

## Final Report: TORONTO inFORM

### INTRODUCTION

TORONTO InFORM is a location-based mobile application that connects users to their surroundings through the lens of architecture. It is a user-generated information source that details Toronto's many buildings in hopes of increasing engagement with these sites for both tourists and locals. The aim is to provide a platform to stimulate dialogue, where architectural sites and physical locations provide a window into the city's political, social, historical and factual details. The end product is an eclectic, fluid, and multifaceted archive that represents the ever-changing urban landscape of Toronto and reflects its diverse communities and neighborhoods. As an application related to the discipline of Museum Studies, TORONTO inFORM acts as a museum of Toronto, one that can grow and change as fast as the city does. Instead of perpetuating an authoritative voice on the history of Toronto, the application 'collects' stories, histories and facts based on public contribution, collaboration and sharing.

### OVERALL DESIGN

When the user starts the application they see the home screen in Figure 1. Each button of the screen is a key functionality of the application. This section will explain the functionalities and provide a high level overview.



Figure 1 - Start screen of application

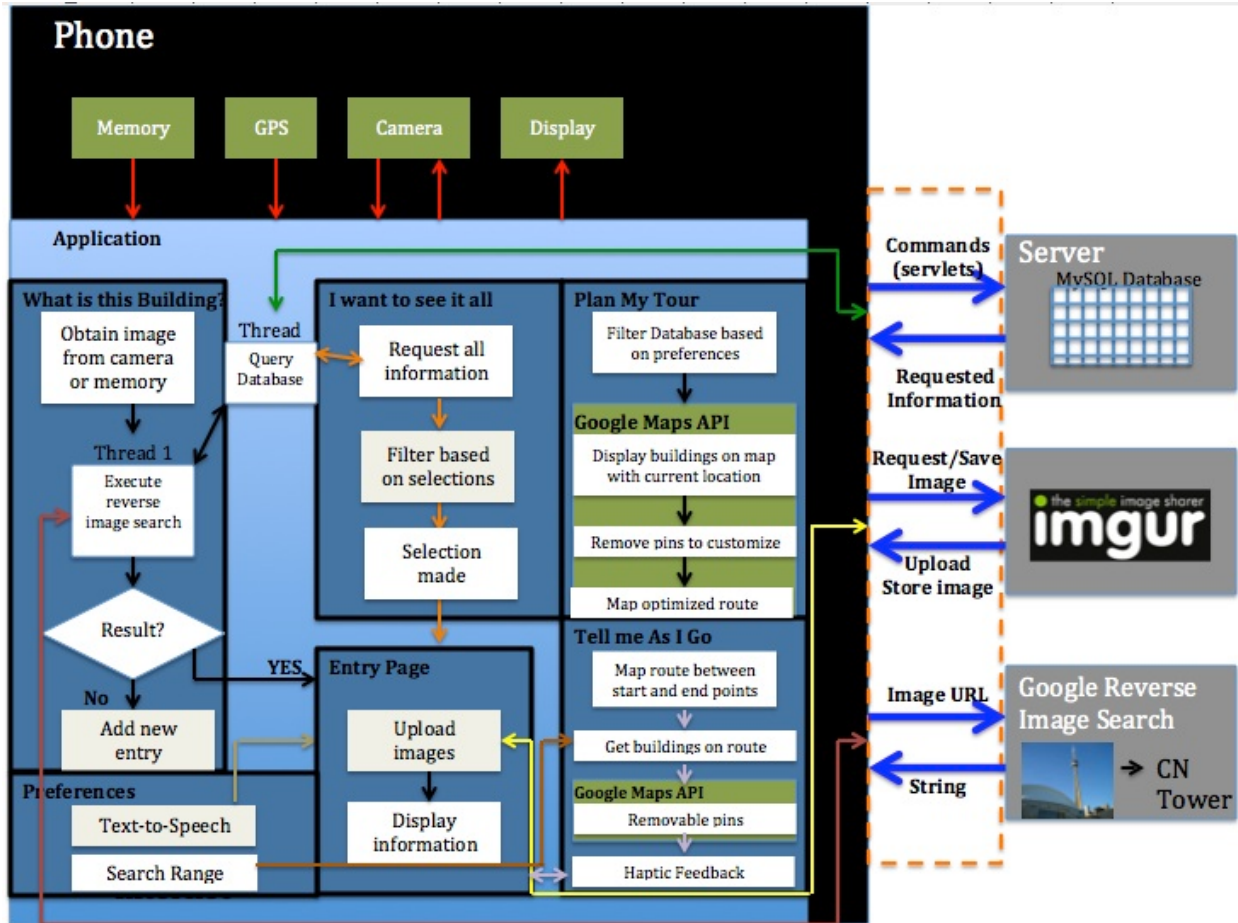


Figure 1b - Block diagram of overall functionality

### 1. What Is This Building

This functionality allows the user to know more about a building in their immediate vicinity. By taking a photograph of the building, image recognition software runs a search to determine the building name, and a search through the online database will pull up information about it. To make it easier for the user, images can be taken through the camera or selected from the phone gallery.



Figure 2 - Screen user is presented with option

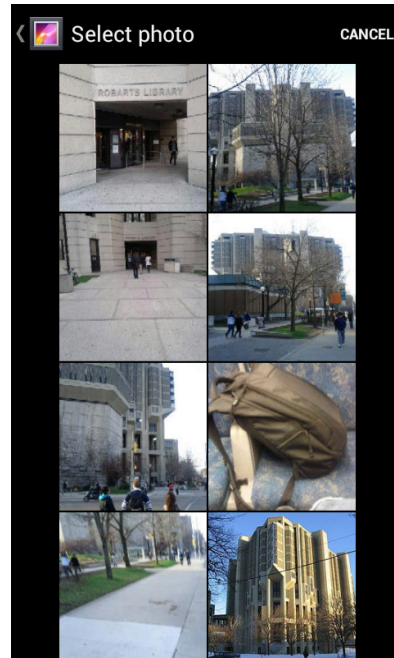


Figure 3 - Album selection screen

### **Image recognition and Reverse Image Search**

To achieve this task, first the image was uploaded to the image hosting website *Imgur*. Google images allows users to perform a reverse image search by inputting the image url into the search bar. Since Google does not provide a public API for this task, a GUI testing tool called *Selenium* is used to simulate this behaviour.

A web project was created to make this functionality easily available to the application. The third party library has a dependency on Windows OS and Firefox being installed on the server.

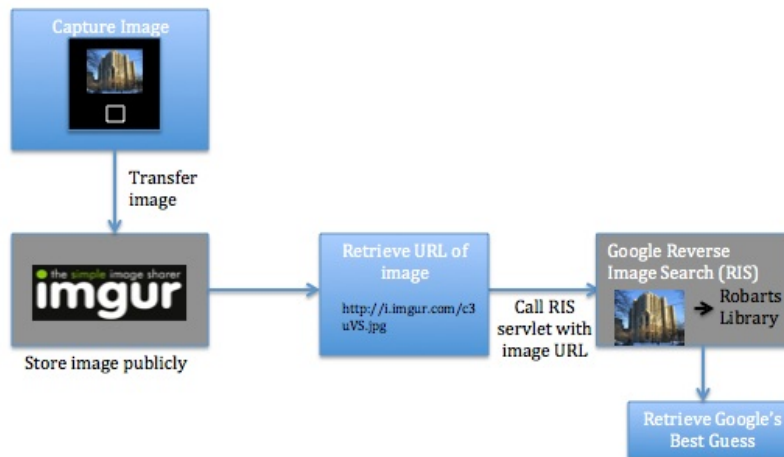


Figure 4 - Flowchart of reverse image search

## GPS search

In the case that the image search does not identify the building in question, a location-based search is performed. The GPS coordinates are noted and the database is searched for all buildings in the vicinity. The search area radius can be changed through preferences.

All buildings found are shown in a list format and the user can view details about any given building. If no buildings are found, the user can add a new entry to the database.

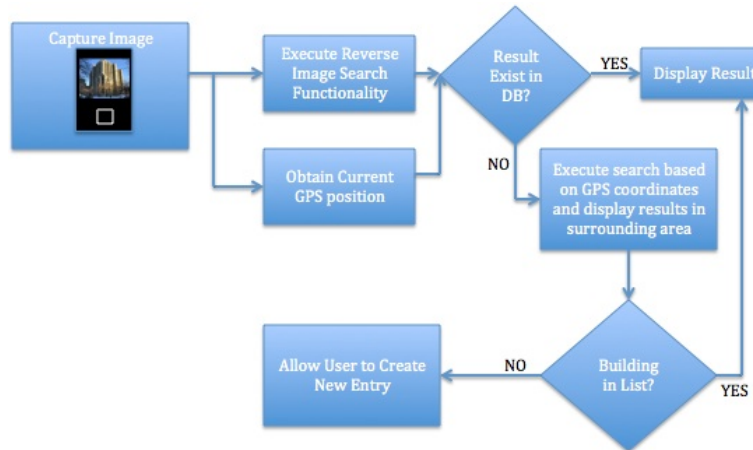


Figure 5 - Simplified flowchart of 'What is This Building'

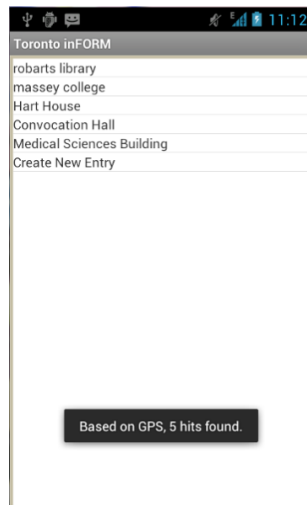


Figure 6 - List of local buildings when no entry found from reverse image search

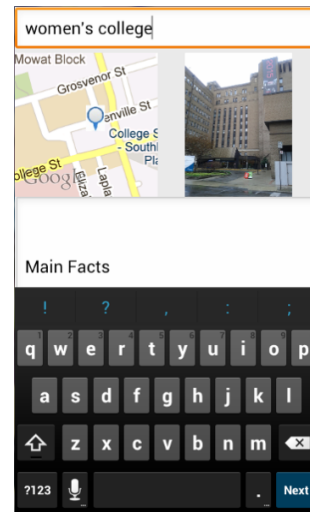


Figure 7 - Example of new entry

## 2. Tell Me As I Go

This functionality allow users to explore their surrounding areas. The user enters the start and end point of their journey, and the app informs the user regarding buildings of interest as they travel.

This feature had three major parts:

1. Obtain google driving directions to mirror the most likely path chosen by a user.
2. Search the database for all buildings on the route.
3. Monitor user location and notifications

### **Obtain driving directions**

Once the user enters the starting and ending point of the journey, the locations are geocached and the coordinates are retrieved. These are then used on google directions API. The results of the API includes the most likely path coordinates with a step size of 50 metres (In cases with multiple possible routes, the best guess is assumed). Having a small step size results in very reliable building search with an added benefit of a cleaner map route overlay.

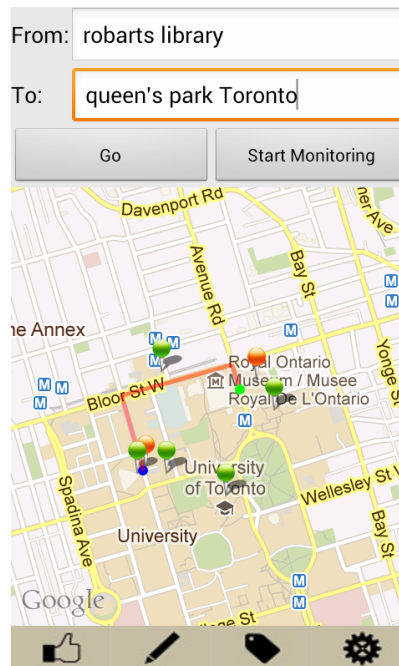


Figure 8 - Example of mapped route with buildings found

### **Search DB for buildings**

All the step points are then uploaded into a table in our database and joined with the entries coordinates. This query returns all the building which are in proximity of the route.

### **Monitor user location and notifications**

The user's location is tracked through GPS and a notification system alerts them when a monitored building is found close to their location. Whenever the location is updated, its proximity to the buildings is calculated. When a building is found, the user is notified through haptic feedback and taken to the display page (see Display Screen section: 'I Want to See it All').

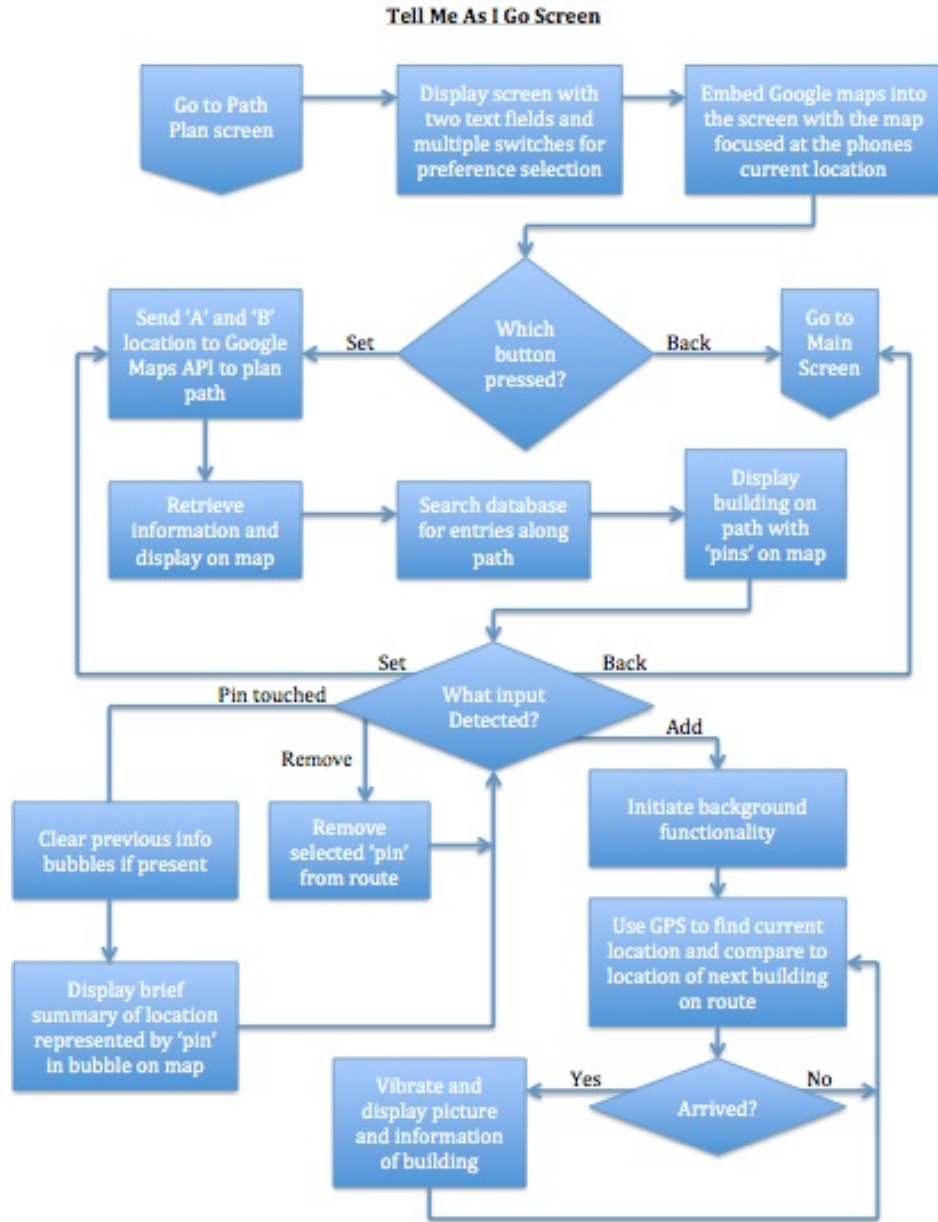


Figure 9a - Flowchart of 'Tell Me as I Go' functionality



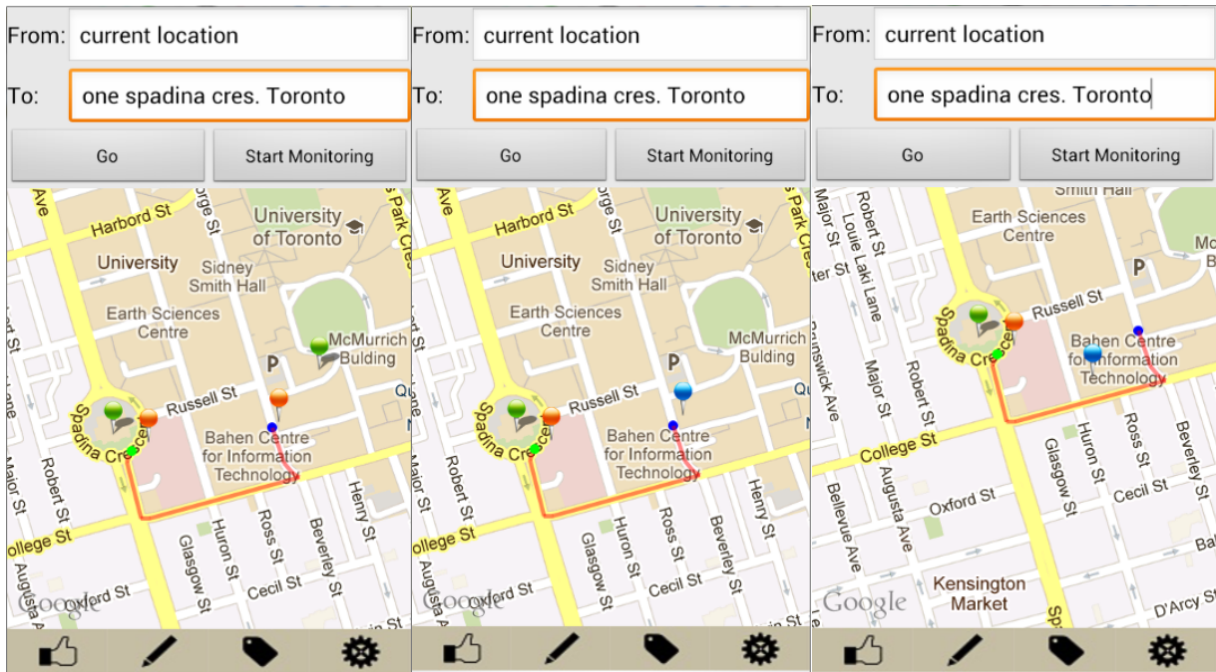


Figure 9b - Snapshots of user travelling. Orange pins: Starting and ending locations, Green pins: Buildings found on the route. Blue pin: Current location

**Text to Speech:** A text to speech service is provided. The benefit of this feature is to read out the relevant information to the user while they are driving or walking. If enabled, this feature will be triggered during user notification.

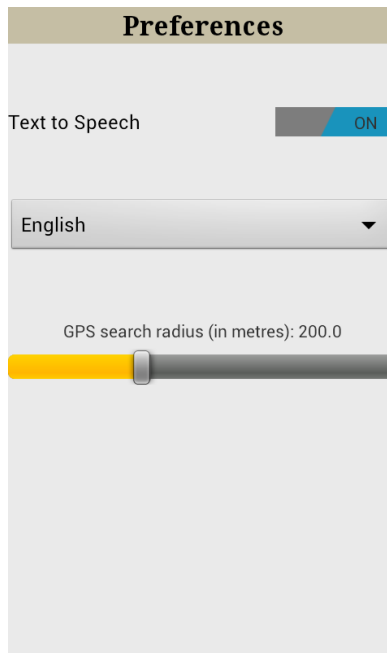


Figure 10 - Preference menu

### 3. Plan A Tour

This feature allows users to create custom tours based on their selected preferences from the dropdown menus.

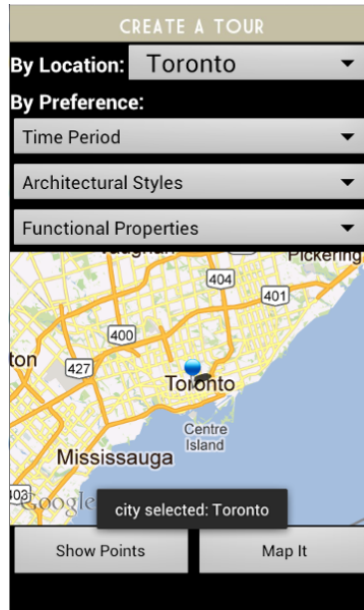


Figure 11 - Filter by City, Time Period, Architectural style, and Function.

#### Show Points

Once preferences are selected, the map is overlaid with pins referencing the buildings. Tapping on the pins shows a menu and allows the user to add/remove the pin from the tour or set it as the endpoint of the tour.

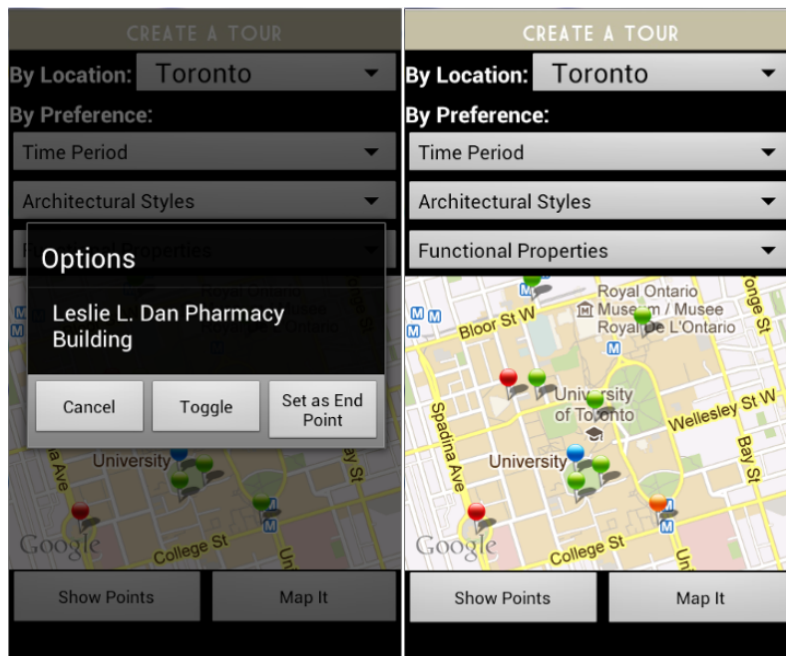


Figure 12 - Popup when user clicks on pin and buildings overlaid on map. Blue pin: current location, green pin: Building on tour, red pin: building removed from tour, orange pin: endpoint



## Map It

When building selections are finalized, the Google Maps API generates an optimized route starting from the user's current location and including the buildings specified by the user. Due to current API restriction, the maximum number of buildings in a tour is eight. Once the map is loaded and overlaid onto the map, step-by-step directions are also displayed for the tour.

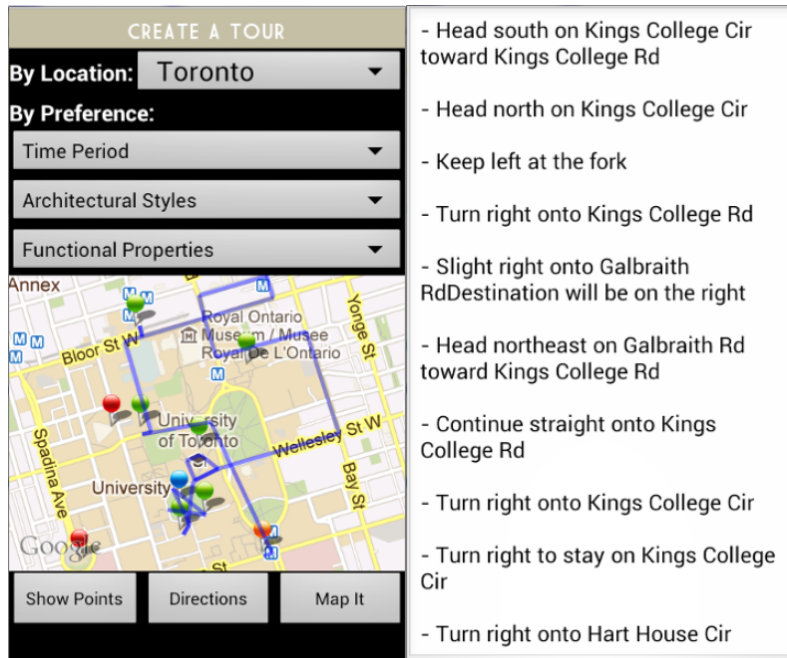


Figure 13 - Tour path overlaid on map with blue lines. Directions seen in text format

## 4. I Want To See It All:

This feature is an archive of the buildings in the database. Users are given multiple dropdown menus to tailor the output according to their interests. Currently, the users can filter buildings based on city, time period and type of architecture. The live search feature filters the result with each keystroke. Results are provided in a list, and clicking on any entry displays the detailed information about the building.

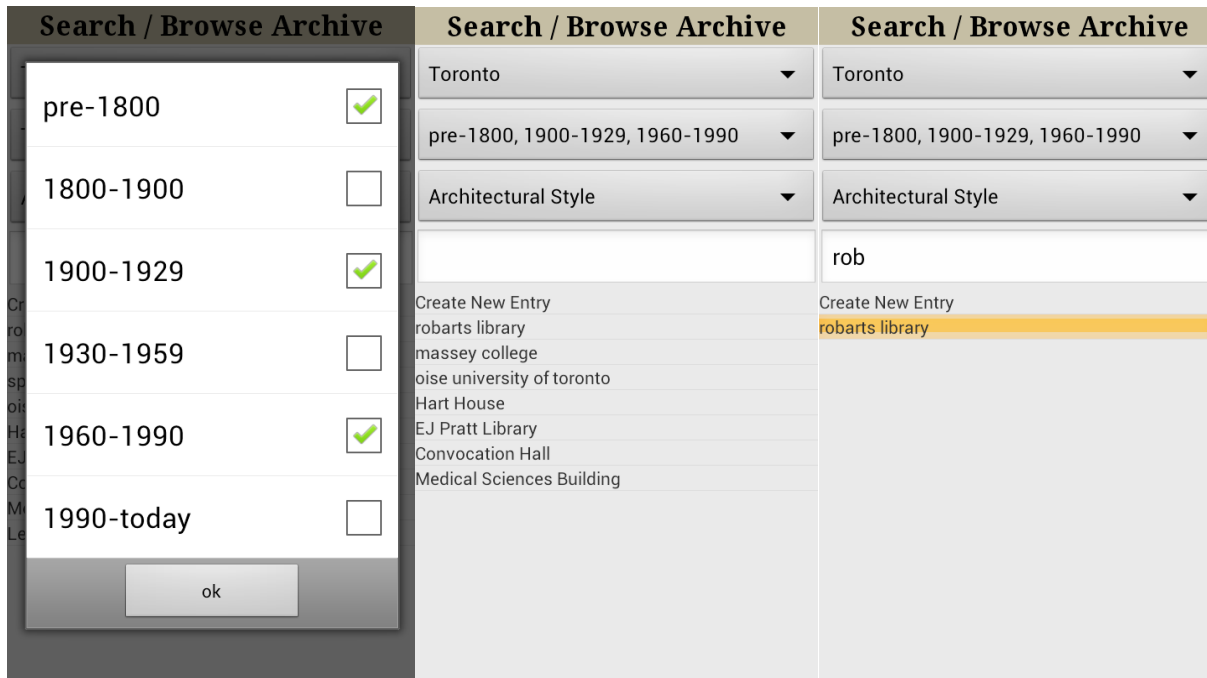


Figure 14 - Buildings in Archive, User Filters and Live Search

### Display Screen

All the functionalities work towards presenting a display screen to the user. All the relevant information about the building is displayed in a user friendly interface. User can click on the screen to toggle between Summary, Architectural, Historical and Social Facts. A photo gallery of the building can be scrolled through, and individual photographs can be tapped and opened up in full screen.

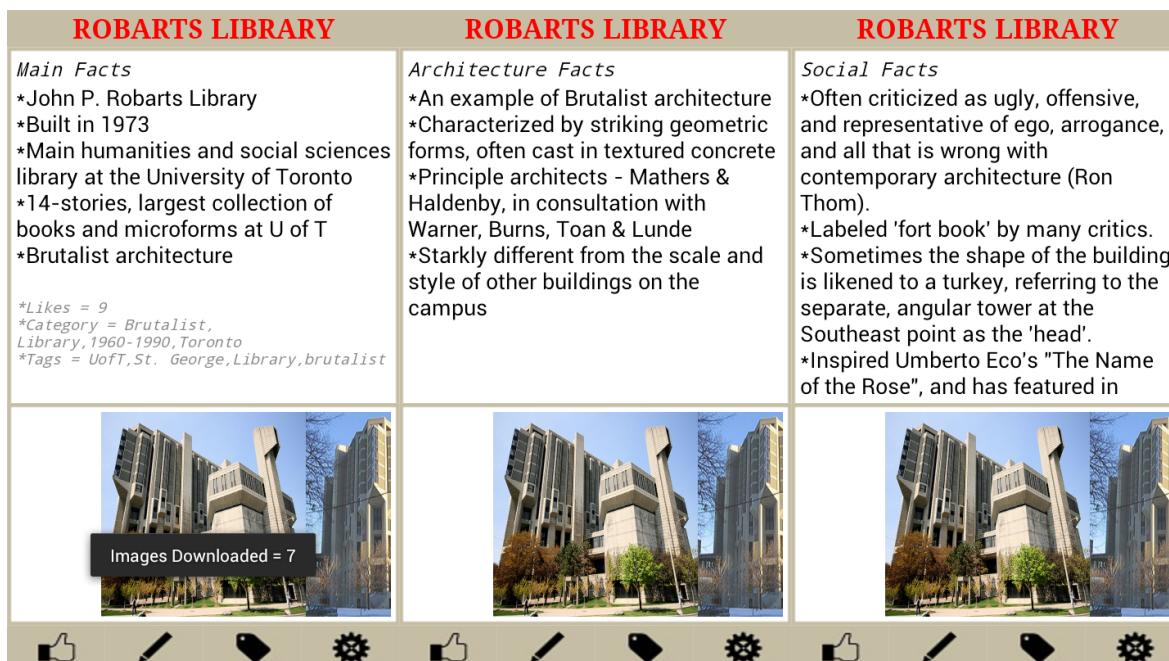


Figure 15 - Multiple Views of Entry

At the bottom of the display screen are multiple buttons with each adding to the user experience.

### Like

A user can like an entry and this will automatically store and affect their preferences.

### Edit/Add

Since the content submitted by a user is unmoderated, it is possible that sometimes irrelevant or incorrect information might be displayed. We give the users the ability to be able to edit an entry or add more information.

### Tags

Users can tag keywords related to the entry and these are used to improve archive search by providing more context.

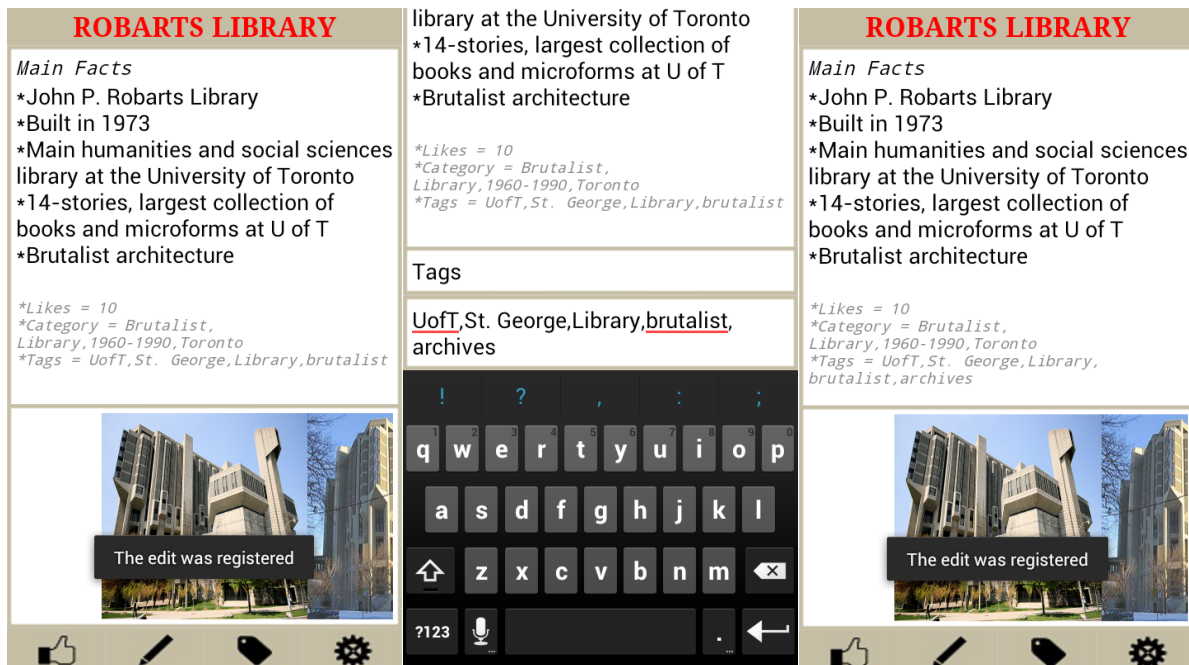


Figure 16 - Like, Edit and Tags

### **What did not work:**

1. During testing, it was found that some entries did not get edited and the changes were not reflected. This happens when the entry is long and the text being sent goes over the max length supported by the database server.

A simple way to prevent this from happening will be to break a single update query into multiple smaller ones but this can impact performance adversely. A safer solution is to send a file to a backend web servlet which can parse and store the data. This was not done due to time constraints.

2. During the final presentation, our app crashed while trying to add a new entry. Upon investigation, this was found to be two problems. The first is due to a timing issue on the database side which gives a timeout error if there are concurrent requests. Consequently, the exception handler in android is invoked but `exception.getMessage()` is an empty string, which if logged causes the application to crash. The second problem is an android bug which was opened in 2010 (<http://code.google.com/p/android/issues/detail?id=9211>). The first problem was not fixed due to time constraints and lack of reproducible test case, however, we were able to workaround the second problem by using `exception.printStackTrace()` method instead.

## **KEY LEARNING**

### ***Apper: Shannon***

I learned about design for mobile technology, including UI and visual capabilities. If I were given the chance to start again, I would have used wireframing to establish a rigid structure for each screen before going ahead with styling. I found I was paying too much attention to how the app looked initially, which did not allow the flexibility we needed for the various APIs.

I also learned how complex information architecture is when designing an application, and how important it is to establish search categories that are consistent and well considered before writing content. It was difficult to think about how information could be searched after the fact, and it would have helped the design of the app to have this functionality confirmed earlier on in the creation process.

### ***Programmer: Abhinav***

This app provided was a huge learning experience for me due to the sheer variety of functionality and complexity of making all the components work together. All the different APIs (Imgur - image upload and download, Google image search, Google driving directions, Google Maps, Text to Speech) presented challenges of their own. The documentation was scarce and not always well written. During the development, we also hit a few known android bugs. On the server side, setting up and hosting a MySQL database and image search web application were things I had never done before and I had to learn php and web javascript.

If I were to do this from scratch, I will most probably start by cutting down on scope. Having a smaller set of functionality will given us enough time to provide a richer user experience without compromising the integrity of the app. Also, writing a test bench for the application would have saved lots of frustration down the line.

### ***Programmer: Sana***

Coming into the course, I had merely a basic knowledge of programming. Throughout the length of the course, I learned the concept of objective programming through Java and the notion of APIs and the application of a number of them. I spent most of my time working with the Google APIs and the ease and functionality that it introduced to the app opened up so many doors. Another key thing I learned from this class was the experience of working with an apper.

As an engineer, I have spend most of my time working with other engineers who understand the functionality and tools at hand. Working with Shannon and having her as project manager, it forced me to use different forms of explanations and try to view and understand things from a different perspective.

If I could go back, what I would do differently is that I would like to do more research before attempting to implement any type of functionalities. I spent hours debugging code that I implemented based off the documentation provided and, for many of these issues, a couple minutes of research would have them solved in much quicker.

## **CONTRIBUTIONS**

Programmer: Abhinav

- What is this building
- Tell me As I Go
- Archive and Preferences
- Display Entry and Text to Speech
- Database
- Reverse Image Search
- Testing and final report

Programmer: Sana

- Google maps implementation
- Map overlay functionality
- Plan my Tour
- Testing and Verification
- Block diagrams
- Final report

Apper: Shannon

- Concept for app
- UI, layout and design
- Content - 15 entries for buildings throughout campus
- Testing
- Visual presentations

## **APPER CONTEXT**

Museums have been an endangered species for the last twenty years, and are trying to reinvent themselves in the digital age to appeal to a broader public. There are now several government-funded initiatives to support online development of heritage and museum collections. While this 'digital heritage boom' is occurring, it will be important for cultural institutions to truly embrace the new form of knowledge sharing that has switched from a linear transmission model (where the all-knowing, authoritative curator tells the generic, passive public what to think), to the dialogic model with multiple entry-points (where audiences are activated

and the relationship with the curator is one of exchange). TORONTO inFORM is an excellent test case for museums as facilitators rather than authors, where a structure is put in place that encourages knowledge-sharing, dialogue and debate, and inspires curiosity in one's surroundings. Instead of trying to draw people through museum doors, it provides a free service to enhance everyday enjoyment of the world, something that museums propose to do in their duty as institutions of public trust. The application is an exciting way for museums to become more relevant to contemporary society, using a personal mobile device to connect individuals to the story of the city.

## **FUTURE WORK**

1. Host reverse image search on a remote platform like amazon web services for a more stable and secure connection.
2. Tell me as I go:
  - a. Incorporate multi-lingual support for tourists.
  - b. Remove the need to enter start and end location and make the monitoring of buildings real time through intelligent location based algorithms.
3. Improve UI / opportunities for interactivity - Fine-tune each screen for better legibility and more enjoyable user experience
4. Allow users to add more than 8 buildings on a tour.
5. User profiles and enhanced preferences - Enable login option for users to save tours, share sites, and contribute to entries as an 'author'

## **INTERESTED IN BUSINESS SCHOOL TAKING IT UP?**

We would certainly be interested. It's important to note that the app wouldn't necessarily be very 'profitable', but might garner a small sum from being marketed as a tourist application rather than a local platform for knowledge-exchange.