Description
Intended User
Features
User Interface Mocks
Screen 1: App Drawer
Screen 2: News Feed
Screen 3: News Detail
Screen 4: Event Diary
Screen 5: Lodging Information
Screen 6: Contact Us
Screen 7: Tablet Layout – News Feed
Screen 8: Tablet Layout – News Detail
Screen 9: Widget
Key Considerations
How will your app handle data persistence?
Describe any corner cases in the UX.
Describe any libraries you'll be using and share your reasoning for including them.
Next Steps: Required Tasks
Task 1: Project Setup
Task 2: Implement UI for Each Activity and Fragment
Task 3: Implement Content Server
Task 4: Set up the SQLite Database
Task 5: Implement Content Provider
Task 6: Implement JSON Client
Task 7: Implement Sync Adapter
Task 8: Tie Things Together
Task 9: Implement Map Functionality
Task 10: Implement Calendar Export
Task 11: Implement Widget
Task 12: Polish the UI
Task 13: Finalize App

GitHub Username: par62mga

MG 2016

Description

This app delivers content and information related to the MG 2016 national event. This event is the fifth national gathering of MG enthusiasts and is hosted by the North American Council of MG Registers.

With this app you can always stay up to date with MG 2016 activities:

- View the latest MG 2016 news, events and lodging details.
- Not miss out on any of the excitement with the “MG 2016 at a Glance” widget and the ability to export events to your calendar.
- See where you are, launch maps and find the way to your hotel and event locations.
- You can also share events and news with your fellow MG enthusiasts via Facebook, Twitter, etc.

Intended User

The intended user base includes MG enthusiasts who are going to our interested in the MG 2016 national event.

Features

The main features of the app include:

News Feed:
- View news listing
- View articles
- Share news

Events Diary:
- View listing of events by date and time
- Select events and export to calendar
- Show event location on map

Lodging Information:

- View list of hotels
- Select hotel to initiate call
- Select hotel and navigate to website
- View hotel location on map

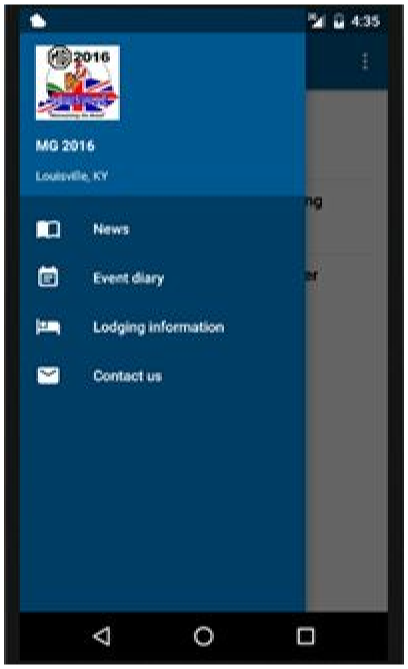
Contact Us:

- View information about the app
- Send email to app support email account

User Interface Mocks

The following mocks cover the core features of the app.

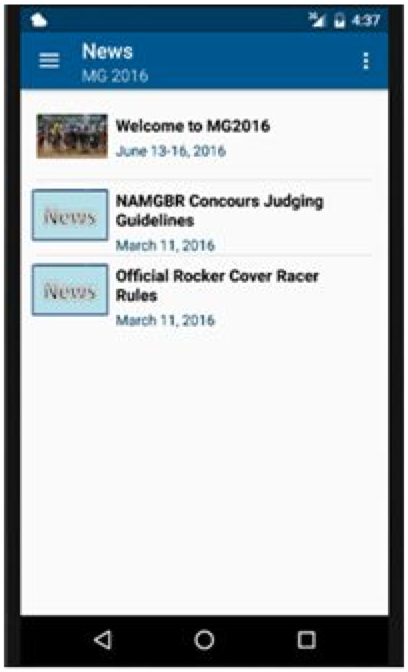
Screen 1: App Drawer



When opened the app drawer presents the main functionality of the application:

- News feed
- Events diary
- Lodging information
- Contact information
- Application settings (not shown in mocks)

Screen 2: News Feed



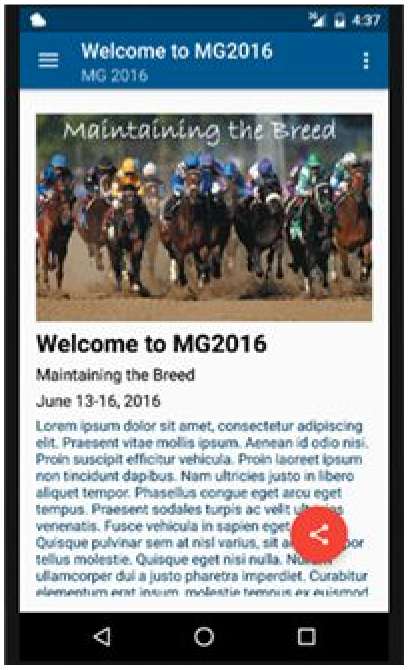
Welcome to MG2016
June 13-16, 2016

NAMGBR Concours Judging Guidelines
March 11, 2016

Official Rocker Cover Racer Rules
March 11, 2016

The news feed shows news articles in a listview. When a news article is selected, the News Detail page is opened.

Screen 3: News Detail



Welcome to MG2016

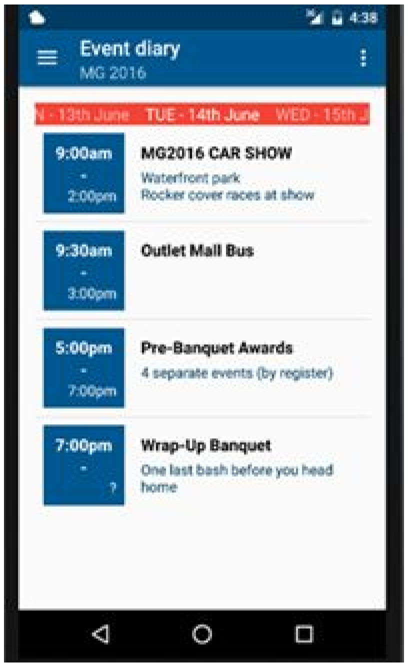
Maintaining the Breed

June 13-16, 2016

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vitae mollis ipsum. Aenean id odio nisi. Proin suscipit efficitur vehicula. Proin laoreet ipsum non tincidunt dapibus. Nam ultricies justo in libero aliquet tempor. Phasellus congue eget arcu eget tempus. Praesent sodales turpis ac velit u[illegible] venenatis. Fusce vehicula in sapien eget [illegible] Quisque pulvinar sem at nisl varius, sit a[illegible]or tellus molestie. Quisque eget nisi nulla. N[illegible] ullamcorper dui a justo pharetra imperdiet. Curabitur elementum erat ipsum, molestie tempus ex euismod

The news detail page shows the selected news article and exposes a floating action button that implements a share provider action that shares the news photo URL and headlines.

Screen 4: Event Diary



9:00am - 2:00pm **MG2016 CAR SHOW**
Waterfront park
Rocker cover races at show

9:30am - 3:00pm **Outlet Mall Bus**

5:00pm - 7:00pm **Pre-Banquet Awards**
4 separate events (by register)

7:00pm - ? **Wrap-Up Banquet**
One last bash before you head home

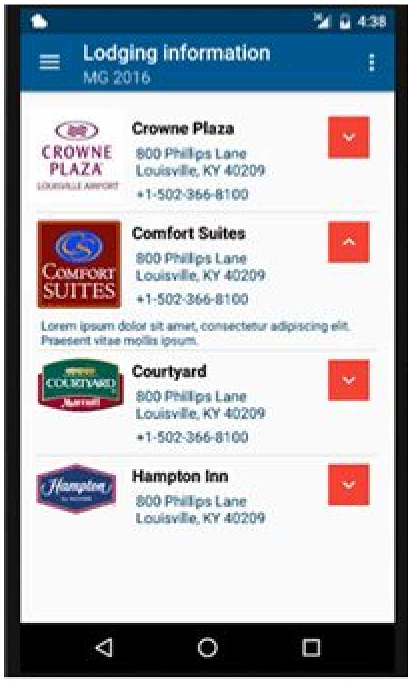
The events diary shows the calendar of events and is managed by a viewpager that shows each day in a separate page.

The page initially shown is based on the current date and time of the device, if the time is less than that first event day, then the first day is shown. Otherwise the day shown corresponds to the current day or the last day if the MG 2016 event is over.

When a particular event on the event page is selected, actionbar icons become available to:

- Share event: implements a share action provider to share event details.
- Show map: opens a map activity that shows both the current location and the event location on map, allowing the user to navigate to the location.

Screen 5: Lodging Information



Lodging information
MG 2016

Crowne Plaza
800 Phillips Lane
Louisville, KY 40209
+1-502-366-8100

Comfort Suites
800 Phillips Lane
Louisville, KY 40209
+1-502-366-8100
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vitae mollis ipsum.

Courtyard
800 Phillips Lane
Louisville, KY 40209
+1-502-366-8100

Hampton Inn
800 Phillips Lane
Louisville, KY 40209

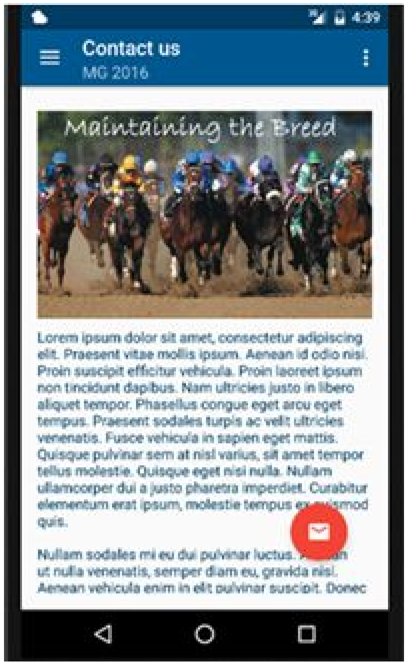
Lodging information shows official lodging details for the MG 2016 event in a listview.

Buttons beside each hotel are used to show or hide details like parking information and restrictions.

When a hotel is selected, actionbar icons become available to:

- Call hotel: launches a phone call to the hotel.
- Show map: opens a map activity that shows both the current location and the hotel location on map, also allowing navigation.

Screen 6: Contact Us

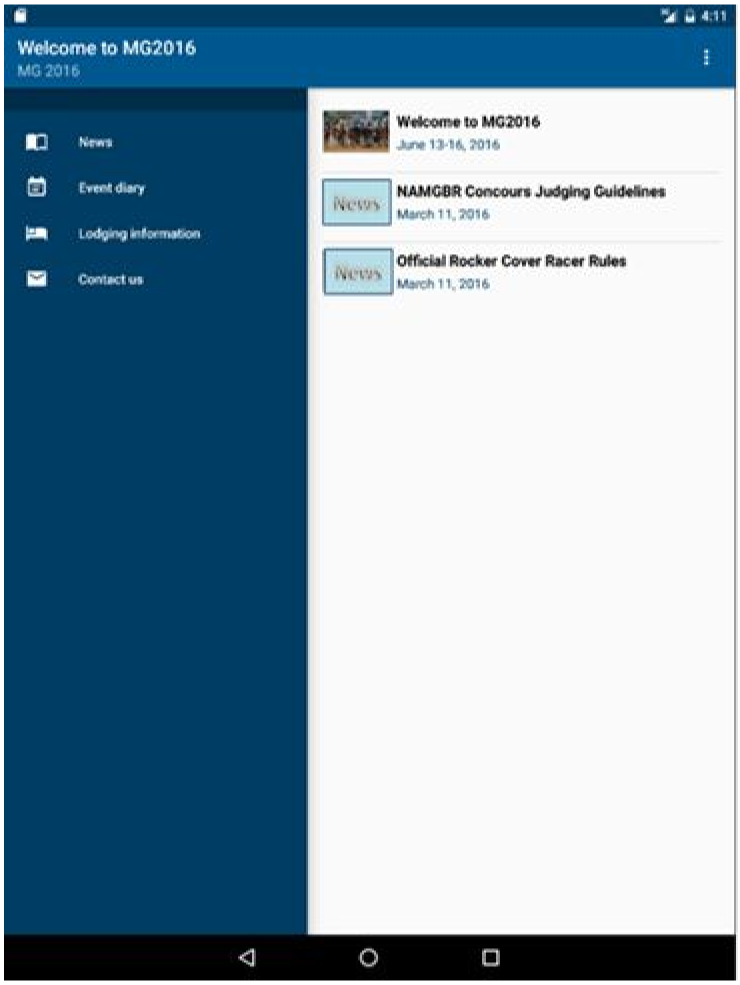


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vitae mollis ipsum. Aenean id odio nisi. Proin suscipit efficitur vehicula. Proin laoreet ipsum non tincidunt dapibus. Nam ultricies justo in libero aliquet tempor. Phasellus congue eget arcu eget tempus. Praesent sodales turpis ac velit ultricies venenatis. Fusce vehicula in sapien eget mattis. Quisque pulvinar sem at nisl varius, sit amet tempor tellus molestie. Quisque eget nisi nulla. Nullam ullamcorper dui a justo pharetra imperdiet. Curabitur elementum erat ipsum, molestie tempus ex euismod quis.

Nullam sodales mi eu dui pulvinar luctus. Aenean ut nulla venenatis, semper diam eu, gravida nisl. Aenean vehicula enim in elit pulvinar suscipit. Donec

The contact us page shows a default image and information about the app. The floating action button is defined to compose an email and send it to the app support email account.

Screen 7: Tablet Layout - News Feed



Welcome to MG2016
MG 2016

Welcome to MG2016
June 13-16, 2016

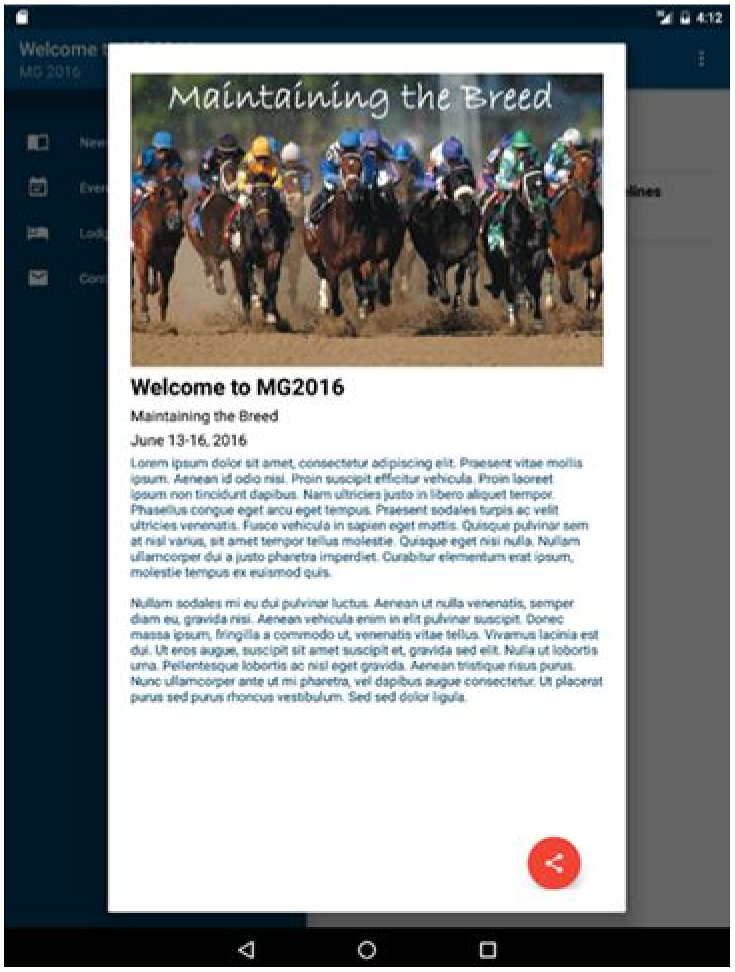
NAMGBR Concours Judging Guidelines
March 11, 2016

Official Rocker Cover Racer Rules
March 11, 2016

The tablet layout includes a fixed navigation pane and shows the news feed, event diary, lodging information or contact us pane based on the currently active selection.

When a detail page is selected such as a news article, a dialog fragment opens on top of the main activity panes as shown in the following screen.

Screen 8: Tablet Layout - News Detail



Welcome to MG2016

Maintaining the Breed

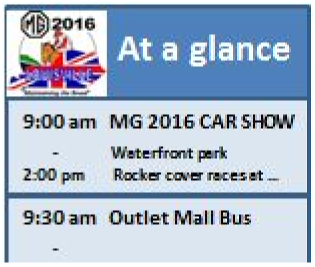
June 13-16, 2016

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vitae mollis ipsum. Aenean id odio nisi. Proin suscipit efficitur vehicula. Proin laoreet ipsum non tincidunt dapibus. Nam ultricies justo in libero aliquet tempor. Phasellus congue eget arcu eget tempus. Praesent sodales turpis ac velit ultricies venenatis. Fusce vehicula in sapien eget mattis. Quisque pulvinar sem at nisl varius, sit amet tempor tellus molestie. Quisque eget nisi nulla. Nullam ullamcorper dui a justo pharetra imperdiet. Curabitur elementum erat ipsum, molestie tempus ex euismod quis.

Nullam sodales mi eu dui pulvinar luctus. Aenean ut nulla venenatis, semper diam eu, gravida nisi. Aenean vehicula enim in elit pulvinar suscipit. Donec massa ipsum, fringilla a commodo ut, venenatis vitae tellus. Vivamus lacinia est dui. Ut eros augue, suscipit sit amet suscipit et, gravida sed elit. Nulla ut lobortis urna. Pellentesque lobortis ac nisl eget gravida. Aenean tristique risus purus. Nunc ullamcorper ante ut mi pharetra, vel dapibus augue consectetur. Ut placerat purus sed purus rhoncus vestibulum. Sed sed dolor ligula.

This screen shows how detail pagers are shown using dialog fragments. The dialog is closed by pressing the back button.

Screen 9: Widget



At a glance

9:00 am - 2:00 pm MG 2016 CAR SHOW
Waterfront park
Rocker cover races at ...

9:30 am - Outlet Mall Bus

The MG 2016 at a glance widget shows upcoming events in a collection widget. When a particular event is clicked, the app is launched and shows the events page.

Key Considerations

How will your app handle data persistence?

Since the content presented by the app may change over time, a server will be set up to deliver content to the app using http: JSON and image files.

Data will be cached locally by the app using SQLite and updated periodically using a SyncAdapter.

A Content Provider will be constructed to retrieve app data from the SQLite database.

Images will be fetched and cached during active user sessions using Picasso.

Describe any corner cases in the UX.

The UX will support a Navigation Drawer to switch between News, Events and Lodging information. When the drawer is used to switch between these pages in the app, transactions will not be added to the back stack since in this app, switching these pages is an example of horizontal navigation.

When detail pages are shown, the main page is added to the back stack and returned to when the detail page is exited using the back button.

A tablet UX is planned where the Navigation Drawer will be persistent and used to change the main page being viewed. Any detail pages will be shown using a dialog fragment. When the dialog fragment is opened, care is taken to ensure only one dialog is active at a time and any user touch/selections outside of the dialog fragment are handled gracefully.

Describe any libraries you'll be using and share your reasoning for including them.

Okhttp – simplify access when fetching remote http/JSON content
Picasso – handle the loading and caching of images

Next Steps: Required Tasks

The following section defines tasks used planned for completing the detailed design and build out of the app. At the completion of each major task, the app will be pushed to GitHub for version control.

Task 1: Project Setup

Complete detailed design of data model for information presented and managed by the app:

- News
- Events
- Lodging
- Contact
- Settings

Create new Android Studio project based on Navigation Drawer Activity.

Configure basic project build scripts and libraries:

- minSdkVersion 16
- target/compileSdkVersion 23
- include appcompat and design support libraries
- include squareup okhttp and picasso libraries

Configure project to use Google Maps API:

- include com.google.android.gms:play-services-maps:8.4.0

Configure project to use the Google Calendar Provider API (as required).

Task 2: Implement UI for Each Activity and Fragment

Initial styles and layouts will be created to define:

- AppCompat styles for activities and text
- Layouts for navigation drawer and main pages: News, Events, Lodging, Contact.
- Layouts for detail pages: News, Events, Lodging
- Alternate layouts for tablet views/dimensions

The basic UI handling will be implemented in java code for the following Activities and Fragments:

- SplashScreen

- MainActivity: to implement navigation drawer and fragment handling to show the different pages within the app
- NewsListFragment
- NewsListAdapter
- NewsDetailFragment, includes a share action provider to share the news article with friends
- EventListPager: a fragment that manages the daily events as a ViewPager since each day will be shown as a separate, swipe-able page.
- EventListFragment, including a share action provider that is active when a specific event is selected
- EventListAdapter
- LodgingListFragment
- LodgingListAdapter
- ContactUsFragment

Dummy data will be populated in the fragments to test drive the application and validate UI look and feel with various handsets and tablets.

Task 3: Implement Content Server

A simple content server will be set up with Dropbox or Amazon Web Services and configured to deliver JSON content and image assets over http.

JSON content will include:
- News feed
- Events feed
- Lodging information
- Contact information

Image assets will include:
- News images
- Lodging logos

Basic testing will be performed using a web browser to ensure content can be retrieved successfully.

Task 4: Set up the SQLite Database

SQLite will be used to persist data fetched from the content server. The remainder of this section provides details for subtasks required to set up the database.

Create DatabaseContract class that defines the database tables, columns and Uri schemes for:
- Storing News feed
- Retrieving News feed, ordered by publish date and time
- Storing Events feed
- Retrieving Events feed by day, ordered by start time
- Storing Lodging information
- Retrieving Lodging information, ordered by hotel name
- Storing and retrieving Contact information

Create DatabaseHelper class that extends SQLiteOpenHelper to:
- OnCreate() to create News feed, Events feed, Lodging information and Contact information tables
- OnUpgrade() to drop any existing tables and create new tables

Create and execute unit test cases to check Uri schemes work properly.

Task 5: Implement Content Provider

A content provider is implemented to abstract how data is accessed by the app and involves the following subtasks:

An AppContentProver class is implemented that extends ContentProvder to:
- Implement Uri handling and operations to store News, Events, Lodging and Contact information
- Implement Uri handling and operations to retrieve News, ordered by publish date and time
- Implement Uri handling and operations to retrieve Events, ordered by start time
- Implement Uri handling and operations to retrieve Lodging information, ordered by hotel name
- Implement Uri handling and operations to retrieve Contact information

Create and execute connected test cases to check that dummy data can be stored and retrieved successfully from each table.

Task 6: Implement JSON Client

This task implements the processing required to pull JSON content from the server. The subtasks include:

A class named AppJSONClient will be constructed that uses the okhttp library to pull down the text/JSON content:

- News feed
- Events feed
- Lodging information
- Contact information

Create and execute unit test cases to pull data from the server and successfully load the data into JSON structures.

Task 7: Implement Sync Adapter

This task implements a sync adapter to pull content from the server at periodic intervals:

AppAuthenticator and AppAuthenticatorService are implemented as they are required by the sync adapter framework. In this case, no authentication is required by the app so these are essentially stub authenticator objects.

The next subtask implements the AppSyncAdapter class by extending AbstractThreadedSyncAdapter:

- Defines synchronization strategy and parameters
- Leverages the AppJSONClient to pull content from the server
- Leverages the AppContentProvider to store/update content received from the server

Next, the AppSyncService class is defined that instantiates an AppSyncAdapter object and supports binding to and running the sync adapter.

Finally, connected test case is defined to exercise the AppSyncAdapter class to validate correct operation.

Task 8: Tie Things Together

This task ties the UI built in Task 2 with the Content Provider built in Task 5 and the Sync Adapter built in Task 7. The following subtasks are involved in this step:

The application is temporarily updated to instantiate and run an AppSyncAdapter object on startup to populate the Content Provider.

The UI adapters and fragments are converted to implement loaders that pull content from the Content Provider:

- NewsListAdapter
- EventListAdapter
- LodgingListAdapter

- ContactUsfragment

Next the adapters are updated to pull images from URLs received from the Content Provider using Picasso:
- NewsListAdapter
- EventListAdapter
- LodgingListAdapter

Now that the UI is basically complete, testing is performed to validate correct operation and appearance:
- Different content and possible exceptions are tried such as missing content or unavailable images.
- Different devices types and sizes are tried to double-check the UI and test proper handling of orientation changes

The final subtask removes the AppSyncAdapter object instantiation and to update the app manifest to declare and run the AppSyncService. At this point the app is retested again for proper operation and to configure the AppSyncAdapter to check that it runs properly with different intervals and sync strategies before setting it to more appropriate, final values.

Task 9: Implement Map Functionality

This task implements mapping functionality in the Events Feed and Lodging information to show the user where they are at in relation to the event or hotel venue and to also support a navigate action. The following subtasks are planned to implement this functionality:

API keys are generated for the GoogleMapsAPI and build.gradle is updated to include the key from the “external” local user gradle properties file to ensure the key is not accidentally shared in the GitHub repo.

The app manifest is updated to use ACCESS_FINE_LOCATION to support showing where the user is at in relation to the destination.

The app is updated to define the layout for the map activity and the ShowMapFragment is implemented using the Google Maps Android API v2 Samples as a guide.

Finally, buttons to launch the ShowMapFragment within the MainActivity are added to the existing UI and the new UI and map functionality are tested for correct behavior.

Task 10: Implement Calendar Export

When an event is selected in the Event feed, a button is enabled that allows the user to export the event to their calendar. This task implements a share intent that will invoke the Calendar app to store the event to the calendar:

The EventListFragment is updated to show a “calendar action” button when an event is selected in the Event feed.

The action button on click handler is defined to create and send an intent to the user’s calendar to complete the action.

Testing is performed to validate that selected events are successfully stored in the user’s Calendar.

Task 11: Implement Widget

This task implements the “MG 2016 at a Glance” widget that presents upcoming events to the user. The following subtasks are required to build out and test the app widget:

Widget layout and XML configuration files are created to define the widget appearance and properties.

The AppWidgetProvdier class is implemented to support RemoteViews, onReceive() and the other widget lifecycle methods.

The widget configuration is added to the android manifest and the widget appearance and operation is tested.

Once the final look and feel of the widget is complete, a snapshot is taken and used as the image for the widget preview and saved as an application resource.

Task 12: Polish the UI

At this point the app is built on the AppCompat library and has default material components and animation. This task involves checking and polishing the following aspects of the app:

Elevation: check and update elevations as needed follow material design guidelines for each page and view presented by the app.

Animation: check and update transitions as needed to improve the flow of the application as the user navigates between pages and elements within the app. This subtask also looks at list item selection to make sure items are properly highlighted when selected.

Accessibility: at this point content descriptions are double-checked (especially for images) and then the application is tested and updated as needed based on how well the content flows when accessibility options are enabled.

RTL: the UI is tested in RTL mode and after any required changes are made, the app is then tested in normal, LTR mode for final validation.

Task 13: Finalize App

At this point the app is functionally complete and is readied for submission:

Update build.gradle to create a signed configuration. The signed build is created and validated for proper operation.

README.md is updated to document the application and any information the user needs to know when building or executing the app such as how to define the Google Maps API keys in their local gradle properties file.