# Applied Project

## MAD9145

# SMART CITY TALK

Missing Pixel Team

Riley Griffith
Designer & Developer

Zaheed Rizwan Jaffer
Designer & Developer

Alison Kapcala
Lead Designer

Yanming Meng
Designer

Robson Miranda
Team Leader

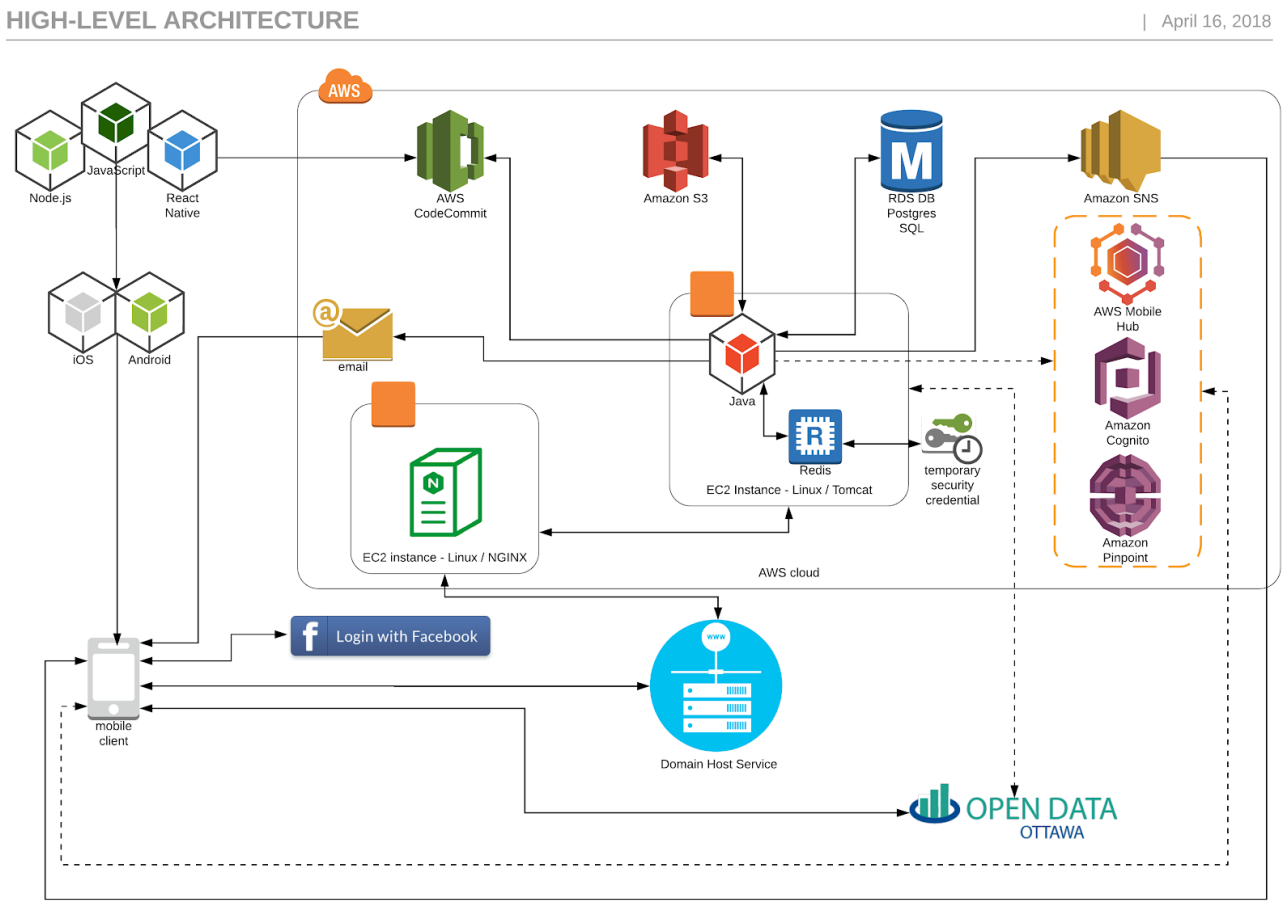
Michel Toutain
Designer & Developer

Jacob Wagner
Lead Developer

# Smart City App – Deployment Document

## Deployment Diagram

Smart City Talk is a Mobile solution built with React-Native technology and deployed natively to iOS and Android devices. It runs on top of Amazon Web Services and uses an EC2 backend engine instance running a Java Spring Restful API. The services provided by the backend lay on Cache Redis for prompt information, backed up by an RDS instance running Postgres RDMS. It also uses a S3 bucket as image storage and SNS and SES as message and email services. Both frontend and backend sources are stored in git repositories saved on CodeCommit Service. This solution is based on the Open 311 API (http://www.open311.org/) and is ready to pull and push data using the city of Ottawa's Open Data API support.



# Smart City App – Deployment Document

## Hardware and Software Requirements

- The iOS app only runs on iOS 9 or higher
- The Android app only runs on Android 5.1 (Lollipop 2) or higher
- The Frontend structure must be hosted within a JavaScript environment:
  - @expo/vector-icons: ^6.2.2
  - axios: ^0.17.1
  - base-64: ^0.1.0
  - class-autobind: ^0.1.4
  - expo: ^22.0.2
  - jest-expo: ^22.0.0
  - native-base: ^2.3.3
  - react-native-drawer: ^2.5.0
  - react-native-easy-grid: ^0.1.15
  - react-native-google-places-autocomplete: ^1.3.6
  - react-native-maps: ^0.17.1
  - react-native-push-notification: ^3.0.2
  - react-native-scripts: 1.7.0
  - react-native-stars-rating: ^0.1.7
  - react-native-vector-icons: ^4.4.3
  - react-native: https://github.com/expo/react-native/archive/sdk-22.0.0.tar.gz
  - react-navigation: ^1.0.0-beta.19
  - react-redux: ^5.0.6
  - react-test-renderer: 16.0.0-beta.5
  - react: 16.0.0-beta.5
  - redux-thunk: ^2.2.0
  - redux: ^3.7.2
  - watchman: ^0.1.8
- The web server must be hosted within Apache Tomcat environment:
  - Server version: Apache Tomcat/8.0.35
  - Server built: May 11 2016 21:57:08 UTC
  - Server number: 8.0.35.0
  - OS Name: Linux
  - OS Version: 3.13.0-74-generic (Ubuntu 4.8.2-19ubuntu1)
  - Architecture: amd64
  - JVM Version: 1.8.0_91-b14
  - JVM Vendor: Oracle Corporation
- The Backend engine must use Java workspace persistence:
  - **JDK Version: 1.8**
  - Aspectj Version: 1.8.9
  - AWS Java SDK Version: 1.11.248
  - Commons Fileupload Version: 1.3.1
  - Commons Pool2 Version: 2.4.2
  - EasyRules Version: 2.5.0
  - Eclipse Persistence JPA Version: 2.6.3-M1

- Eclipse Persistence Version: 2.1.1
- Ehcache Version: 2.10.1
- Fasterxml Jackson Version: 2.7.3
- Geotools Version: 15-M0
- Google Code GSON Version: 2.8.2
- Google Guava Version: 19.0
- Hibernate C3P0 Version: 5.1.0.Final
- Hibernate Spatial Version: 5.1.0.Final
- Hibernate Version: 5.1.0.Final
- HTTPClient Version: 4.5.5
- HTTPCore Version: 4.4.9
- JavaX Mail Version: 1.5.0-b01
- JavaX Validation API Version: 1.1.0.Final
- Jedis Client Version: 2.8.1
- JSTL Version: 1.2
- LOG4J Version: 1.2.17
- NET PostGIS JDBC Version: 2.2.0
- Openimaj Version: 1.3.1
- Postgresql Version: 9.4-1200-jdbc4
- SLF4J Version: 1.7.19
- Spring Security Oauth2 Version: 2.0.9.RELEASE
- Spring Security Version: 4.1.0.RELEASE
- Spring Version: 4.3.4.RELEASE
- Spring-Data-JPA: 1.9.4.RELEASE
- Stripe Version: 5.21.0
- XML-APIs Version: 2.0.2

- The Backend cache must use Redis:
  - Server Version: 3.2.0
  - Client Version: 3.2.0
- The relational database should be PostgreSQL Version 9.4.7 or higher.
- The subsequent AWS Services must be available:
  - S3: Image hosting service
  - SNS: SMS Text Message feature
  - SES: Email Service

# Smart City App – Deployment Document

## Installation

This project should be hosted on MGIS' current structures. For the source code control, CodeCommit on AWS should be used. The client will be provided with a new repo called "smartcity" and all the frontend source code will be directly deployed in it (https://git-codecommit.us-east-1.amazonaws.com/v1/repos/smartcity). The final version provided there should be built and prepared for further deployment through Xcode (iOS) and Android SDK to their respective Apps in the MGIS' accounts with both stores.

All the backend changes are hosted in a separate branch called "SmartCity" within the Registration Service repository (https://git-codecommit.us-east-1.amazonaws.com/v1/repos/registration-service). This new branch should be merged and validated in the Master Branch (Trunk) and then tested before being deployed to the Production server in the AWS EC2.

Database changes will also be hosted in the "SmartCity" branch and MUST be applied to the RDS production instance BEFORE the backend starts be tested. The changes in the database relate to the User's Request process and adapts the table USER_REQUESTS to store the Open311 API standard attributes. Also, the basic settings, as types, categories, and app configuration were set in the DEV environment and will be available through the "SmartCity" Branch. Update and Insert statements should be executed in order to add all these settings to the Production environment as well.

Regarding to the City of Ottawa's Open Data, which provides all the 311 information for the project, the MGIS' key is being used and there is no need to change it during the test period. However, as long as the application is stable and will be pushed to the public, a change in the domains in the Open311Service.java should be done. The main domain "https://city-of-ottawa-dev.apigee.net/" should be changed to "https://city-of-ottawa-prod.apigee.net/" in order to post and receive requests from the production environment, instead of the development one.

All the documents supporting the implementation and structure of the project will be included along with the source code (https://git-codecommit.us-east-1.amazonaws.com/v1/repos/smartcity). There will be a folder called "docs" in the repository containing all the documents produced along the project. Any other detailed information needed in the future, please, contact us.

# Smart City App – Deployment Document

## Deployment

This document provides a brief overview for the ways to deploy your mobile solution:

This document explains the following ways to deploy a mobile solution:

1. Android Deployment
2. Apple's App Store

The above platforms were selected based on the collective experiences of project teams from previous editions of this course. For other platforms, such as BlackBerry or Windows Phone, your team will need to research online for how to deploy an app to your team's unique platform.

As known by your team, you should follow the steps below to add the app to the stores:

**Google Play**

1. Build the React Native Project
2. Eject the React Native Project to generate the Android Platform Folder
3. Open the Android Project with Android Studio and generate a new KeyStore for the Smart City Talk Project
4. Give this key a password and store it in a safe place and close Android Studio
5. Generate the Release Android APK file using the command line.
6. Sign the APK file using the jarsigner command line
7. Align the APK using the zipalign and generate the aligned version of your APK
8. On Play Store, create a new Application and fill out all the required information (Exclamation marks are posted to the required sections you must complete)
9. Once all required information is filled, jump to the Release Management section, and then to App Release.
10. Choose your Release Type (Alpha, Beta or Production), browse and upload your APK and fill out the comments for this release. Review it and Roll out to the Store.

**Apple Store**

1. Build the React Native Project
2. Eject the React Native Project to generate the iOS Platform Folder
3. Open the iOS Project with XCode, clean and Build the Project
4. Go to the iTunes Connect, create a new Application and fill out all the required information (Exclamation marks are posted to the required sections you must complete)
5. Go back to Xcode, select Devices, and then Generic Device.
6. Go to Product and then click on Archive. Follow the instructions, check the Application that will receive the archive (It should be the same one you created in the step above) and upload to the store.
7. Once the archive is processed, go to TestFlight Tab in the iTunes Connect Page and fill out the compliance for test version. This step allows selected users to start testing the application on the TestFlight App provided in the Apple Store.

8. Go back to the App Information, select the iOS Version you are about to release and look for the Build Section. Select the Build you've just uploaded.
9. Review the information provided and as long as everything is ok, save and put the App up for Review.
10. Once it's accepted by Apple, the app will be available in the Apple Store.

**Accounts, Usernames and Passwords**

All the accounts and keys used in this project were property of MGIS Inc. and can be found within the cloud services provided.

## Outstanding Issues

All the known issues listed below should be addressed before the final release. They were found during the test and presentation periods.

- Provide Privacy Policy Info in the About/Register/Profile Page:
    - Subject to the City's Election Related Resources Policy (http://ottawa.ca/en/city-hall/your-city-government/policies-and-administrative-structure/administrative-policies#election-related-resources-policy), I consent to provide my personal information in this service request to the Councillor of the ward to which the service request relates for the purpose of receiving communications relating to City business or activities within that ward.
- Provide checkbox with text:
    - I consent to provide my personal information
- Redundant API calls on data update. Revise the frequency of calls in the frontend.
- Distance matrix calls too often. Prevent calls once the user does not move more than a certain distance in radius.
- Login with Facebook is missing. Application with Facebook is created, but it needs to be implemented in the frontend.
- If the user creates an account in the app and logs out the only way back in is to make a new user with a new number or using the same number you can't change between users.
- Image sizing should be reduced due to the final file size stored in the S3 service.
- Distance sorting needs to be defined according to the company premises.
- Image rotation (on iOS). It shows the image in landscape, always rotating 90°.
- Style in all pages needs to be revised, as some of the pages have issues.
- Constant fetch of user location causes issues on Android.
- If entering camera for 2nd time the old image is still displayed.
- Image cropping to be shown on the Request Detail view.
- Need callouts for stacked pins.
- Fix interactions with iOS status bar.
- Search auto-complete parameters need to be reviewed.
- Profile Form needs to be validate before posting.
- Needs password change implementation

## Future Implementation

Below is a list of future features that could be added to the app:

- French and English Toggle (Strings are currently externalized)
- Worker Dashboard
- Pin Clusters on the map
- Heat Map layer
- Video submissions
- Edit Pictures before sending request
- Specific features for city workers
    - Add status updates
    - Add notes for the city and other service workers
    - Close requests

## About Us

In the case of future available positions with MGIS, or further recommendation to your partners, please keep our professional information. We will be glad to keep in touch and participate in the Smart City Talk next steps.

Riley Griffith
griffith.riley@gmail.com
613-866-6839

Zaheed Jaffer
Jaffer.zaheed@gmail.com
613-355-5461

Alison Kapcala
alisonkapcala@gmail.com
613-863-2688

Robert Yanming
robertyanming@gmail.com
613-866-2676

Robson Miranda
robsonscm@gmail.com
613-700-0017

Michel Toutain
toutain.michel@hotmail.com
613-266-4177

Jacob Wagner
jacobwagner613@gmail.com
613-608-6608

Finally, we would like to say thank you for the opportunity and for all the support we received from Marc Guindon and Matt Page in name of MGIS Inc.

Team Missing Pixel