UrbanEyes: Final Report

Word count	
Body	1964
Apper section	472

Sheraz Khan, Ravi Kalyani, Miroslav Cupák
Professor Jonathan Rose
ECE1778H1S
April 12, 2013

1. Introduction

UrbanEyes is an app that allows researchers of urban settings to collect data for mapping. Urban planning researchers make maps to communicate data about cities and make recommendations for change. For example, researchers may want to know the locations of wheelchair-accessible building entrances to understand the differences in travel experiences of people living with disabilities. UrbanEyes can facilitate projects like this (which are based on spatial data) by simplifying the data collection process.

UrbanEyes works through a mobile and web application, where researchers create surveys through the UrbanEyes web app, and surveyors respond to surveys by collecting data using their smartphones. The researcher can then download the data from the web in a ready-to-use format (a comma delimited file) for mapping. This workflow is captured in the diagram below.

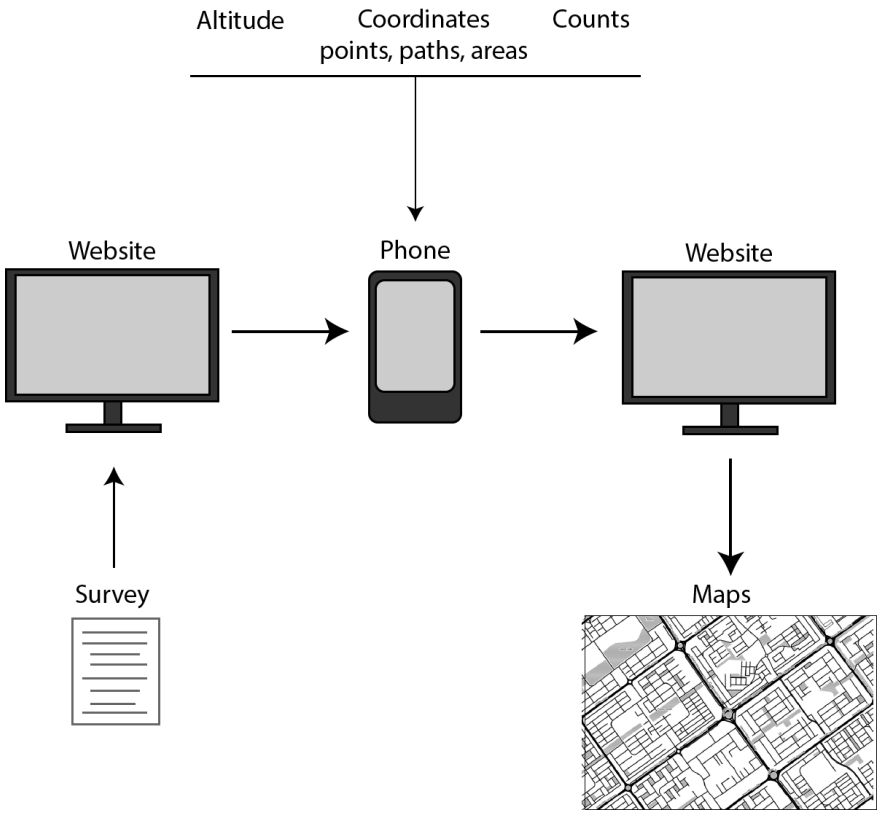

Figure: The UrbanEyes workflow. Researchers input surveys into the web application. These are sent to surveyors' smartphones for data collection. Data is returned to the web application for download.

This report outlines the functionality of UrbanEyes and the lessons learned through its development.

2. Design and Implementation

Below is a block diagram describing UrbanEyes. As can be seen, the application consists of a client (mobile application) and a server (web application). On the client side, the app allows users to collect GPS and altitude data in three geometries (point, path and area), and allows survey questions to be attached to those geometries. On the server side, the application allows users to login, sign up, create/edit/delete surveys, and download data in a suitable format.

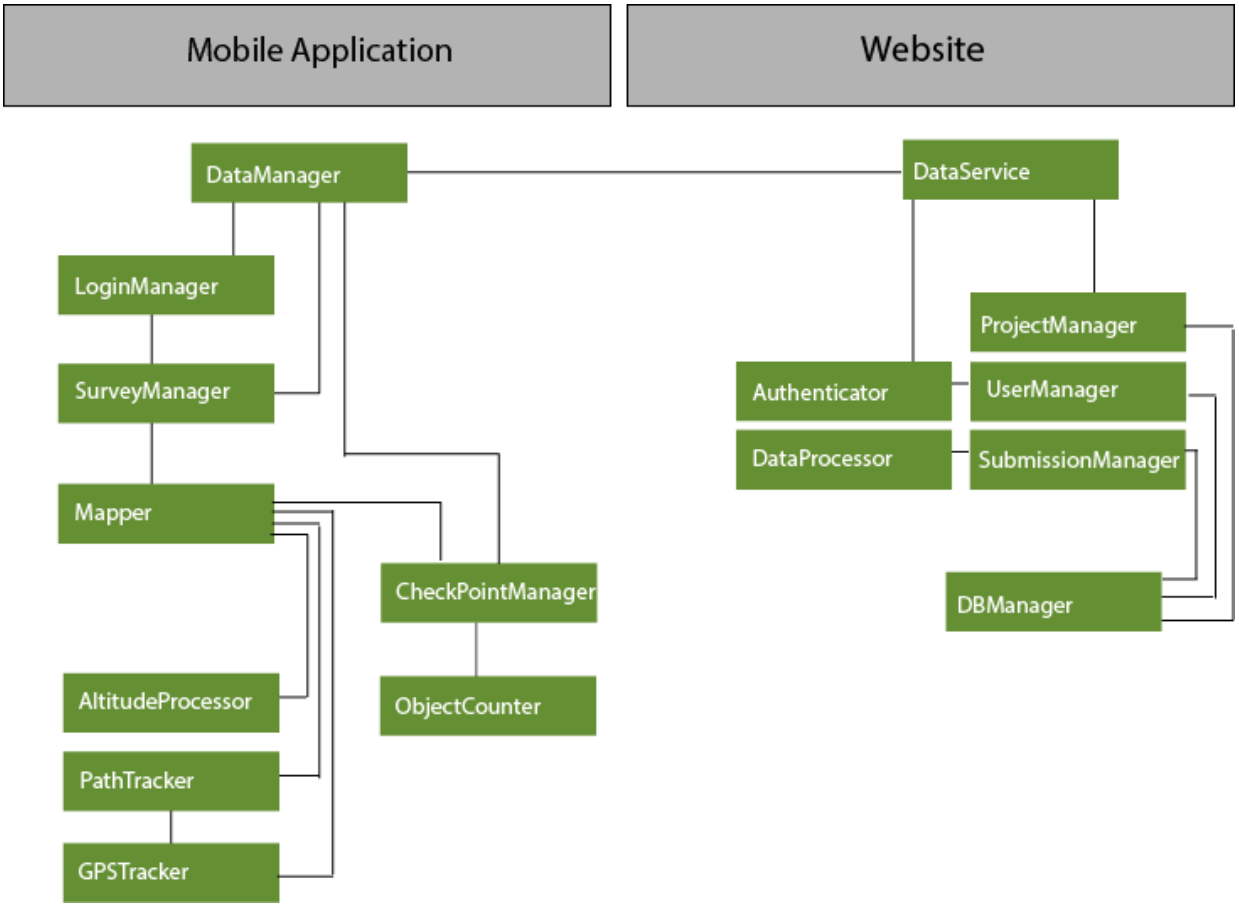

Figure: Block diagram describing UrbanEyes. On the left are blocks related to the mobile app, while on the right are blocks describing the web app.

Delving further, the mobile application contains the tools for collecting data. The GPSTracker provides the latitude and longitude of the phone when a survey is taken. The PathTracker uses

GPS coordinates taken at short intervals (10 meters) to draw paths, or areas. The ObjectCounter is used to collect counts using a simple clicker-like interface.
The AltitudeProcessor uses the phone's barometer to compute the elevation - a value which is important to urban planners. The measurement is based on a comparison of the phone's measured atmospheric pressure a reference sea-level pressure. This yields the elevation in meters above the sea level. The following formula is used to calculate the altitude (based on the hypsometric equation, AMS 2012):

$$h = c \ \ln \frac{P_{sea}}{P_{phone}}$$

Here, P_{phone} is the pressure at the phone's location, and P_{sea} is the value at sea level. The pressure at sea level is not a constant value and has to be obtained in real-time. We obtain it from the WeatherBug[1] service, which enables queries for current weather values based on GPS coordinates. We verified the values we measured using an elevation map of Toronto (see figure below).The pressure measured by our Nexus 4 phone varied by up to 0.2 mBar which corresponds to a possible error of about 1.6 m in altitude. We were not able to compare values between phones only one phone with a barometer was available.

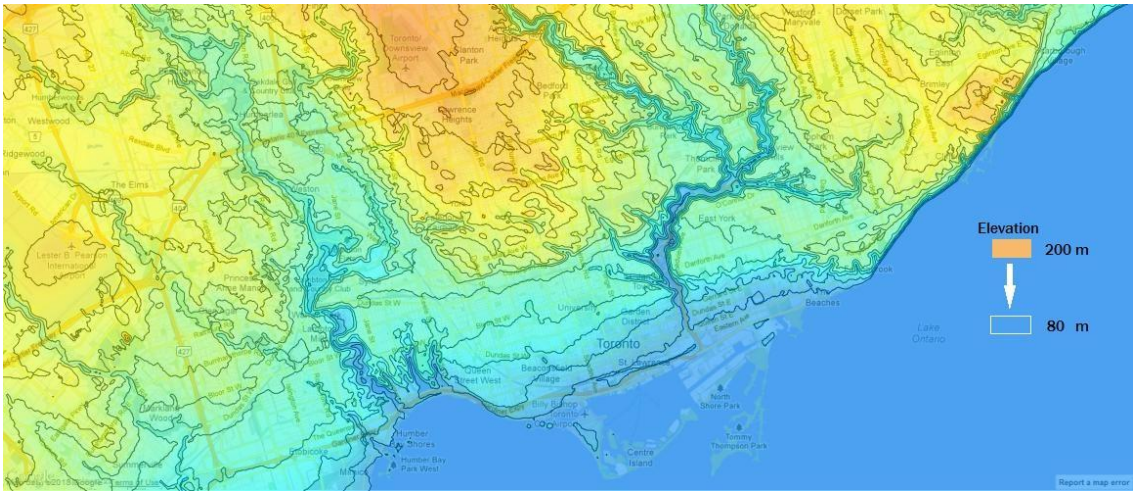

Figure: Elevation map of Toronto and region. Each line corresponds to 10 m in elevation (University of Toronto, 2010).

Whenever a user answers a survey, their answers are associated with the spatial location provided by GPSTracker and AltitudeProcessor and form a submission managed by the CheckPointManager. This information is visualized through the Mapper with respect to the currently selected survey, which is managed by the SurveyManager. The data collected by the

[1] http://weather.weatherbug.com/

phone is communicated by the DataManager to the server. The Login Manager authenticates users, since we require users to create an account in order to manage the submissions properly and provide access control.

The web application communicates with the mobile application through a collection of services represented by the DataService component. It allows the users to create an account and sign in as well as authenticate via phone through the Authenticator and UserManager components. SubmissionManager takes care of all the operations on submissions including searching, filtering and saving. Management functionalities on surveys as well as editing, removing or creating of surveys is done through the ProjectManager. As for the data model, submissions and surveys are associated with several entities (submission, survey, user, question, answer, answer type, point, option), the persistence and retrieval of which is provided by the DBManager component. The data can be exported through the application in XML and CSV formats using the services of the DataProcessor.

Implementation

The client is implemented as an Android application using Google Maps Android API v2[2] to provide the main mapping view and ActionBarSherlock[3] library to achieve a uniform minimalistic look and feel across all Android devices.

The server is implemented as a Java EE 6[4] web application and uses several technologies from the Java EE stack. The frontend is based on JavaServer Faces 2[5] and RichFaces 4[6] framework and uses AJAX heavily. The application exposes the data via REST in XML using Java API for RESTful Web Services (JAX-RS)[7] and Java Architecture for XML Binding (JAXB)[8], which is used to get data from the server to the client. Communication in the other direction is facilitated using Java Servlet 3[9]. Main application logic is provided by Enterprise JavaBeans 3[10] facilitating Context and Dependency Injection (CDI)[11]. Authentication and authorization is done via Seam Security[12]

---

[2] https://developers.google.com/maps/documentation/android/
[3] http://actionbarsherlock.com/
[4] http://http://www.oracle.com/technetwork/java/javaee/overview/
[5] http://javaserverfaces.java.net/
[6] http://www.jboss.org/richfaces
[7] http://jax-rs-spec.java.net/
[8] http://jaxb.java.net/
[9] http://jcp.org/aboutJava/communityprocess/final/jsr315/
[10] http://www.oracle.com/technetwork/articles/entarch/ejb-3-085455.html
[11] http://www.oracle.com/technetwork/articles/java/cdi-javaee-bien-225152.html
[12] http://www.seamframework.org/Seam3/SecurityModuleHome

and persistence is implemented via Java Persistence API 2[13] with object-relational mapping provided by Hibernate[14].

3. Functionality

This section delves further into the functions of UrbanEyes. To start off, in order to be able to collect data and create surveys, a user has to create an account through the web application.

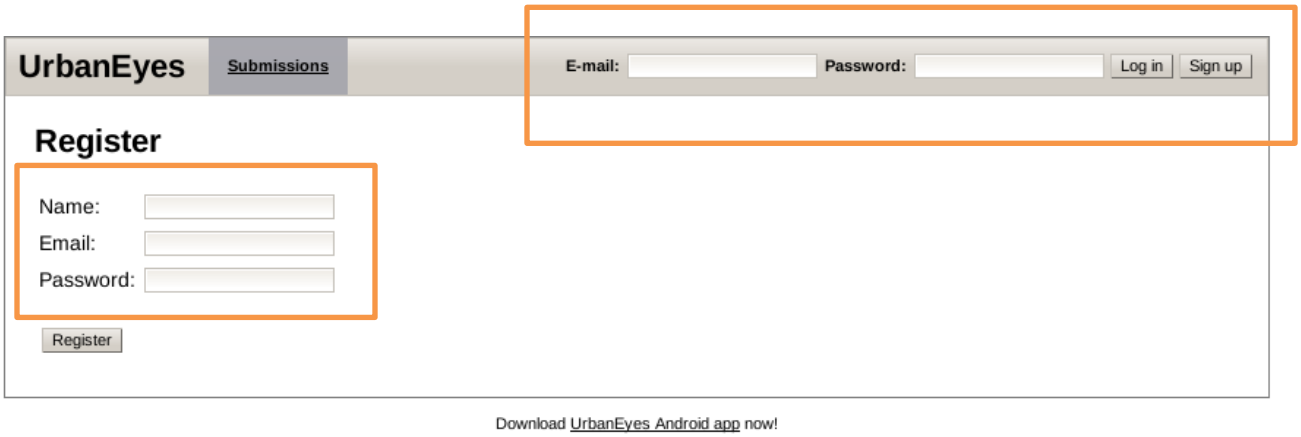

Figure: Sign-up and login dialogs in the web application.

After signing in, the user is presented with several views. The basic view shows the user's surveys allowing them to view, edit, delete or export submissions to XML and CSV and an option to create a new survey.

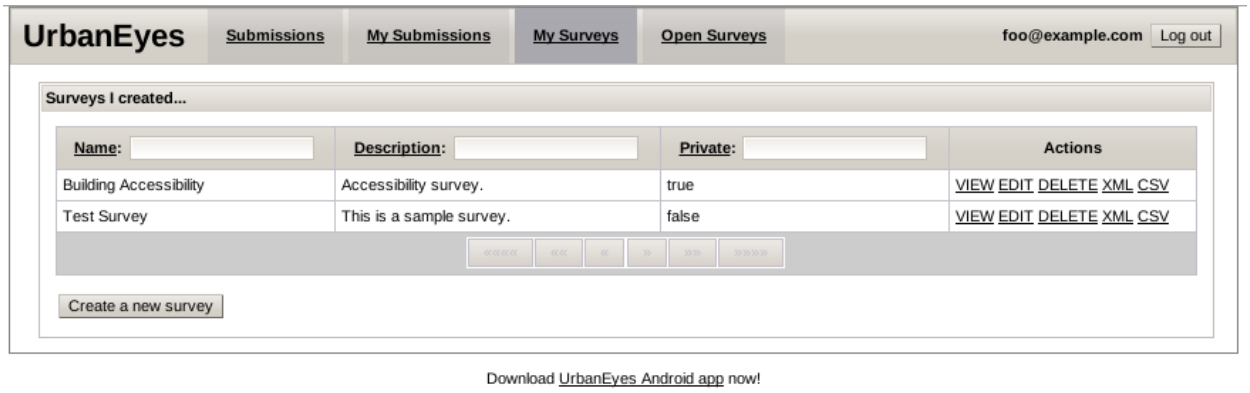

Figure: My Surveys view in the web application. The data is shown in a table with sorting and filtering capabilities.

[13] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[14] http://www.hibernate.org/

When creating a survey, a user can specify its name, description, privacy status, questions and contributors. Privacy status determines the access rights to a survey - a survey can be public (accessible to every user) or private (accessible to specific users). Questions are typed in and the type of the answer is offered, which is used on the client side to generate the correct input. When editing an existing survey, a user is allowed to change anything except the questions to prevent inconsistencies with existing submissions.

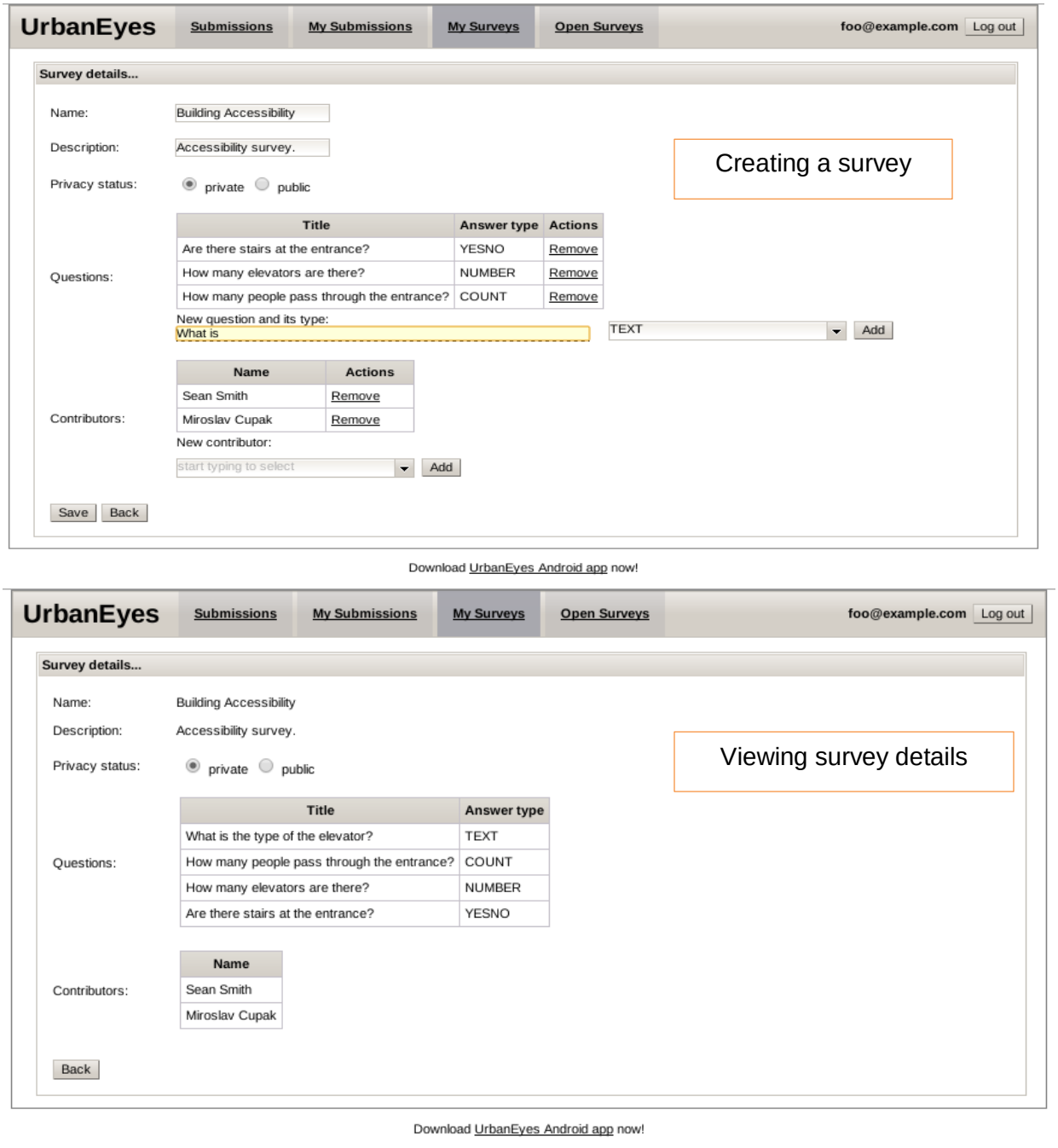

Figure: Pages for creating and displaying surveys.

The application allows for viewing real-time survey submissions from the client. The user can choose to view personal submissions or view all submissions for a survey they have access to. More detailed information can be displayed within the application or exported to XML or CSV.

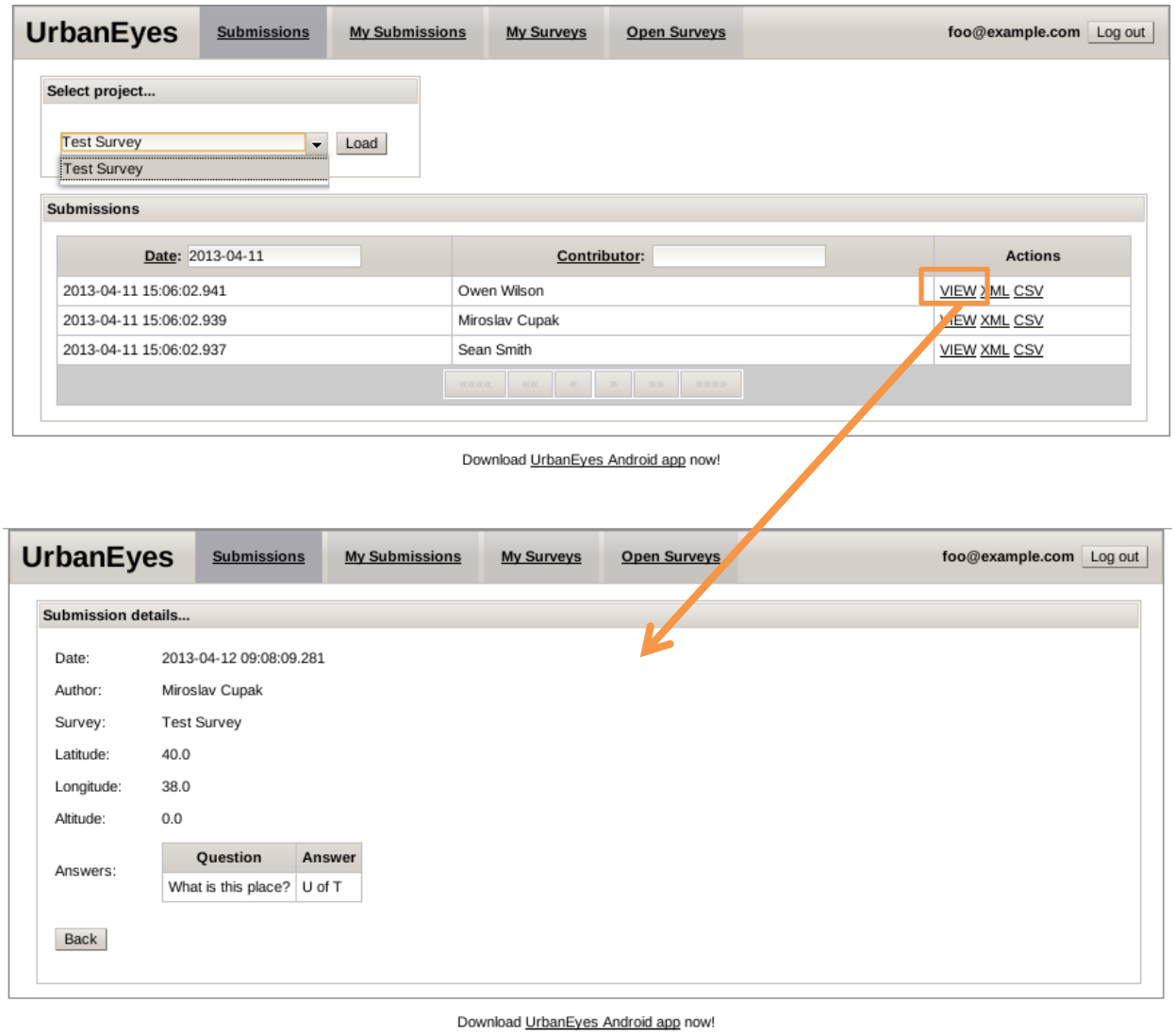

Figure: Submissions and detailed view. The data is shown in a table with sorting and filtering capabilities.

When using the mobile application, the user is first presented with the login screen prompting them for their username and password:

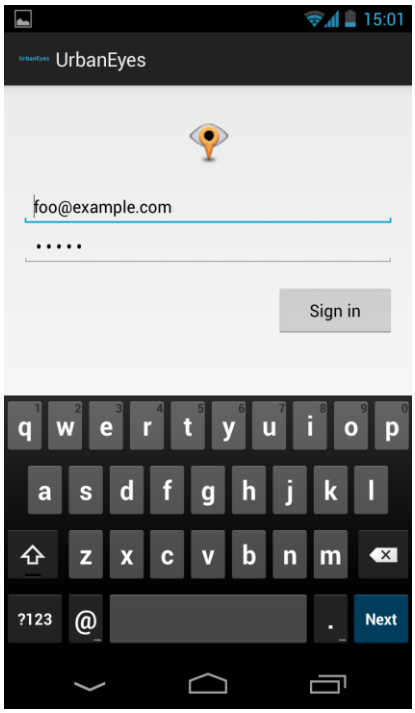

Figure: App's login screen.

After logging in, the user is presented with a list of surveys they are allowed to contribute to. The list is pulled from the server and contains surveys of 3 types – point, path and area surveys.

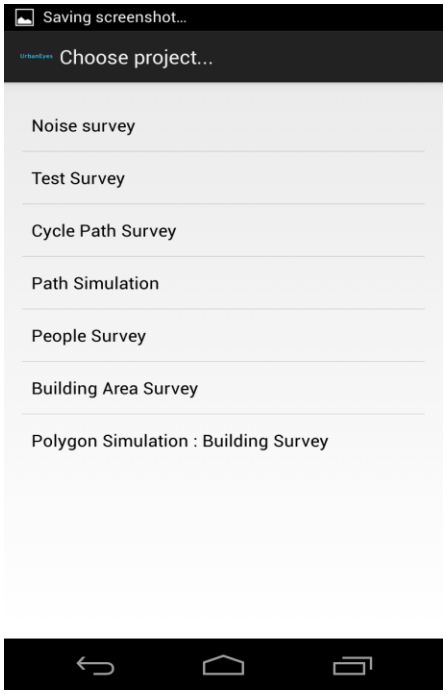

Figure: The list of surveys obtained from the server.

After selecting a survey, the user can choose to collect data by clicking the button “ADD POINT”.

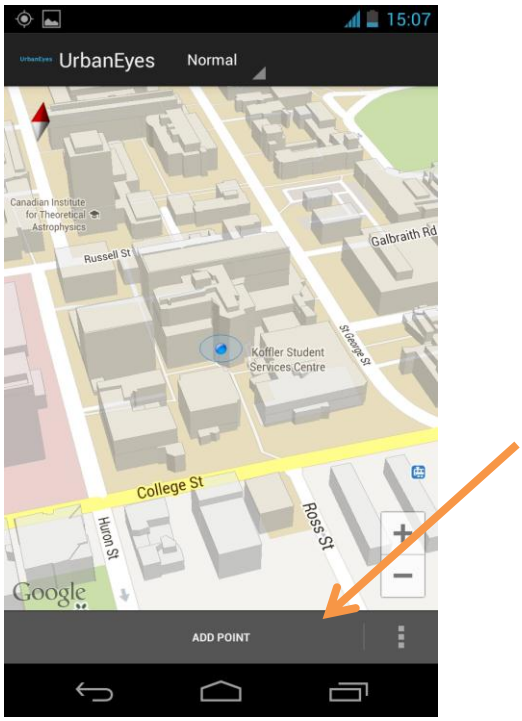

Figure: UrbanEyes Map view.

Subsequently, the survey is brought up. The forms are generated based on the question’s type e.g. the input to a ‘YESNO’ question is presented as 2 radio buttons (yes and no), the input to a ‘COUNT’ question is presented with a simple clicker-like screen etc.

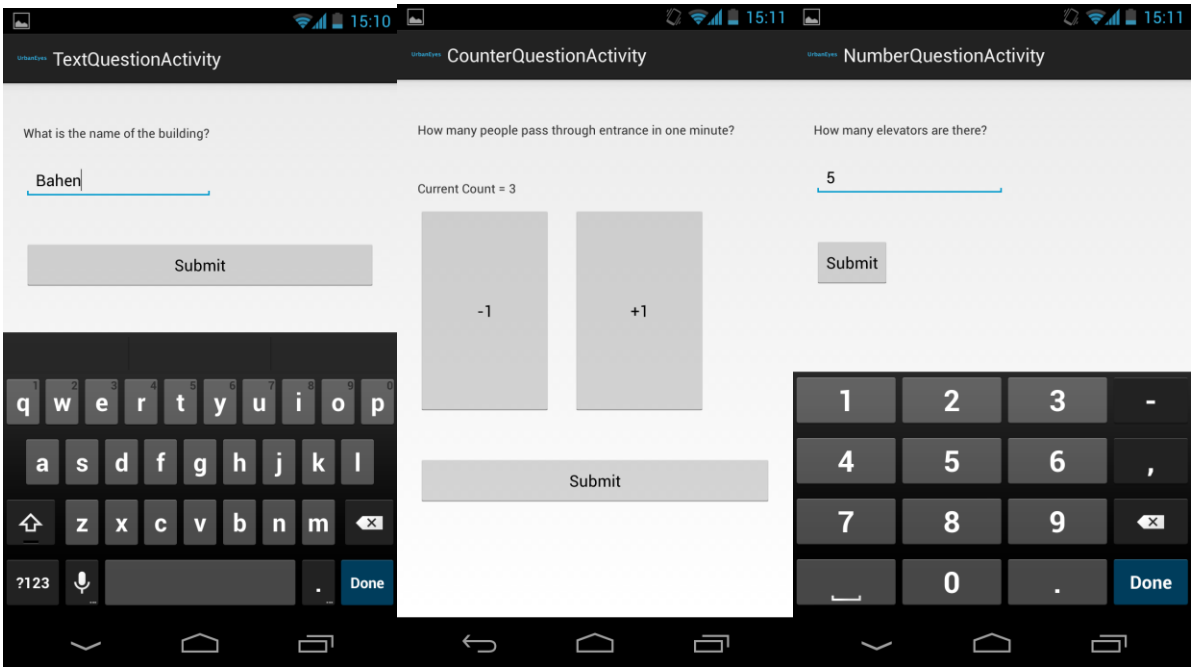

Figure: Different types of survey questions (text, count and number).

After submitting answers, a point is marked on the map and the data is sent to the server. This process is similar for paths and areas, the only difference is that the user must move around to draw these shapes.

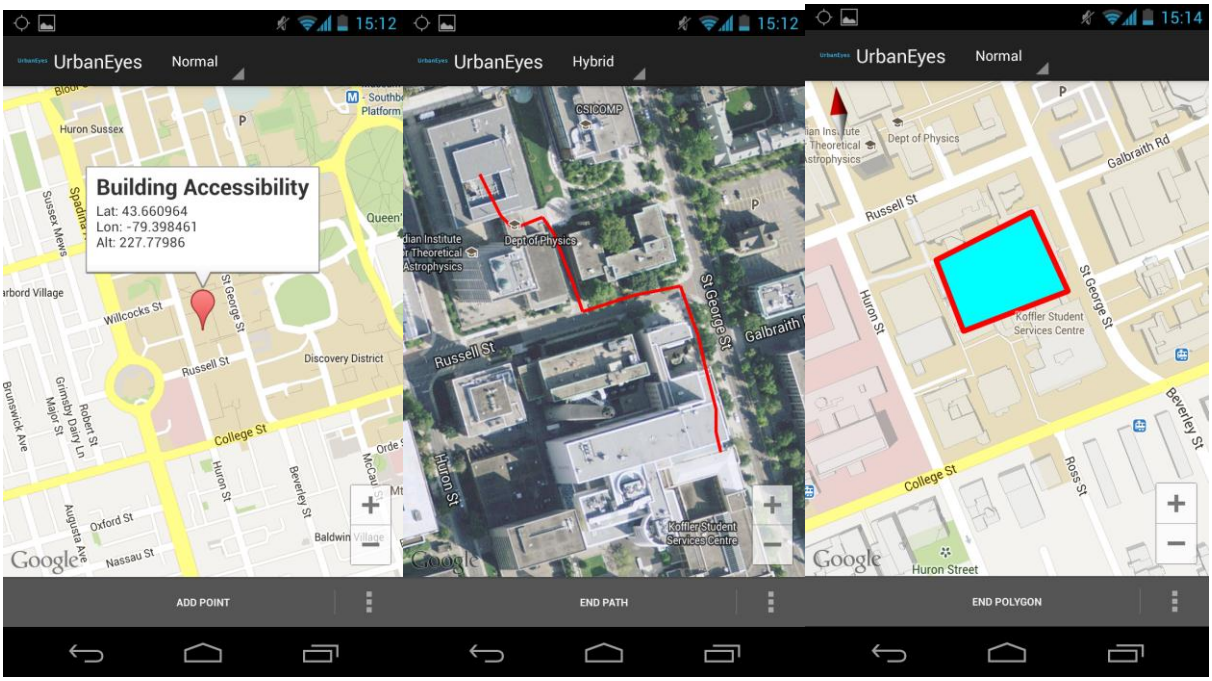

Figure: Points, paths and areas as shown on the map.

Seeing as mistakes and misplacement of points are possible, points can be moved and edited to update the data. This is especially important in urban settings, where GPS sensors can be affected by tall buildings.

The data we measure can be used as an input to mapping programs. Currently, our web application is able to export data in XML and CSV.

ID	DATE	SURVEY_ID	LAT	LON	ALT	QUESTION_4	USER
13	2013-04-12 09:08:09.28	6	40	38	0	I don't know.	foo@example.com
14	2013-04-12 09:08:09.281	6	40	38	0	U of T	mirocupak@gmail.com
15	2013-04-12 09:08:09.292	6	40	38	0	Toronto	ow@gmail.com

Figure: Sample submissions exported to CSV and XML.

4. Lessons Learned

The lessons and key learning related to project management, communication, feedback, and GPS data on mobile devices. The main problems we encountered stemmed from the initial change in direction of the app, the establishment of data formats and the work required for the web application.

At the beginning of the project, our idea for UrbanEyes included a counter for moving objects using the camera. This was not feasible and as such, significant time was spent reorienting the project. For the programmers, this required an understanding of OpenCV. For the apper, this required rethinking the relevance of the app to urban planning. This aspect of the app would not have been included if done differently.

In addition to this, it was clear from the beginning that there were problems with explaining why the app was relevant and related to the lack of an engaging use-case. The apper learned the value of having clear, real and impactful use cases when proposing apps and the value of clear and simple communication. Furthermore, the programmers learned to question ideas and think critically about the purpose of technologies they are developing.

Key learning for all involved was the value of regular, task-oriented meetings. Meetings were held weekly after class and before presentations to practice. After them, action plans and notes were shared. The value of this direct communication was essential in moving the project forward and implementing feedback.

The programmers had to gain significant technical knowledge. This involved understanding of Java EE stack and all the related technologies (see implementation) as well as creating a full-featured web application. Significant effort was put into understanding altitude computation using a barometer, understanding Google Maps API and integrating Google Maps with Android. The apper had to understand how shapefiles (the industry standard file format for geographic information) are encoded and can be created, as well as how Google Maps encode coordinates. For all, understanding the strengths and weaknesses of GPS was necessary.

5. Breakdown of Work

Ravi

- client-server information interchange spec
- line and polygon setting tools, mobile user interface, survey generation, communication with server

Sheraz

- how to convert coordinates into Shapefiles and formatting CSV files for import
- researching geographic coordinate system transformations required for GPS data from smartphone
- altitude measurement calculation and verification
- testing and user interface recommendations
- project management and meeting coordination

Miroslav

- complete web application including data management, access control, communication with the client, sorting, filtering and export of data to XML and CSV, user interface design
- mapping part of the mobile application including a point marking tool
- altitude measurement based on barometer and data from weather website

6. Future Work

Some features should be added before the UrbanEyes is launched. On the mobile app, adding sound measurement and better organizing project types would be helpful additions. On the web application, converting CSV files to shapefiles would be an extremely useful addition. Another useful addition would be to allow researchers to upload locations for surveyors to collect data from. Choosing layout and colours for the website and embedding a Google map could be useful additions as well.

7. Relevance to Urban Planning Research

At its core, UrbanEyes achieves two purposes: (1) it makes the process of collaboratively collecting data simple and streamlined, and (2) it makes this capability available to a wide audience through a very simple user interface on both the web and the smartphone.

While it might be logical to assume that people studying cities (especially planners) would be well versed in mapping, this is generally untrue. A sort of paradox exists in the urban research fields, in that planners and researchers require spatial data to make recommendations about urban settings, but not all planners have access to mapping and GPS tools. Often planners are not trained in collecting GPS data using technical instruments, and mapping software can be prohibitively expensive. With the development of Google maps and open source software for mapping, more community groups, non-profits and even governments have been *allowed* to map, but significant barriers still exist.

Problems still arise in that, while tools exist to create maps, the data to create the maps are unavailable. This is especially true in Canada, where geographic data control has had a long and generally proprietary nature (i.e., data is often not shared). While this is changing with the development of open data portals in Canadian municipalities, there are still many types of data that have never been collected. UrbanEyes makes this possible, especially on the small scale.

Cases of this can be seen throughout cities. For example, when non-profits concerned with the accessibility of buildings in a region of a city want to make recommendations to city council, having a list of buildings with and without accessible entrances would be useful. Having the locations of the entrances can allow the non-profit to estimate how much farther people with disabilities have to travel in comparison to people who can walk. This information can inform policy, but does not currently exist.

While Canadian practice lacks spatial data, other locations may lack data in even more pronounced ways. In cities in developing countries, data is also generally unavailable for both the city and for the residents. Cities need data to efficiently address problems present in their jurisdiction, while residents need data to hold their governments accountable. While UrbanEyes may not be a world-wide sensation, the app contributes to the solution of these problems. Using common and familiar tools, data can be used to inform and make recommendations about cities based in evidence. With mobile internet service increasing yearly and 3G becoming increasingly available (ex, Brennan 2012), this will become even more important in the future.

What was achieved in the app is a way for people to share the data they have collected, especially as contributions to open data. Opening ownership to spatial data made by people is important for keeping governments and planners accountable as projects move forward.

8. Business School Interest

We are not interested in having Business School class on marketing/entrepreneurship take up UrbanEyes.

9. Source Code

We have made the source code for both mobile and web application available online at https://github.com/mcupak/urbaneyes, where we plan to develop it further after the course.

10. Technical Resources

We found several resources very helpful during the implementation process.

Drawing maps on Android

- Google Maps Android API v2 Documentation and Samples: https://developers.google.com/maps/documentation/android/ (Apr 2013).

Barometer usage for altitude measurements

- Samsung Developers Technical Docs: http://developer.samsung.com/android/technical-docs (Apr 2013).
- WeatherBug API: http://developer.weatherbug.com/docs/read/WeatherBug_Rest_XML_API (Apr 2013).

Authentication and authorization

- Seam Security Examples: https://github.com/seam/examples (Apr 2013).

RichFaces

- RichFaces Showcase: http://showcase.richfaces.org/ (Apr 2013).
- RichFaces Developer Guide: http://docs.jboss.org/richfaces/4.3.X/4.3.1.Final/Developer_Guide/en-US/html/ (Apr 2013).

Java EE 6

- JBoss Quickstarts: http://www.jboss.org/developer/quickstarts.html (Apr 2013).
- Java EE 6 Specifications: http://www.oracle.com/technetwork/java/javaee/tech/index.html (Apr 2013).
- Java Servlet Tutorial: http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html (Apr 2013).

Web Application Deployment

- OpenShift User Guide: https://access.redhat.com/site/documentation/en-US/OpenShift/2.0/html/User_Guide/ (Apr 2013).

References

American Meteorological Society. 2012. Hypsometric Equation. Retrieved online at http://glossary.ametsoc.org/wiki/Hypsometric_equation. (Apr 2013).

Brennan, C. (Oct 31, 2012). Satellite tech powers 3G revolution for rural Africa. CNN. Retrieved online at http://edition.cnn.com/2012/10/29/business/3g-africa-mobile-broadband (Apr 2013).

Ipsos. 2013. Close to Half of Canadians Now Own a Smartphone. Retrieved online at http://www.ipsos-na.com/news-polls/pressrelease.aspx?id=6005. (Apr 2013).

University of Toronto. 2010. Toronto Digital Elevation Model. Retrieved online at http://maps.library.utoronto.ca/. (Apr 2013).