

Lab 10 - Using the Mobile Quality Assurance Service

In this lab, you will get a chance to use the Mobile Quality Assurance service in Bluemix to test mobile application builds on device, which enables you to report bugs and feedback in testing or in production. It also provides key views into user sentiment of your application once it is has been published to public apps stores, which helps you maintain and continually enhance your app based on user sentiment and feedback.

Lab 10 - Objectives

In the following lab you will learn to:

- Instantiate the Mobile Quality Assurance (MQA) service on IBM Bluemix
 - Instrument an application to leverage MQA
 - Review bug reports and feedback using the MQA service on IBM Bluemix
 - Upload application binary for distribution
 - Distribute an application build using MQA
 - Configure and view user sentiment of Consumer Applications
-

Lab 10 - Before you begin

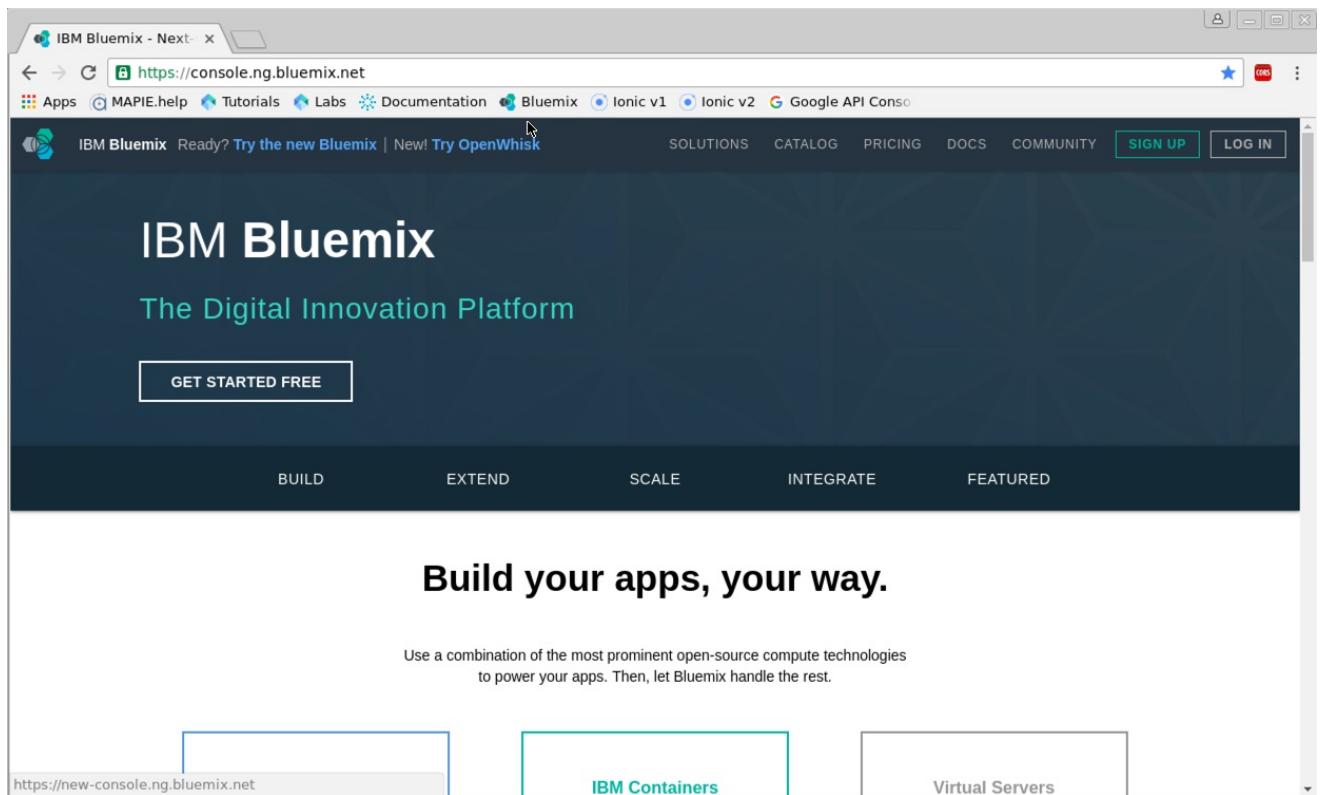
In this tutorial you will need

- A Bluemix account
- Source code for labs 1-8

Lab 10 - Step by Step Lab Instructions

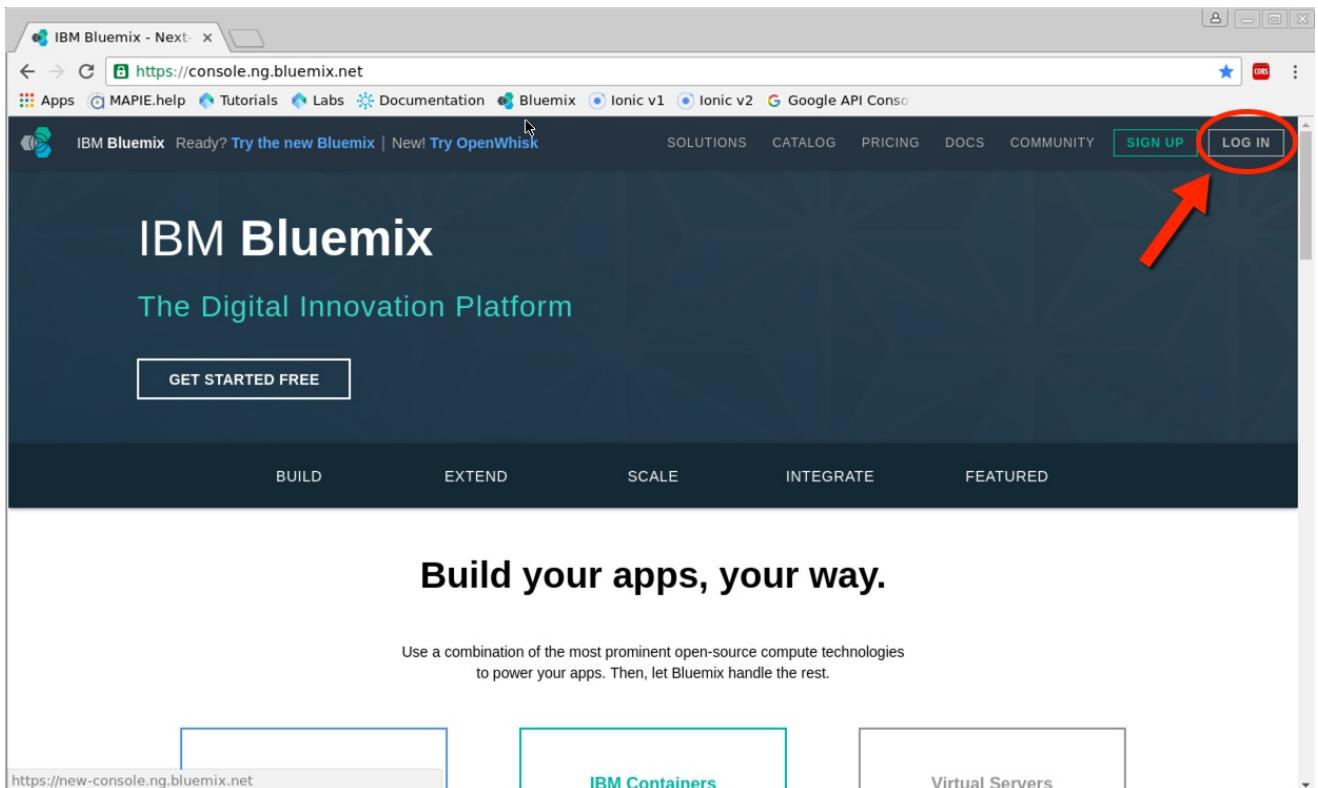
Section 1 - Instantiate Mobile Quality Assurance service on IBM Bluemix

1. Start Bluemix by opening a browser and navigating to
<http://www.ng.bluemix.net>

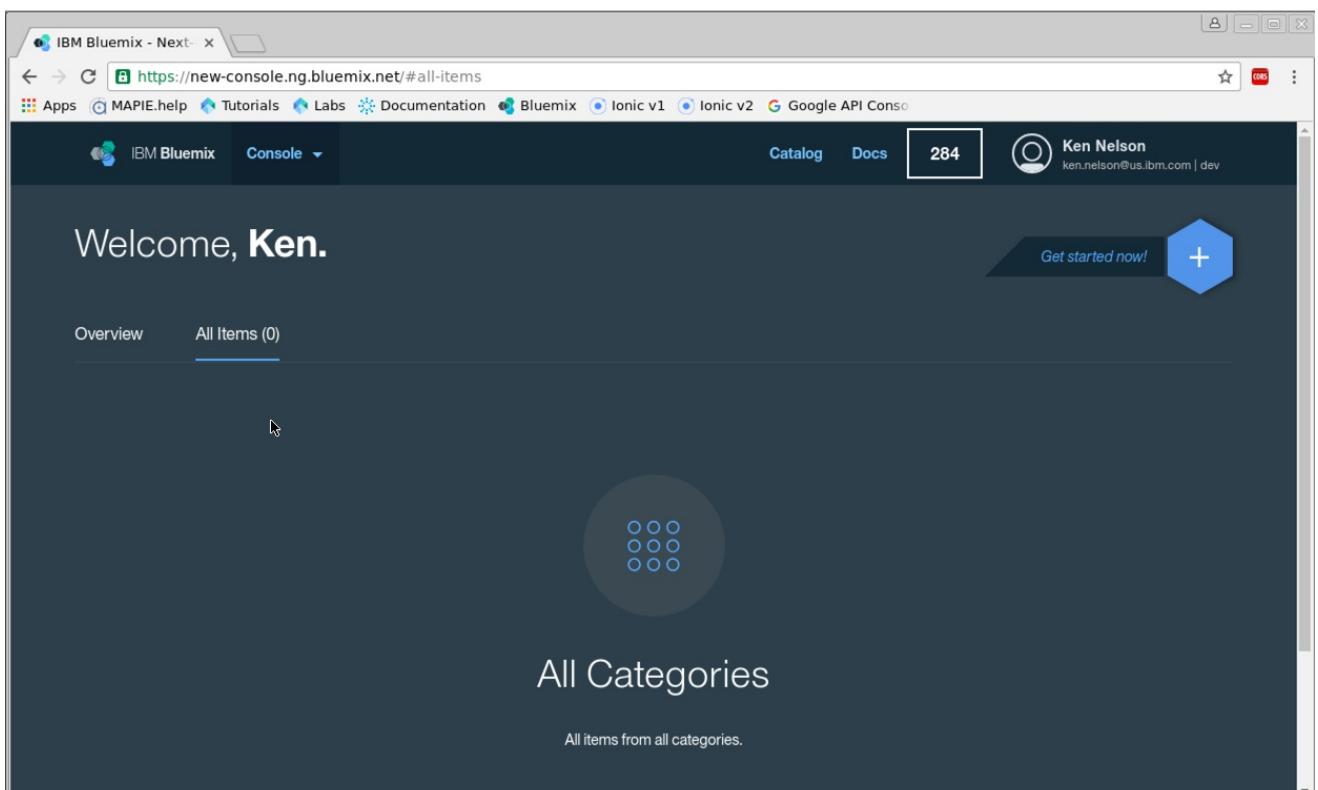


2. Login to your Bluemix account by selecting the **Log In** button in the

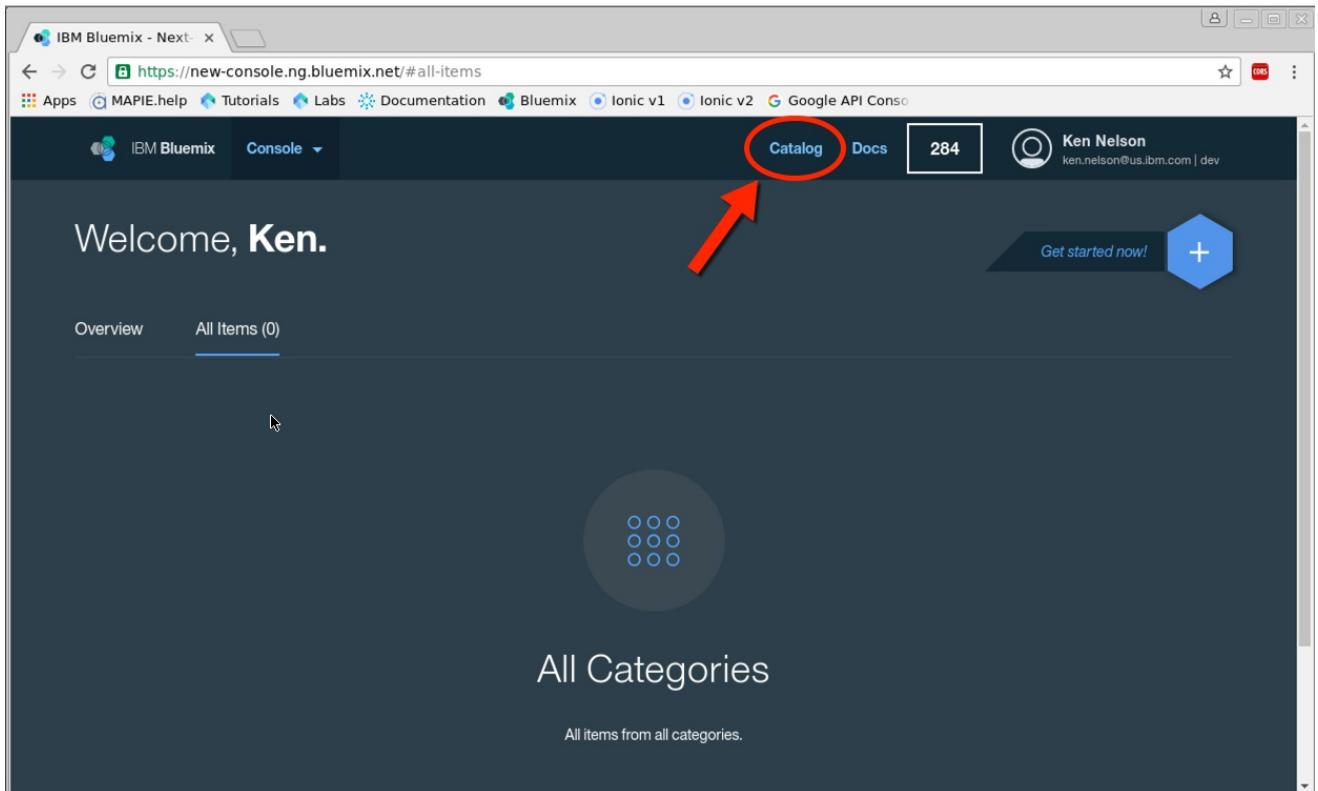
upper right hand corner of the page.



3. Follow instructions to login. When you have successfully logged in you should see a Welcome page.



4. In the upper right hand corner of the page, click the **Catalog** link. This will open the catalog of available Bluemix Services.



- Once the **Catalog Page** is open, click the **Mobile** link from the list of categories on the left side of the page. This will filter the available services on Bluemix to services that are categorized as **Mobile** types of services.

Note: You may have to scroll to see the Mobile link on the left side of the page.

The screenshot shows the IBM Bluemix Catalog interface. On the left, a sidebar lists various service categories: Compute, Network, Storage, Data & Analytics, Watson, Internet of Things, APIs, DevOps, Security, Application Services, and Mobile. The 'Mobile' category is highlighted with a red arrow pointing to it. The main content area is titled 'BOILERPLATES' and displays several service starters, including 'ASP.NET Core Cloudant Starter', 'Internet of Things Platform Starter', 'IoT for Electronics Starter', 'Java Cloudant Web Starter', 'Java Workload Scheduler Web Starter', and 'LoopBack Starter'. A search bar labeled 'Search' is located at the top of the main content area.

- With the **Mobile** services displayed, find and click on the **Mobile Quality Assurance** service.

The screenshot shows the IBM Bluemix Catalog interface with the 'Mobile' category selected. The main content area displays several mobile-related services, each with a thumbnail icon and a brief description. The 'Mobile Quality Assurance' service is circled with a red oval. Other visible services include 'Mobile Analytics', 'Mobile Application Content Manager', 'Mobile Client Access', 'Mobile Foundation', 'Push Notifications', 'Testdroid Cloud', and 'Twilio'. A search bar labeled 'Search' is located at the top of the main content area.

- You are now on the landing page to create a service that can be used to configure Mobile Quality Assurance for your application.

The screenshot shows the IBM Bluemix Catalog interface. At the top, there's a navigation bar with links for Apps, MAPIE.help, Tutorials, Labs, Documentation, Bluemix, Ionic v1, Ionic v2, and Google API Console. On the right side of the header, there are links for Catalog, Docs, and a user profile for Ken Nelson (ken.nelson@us.ibm.com | dev). The main content area has a title "Mobile Quality Assurance" with a "View All" link. Below the title, there's a description of the service: "Mobile Quality Assurance enables mobile beta management, mobile app testing, user validation, and streamlined quality feedback with sentiment analysis, over-the-air build distribution, automated crash reporting, in-app bug reporting and user feedback." To the right, there's a "Service name:" field containing "Mobile Quality Assurance-jp". Under the heading "Features", there are four bullet points: "Mobile Beta Management" (Create, manage and run an efficient mobile beta program. Solicit prospective beta users, customize beta programs and continuously engage with beta participants.), "Over the Air Build Distribution" (Distribute builds to devices via Wi-Fi or cellular networks), "In-app Bug/Feedback Reporting" (Submit defects and provide feedback in seconds while using the application from your mobile device. Add screenshots and in app screen videos.), and "Automated Crash Reporting" (Get notified about crashes and analyze them to fix them faster). At the bottom left, there are links for "Need Help?", "Contact Bluemix Sales", "Estimate Monthly Cost", and "Cost Calculator". On the far right, there's a large blue "Create" button.

Section 2 - Configure Mobile Quality Assurance

1. You can *optionally* change the service name if you do not like the default Service name generated. In this example the Service name is set to advancedMessengerMQA

The screenshot shows the IBM Bluemix Catalog page for the 'Mobile Quality Assurance' service. A red circle highlights the 'Service name:' input field, which contains the value 'advancedMessengerMQA'. The page also displays a summary of features including Mobile Beta Management, In-app Bug/Feedback Reporting, Over the Air Build Distribution, and Automated Crash Reporting.

2. Scroll to the bottom and select the **Standard plan** in the Pricing plans section.

The screenshot shows the 'Pricing Plans' section of the IBM Bluemix Catalog. A red circle highlights the 'Standard plan' row, which includes the feature '100 free devices per application-platform' and the price '\$199.00 USD/application-platform \$0.02 USD/addressable device'. Below this, a note states: 'Mobile Quality Assurance (MQA) services are charged based on the number of application-platform combinations and the number of devices on which each app is running.' The 'Premium plan' row is also visible, offering 'Unlimited devices for an application-platform' at a price of '\$3,000.00 USD/application-platform'.

3. Click the **Create** button in the lower right hand side of the page to create the Mobile Quality Assurance Service. This will create the service.

The screenshot shows the IBM Bluemix mobile quality assurance pricing plans. It displays two plans: 'Standard plan' and 'Premium plan'. The Standard plan includes 100 free devices per application-platform and costs \$199.00 USD/application-platform, plus \$0.02 USD/addressable device. The Premium plan includes unlimited devices for an application-platform and costs \$3,000.00 USD/application-platform. A red arrow points to the 'Create' button at the bottom right of the page, which is highlighted with a red oval.

Plan	Features	Pricing
✓ Standard plan	100 free devices per application-platform	\$199.00 USD/application-platform \$0.02 USD/addressable device
Premium plan	Unlimited devices for an application-platform	\$3,000.00 USD/application-platform

Mobile Quality Assurance (MQA) services are charged based on the number of application-platform combinations and the number of devices on which each app is running.

Need Help? Contact Bluemix Sales Estimate Monthly Cost Cost Calculator

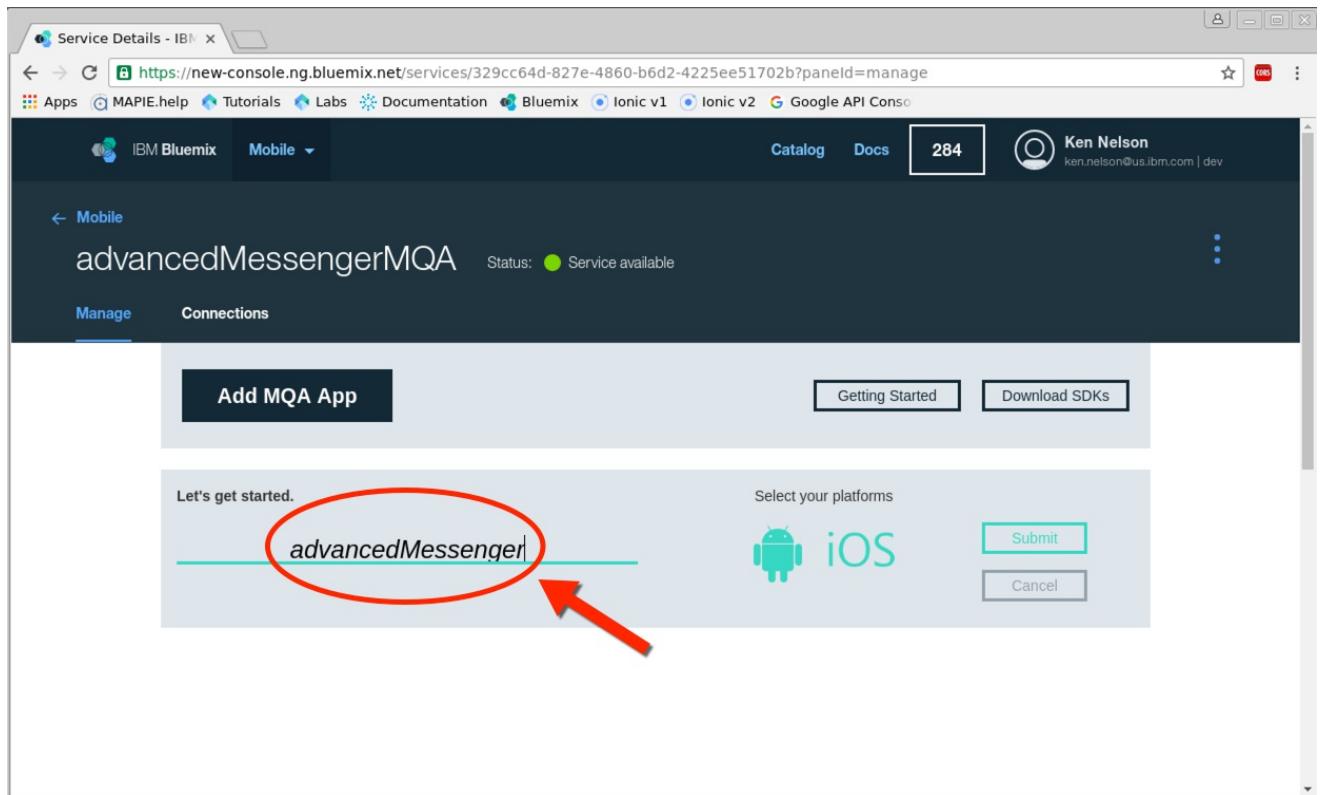
Create

Section 3 - Configure Mobile Quality Assurance Service

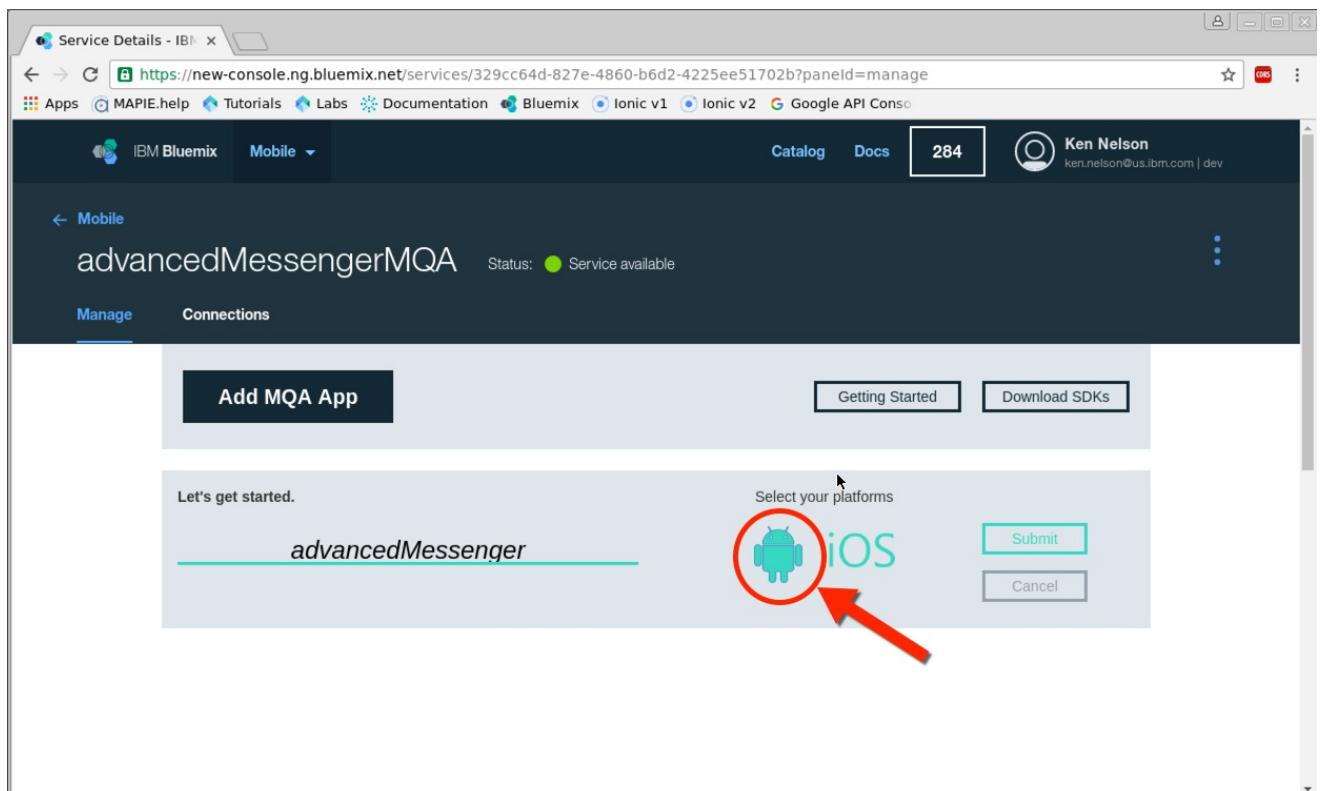
1. Click the **Add MQA App** button to begin the configuration process.

The screenshot shows the service details for 'advancedMessengerMQA'. The status is listed as 'Service available'. There are two tabs: 'Manage' (which is selected) and 'Connections'. A red arrow points to the 'Add MQA App' button, which is highlighted with a red oval. Other visible buttons include 'Getting Started' and 'Download SDKs'.

2. Enter an App name such as **advancedMessenger**.



3. Select platforms to register. For this lab select only the Android platform.



4. Press the **Submit** to complete the registration

The screenshot shows the 'Service Details - IBM' page for the 'advancedMessengerMQA' service. At the top, there's a navigation bar with links like 'Catalog', 'Docs', and '284'. On the right, it shows 'Ken Nelson' and an email address. Below the header, the service name 'advancedMessengerMQA' is displayed with a status indicator 'Service available'. There are two tabs: 'Manage' (selected) and 'Connections'. A large button labeled 'Add MQA App' is prominent. To its right are 'Getting Started' and 'Download SDKs' buttons. Below these, there's a section titled 'Let's get started.' with a text input field containing 'advancedMessenger'. To the right, there's a 'Select your platforms' section with icons for Android and iOS, and a 'Submit' button which is circled in red with a red arrow pointing to it.

5. Your MQA configuration is complete when you see the following screen:

The screenshot shows the 'Service Details - IBM' page for the 'advancedMessenger' service. The interface is similar to the previous one, with a 'Catalog', 'Docs', and '284' button at the top right. It shows 'Ken Nelson' and an email address. Below the header, it says 'Data was loaded at: 22:25:43'. The main area displays the 'advancedMessenger' service with a title 'advancedMessenger'. It features three cards: 'Preproduction (Last 7 days)' with 0 sessions and sentiment scores (0 red, 0 yellow, 0 blue), 'Production (Last 7 days)' with 0 sessions and sentiment scores (0 red, 0 blue), and 'Registered Devices (Oct 1, 2016 to Oct 3, 2016)' with 0 devices. Below these cards, it says 'Sentiment Analysis is not configured.' and has a link 'Configure Sentiment Analysis.'. At the bottom, there are four buttons: 'Upload Builds', 'Show app key', 'Distribute to Test', and 'Configure Integrations'.

Section 4 - Instrument Application (Select Android)

1. Scroll to the top of the page and **Click the Download SDKs button.**

The screenshot shows the 'Service Details - IBM' page for a service named 'advancedMessengerMQA'. The service status is 'Service available'. Below the service name, there are two tabs: 'Manage' (which is selected) and 'Connections'. Under the 'Manage' tab, there is a large 'Add MQA App' button. To its right are two smaller buttons: 'Getting Started' and 'Download SDKs'. A red arrow points from the left towards the 'Download SDKs' button, which is also circled in red. Below these buttons, a message says 'Data was loaded at: 22:25:43'. Further down, there is a section titled 'advancedMessenger' with an Android icon. It displays three metrics: 'Preproduction (Last 7 days)' with 0 sessions, 'Production (Last 7 days)' with 0 sessions, and 'Registered Devices (Oct 1, 2016 to Oct 3, 2016)' with a progress bar showing approximately 25% completion. There is also an 'Add Platform' button.

2. This will open a new browser window or tab with links to download the appropriate SDK.

The screenshot shows the 'IBM Knowledge Center' page for 'IBM Mobile Quality Assurance for Bluemix'. The current section is 'Downloading the libraries'. It provides instructions for downloading the library for the platform your app runs on. Below this, there are sections for 'Android SDK for download', 'iOS SDK for download', 'Downloads for JavaScript SDK for IBM MobileFirst Platform Foundation', and 'Mobile Quality Assurance plug-in for Apache Cordova for download'. At the bottom of the page, there are links for 'Contact', 'Privacy', 'Terms of use', 'Accessibility', 'Feedback', 'Cookie preferences', a language selector set to 'English', and a 'Contact Us' button.

3. Download Mobile Quality Assurance plug-in for Apache Cordova SDK. This will open a second page with the actual download link.

Service Details - IBM > IBM Knowledge Center

www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/download_sdks.html

IBM Knowledge Center

Search Content Products

IBM Mobile Quality Assurance for Bluemix > IBM Mobile Quality Assurance for Bluemix 6.0.0 > Preparing apps for reporting > Downloading the libraries

Downloading the libraries Version 6.0.0

After you register your app, you must download and install the library for the platform that your app runs on.

About this task

Download the library for one of the following platforms:

[Android SDK for download](#)
After you establish an account with IBM® Mobile Quality Assurance for Bluemix®, you can download and install the Android SDK. The Android SDK is available in two formats, one for the new official Android Studio IDE, and one for the older Eclipse Android Developer Tools IDE.

[iOS SDK for download](#)
After you establish an account with IBM Mobile Quality Assurance for Bluemix, you can download and install the iOS SDK.

[Downloads for JavaScript SDK for IBM MobileFirst Platform Foundation](#)
After you establish an account with Mobile Quality Assurance, you can download and install the JavaScript SDK component for IBM MobileFirst™ Platform Foundation and the other required downloads.

[Mobile Quality Assurance plug-in for Apache Cordova for download](#)
After you establish an account with Mobile Quality Assurance, download and extract the Mobile Quality Assurance plug-in.

Contact Privacy Terms of use Accessibility Feedback English Contact Us

4. Click the Mobile Quality Assurance plug-in for Apache Cordova link to begin the download.

Service Details - IBM > IBM Knowledge Center

www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/c_MQAPluginForCordova.html

IBM Knowledge Center

Search Content Products

IBM Mobile Quality Assurance for Bluemix > IBM Mobile Quality Assurance for Bluemix 6.0.0 > Preparing apps for reporting > Downloading the libraries > Mobile Quality Assurance plug-in for Apache Cordova for download

Mobile Quality Assurance plug-in for Apache Cordova for download Version 6.0.0

After you establish an account with Mobile Quality Assurance, download and extract the Mobile Quality Assurance plug-in.

Plug-in	Purpose	Details
Mobile Quality Assurance plug-in for Apache Cordova	The plug-in is installed into your Apache Cordova app and provides 2 modes: preproduction and production. Use the preproduction mode for internal testing during development. Use production mode to gather user information after an app is released.	This plug-in is supported on the following versions of Apache Cordova: Android 4.0, or later iOS 3.7 Important: The latest version of the Cordova runtime for iOS that is supported by the Mobile Quality Assurance Cordova plug-in is 3.7. The minimum version of the Cordova runtime for iOS that is supported by IBM MobileFirst™ Platform Foundation is 4.1. Consider these limitations when you run Mobile Quality Assurance and IBM MobileFirst Platform Foundation.

Requirements

Mobile Quality Assurance works with projects that are developed with the supported versions of the Apache Cordova plug-in. You must have the Cordova command-line interface and all of its dependencies that are installed to have access to the Cordova commands that are required to use Mobile Quality Assurance.

If you are using the MobileFirst Platform commands with Cordova, then you must install the IBM MobileFirst Platform Foundation command-line interface SDK.

Contact Us

Notice the Download progress. The file is around 130Mb

Service Details - IBM > IBM Knowledge Center > www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/c_MQAPluginForCordova.html

IBM Knowledge Center

Search Content Products

IBM Mobile Quality Assurance for Bluemix > IBM Mobile Quality Assurance for Bluemix 6.0.0 > Preparing apps for reporting > Downloading the libraries > Mobile Quality Assurance plug-in for Apache Cordova for download

Mobile Quality Assurance plug-in for Apache Cordova for download

After you establish an account with Mobile Quality Assurance, download and extract the Mobile Quality Assurance plug-in.

Plug-in	Purpose	Details
Mobile Quality Assurance plug-in for Apache Cordova	The plug-in is installed into your Apache Cordova app and provides 2 modes: preproduction and production. Use the preproduction mode for internal testing during development. Use production mode to gather user information after an app is released.	<p>This plug-in is supported on the following versions of Apache Cordova:</p> <p>Android 4.0, or later</p> <p>iOS 3.7</p> <p>Important: The latest version of the Cordova runtime for iOS that is supported by the Mobile Quality Assurance Cordova plug-in is 3.7. The minimum version of the Cordova runtime for iOS that is supported by IBM MobileFirst™ Platform Foundation is 4.1. Consider these limitations when you run Mobile Quality Assurance and IBM MobileFirst Platform Foundation.</p>

Requirements

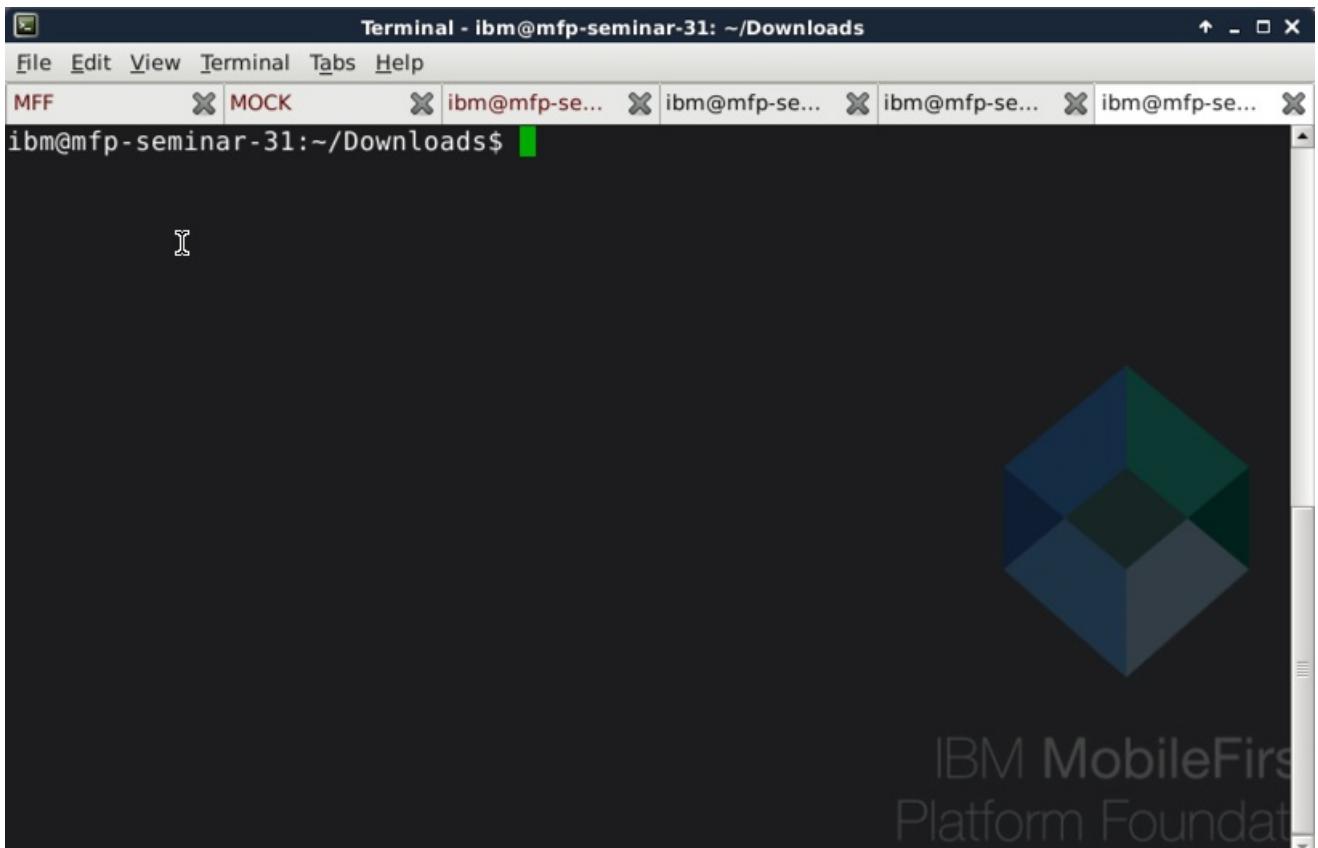
Mobile Quality Assurance works with projects that are developed with the supported versions of the Apache Cordova plug-in. You must have the Cordova command-line interface installed to have access to the Cordova commands that are required to use Mobile Quality Assurance.

[CordovaPlugi....zip](#) 50.4/130 MB, 27 secs left

Contact Us SHOW ALL

- When the download is complete, open a terminal session and navigate to the **Downloads** directory.

```
cd ~/Downloads
```



6. Make a directory to unzip the file that was just downloaded.

```
mkdir mqa
```

A screenshot of a Linux terminal window titled "Terminal - ibm@mfp-seminar-31: ~/Downloads". The window has multiple tabs at the top, one of which is labeled "MOCK". The main pane shows the command "mkdir mqa" being typed at the prompt "ibm@mfp-seminar-31:~/Downloads\$". The background of the terminal window features the IBM MobileFirst Platform Foundation logo, which consists of a stylized blue and green cube.

7. Unzip the file that was downloaded (CordovaPlugin-3.0.18.zip) to the mqa directory.

```
unzip CordovaPlugin-3.0.18.zip -d mqa/
```

```
Terminal - ibm@mfp-seminar-31: ~/Downloads
File Edit View Terminal Tabs Help
MFF      X MOCK      X ibm@mfp-se... X ibm@mfp-se... X ibm@mfp-se... X ibm@mfp-se...
ibm@mfp-seminar-31:~/Downloads$ mkdir mqa
ibm@mfp-seminar-31:~/Downloads$ unzip CordovaPlugin-3.0.18.zip -d mqa/
```



IBM MobileFirst
Platform Foundation

8. Change directory back to the advancedMessenger directory

```
cd ~/dev/workspaces/am/advancedMessenger
```

```
Terminal - ibm@mfp-seminar-31: ~/Downloads
File Edit View Terminal Tabs Help
MFF      X MOCK      X ibm@mfp-se... X ibm@mfp-se... X ibm@mfp-se... X ibm@mfp-se...
ibm@mfp-seminar-31:~/Downloads$ cd ~/dev/workspaces/am/advancedMessenger/
```



IBM MobileFirst
Platform Foundation

9. Add Cordova MQA Plugin.

```
cordova plugin add ~/Downloads/mqa/
```



IBM MobileFirst
Platform Foundation

```
Terminal - ibm@mfp-seminar-31: ~/dev/workspaces/am/advancedMessenger
File Edit View Terminal Tabs Help
MFF      MOCK      ibm@mfp-se... ibm@mfp-se... ibm@mfp-se... ibm@mfp-se...
ibm@mfp-seminar-31:~/dev/workspaces/am/advancedMessenger$ cordova plugin add ~/D
ownloads/mqa/
```

Section 5 - Instrument Application

1. Open your **app.ts** file using Visual Studio Code, or IDE of your choice.

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** app.ts - am - Visual Studio Code
- File Menu:** File Edit View Goto Help
- Explorer View:** Shows the project structure under "OPEN EDITORS". The "app.ts" file is selected.
- Code Editor:** Displays the content of the "app.ts" file. The code is a TypeScript file for an Ionic application, defining the "MyApp" component and its constructor logic, including MFP API initialization and StatusBar styling.
- Status Bar:** Shows "Ln 24, Col 30" and other file metadata.

Note: For more information on instrumenting your application with the MQA Plug-in for Apache Cordova, visit [here](#).

2. Make a declaration for MQA. Under the import statements in the app.ts file enter the following:

```
declare var MQA: any;
```

```
app.ts
1 import {Component, Renderer} from '@angular/core';
2 import {Platform, Alert, App, ionicBootstrap} from 'ionic-angular';
3 import {StatusBar} from 'ionic-native';
4 import {TabsPage} from './pages/tabs/tabs';
5 import {PushProvider} from './providers/push-provider/push-provider';
6
7 declare var MQA: any;
8
9 @Component({
10   template: '<ion-nav [root]="rootPage"></ion-nav>',
11   providers: [PushProvider]
12 })
13 export class MyApp {
14   private rootPage:any;
15   private AuthHandler:any;
16   private nav:any;
17
18   constructor(private platform:Platform, renderer:Renderer, private app: App, private push:PushProvider) {
19     console.log('constructor done');
20
21     renderer.listenGlobal('document','mfpjsloaded', () => {
22       console.log("--> MFP API init complete");
23
24       this.MFPInitComplete();
25     })
26
27     platform.ready().then(() => {
28       StatusBar.styleDefault();
29     });
30
31   }
32
33   ngAfterViewInit(){
34     console.log('--> ngAfterViewInit fired');
35 }
```

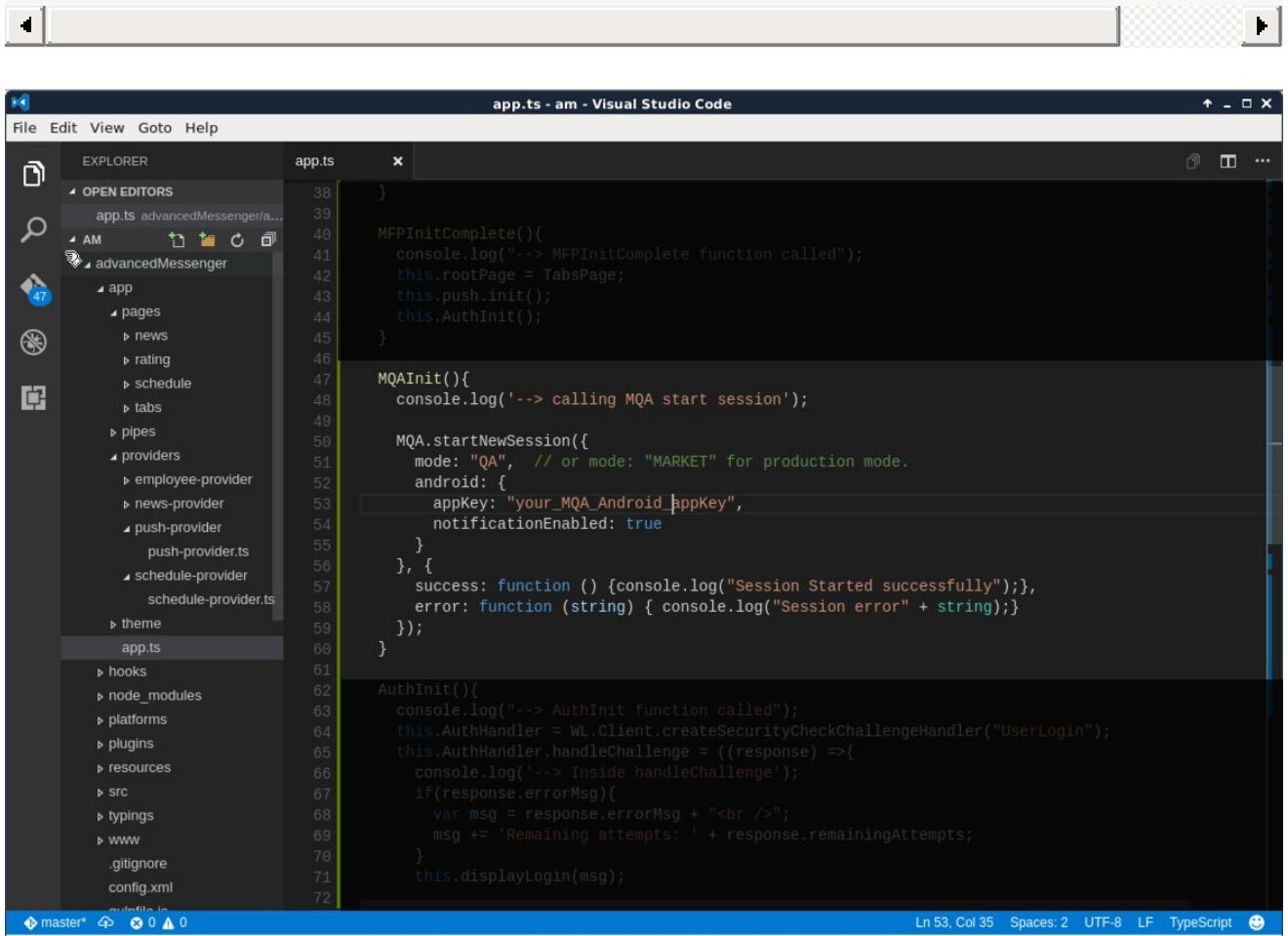
This will prevent the compiler from throwing errors.

3. Next, after the MFPInitComplete() function, create a new function called MQAInit().

```
MQAInit(){
  console.log('--> calling MQA start session');

  MQA.startNewSession(
    {
      mode: "QA", // or mode: "MARKET" for production
      android: {
        appKey: "your_MQA_Android_appKey" ,
        notificationsEnabled: true
      }
    },
    {
      success: function () {
        console.log("Session Started successfully")
      },
      error: function (string) {
        console.log("Session error" + string);
      }
    }
  )
}
```

```
    }
  );
}
```



The screenshot shows the Visual Studio Code interface with the title bar "app.ts - am - Visual Studio Code". The left sidebar is titled "EXPLORER" and shows a tree view of the project structure. The "OPEN EDITORS" section contains "app.ts". The main editor area displays the following TypeScript code:

```
38  }
39
40 MFPInitComplete(){
41   console.log("--> MFPInitComplete function called");
42   this.rootPage = TabsPage;
43   this.push.init();
44   this.AuthInit();
45 }
46
47 MQAInit(){
48   console.log('--> calling MQA start session');
49
50   MQA.startNewSession({
51     mode: "QA", // or mode: "MARKET" for production mode.
52     android: {
53       appKey: "your_MQA_Android_appKey",
54       notificationEnabled: true
55     }
56   },
57   success: function () {console.log("Session Started successfully");},
58   error: function (string) { console.log("Session error" + string);}
59 );
60
61 AuthInit(){
62   console.log("--> AuthInit function called");
63   this.AuthHandler = WL.Client.createSecurityCheckChallengeHandler("UserLogin");
64   this.AuthHandler.handleChallenge = ((response) =>{
65     console.log('--> Inside handleChallenge');
66     if(response.errorMsg){
67       var msg = response.errorMsg + "<br />";
68       msg += 'Remaining attempts: ' + response.remainingAttempts;
69     }
70     this.displayLogin(msg);
71
72 }
```

The status bar at the bottom indicates "Ln 53, Col 35 Spaces: 2 UTF-8 LF TypeScript".

4. Next you will need to replace the "your_MQA_Android_appKey" with the app key found in the MQA Service. To do this, first go to your MQA Service you created earlier. Then **Click the Show App Key button.**

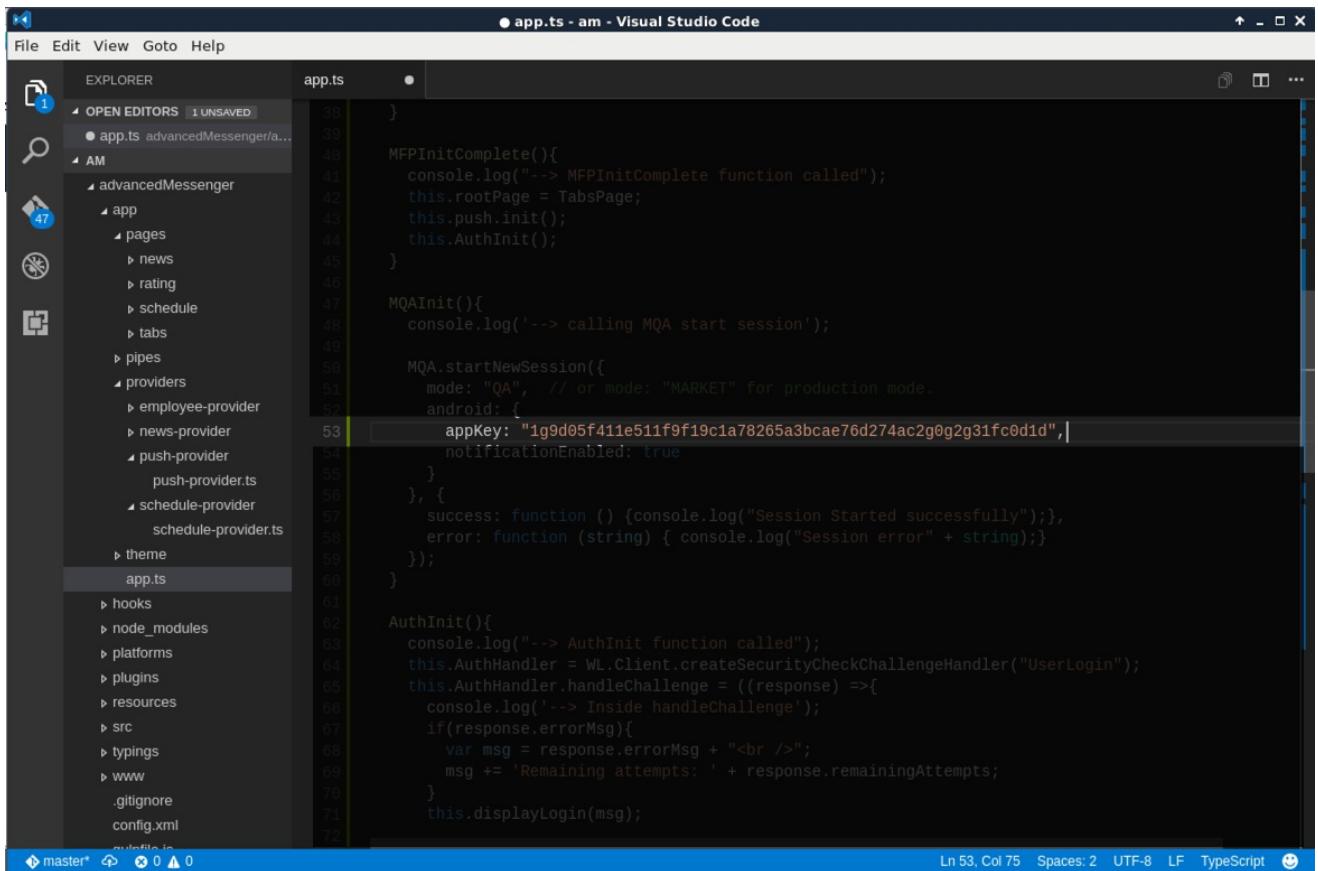
The screenshot shows the 'advancedMessenger' service details page in the IBM Bluemix console. The page has a header with the service name and a navigation bar with links like Catalog, Docs, and 284. Below the header is a summary card for the 'advancedMessenger' app. The card displays three sections: 'Preproduction (Last 7 days)' with 0 sessions, 'Production (Last 7 days)' with 0 sessions, and 'Registered Devices (Oct 1, 2016 to Oct 3, 2016)' with 0 devices. Below these sections is a 'Sentiment Score' section with a note that 'Sentiment Analysis is not configured.' and a link to 'Configure Sentiment Analysis.' At the bottom of the card are four buttons: 'Upload Builds', 'Show App Key' (which is circled in red), 'Distribute to Test', and 'Configure Integrations'. A red arrow points from the left towards the 'Show App Key' button.

- With the App Key showing, select and copy the key to the clipboard.

The screenshot shows the same 'advancedMessenger' service details page in the IBM Bluemix console. The 'Show App Key' button has been clicked, and the resulting app key value ('1g9d05f411e511f9f19c1a78265a3bcae76d274ac2g0g2g31fc0d1d') is displayed in a white box with a red border. A red arrow points from the left towards this box. Below the box are the original four buttons: 'Upload Builds', 'Hide App Key' (which is now visible), 'Distribute to Test', and 'Configure Integrations'.

- Replace the "your_MQA_Android_appKey" with the App Key that you copied to the clipboard.

Note: Make sure you use your app key, or you will not be able to view any bugs or feedback later.



```
File Edit View Goto Help
OPEN EDITORS app.ts
AM
advancedMessenger
app
  pages
    news
    rating
    schedule
    tabs
  pipes
  providers
    employee-provider
    news-provider
  push-provider
    push-provider.ts
  schedule-provider
    schedule-provider.ts
theme
app.ts
hooks
node_modules
platforms
plugins
resources
src
 typings
www
.gitignore
config.xml
Ln 53, Col 75 Spaces: 2 UTF-8 LF TypeScript
master* 0 0 ▲ 0
```

```
        }
      }

      MFPIInitComplete(){
        console.log("--> MFPIInitComplete function called");
        this.rootPage = TabsPage;
        this.push.init();
        this.AuthInit();
      }

      MQAInit(){
        console.log('--> calling Mqa start session');

        Mqa.startNewSession({
          mode: "QA", // or mode: "MARKET" for production mode.
          android: {
            apiKey: "1g9d05f411e511f9f19c1a78265a3bcae76d274ac2g0g2g31fc0d1d",
            notificationEnabled: true
          }
        }, {
          success: function () {console.log("Session Started successfully");},
          error: function (string) { console.log("Session error" + string);}
        });
      }

      AuthInit(){
        console.log("--> AuthInit function called");
        this.AuthHandler = WL.Client.createSecurityCheckChallengeHandler("UserLogin");
        this.AuthHandler.handleChallenge = ((response) =>{
          console.log('--> Inside handleChallenge');
          if(response.errorMsg){
            var msg = response.errorMsg + "<br />";
            msg += 'Remaining attempts: ' + response.remainingAttempts;
          }
          this.displayLogin(msg);
        });
      }
    }
  }

  ngAfterViewInit(){
    console.log('--> ngAfterViewInit fired');

    this.nav = this.app.getActiveNav();
  }

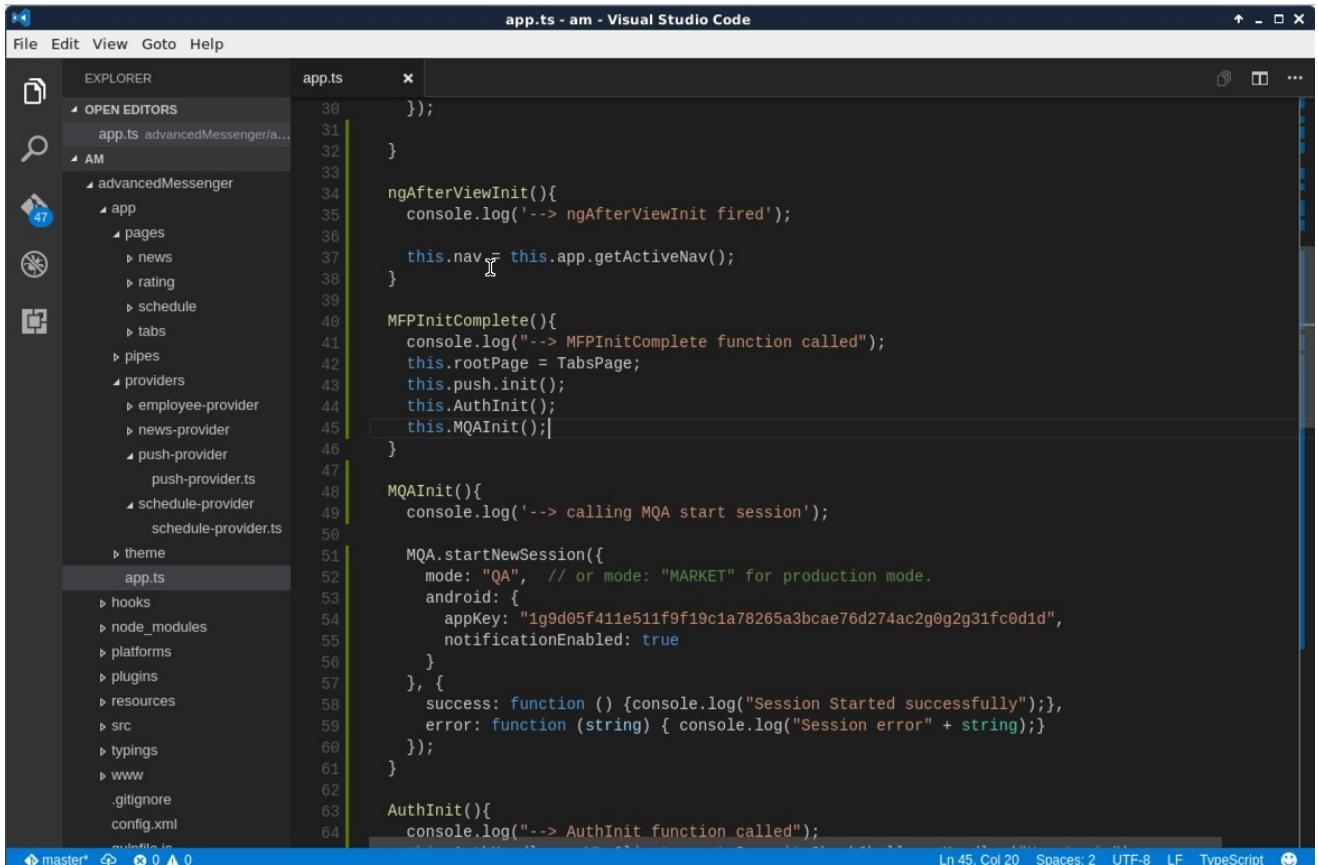
  MFPIInitComplete(){
    console.log("--> MFPIInitComplete function called");
    this.rootPage = TabsPage;
    this.push.init();
    this.AuthInit();
    this.MQAInit();
  }

  MQAInit(){
    console.log('--> calling Mqa start session');

    Mqa.startNewSession({
      mode: "QA", // or mode: "MARKET" for production mode.
      android: {
        apiKey: "1g9d05f411e511f9f19c1a78265a3bcae76d274ac2g0g2g31fc0d1d",
        notificationEnabled: true
      }
    }, {
      success: function () {console.log("Session Started successfully");},
      error: function (string) { console.log("Session error" + string);}
    });
  }

  AuthInit(){
    console.log("--> AuthInit function called");
  }
}
```

- Finally you will need to add the call to the MQAInit() function inside the MFPIInitComplete() function.



```
File Edit View Goto Help
OPEN EDITORS app.ts
AM
advancedMessenger
app
  pages
    news
    rating
    schedule
    tabs
  pipes
  providers
    employee-provider
    news-provider
  push-provider
    push-provider.ts
  schedule-provider
    schedule-provider.ts
theme
app.ts
hooks
node_modules
platforms
plugins
resources
src
 typings
www
.gitignore
config.xml
Ln 45, Col 20 Spaces: 2 UTF-8 LF TypeScript
master* 0 0 ▲ 0
```

```
        }
      }

      MFPIInitComplete(){
        console.log("--> MFPIInitComplete function called");
        this.rootPage = TabsPage;
        this.push.init();
        this.AuthInit();
        this.MQAInit();
      }

      MQAInit(){
        console.log('--> calling Mqa start session');

        Mqa.startNewSession({
          mode: "QA", // or mode: "MARKET" for production mode.
          android: {
            apiKey: "1g9d05f411e511f9f19c1a78265a3bcae76d274ac2g0g2g31fc0d1d",
            notificationEnabled: true
          }
        }, {
          success: function () {console.log("Session Started successfully");},
          error: function (string) { console.log("Session error" + string);}
        });
      }

      AuthInit(){
        console.log("--> AuthInit function called");
      }
    }
  }

  ngAfterViewInit(){
    console.log('--> ngAfterViewInit fired');

    this.nav = this.app.getActiveNav();
  }

  MFPIInitComplete(){
    console.log("--> MFPIInitComplete function called");
    this.rootPage = TabsPage;
    this.push.init();
    this.AuthInit();
    this.MQAInit();
  }

  MQAInit(){
    console.log('--> calling Mqa start session');

    Mqa.startNewSession({
      mode: "QA", // or mode: "MARKET" for production mode.
      android: {
        apiKey: "1g9d05f411e511f9f19c1a78265a3bcae76d274ac2g0g2g31fc0d1d",
        notificationEnabled: true
      }
    }, {
      success: function () {console.log("Session Started successfully");},
      error: function (string) { console.log("Session error" + string);}
    });
  }

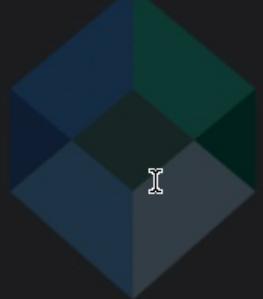
  AuthInit(){
    console.log("--> AuthInit function called");
  }
}
```

- Save your changes.

Section 6 - Build and Run App

1. Next, build the application by opening a terminal session, if not already open. Navigate to your `~/dev/workspaces/am/advancedMessenger` directory. Then type:

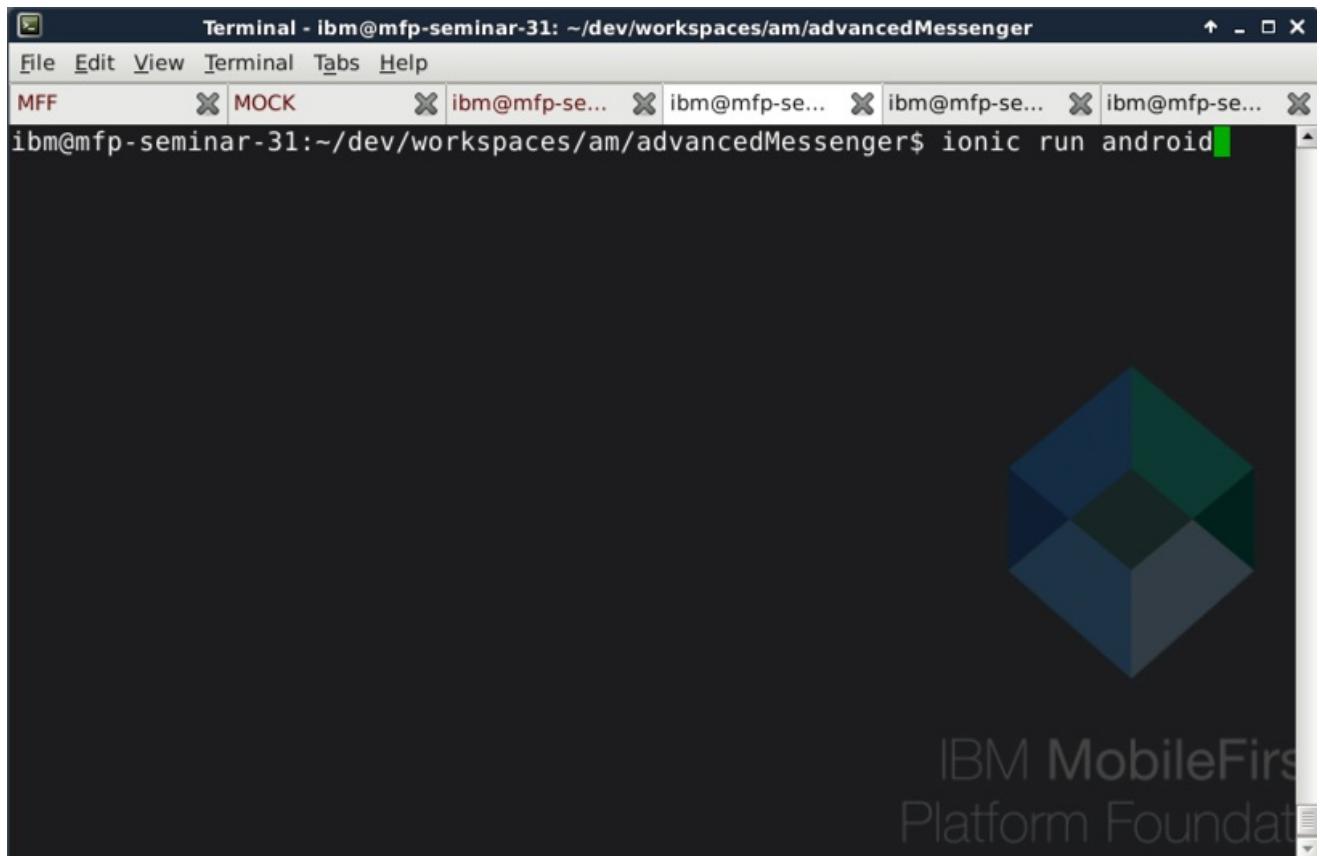
```
cordova prepare
```



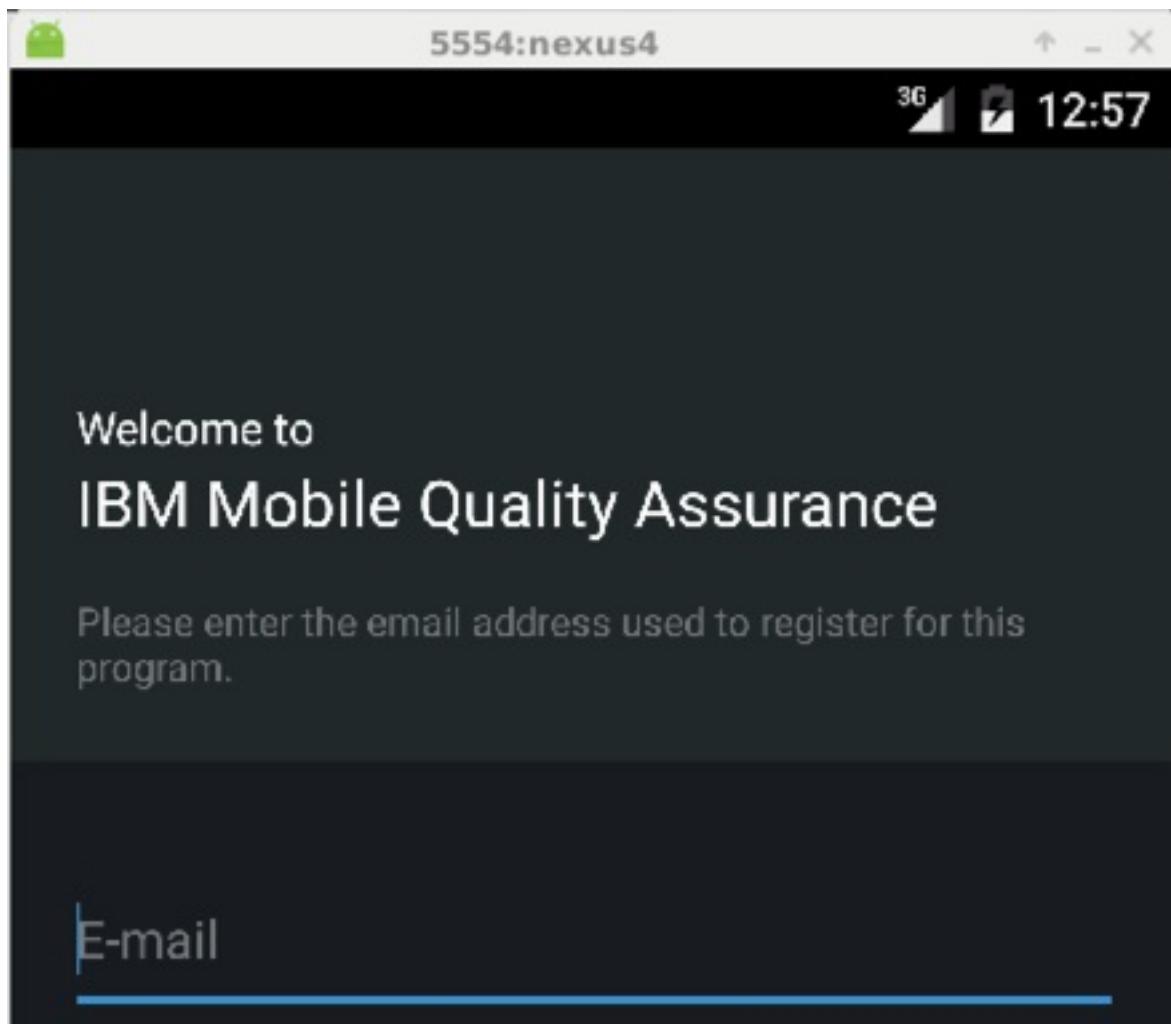
A screenshot of a terminal window titled "Terminal - ibm@mfp-seminar-31: ~/dev/workspaces/am/advancedMessenger". The window shows several tabs at the top, one of which is labeled "MOCK". The main pane of the terminal displays the command "ibm@mfp-seminar-31:~/dev/workspaces/am/advancedMessenger\$ cordova prepare" followed by a green prompt character. In the bottom right corner of the terminal window, there is a watermark for "IBM MobileFirst Platform Foundation".

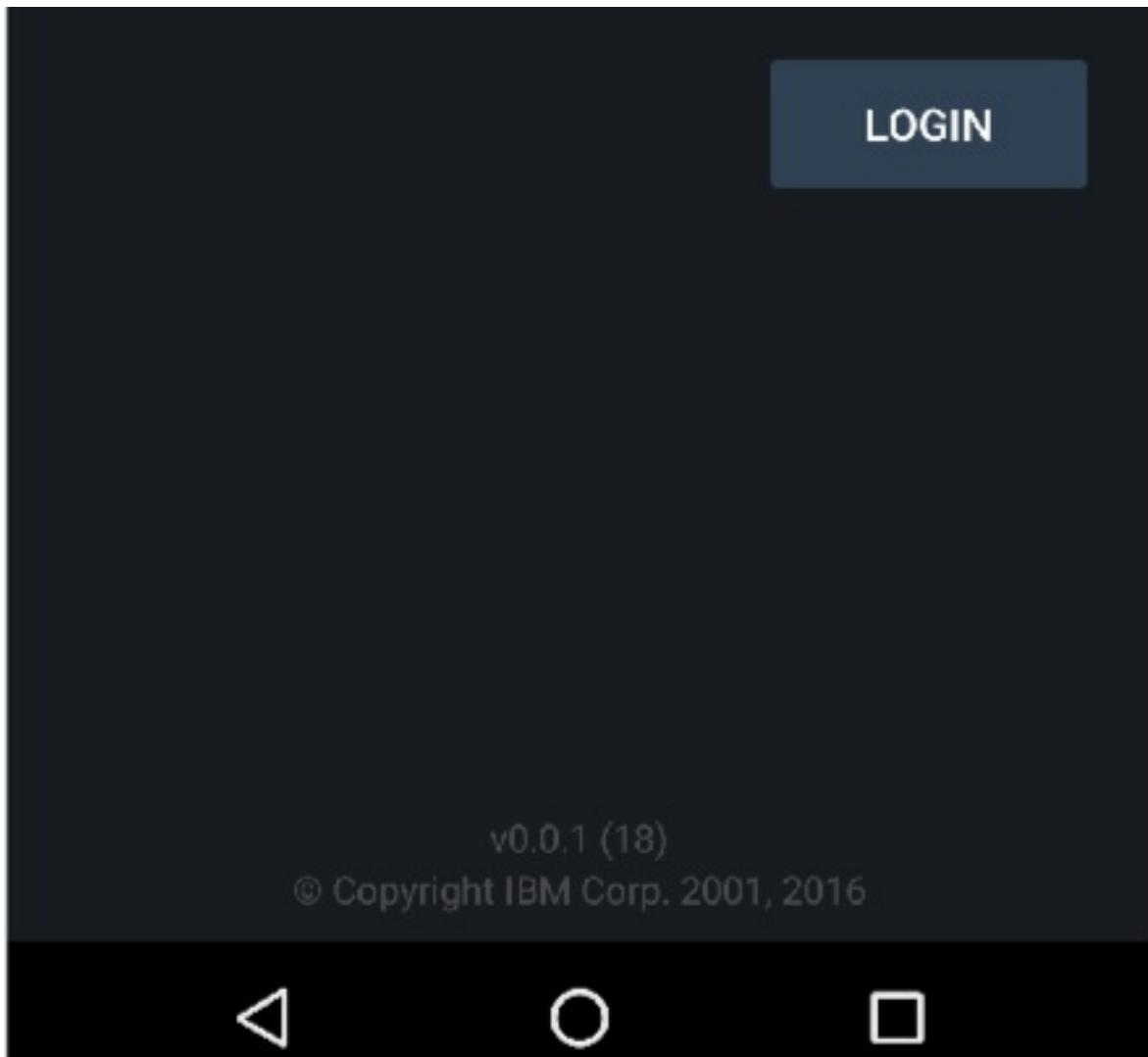
2. Next, run the application using ionic. This will push the latest application to the mobile device emulator.

```
ionic run android
```

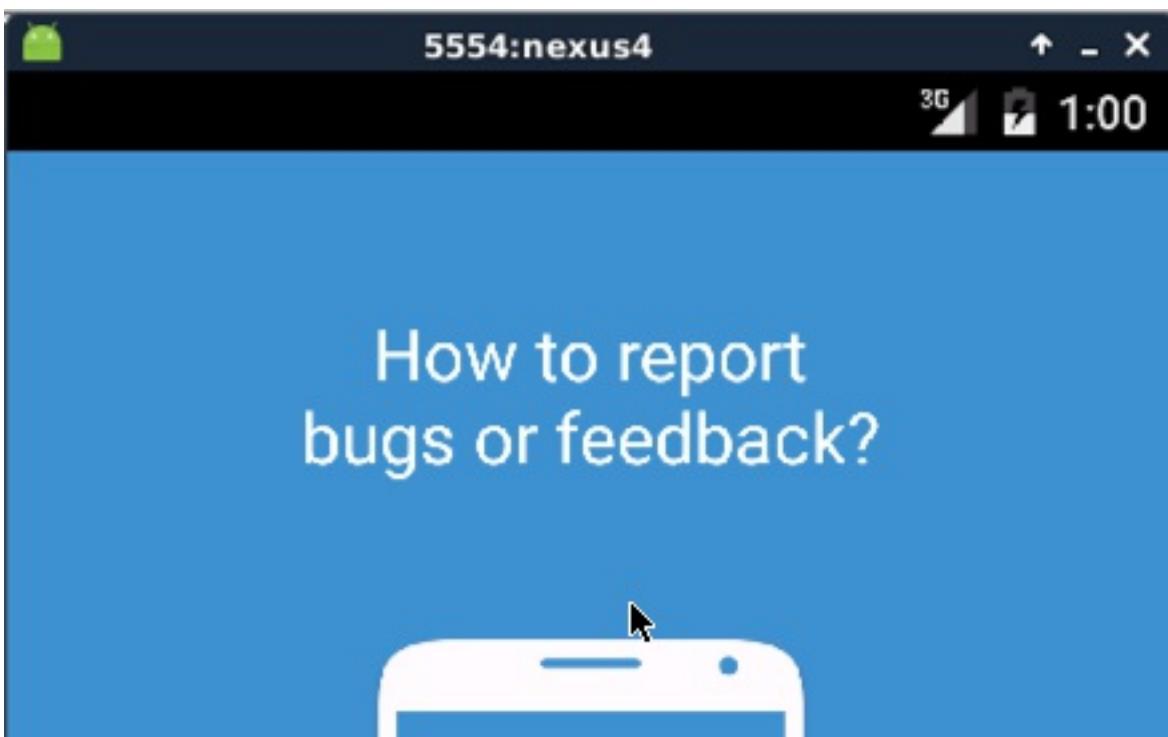


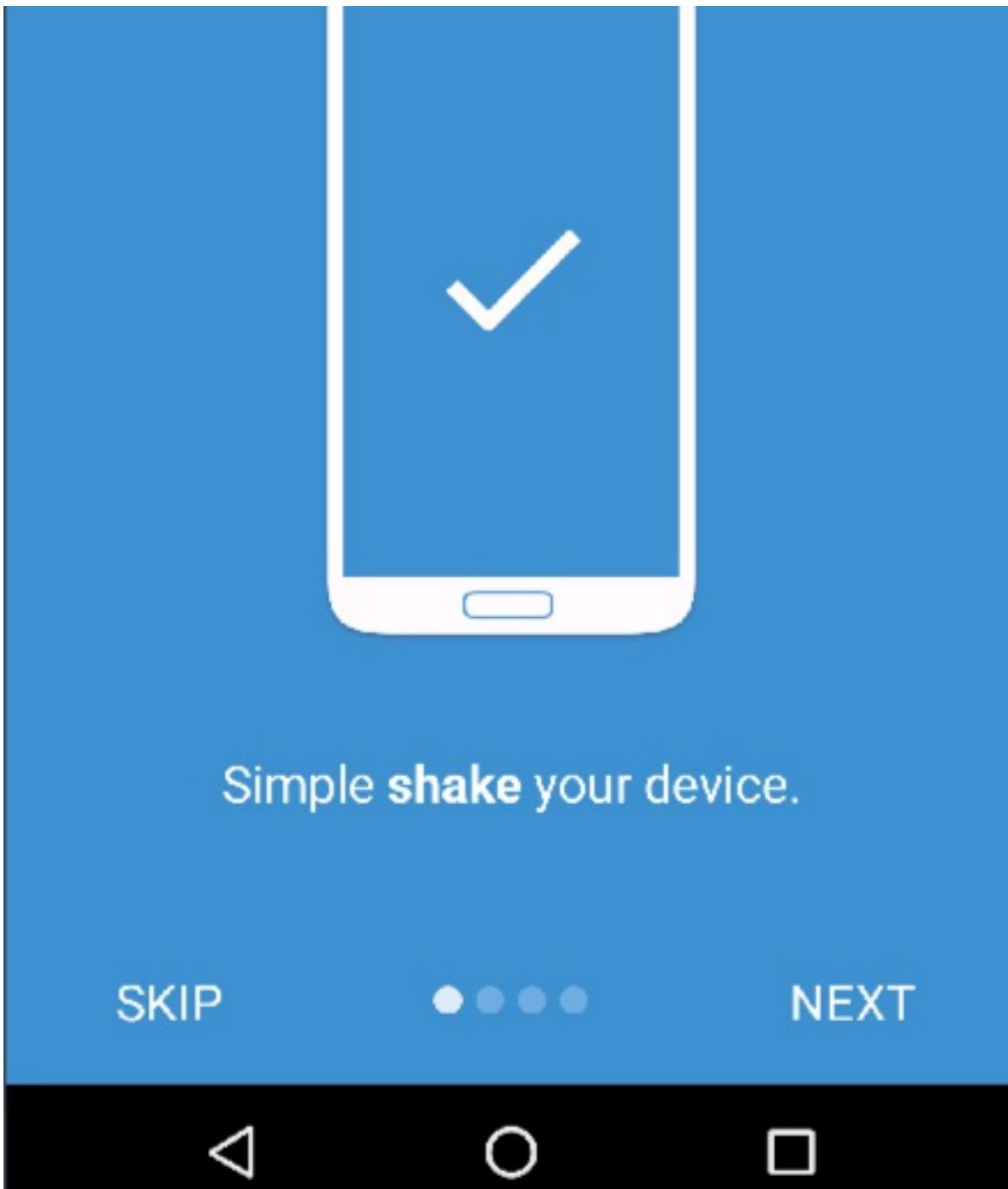
3. The first time the application runs, it will require you to provide an email address so that the application can be registered.





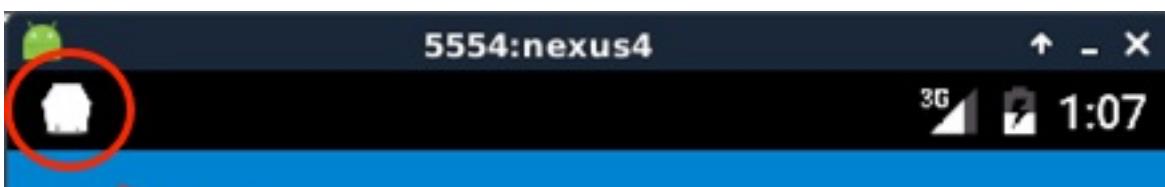
4. Once you have provided an email address and MQA has registered the user, a wizard screen will open to guide you through the process of using MQA.





Section 7 - Report a Bug

1. Device shaking is not supported in the Android emulator. There is however a button at the top of the screen that can be used to report a bug or provide feedback. Press and hold the icon in the upper left hand corner of the device.



Last news



SCHEDULE



NEWS



RATING

Jun 18, 2016

We are looking forward to use MFP v8.0!

Jun 17, 2016

ALPHA version contains some bugs, please be aware

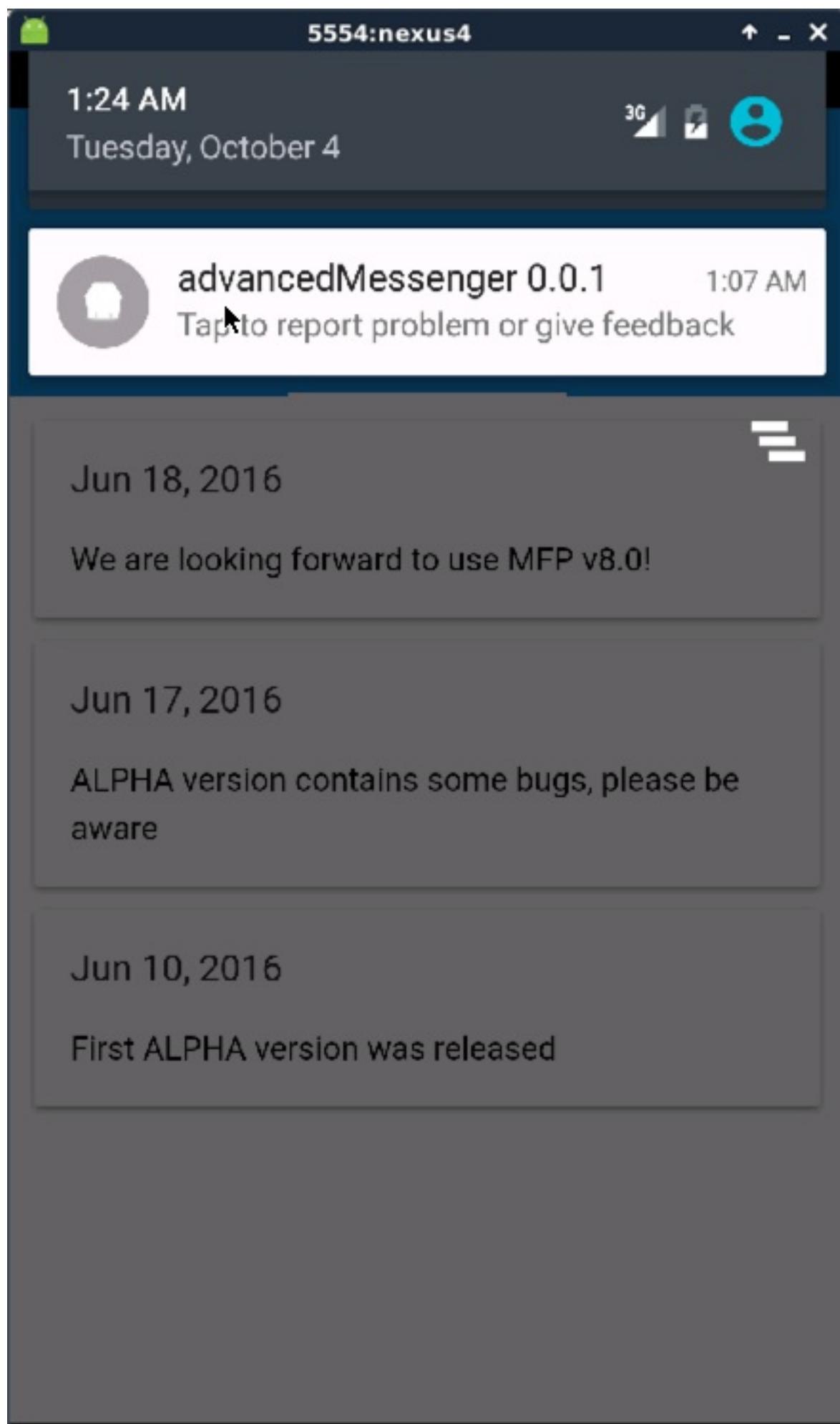
Jun 10, 2016

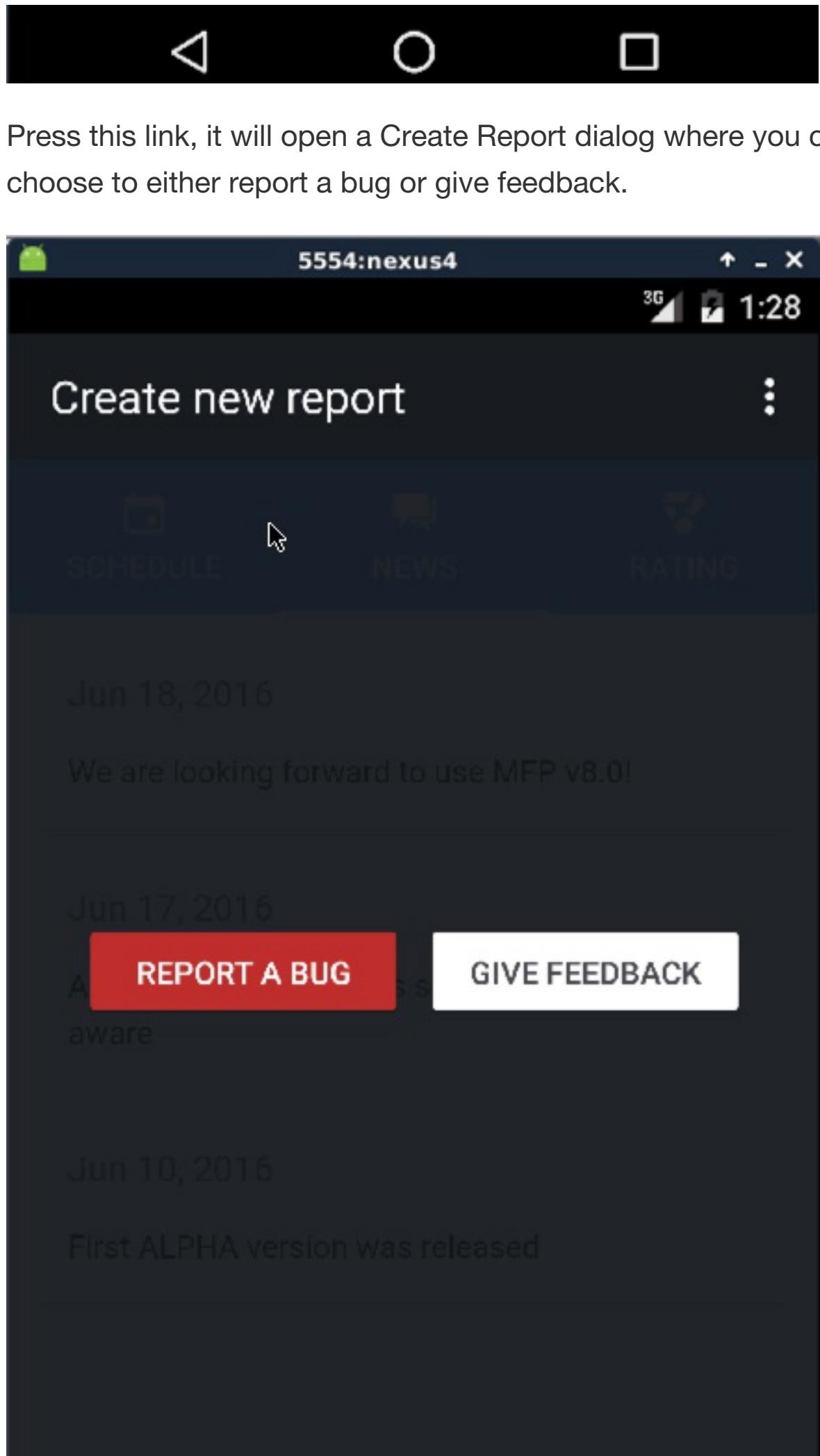
First ALPHA version was released



This will open a drop-down with a link for advancedMessenger

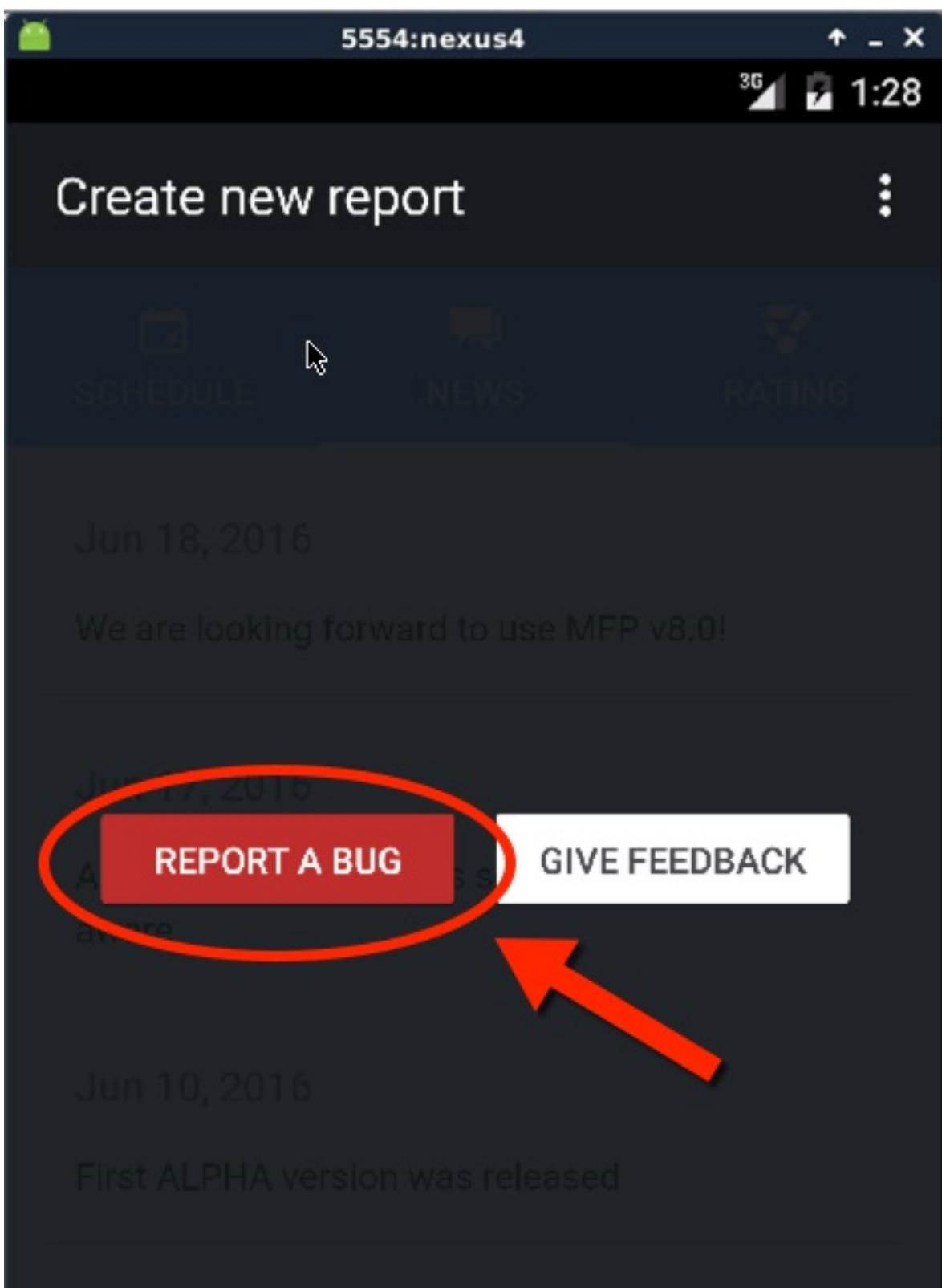
0.0.1, Tap to report problem or give feedback.





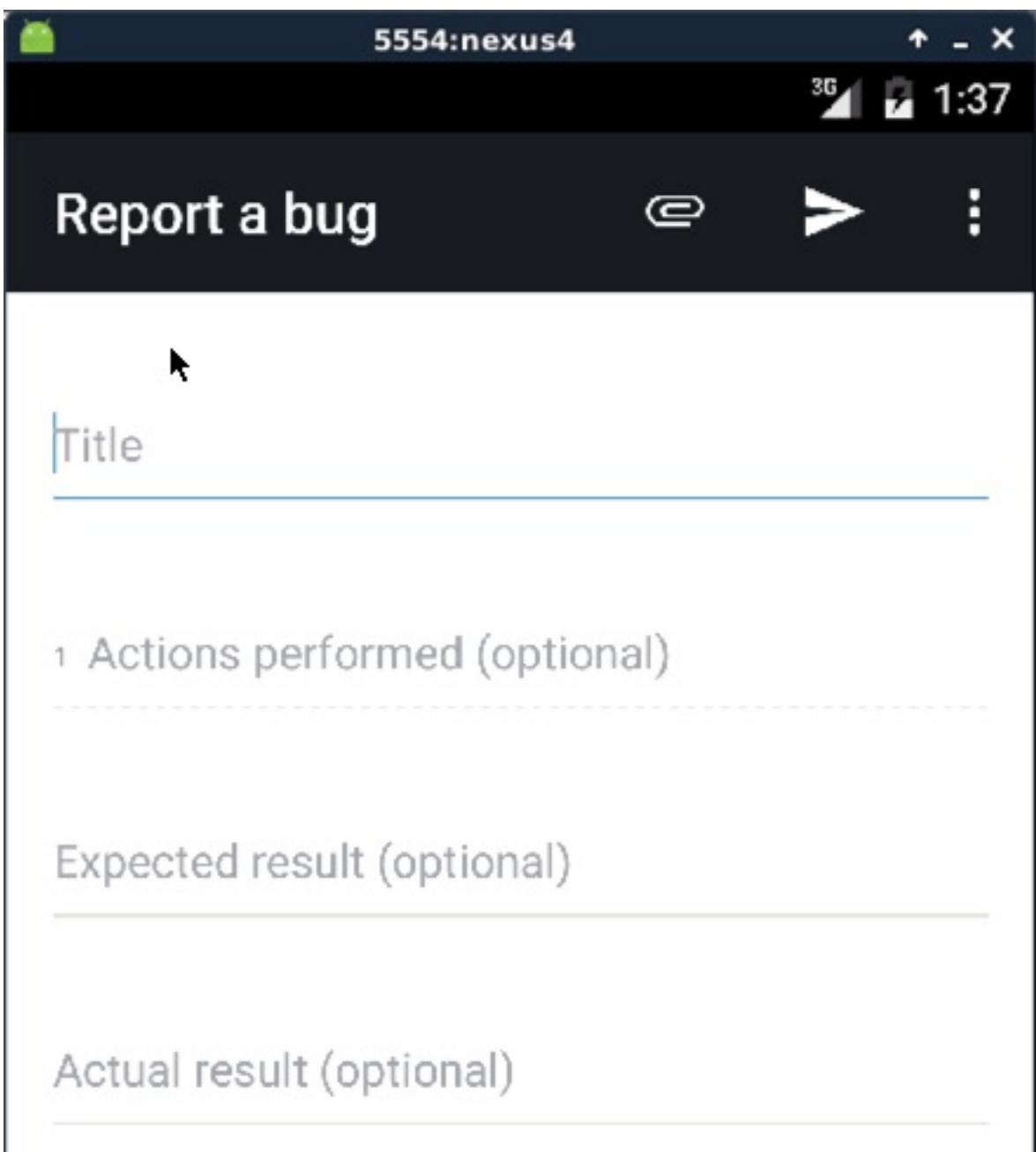


2. Press the **REPORT A BUG** button.

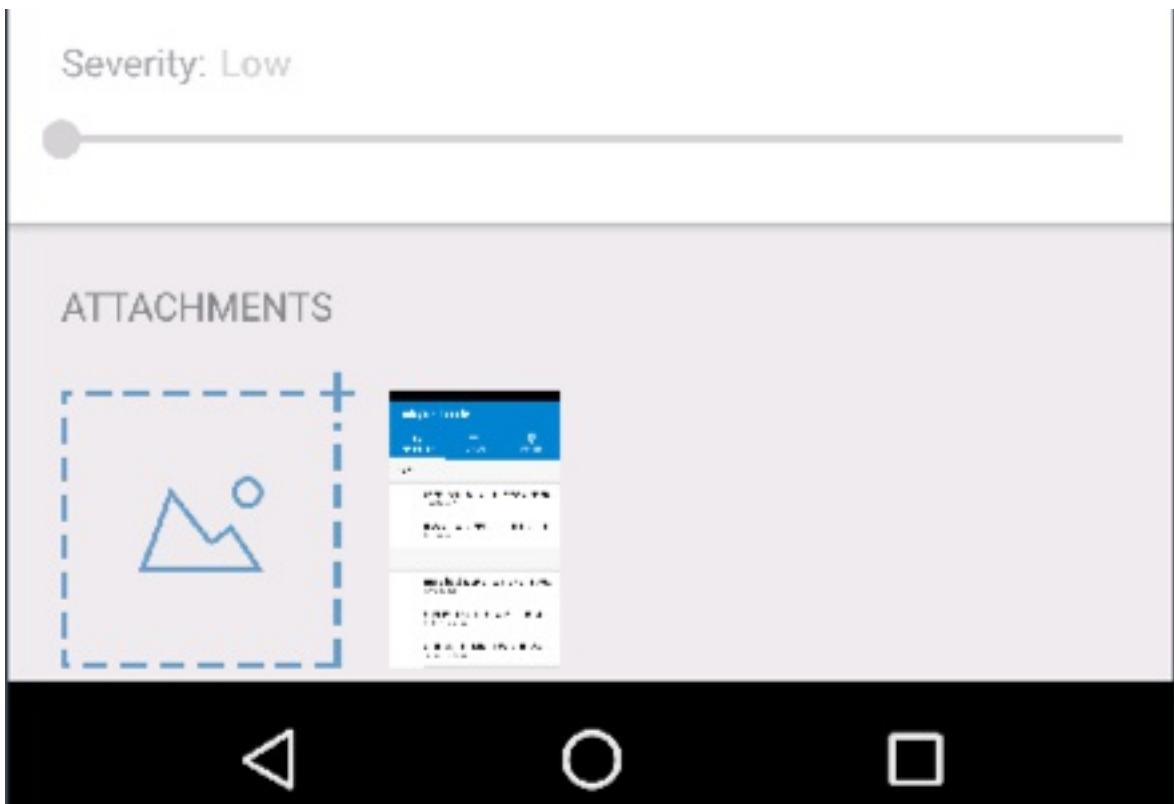




3. This will open the bug report form. Here you will need to enter a **Title** for the report. Optionally you can enter one or more **Actions Performed**, **Expected Results**, **Actual Results**, and **Severity**.



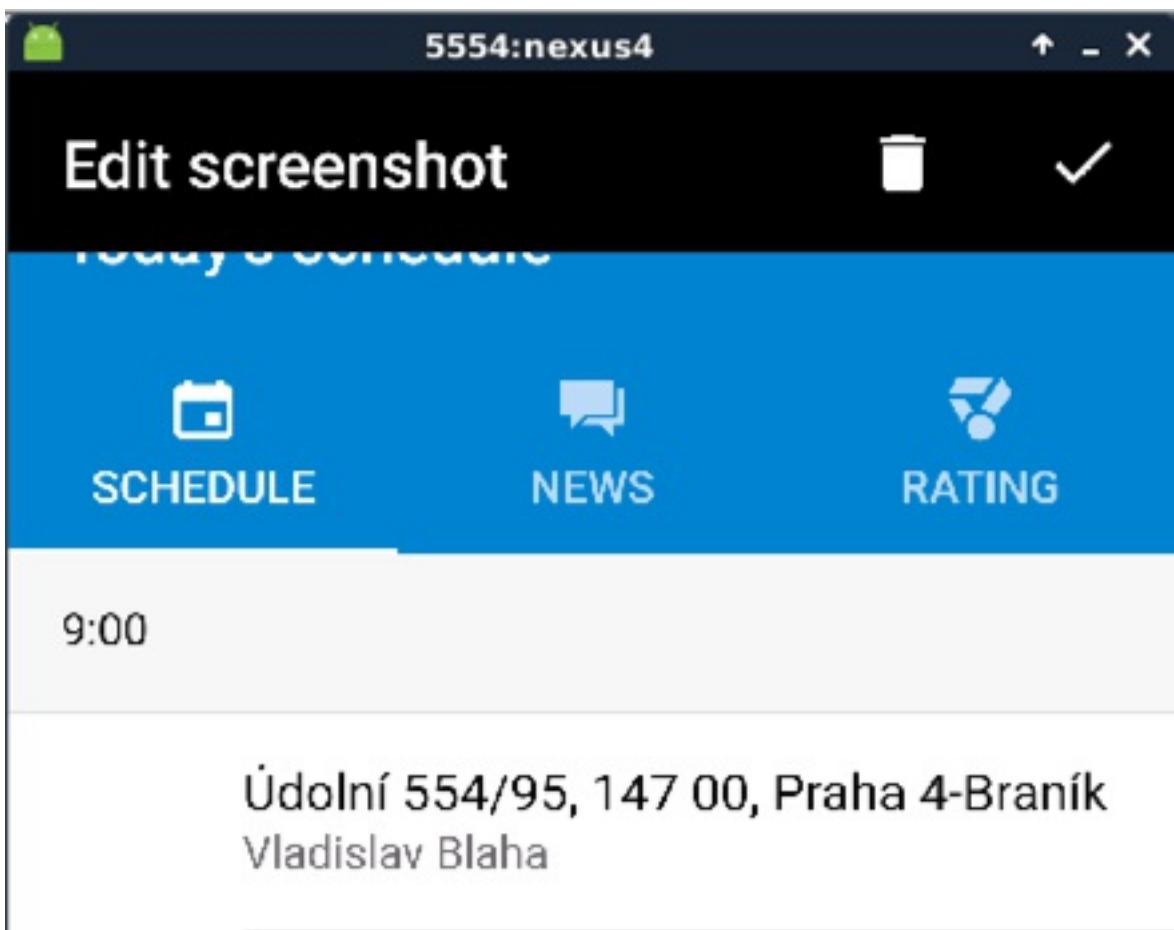
Severity: Low



You can annotate the default image by double clicking it.



With the image open you can draw on the screen and blur fields.



Na Vrstvách 259/37, 140 00, Praha 4...

Ivo Máta

Reset changes



10:00

Blur



Na Farkáně IV 250/60, 150 00, Praha...

René Kuba

Brush



nám. Před Bateriemi 674/20, 162 00,...

Běta Ciháková

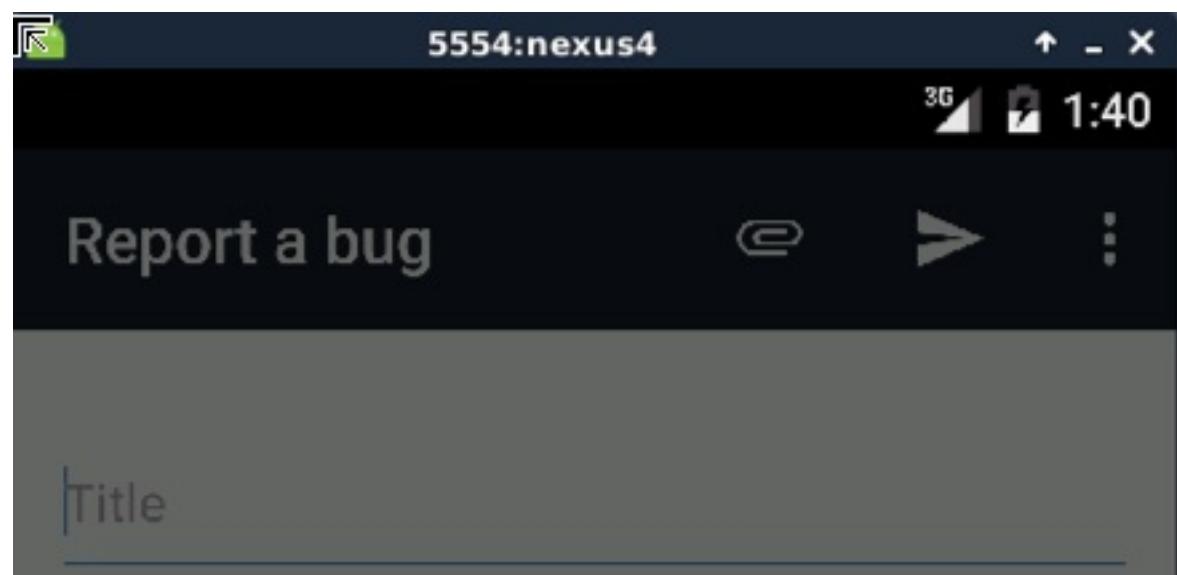


Charlese de Gaulla 465/20, 162 00, ...

Jiřina Hádková



You can add additional attachments such as additional images or even a video by either pressing the MQA menu item that looks like a paperclip or by pressing the New Image button at the bottom of the screen.



Add attachment

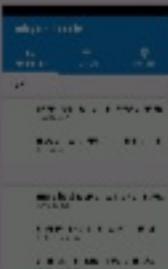
TAKE SCREENSHOT

START IN-APP VIDEO CAPTURE

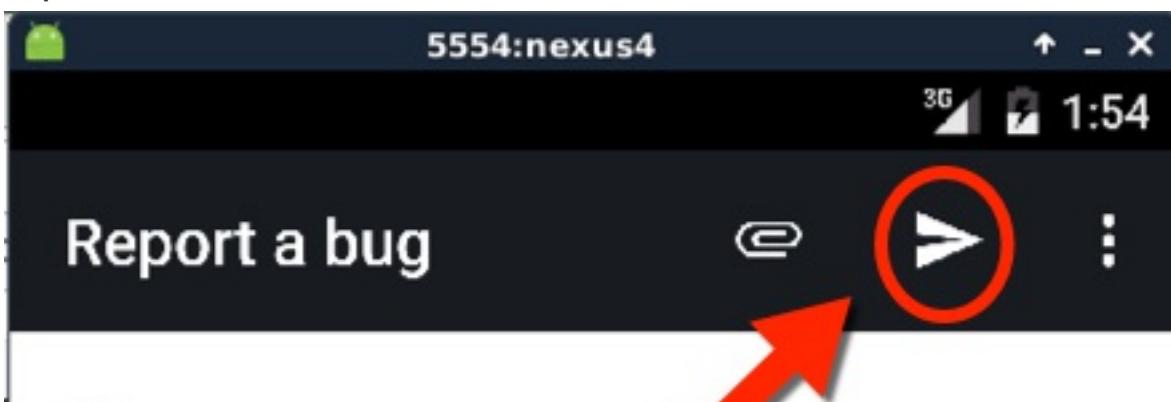
Videos can be large. Recommended to use WiFi.

CANCEL

ATTACHMENTS



4. To complete the bug report, use the MQA Menu to Upload the report.



Title

Test Bug

Actions performed (optional)

1 Action 1

+

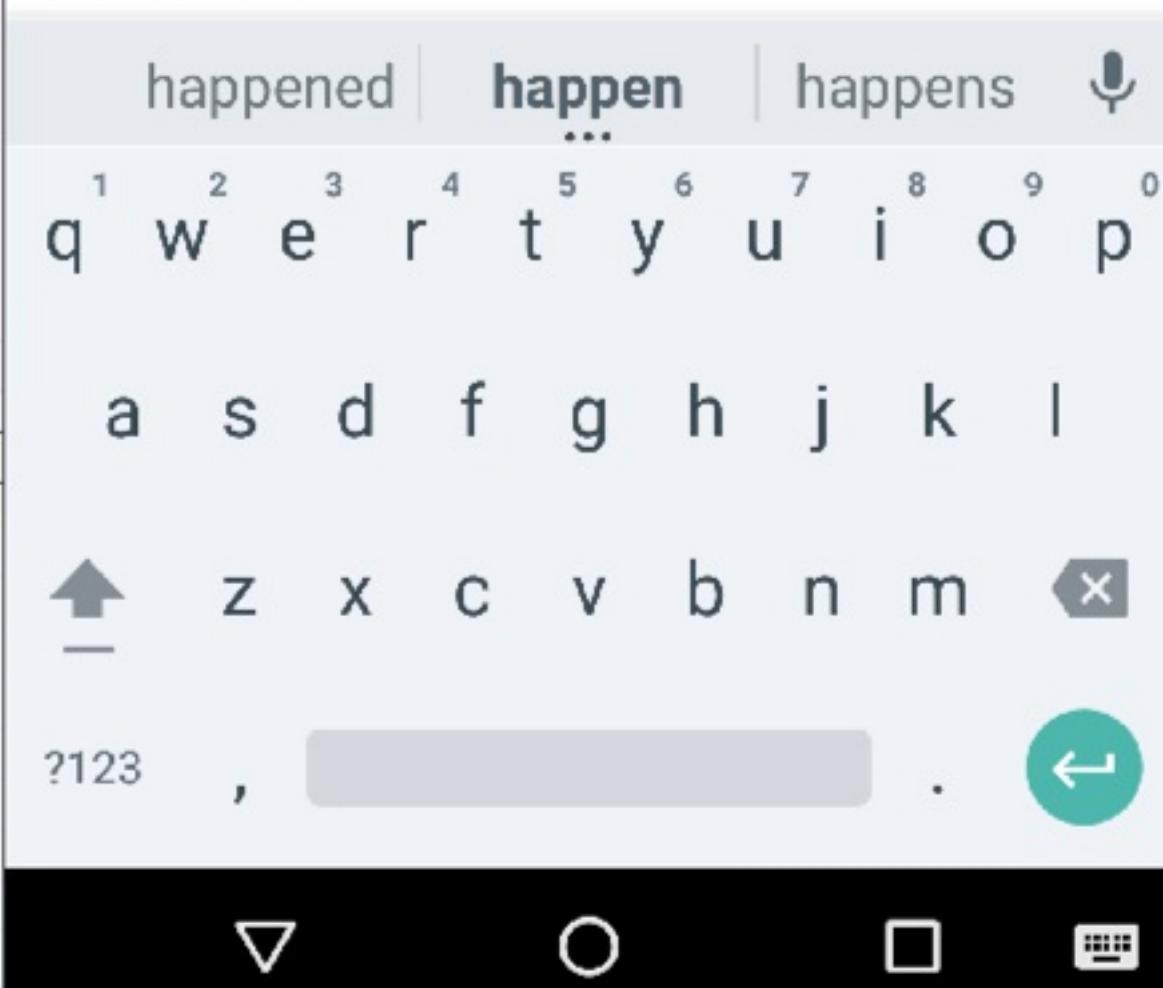
2 Action 2

+

3 Next Action

+

4 Final Action



Review the options available via the MQA menu by clicking the option icon. If the "Upload only via WiFi" checkbox is checked, then the report will only be upload if the device is using WiFi.



Report a bug



Title

Test Bug

Discard



Upload only via WiFi

Actions performed (optional)

1 Action 1



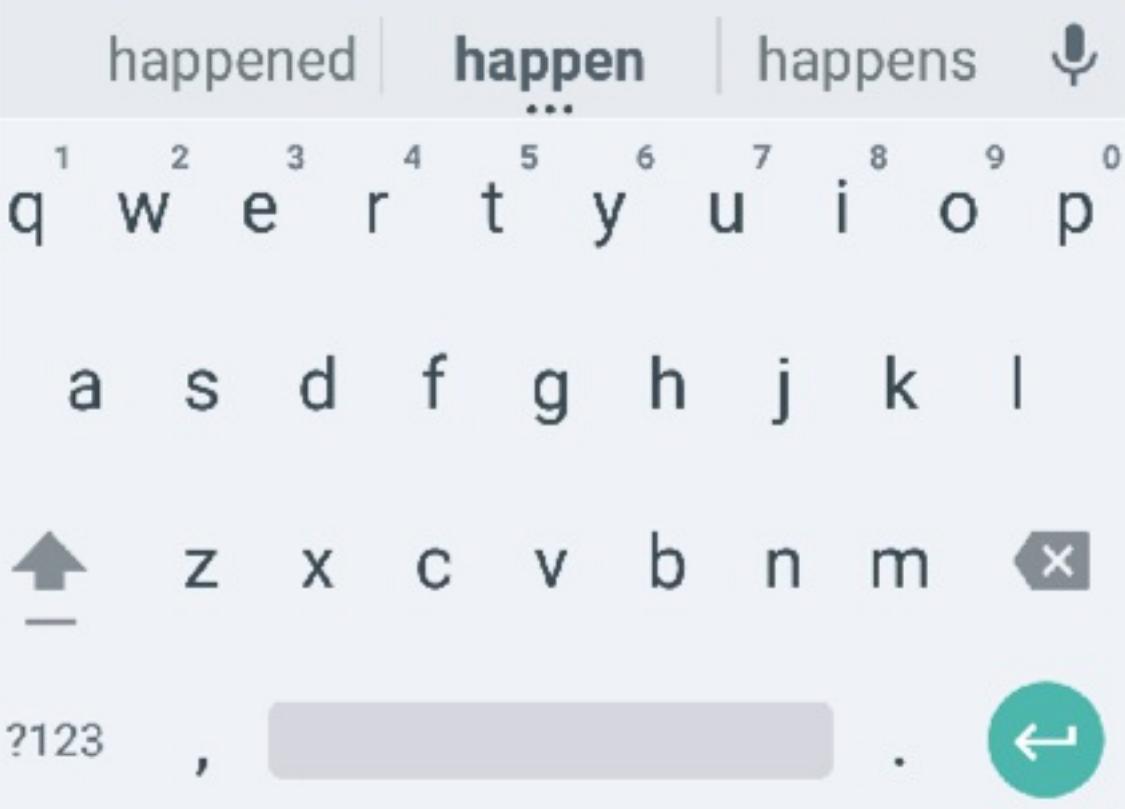
2 Action 2



3 Next Action



4 Final Action

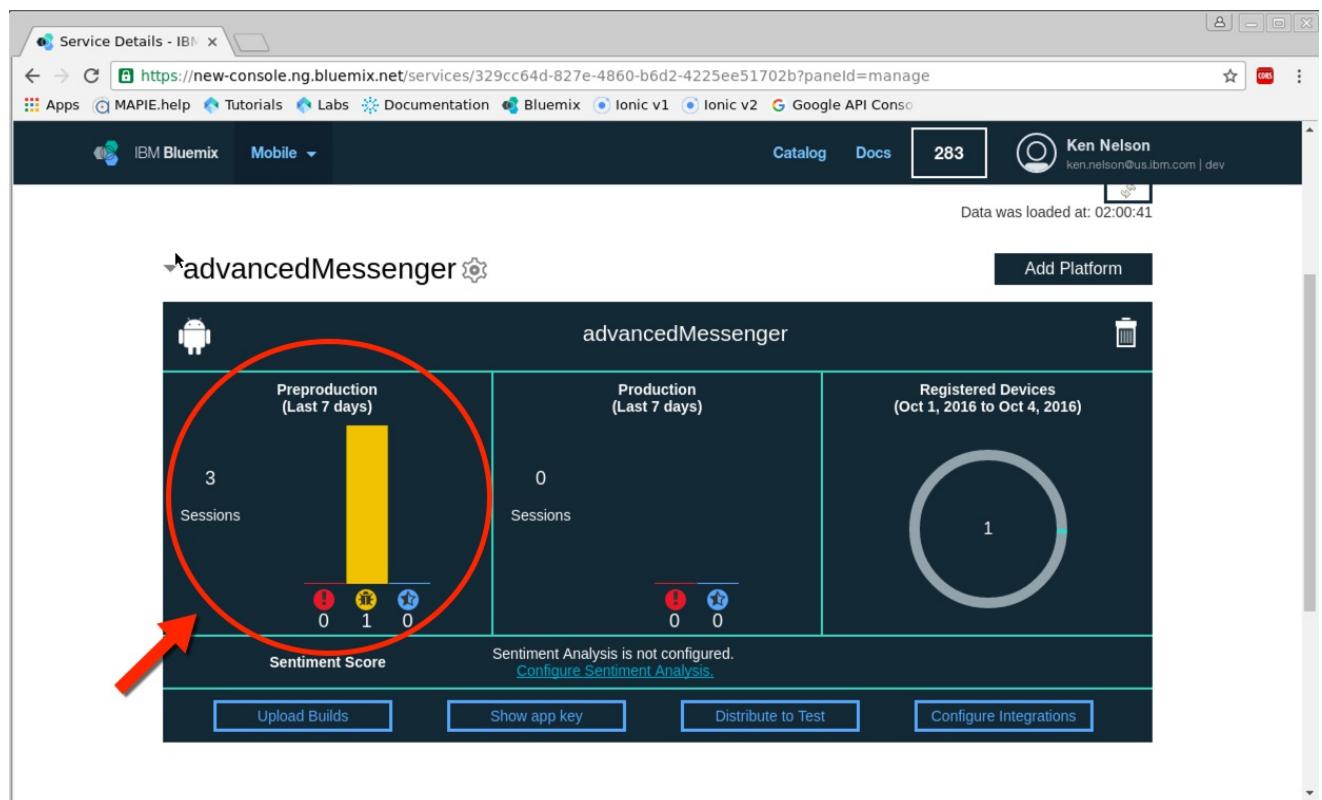


Note: With the Android emulator you can only submit one bug report at time. Because of this, you have to stop and restart the application to submit another bug.

8 - Review Bug Reports

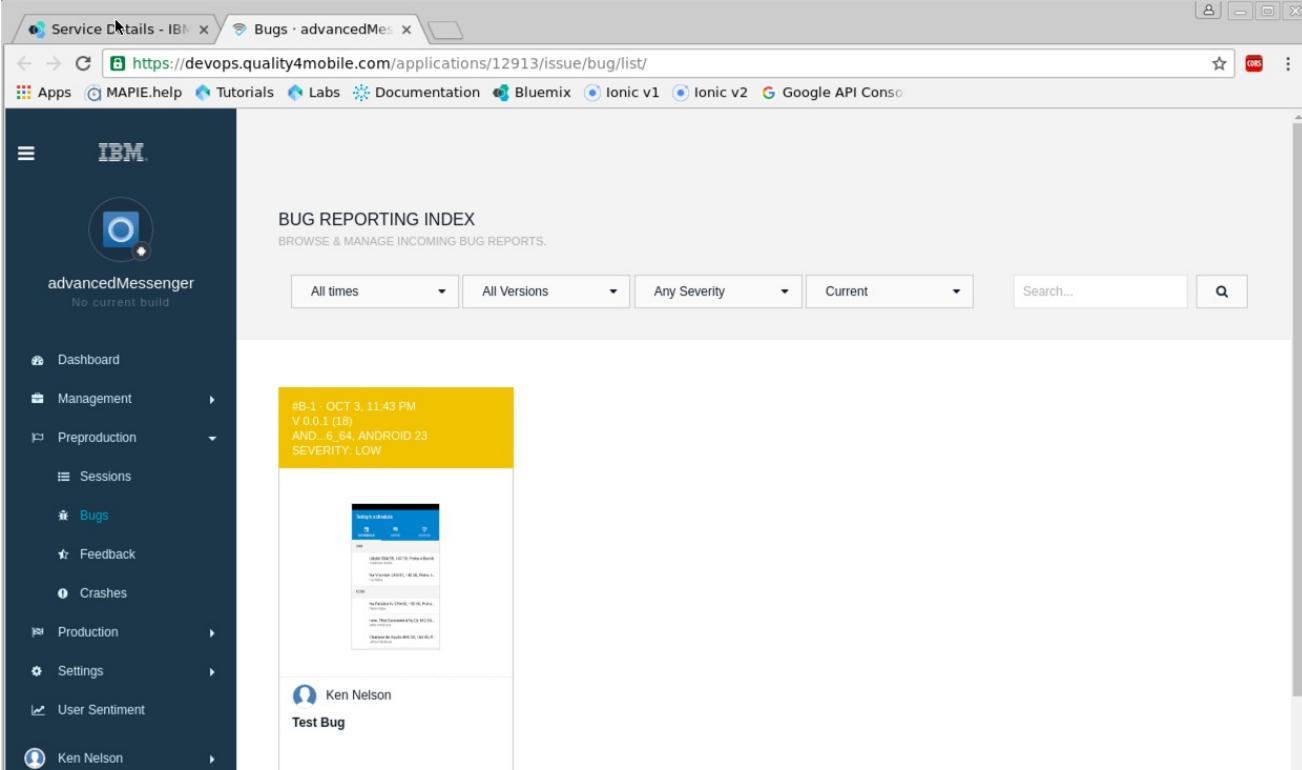
Now that a bug has been created it is time to review the bug.

1. If not already open, open a browser, login, and select the MQA service that was created in step 1. If your browser is still open and on the MQA service that was created in step 1, refresh your page. You should notice that you have a bug identified in the dashboard.



2. Click on the bug count to open Bug Reporting Index screen. This will show a list of the bugs that have been identified. On the left-hand side you will see a menu with options such as **Sessions**, **Bugs, Feedback, & Crashes** for Preproduction. There are similar options for Production. You can also modify the settings for MQA such as branding, notifications, and integrations with other software

such as JIRA and GitHub. You can also configure User Sentiment. For now we will work with the Bug Reporting Index.



The screenshot shows a web browser window with the URL <https://devops.quality4mobile.com/applications/12913/issue/bug/list/>. The page title is "Bugs · advancedMessenger". The left sidebar for the "advancedMessenger" service includes options like Dashboard, Management, Preproduction, Sessions, Bugs (which is selected), Feedback, Crashes, Production, Settings, User Sentiment, and Ken Nelson. The main content area is titled "BUG REPORTING INDEX" with the sub-instruction "BROWSE & MANAGE INCOMING BUG REPORTS.". It features several filter dropdowns: "All times", "All Versions", "Any Severity", and "Current". A search bar and a magnifying glass icon are also present. A yellow callout box highlights a specific bug entry: "RB-1 OCT 3, 11:43 PM V 0.0.1 (1.8) AND_ 6_64 ANDROID 23 SEVERITY: LOW". Below this is a screenshot of a mobile device displaying a bug report. At the bottom of the bug entry is the user "Ken Nelson" and the label "Test Bug".

Note: If pop-ups are blocked, you will not be able to see the list of bugs as the link opens a new window/tab. To view the list of bug please ensure that pop-ups are not blocked.

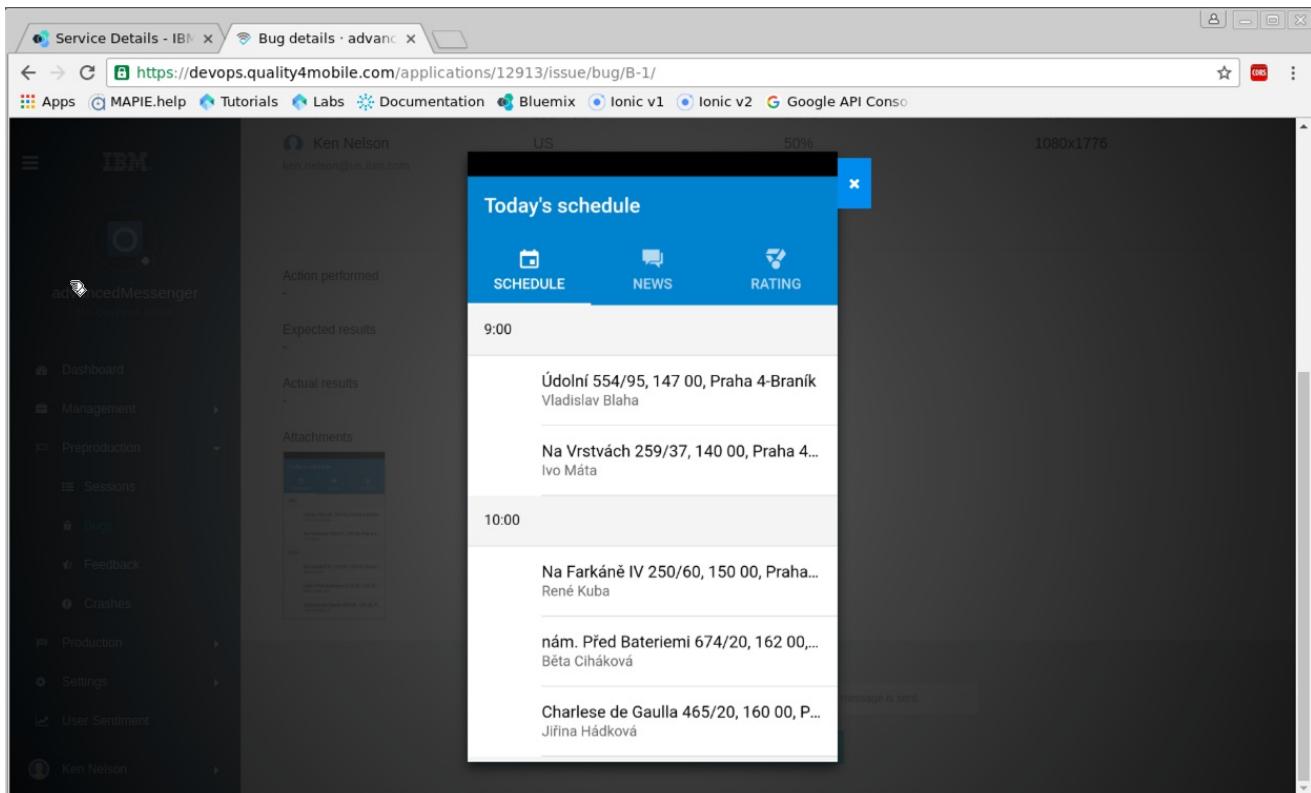
3. With the **Preproduction | Bugs** menu option selected, scroll until you see the **DETAILS** link for the recently created bug. **Click the DETAILS link** to open the bug details.

The screenshot shows the 'BUG REPORTING INDEX' section of the IBM Service Details interface. On the left, there's a sidebar with various navigation options like Dashboard, Management, Preproduction, Bugs, Feedback, Crashes, Production, Settings, User Sentiment, and a profile for Ken Nelson. The main area displays a card for a bug report. The card header includes the bug ID (#B-1), date (OCT 3, 11:43 PM), version (V 0.0.1 (18)), and device (AND_6_64, ANDROID 23). The severity is listed as LOW. Below the header is a screenshot of a mobile application interface. At the bottom of the card, there's a link labeled 'Test Bug' and a blue button labeled 'DETAILS' which is circled in red. A red arrow points from the 'Test Bug' link towards the 'DETAILS' button.

4. With the Bug Report Detail page open, you will get some basic information about the bug. You can scroll up and down the page to view the basic information.

The screenshot shows the 'BUG #B-1' detail page. The top bar includes the URL https://devops.quality4mobile.com/applications/12913/issue/bug/B-1/. The sidebar on the left is identical to the one in the previous screenshot. The main content area starts with 'BUG #B-1'. Below it, there's a timestamp (OCT 3, 2016 11:43 PM), a version (0.0.1 (18) / Android 23 / Android SDK built for x86_64), and a severity (Low). The title of the bug is 'Test Bug'. To the right of the title is a 'Archive' button. Below this, there are two tables of detailed information. The first table includes columns for Version, Session Start, Network / Operator, and Device / OS. The second table includes columns for Participant, Location, Battery, and Screen. At the bottom of the page, there are sections for 'Action performed', 'Expected results', and 'Actual results', each with a minus sign indicating expandable content. A blue link labeled 'FULL SESSION DETAIL' is located between the two tables. A red arrow points to this link.

At the bottom of the page you can click the images to expand them for a more detailed view.



To view the session details of the bug, click the **FULL SESSION DETAILS** link.

A screenshot of the same application interface, but now focused on a specific bug report. The sidebar shows 'Sessions' is still selected. The main area displays 'BUG #B-1' with details: 'OCT 3, 2016 11:43 PM', '0.0.1 (18) Android 23', 'Android SDK built for x86_64', 'Low Test Bug', and an 'Archive' button. Below this, there's a table with device information: VERSION 'v 0.0.1 (18)', SESSION START 'Oct 3, 2016 11:36 PM', NETWORK / OPERATOR 'Unknown', DEVICE / OS 'Android SDK built for x86_64', PARTICIPANT 'Ken Nelson', LOCATION 'US', BATTERY '50%', and SCREEN '1080x1776'. A large red arrow points from the text above to the 'FULL SESSION DETAIL' link, which is highlighted with a red oval. The link is located in the 'Action performed' section of the bug details table.

This will display additional information about the state of the device, networking, etc.

The screenshot shows the 'Service Details - IBM' interface. On the left is a sidebar with the IBM logo and navigation links like Dashboard, Management, Preproduction, Production, Settings, and User Sentiment. A user profile for Ken Nelson is also present. The main area displays session details. At the top right is a checkbox labeled 'Show Initial Conditions'. Below it is a table with columns 'TYPE', 'TIME', and 'CODE'. The first row is a 'CONDITION' entry for 'power:' occurring at '1s' with the code: battery: voltage: 0, temperature: 0, level: 50, technology: Li-ion, health: good; plug: ac, state: charging. The second row is a 'CONDITION' entry for 'networking: interfaces:' occurring at '1s'. The third row is a 'BUG' entry for '#B-1 - OCT 3, 11:43 PM' with a yellow background, V 0.0.1 (18), AND..._6_64, ANDROID 23, SEVERITY: LOW. It includes a screenshot of a mobile application's schedule screen.

Finally, click the **Show Initial Conditions** checkbox to show information about the initial state of the device before the application began running.

This screenshot is identical to the one above, but the 'Show Initial Conditions' checkbox in the top right corner is circled in red, and a large red arrow points to it from the bottom left, indicating where the user should click.

Checking the checkbox will present the following:

TYPE	TIME	CODE
BUILD	Initial	launch_count: 2 installation: false
LOCATION	Initial	cell:
NETWORKING	Initial	interfaces:
POWER	Initial	battery: level: 50 voltage: 0 health: good temperature: 0 technology: Li-ion state: charging plug: ac
SCREEN	Initial	colors: 16777216 height: 1776 width: 1080 density: xxhdpi orientation: portrait rotation: 0
SYSTEM	Initial	os: version: 23 name: Android info: 6.0 REL (2872745) environment: variables: ANDROID_BOOTLOGO: 1 BOOTCLASSPATH: /system/framework/core-libart.jar:/system/framework/conscrypt. ANDROID_PROPERTY_WORKSPACE: 9,0 PATH: /sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin

Section 8 - Upload Application Binary for Distribution (Optional)

1. Another feature that Mobile Quality Assurance provides is the ability to distribute application builds to testers. To do this, you will need to upload your application binary to MQA. Start by expanding the **Management** menu option, then **click** the **Builds** menu option.

The screenshot shows the 'Service Details - IBM' application interface. On the left, a sidebar menu for 'advancedMessenger' includes options like Dashboard, Management (Builds, Participants, Distributions, NDAs, Statistics, Devices), Preproduction, Production, and Ken Nelson. The 'Builds' option is circled in red, and a red arrow points to it from below. The main content area is titled 'BUG REPORTING INDEX' and displays a summary of a build: '#B-1 OCT 3, 11:43 PM V 0.0.1 (18) AND. 6_64, ANDROID 23 SEVERITY: LOW'. Below this is a screenshot of a mobile application interface with a 'Select database' screen. At the bottom, there's a user profile for 'Ken Nelson' and a note 'Test Bug'.

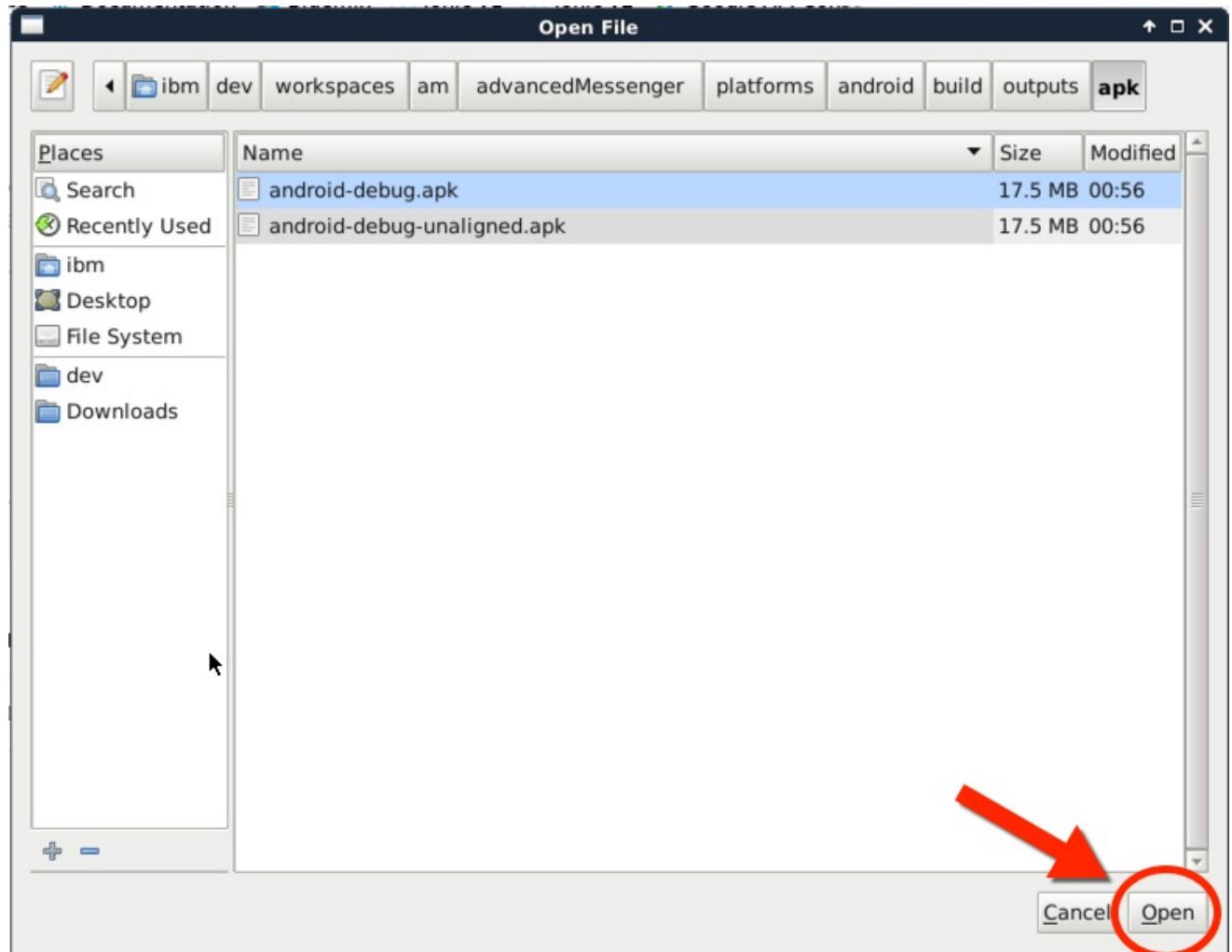
2. This opens a page to upload your application binary either by dragging and dropping the file on the space indicated by the dashed-lined box, or by pressing the **Choose file to upload** button.

The screenshot shows the 'Builds · advancedMessenger' management page. The sidebar menu is identical to the previous screenshot. The main content area is titled 'BUILD MANAGEMENT' and has a section for uploading builds with the text 'Drop your build here or just...'. A blue button labeled 'Choose file to upload' is highlighted with a red circle and a red arrow pointing to it. Below this section, there's a table with columns for VERSION, BUILD FILES, RECORD LOGS, and OPTIONS. One row is shown for 'v 0.0.1 (18)' with 'No files available.' under BUILD FILES and a green 'Enable' button under OPTIONS.

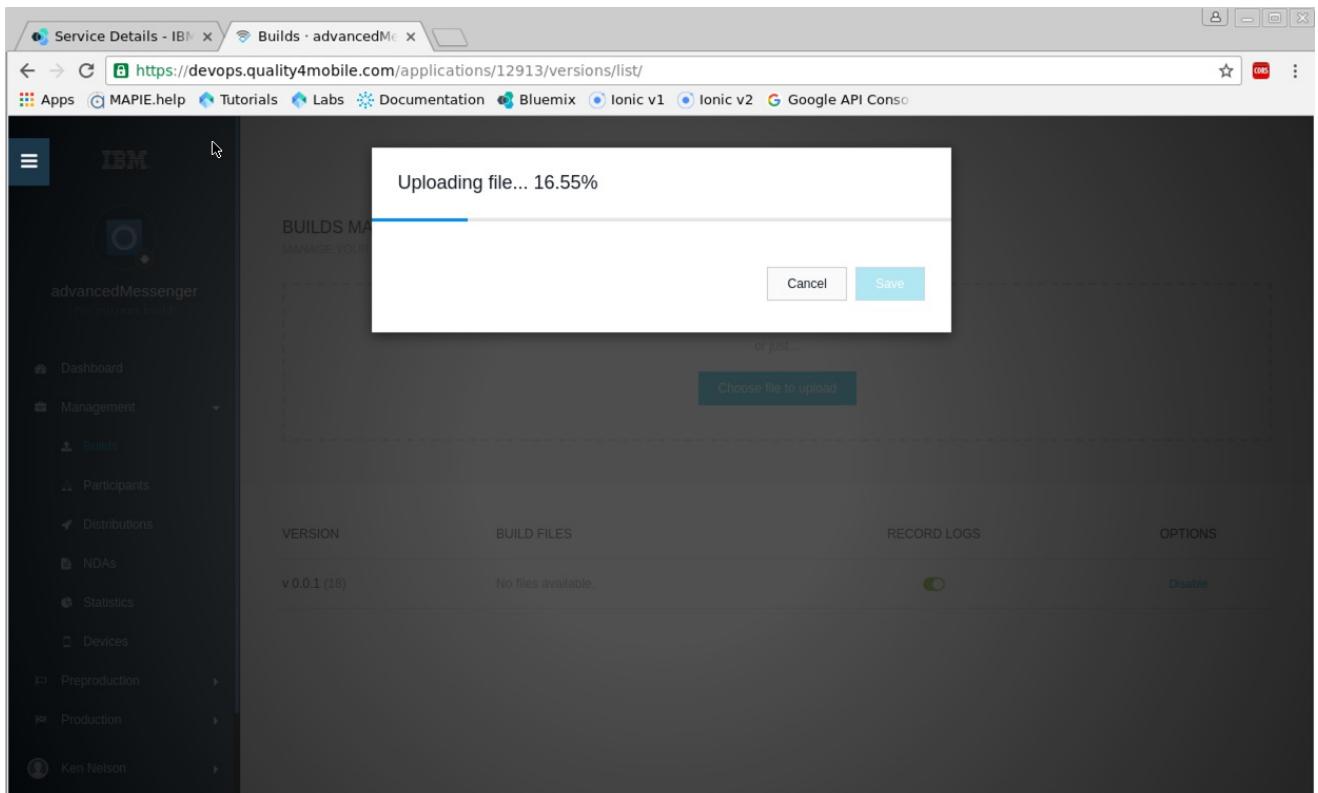
3. Click the **Choose file to upload** button and then navigate to the location of the Android Application Binary. If you have completed

labs 1-8, this should be located in

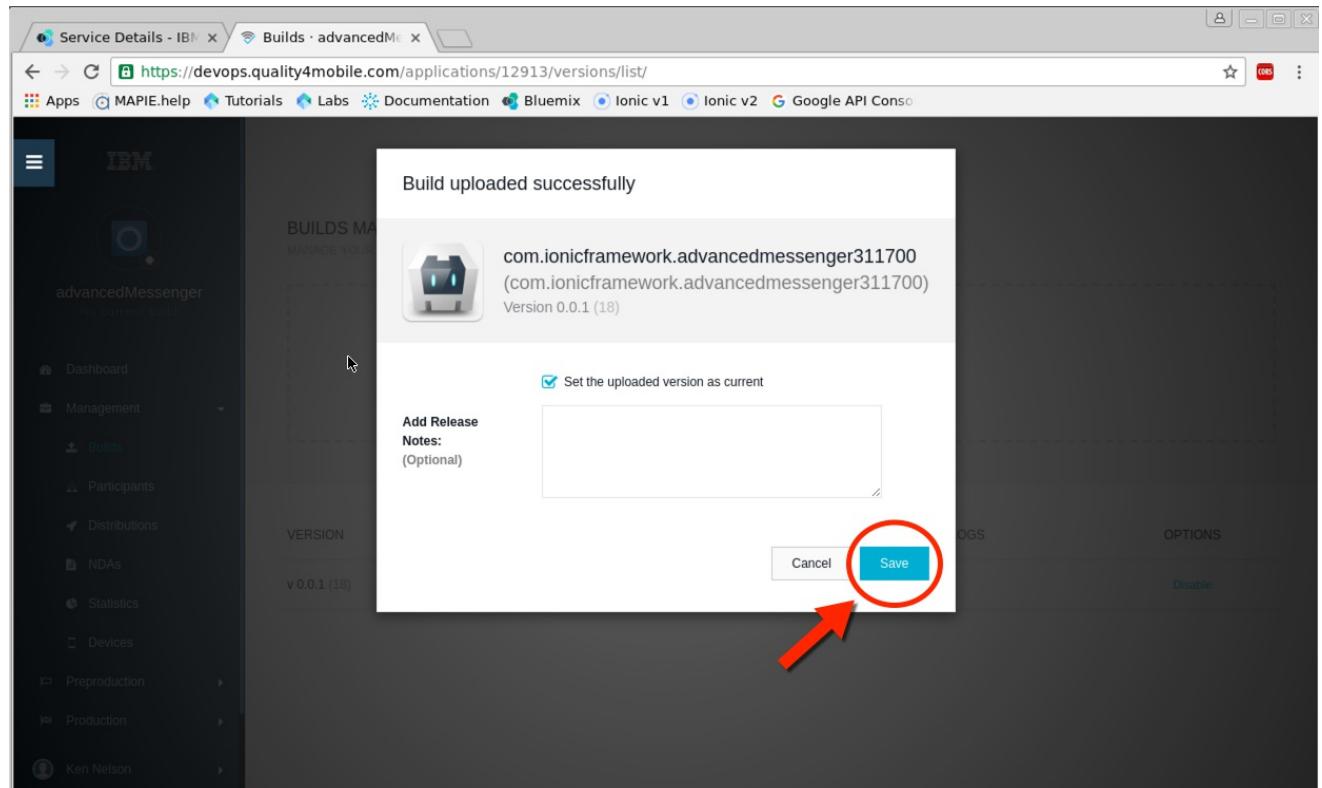
`~/dev/workspaces/am/advancedMessenger/platforms/android/outputs/apk`. Select the `android-debug.apk` file and **click the Open button.**



The upload starts and you will see a progress dialog showing the current state of the file upload.



The upload is complete when you see the **Build uploaded successfully** dialog. Click the **Save** button to complete the upload process.



Notice that you now have a build file associated with version 0.0.1.

The screenshot shows the IBM Build Management interface. On the left, a sidebar menu for 'advancedMessenger' includes 'Dashboard', 'Management' (with 'Builds' selected), 'Participants', 'Distributions', 'NDAs', 'Statistics', 'Devices', 'Preproduction', 'Production', and 'Ken Nelson'. The main area is titled 'BUILD MANAGEMENT' with the sub-instruction 'MANAGE YOUR APP VERSIONS & ADD NEW BUILD FILES.' It features a dashed box for dropping files, a 'Choose file to upload' button, and a 'Current version' dropdown set to 'v (18)' with 'Build file: android-debug.apk - Oct 4, 2016 6:15 PM'. A table lists one build entry: 'v 0.0.1 (18)' with 'android-debug.apk - Oct 4, 2016 6:15 PM' under 'BUILD FILES', and 'Record Logs' and 'Options' columns.

4. Next you can register participants & groups. To create a participant, **click the Participants** menu item under the **Management** menu category.

The screenshot shows the IBM Participants & Groups Management interface. The left sidebar is identical to the previous screenshot. The main area is titled 'PARTICIPANTS & GROUPS MANAGEMENT' with the sub-instruction 'MANAGE YOUR USERS, ADD OR REMOVE GROUPS.' It has tabs for 'PARTICIPANTS' (selected), 'GROUPS', and 'WAITING LIST'. A search bar is at the top right. Below, there's a '+ Add Participants' button and a list of users. The user 'ken.nelson@us.ibm.com' is listed with the group 'internal-admin' and a 'Remove' link. A red circle highlights the 'Participants' menu item in the sidebar, and a red arrow points to it from below.

When the **PARTICIPANTS & GROUP MANAGEMENT** page displays, **click the Add Participants** button.

The screenshot shows the 'Participants & Groups Management' section of the IBM application interface. On the left, there's a sidebar with various navigation options like Dashboard, Management, Participants, Distributions, NDAs, Statistics, Devices, Preproduction, Production, and a user profile for Ken Nelson. The main area is titled 'PARTICIPANTS & GROUPS MANAGEMENT' with the sub-instruction 'MANAGE YOUR USERS, ADD OR REMOVE GROUPS.' Below this, there are tabs for 'PARTICIPANTS' (which is selected), 'GROUPS', and 'WAITING LIST'. A search bar is at the top right. In the center, there's a list of participants starting with 'ken.nelson@us.ibm.com' and an 'internal-admin' entry with a 'Remove' link. At the bottom left of this list is a button labeled '+ Add Participants' with a red circle and a red arrow pointing to it.

When the **Add Participants** dialog opens, enter a comma separated list of email addresses to invite participants. Optionally you can create and add them to a group by typing a group name in the **Add a group** field. To finish adding participants, **click the Add Participants button**. For this lab, simply **click the Cancel** button as we will not be adding participants.

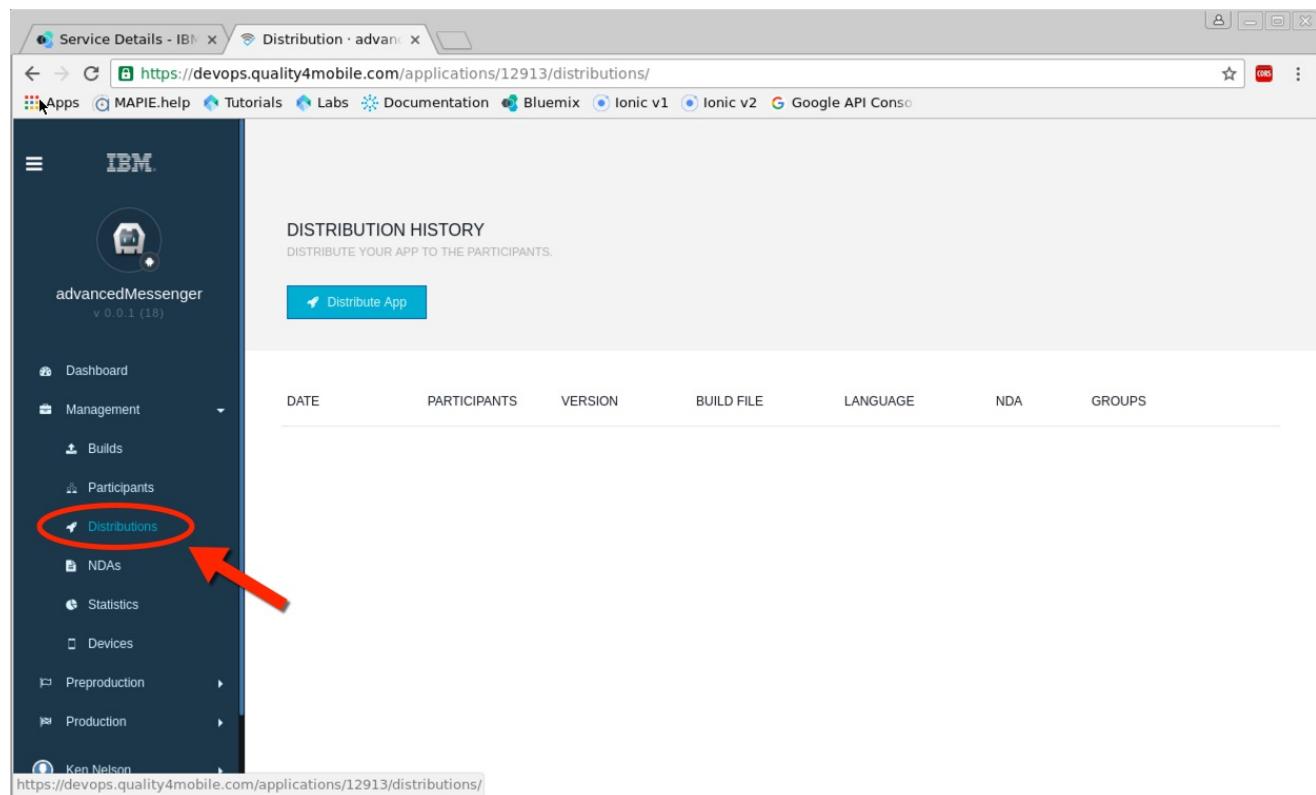
This screenshot shows the 'Add participants' dialog box. It has a header 'Add participants' and a message box stating 'The email addresses provided here will be added to Participants list.' Below this is an 'Emails:' input field with a placeholder 'Use commas or spaces between email addresses.' and a note 'You can assign groups to your participants.' The 'Groups:' section contains a 'Add a group' button. At the bottom, there are two buttons: 'Cancel' and 'Add Participants', with a red circle and a red arrow pointing to the 'Cancel' button.

Adding groups works in a similar fashion. If you wish to explore, you may **click** the **Groups** tab and then **click** the **Add Groups** button.

One last option is to copy the **Waiting Room API URL** and send it in an email distribution list. As members register they will show in the waiting room awaiting an administrator's approval.

Step 9 - Distributing your Application (Optional)

1. To perform a distribution, **click** the **Distributions** menu item under the **Management** menu category. This will open the **Distributions** page.



2. **Click** the **Distribute App** button to open the **App Distribution** dialog.

The screenshot shows the 'Distribution History' section of the IBM app management interface. On the left, there's a sidebar with navigation links like Dashboard, Management, and Distributions. The main area displays a table with columns: DATE, PARTICIPANTS, VERSION, BUILD FILE, LANGUAGE, NDA, and GROUPS. At the top of this table, there's a blue button labeled 'Distribute App' with a red circle around it. A red arrow points from the bottom-left towards this button.

- With the **App Distribution** dialog open, you can select the build you want to send, attach an NDA (if one has been created), select the group or groups to distribute the app to, and add additional email addresses to the distribution list. Once everything has been configured you can **click the Distribute App button**. For now simply **click the Cancel button** as we will not be distributing apps.

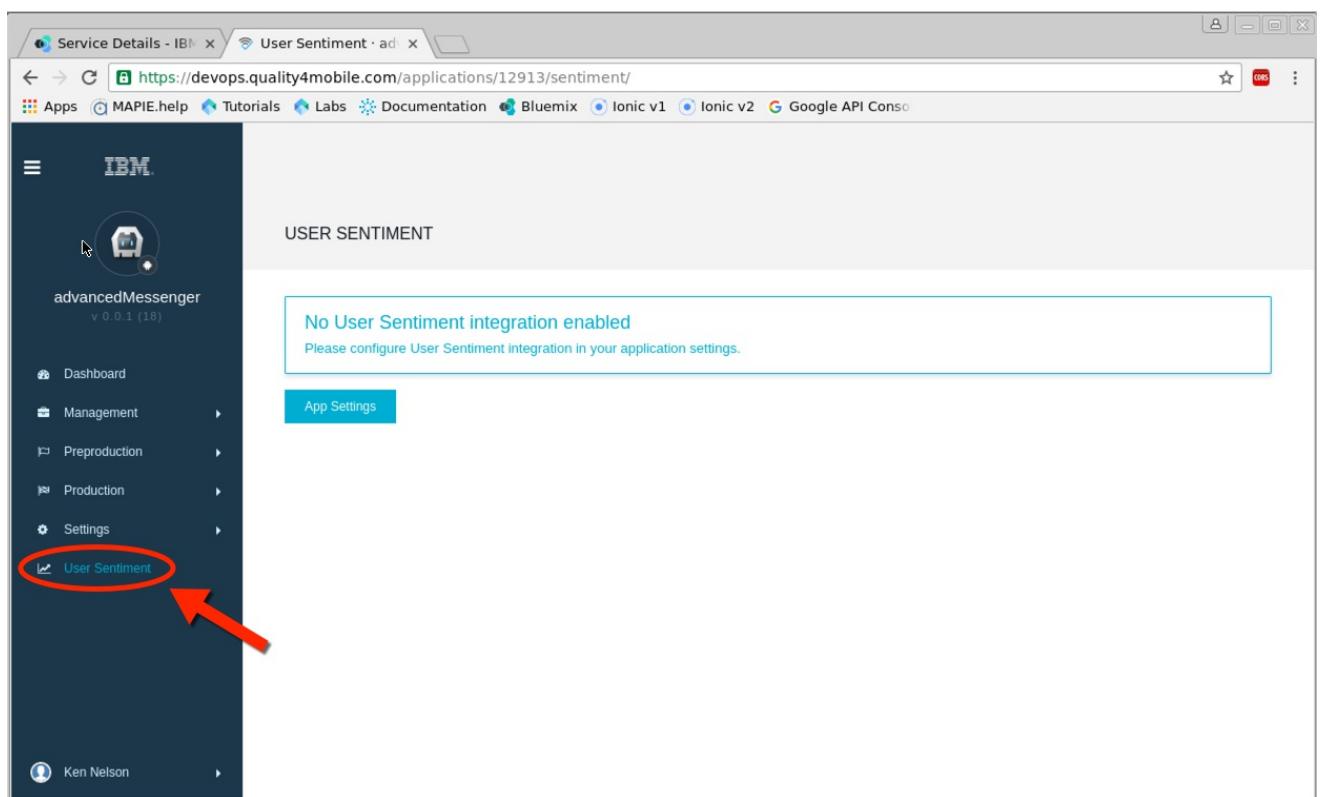
The screenshot shows the 'App Distribution' dialog box. It contains fields for 'Sender address' (set to 'ken.nelson@us.ibm.com'), 'Build file' (set to 'android-debug.apk'), 'NDA' (set to 'No NDA'), 'Select groups' (with a checkbox for 'internal-admin' which is unselected), and 'Extra emails' (an optional text area). At the bottom of the dialog, there are two buttons: 'Cancel' (highlighted with a red circle and a red arrow pointing to it) and 'Distribute App'.

Section 10 - Configure and View User Sentiment of Consumer Applications

Consumer applications are applications that are downloadable via the various public app stores such as iTunes and Google Play. The User Sentiment Analysis feature of Mobile Quality Assurance will analyze user feedback and provide recommendations for future enhancements.

To use User Sentiment Analysis you will need to configure it with an app that is in one or more of the public app stores.

1. To start, **click the User Sentiment menu item.**



2. Next **click the App Settings button** to open the **App Settings** page to configure the application.

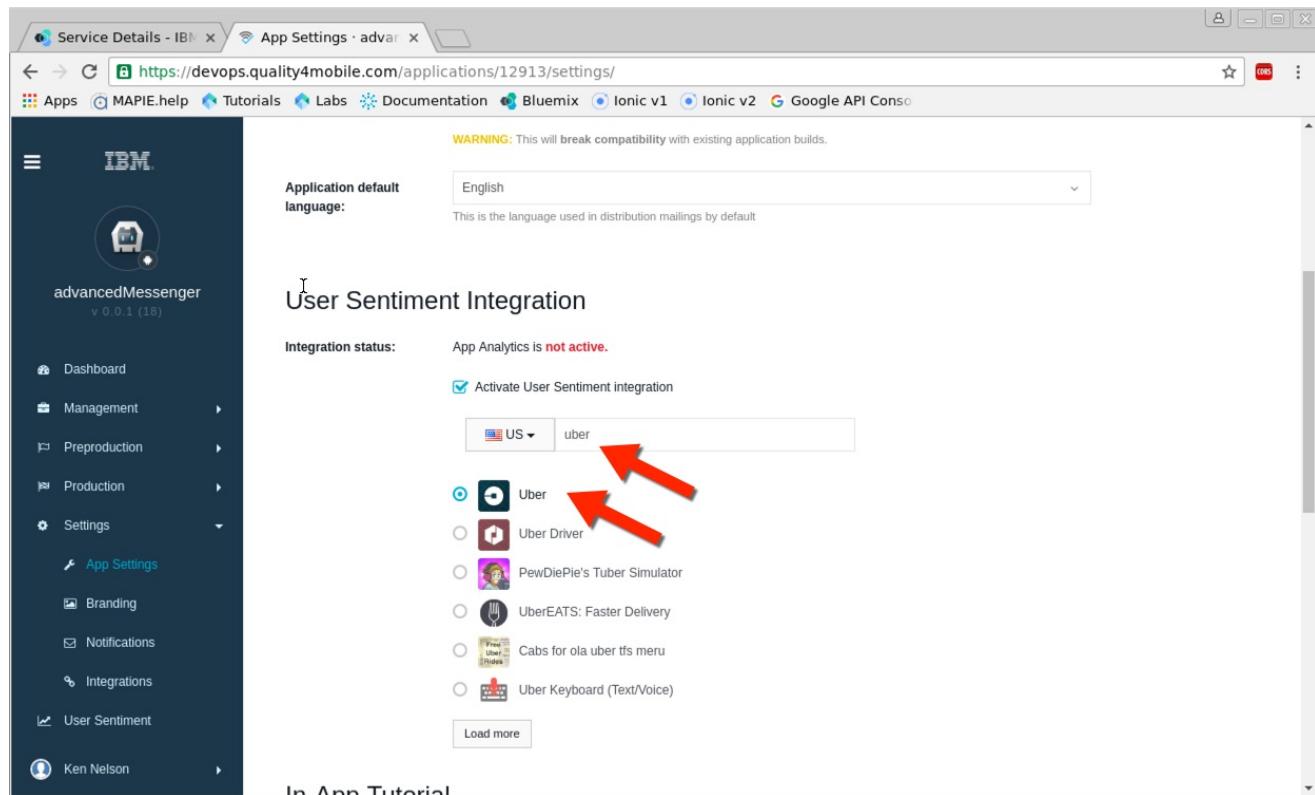
The screenshot shows the IBM Bluemix interface for the advancedMessenger application. On the left, there's a sidebar with navigation links like Dashboard, Management, Preproduction, Production, Settings, and User Sentiment. The main content area is titled 'USER SENTIMENT'. It displays a message: 'No User Sentiment integration enabled' with the sub-instruction 'Please configure User Sentiment integration in your application settings.' Below this message is a blue rectangular button labeled 'App Settings'. A red circle highlights this button, and a red arrow points to it from the bottom right.

- With the App Settings page open, check the Activate User Sentiment integration checkbox found in the User Sentiment Integration section.

The screenshot shows the 'App Settings' page for the advancedMessenger application. The left sidebar includes 'App Settings' under the 'Settings' category. The main area has a section titled 'User Sentiment Integration'. Inside this section, the status is shown as 'App Analytics is not active.' Below the status is a checkbox labeled 'Activate User Sentiment integration', which is currently unchecked. A red box surrounds the entire 'User Sentiment Integration' section, and a red arrow points to the 'Activate User Sentiment integration' checkbox.

- With the Activate User Sentiment integration checkbox checked, a text input will be displayed showing the advancedMessenger

application name. Because the **advancedMessenger** application does not exist on a public app store you will not be able to configure the rest of the integration. However, if you enter a known application such as **Uber** you will get a list of potential matches. **Check the radio button next to the option that says **Uber**.**



5. Save your settings by scrolling to the bottom of the page and click the **Save settings** button.

The screenshot shows the 'SDK Logging' configuration page. On the left is a sidebar with the 'advancedMessenger' app logo and version 'v 0.0.1 (18)'. The main area has two sections: 'Minimum logging level:' with radio buttons for Verbose, Info (selected), Warning, Error, and Fatal, and a checked checkbox for 'Include logs from emulators'. Below this is a note: 'Changing above settings won't affect currently running sessions.' The second section is 'Log messages about:' with checkboxes for Telephony, Power (checked), Screen (checked), Networking (checked), System (checked), Location, and Custom. A note below says 'Changing above settings will affect currently running sessions.' At the bottom is a blue 'Save settings' button, which is circled in red with a large red arrow pointing to it.

- After saving the settings, scroll back to the **User Sentiment Integration** section and you will notice that the Integration Status has changed from **Not Active** to **Active**.

The screenshot shows the 'User Sentiment Integration' configuration page. The left sidebar is identical to the previous screenshot. The main area starts with 'Application default language:' set to English. Below this is the 'User Sentiment Integration' section. It shows 'Integration status:' as 'Active - Uber (com.ubercab)' with a green checkmark, and a checked checkbox for 'Activate User Sentiment integration'. A dropdown menu shows 'US' selected. To the right is a list of integration options with radio buttons: Uber (selected), Uber Driver, PewDiePie's Tuber Simulator, UberEATS: Faster Delivery, Cabs for ola uber tfs meru, and Uber Keyboard (Text/Voice). A 'Load more' button is at the bottom. At the very bottom is an 'In-App Tutorial' section with a checked checkbox for 'Enable in-application SDK tutorial.'

- Click the **User Sentiment** menu item, and this time notice that there is available feedback.

The screenshot shows the 'User Sentiment' dashboard for the Uber app. At the top, there's a navigation bar with tabs for 'Dashboard', 'Compare', 'Clusters', 'Reviews', and 'Stats'. Below the navigation is a section titled 'Top Clusters' which lists user comments categorized by sentiment. To the right of this is a 'Cluster Stats' section featuring a donut chart and a table of average cluster counts.

Average Cluster Type	Count
5 Star average clusters	20
4 Star average clusters	3
3 Star average clusters	11
2 Star average clusters	10
1 Star average clusters	4

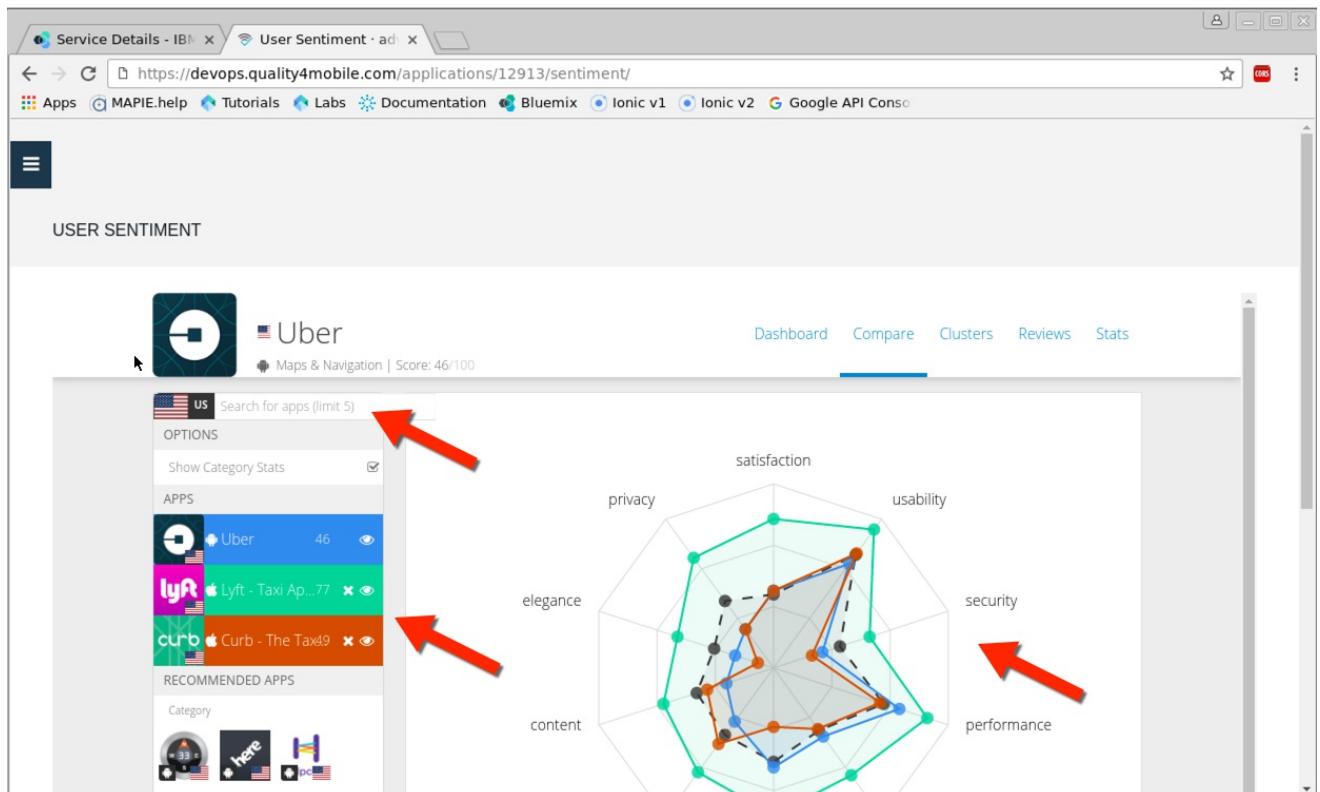
Scroll to the **Attributes** section and you will see how the application breaks down with respect to 10 specific attribute categories (Privacy, Content, Interoperability, Elegance, Security, Pricing, Usability, Stability, Satisfaction, and Performance). Not all applications will have values for each category as comments may not be identified that match a particular category.

The screenshot shows the 'User Sentiment' dashboard for the Uber app, specifically focusing on the 'Attributes' section. This section displays four cards, each representing a different attribute: Privacy, Content, Interoperability, and Elegance. Each card includes a red circular icon with a white symbol (representing the attribute), a progress bar indicating the number of reviews, and a detailed description of the attribute and its impact on user perception.

Attribute	Reviews	Description
Privacy	Based on 2498 reviews	Even the perception of privacy issues can cause alarm. Do you ask for more information than you need? Does the app link to questionable external content or appear to be fraudulent or
Content	Based on 7260 reviews	Ads, incorrect data, or content download issues can seriously affect users perception of app quality.
Interoperability	Based on 6801 reviews	Interaction with the device, camera, other apps, or audio and video output issues can be frustrating to users. Test how your app interacts with everything outside of your app.
Elegance	Based on 2533 reviews	The elegance of an applications UX is very important on mobile. Take a look at top apps that score high in excellence such as Path, and Google+

8. Continue exploring the User Sentiment for the application by clicking the **Compare**, **Clusters**, **Reviews**, and **Stats** tabs.

The **Compare** allows you to compare this app with up to 5 additional applications. Simply enter the name of the application or applications you want to compare in the **Search for apps** textbox and hit enter. It will be added to the list of apps. The stats for all apps identified will be shown in the graph as well as stats for the category of apps.



The **Clusters** shows similar keywords identified in reviews. This will tell you how the same clusters or groupings of words or phrases have been identified in multiple reviews. You can limit the reviews to 1, 3, 14, 30, or 90 days.

The screenshot shows the 'Reviews' tab for the Uber application. At the top, there's a navigation bar with tabs: Dashboard, Compare, Clusters, Reviews (which is highlighted in blue), and Stats. Below the navigation bar, there's a search bar and a date range selector. The main area displays a list of user reviews with their dates and scores. The reviews are as follows:

- When compared to others, Uber charging **high prices** (9/14/2016 - 9/22/2016) • ★ 2.60
- I am unable to **book a cab** by this app (9/10/2016 - 9/25/2016) • ★ 2.71
- Always telling me to **reauthorize** my **Paytm** wallet. And despite multiple attempts there is no way to **book** a cab. It's really frustrating. Deve... (9/17/2016 - 9/22/2016) • ★ 2.75
- Amazing .. rocking .. at time .. best **services** .. #alltimefavourite (9/15/2016 - 9/24/2016) • ★ 5.00

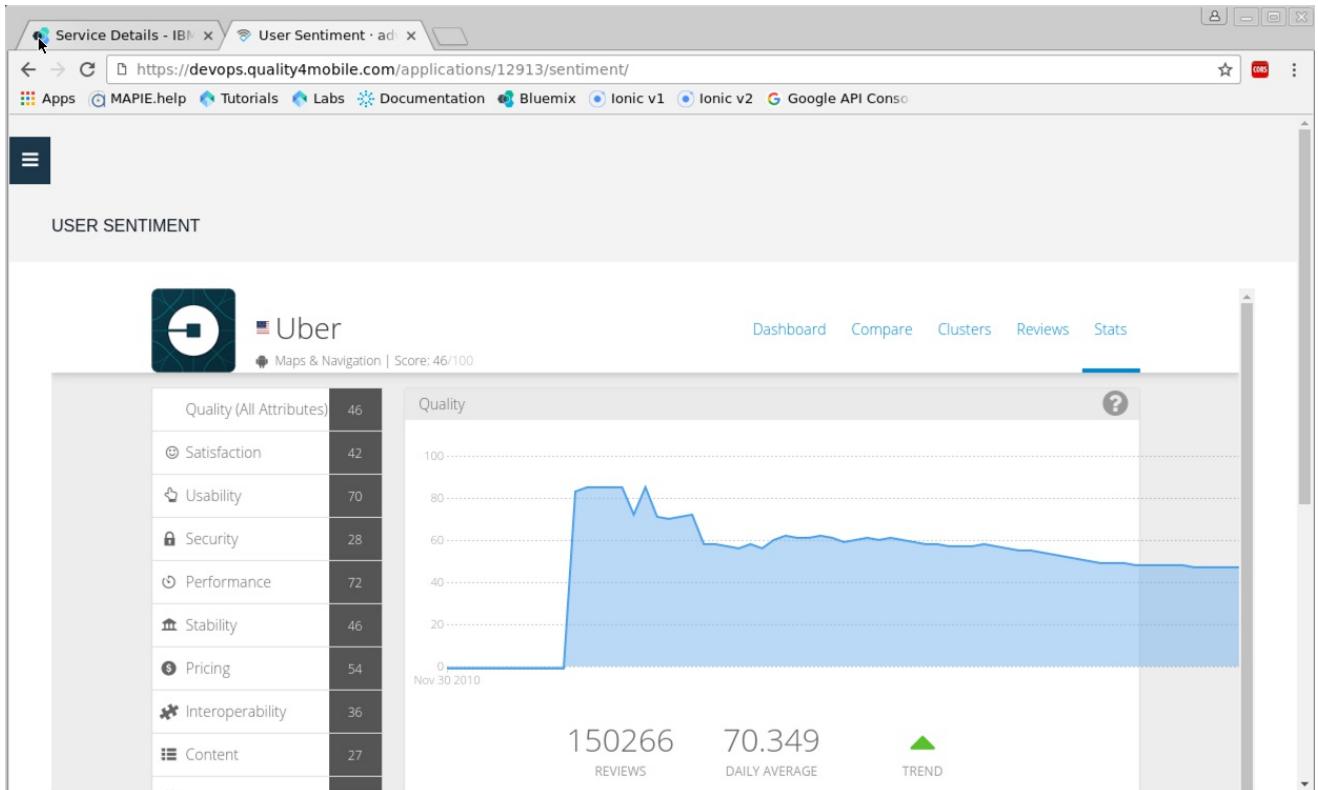
At the bottom right, there's a link to 'Export to CSV'.

The **Reviews** will show all the reviews from the the public app store as well as graph on a daily basis the positive and negative reviews. You can limit the reviews to specific categories such as Content, Elegance, etc. as well as the date range of reviews.

The screenshot shows the 'Reviews' tab for the Uber application. At the top, there's a navigation bar with tabs: Dashboard, Compare, Clusters, Reviews (which is highlighted in blue), and Stats. Below the navigation bar, there's a search bar and a date range selector. The main area features a stacked bar chart showing the daily count of positive (green) and negative (red) reviews. The chart spans from Sep 4 to Sep 29. To the left of the chart, there's a sidebar with filters for attributes like Content, Elegance, Interoperability, Performance, Pricing, Privacy, Satisfaction, Security, Stability, and Usability, along with a date range selector. At the bottom, there's a single review displayed with its author and date.

Finally the **Stats** tab shows running average score for a particular

category such as Quality, Satisfaction, Usability, etc. You can change the category by select the desired category from the list on on the left hand side.



Lab 10 - Conclusion

Congratulations! You have completed the Mobile Quality Assurance lab.

In this lab you learned to:

- Instantiate Mobile Quality Assurance (MQA) service on IBM Bluemix
- Instrument an application to leverage MQA
- Review bug reports and feedback using the MQA service on IBM Bluemix
- Upload application binary for distribution
- Distribute an application using MQA
- Configure and view user sentiment of consumer applications