SqlBuddy : 1.3.3 XDebug : 2.2.5
changelog

4. Install Genymotion (https://www.genymotion.com) for a faster Android emulator to use with Android Studio. It can be downloaded for free for personal use, you just have to register. This is has recently change, because it is no longer have a free personal usage download.



Make better apps with the fastest emulator in the world

More than 3000 Android configurations to test your apps!

Development environment
Plugins for Android Studio and Eclipse. Compatible with all Android SDK tools.

Easy integration
Genymotion integration into your testing and continuous integration server is simple and powerful.

Powerful features
GPS, Battery, Screencast, Network quality and

High performance
Built from AOSP, optimized for speed and validated against the Compatibility Test Suite.

Large community
4 million active accounts and 10 000 companies in less than 2 years.

Developer tools
Command line tool, Java API, Gradle plugin.

5. Install Android Volley.

Android Volley is an HTTP library that makes networking for Android apps easier and faster. Volley simplifies networking task in Android and creates and easy way to use request management, provides an efficient network management and is easily customizable. Steps to download this library was found at this website: https://www.simplifiedcoding.net/android-volley-tutorial-to-get-json-from-server/

References

Samson, M. (2015, April 7). Why Mobile Apps May Become More Important Than Your Website. Retrieved October 17, 2015, from https://www.linkedin.com/pulse/why-mobile-apps-may-become-more-important-than-your-website-samson

Number of apps available in leading app stores 2015 | Statistic. (n.d.). Retrieved October 17, 2015, from http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores

Rowinski, D. (2013, January 9). How Long Does It Take To Build A Native Mobile App. Retrieved October 14, 2015, from http://readwrite.com/2013/01/09/how-long-does-it-take-to-build-a-native-mobile-app-infographic

Insight of Mobile App Development Process [Infographic]. (2015, September 7). Retrieved October 14, 2015, from http://theninehertz.com/insight-of-mobile-app-development-process

How Long Does it Take to Make an App?: An Infographic - Idea to Appster. (2013, February 13). Retrieved October 17, 2015, from http://ideatoappster.com/how-long-does-it-take-to-make-an-app/