Contents

Personal Android Project Milestone 3 Rubric

Overview

Content

This is the final deliverable of the personal Android project. This milestone is cumulative: It include elements delivered in earlier milestones, but these should be updated as necessary, and they won't be weighted as heavily in the grading. (**Note**: If these elements from earlier milestones have not been updated, and do not correspond to the application in its current state, you will not get full credit for them.) Thus, the content for this milestone is expected to include project summary elements, intended user identification & stories, data model documentation, data model implementation, connections to external data sources or other services, and a functioning user interface.

Deadline

The early deadline is 19 November, 2019 @ 8:00 AM MST. Submissions received by that deadline will receive a 10% grading bonus for the final milestone.

The normal project deadline is 20 November, 2019 @ 8:00 AM MST. Submissions after that deadline are subject to a grading penalty of up to 10% per full or partial day late.

Grading eligibility

Important: We can only grade you on what you've pushed to GitHub. All code, project metadata, and documentation (with the exception of sensitive content) must therefore be committed to Git and pushed to GitHub, with the associated URLs (for the repository and its GitHub Pages web site) posted as directed to https://learn.cnm.edu. (While this element is not worth any points by itself, without it you can't earn any of the points below.) If the version of the code you want to submit for grading is not the one most recently committed, then post a link to the specific commit to be graded; on the other hand, any commits made after the deadline will be ignored in the grading (unless the student wants to accept the late penalty).

Sensitive content

Important: Sensitive content needed for building and running your app—e.g. API keys or other credentials for external services—**must not** be committed to your Git repository and pushed to GitHub. Instead, files containing such content should be submitted in https://learn.cnm.edu, along with your GitHub repository URL. (Be sure to include any instructions for incorporating this content into the app in your documentation for building or using your app.) If you don't know whether some key or credential used in your project constitute sensitive content, please discuss it with the instructors *before* committing it to your Git repository. (Please **do not assume** that committing a file containing sensitive content and then deleting that file later will make the file inaccessible on GitHub; it won't—even if that deletion occurs before the first time you push to GitHub.)

Grade value

The nominal total value of this deliverable is 100 points; in addition, up to 10 points of extra credit can be earned for early submission, and up to 10 more points can be earned for exemplary execution of any or all of the required elements.

Graded elements

All of the items listed below must be accessible—directly or indirectly—from the project's GitHub Pages web site. In other words, it should all be reachable by starting from the web page at `https://{username}.github.io/{project-repo-name}` (that URL must be included in your final submission on https://learn.cnm.edu), reading that page, and following links from that page. **Note**: This requirement includes the Android project repository itself, to allow the instructors to clone and build your project.

Documentation: 44 points

Project introduction & description: 4 points

This should consist of at least 2 paragraphs (please use complete sentences and proper grammar):

- 1 or more paragraphs summarizing the project topic in general terms, with an overview of intended functionality.
- 1 or more paragraphs summarizing your aims or motivations for selecting the given topic for your app. That is, why did you choose to develop this particular app, and why is this (at least potentially) a useful or interesting app?

Intended users: 2 points

This may be given as a paragraph (or more) of descriptive text, or a bullet list. Be as specific as possible—for example, something along the lines of "Anyone who likes games" is not sufficient.

User stories: 2 points

On a separate page (reachable via a link from the main page of the GitHub Pages site), provide user stories for at least two user *personas*, in the form of a bullet list. Make these specific, but not excessively detailed; 1 or 2 sentences per persona is generally appropriate.

Summary of current state of the app: 4 points

In grading a prototype, understanding what is does and doesn't (yet) do is important preparation for evaluating it in operation. Please provide the following:

- A description of the current state of completion/readiness of your app. This should include a "hit list" of deficiencies: any unimplemented/incomplete elements, and known bugs, that would have to be implemented or corrected for a usable prototype (i.e. one that could be given to a skilled user for testing and feedback), ordered with the most urgent items first.
- A list of aesthetic/cosmetic (not functional) improvements that you think would improve your app. This list should be ordered, with those that would give the most improvement (in your opinion) listed first.
- A list of functional stretch goals. These should be sorted either with those that would add the most utility at the top, or with those that would be the simplest to implement at the top.

Wireframe diagram: 2 points

To the extent that the screens in your wireframe are implemented in your app, the diagram should correspond closely to the implementation. On the other hand, screens in the wireframe that are not yet implemented in the app should be identified as such, by annotations in the diagram.

ERD: 4 points

This should be an **up-to-date** physical ERD, showing all entities, attributes, and relationships.

- The ERD must be displayed as a single PNG, in a page dedicated to this purpose, accessible via links from the main page of your GitHub pages site. This PNG image should be a clickable link that opens a PDF form of the same image.

- Each entity in the ERD must be named using `UpperCamelCase`, and must include (at least) a primary key attribute.
- Each attribute must have:
  - name, in `lower_snake_case`;
  - data type (either as a Java type or a SQL type—but be consistent);
  - any relevant primary key (`PK`), foreign key (`FK`), unique constraint (`UQ`), and/or index (`IX`) indicators (the last 3 should be numbered—e.g. `FK1`, `IX1`, `UQ1`, etc.);
  - nullability indicator, if the attribute may have a `null` value (use either `N`, `null`, `NULL`, or `Null`).
- All relationship lines must include the appropriate crow's foot symbol at each end.
- For a many-to-many relationship, the join entity **must** be included in the ERD, since a corresponding Room entity class must be defined in the data model implementation.

DDL: 4 points

- Your repository must have an **up-to-date** `docs/ddl.sql` file, containing all of the SQL statements defining the tables, primary keys, foreign keys, unique constraints, and other indices declared in your entity classes.
- The contents of `docs/ddl.sql` must either be formatted manually (following https://www.sqlstyle.guide) or formatted via the IntelliJ **Code/Reformat** feature. (If you choose the latter option, don't forget to set the dialect of the file to SQLite, and use semicolons after each SQL statement; otherwise, the formatting will not work as expected.)
- Additionally, your repository must contain a `docs/ddl.md` file. This file must contain a *fenced code block*, showing the same content as your DDL file.
- The `docs/ddl.md` file must be linked to from the main data model documentation page, and must itself link to the `docs/ddl.sql` file.

Technical requirements & dependencies: 4 points

Your project documentation must include the following:

- A list of Android API versions and hardware (including emulators) on which you've tested the submitted version of your app, the minimum Android API required, and any other hardware/software/orientation restrictions that you're aware of. (This includes restrictions on device language, orientation, etc.)

- A list of the 3rd-party libraries (i.e. anything beyond the Android standard and support libraries) used by your app.
- A list of the external services (including Google services such as Sign In, Calendar, Maps, etc.) consumed by your app.

Javadoc-generated technical documentation: 6 points

The project must include Javadoc documentation. Please keep in mind that including this requires 4 distinct, high-level actions:

- Add Javadoc comments to all public classes, and all public and protected members (fields, constructors, methods, and nested classes and enums) of those classes. The only allowable exceptions to this are methods with the `@Override` annotation, which do not *necessarily* require additional comments.
- Use the appropriate IntelliJ, Ant, or direct (command line) `javadoc` commands and options to generate Javadoc HTML/CSS/JavaScript artifacts from the Javadoc comments. (These should be generated into a `docs/api` subdirectory of your project directory. Also, don't forget to verify that *all* of the Java files in your project—aside from test classes—are included in this generation step; it's easy to overlook a default selection by IntelliJ that does not include all of your files. Finally, remember that for an Android project, you will typically need to include `-bootclasspath` and `-linkoffline` options to resolve references to the Android standard library in the generated documentation.)
- Add the generated files to your Git repository.
- Commit and push the generated files to GitHub.

Copyrights & licenses: 4 points

The project must include copyright statements and license information—for your app itself (you may optionally choose an open source license for your app; this is not required), as well as for *all* relevant 3rd-party libraries.

Note that it is **not** sufficient simply to include a link to the license information on each library's publisher's site. Most open source licenses specifically state that software incorporating the licensed software must include copyright and license information in the documentation, include the relevant license statements in the source code repository, display copyright & license info in a screen displayed by executable software, etc.

Build instructions: 4 points

These should include the steps required for cloning/downloading the repository, importing the project in Android Studio/IntelliJ IDEA, synchronizing the Gradle build, and executing the build. If any content not included in the repository (e.g. an API key) must be incorporated for a successful build, instructions for doing so must be included.

Basic user instructions: 4 points

In writing these instructions (another Markdown file), it should be assumed that the user is reasonably experienced with the Android OS, familiar with most standard navigational devices (e.g. navigation drawer, action bar, overflow menu, back button). The user instructions should thus focus primarily on app-specific operations for exercising the implemented functionality.

Implementation components: 56 points

In order to grade the app itself, the instructors must be able to build the app, install it on a physical device or an emulator, and run it. In doing so, the build instructions and basic user instructions provided in the documentation will be followed. Of course, when following those instructions, if there are serious, failures, this can prevent the evaluation of the app's functionality and technical elements. For this reason, the steps of building, installing, and launching the app—like committing and pushing the project repository to GitHub—have no direct point values associated with them; but any issues encountered here will almost certainly affect the grade indirectly.

Multiple activities and/or multiple fragments, with rational stack management: 10 points

In navigating through your app, does the back/up button work in a sensible manner with the rest of the navigational elements (e.g. bottom button navigation, navigation drawer, action bar items)?

For additional information on general navigation principles, see the "Designing effective navigation" section of the "User Interface & Navigation" developer guide.

Shared preferences for maintaining application configuration/profile data: 6 points

See Shared preferences for more information. (This requirement may be satisfied through the use of SQLite, but shared preferences are preferred for this purpose.)

Use of the SQLite database, either directly via the SQLite library or via the Room ORM persistence library, for the app's data store: 10 points

For an overview of SQLite use in Android apps, see the "Databases" section of the "App data and files" developer guide.

Dynamic, database-driven UI container components (e.g. `ListView`, `RecyclerView`, `ViewPager`, `GridView`): 10 points

Virtually all apps that make use of a data store such as SQLite or Firebase will need to provide the user with views of the stored data. For some purposes, looking at a single record at a time is sufficient; however, letting the user view multiple records at once (e.g. in a scrolling list or grid) is generally an important feature. Even an app that uses the tabbed activity with the swipe left/right navigation style will typically need to construct the set of available pages dynamically, by querying the database.

For credit in this category, your app must use one or more instances of data-driven view groups such as `ListView`, `RecyclerView`, `ViewPager`, `GridView`, or some functionally equivalent alternative (possibly a subclass of one of the above), populated by data from the data store, to present some portion of the app's data to the user.

Consumption of external services: 10 points

While the standalone Android project requirements do not include (and in fact, specifically consider out-of-scope) a server-side development component, it is very common for an Android app to consume data from, or otherwise integrate with, existing web services, and doing so *is* part of the grading rubric. This might take the form of integrating Google Maps with your app, implementing a component in your app that consumes relevant data from an outside service, etc.

(If the app doesn't consume any additional external data or services in real time, a data pre-load, using data obtained from an external source, may be used to satisfy a portion of this requirement. Google Sign In may also be used to satisfy a portion of this requirement. However, neither of these, on its own, will earn full credit in this grading category.)

Custom drawable assets for launcher icons, as well as in-app drawables (as appropriate): 4 points

Don't forget to include launcher icons (mipmap assets) in your app! Additionally, drawable assets are typically used in bottom navigation buttons, and sometimes in action bar items and navigation drawer items. Of course, games will often use drawables extensively. For all of these, keep in mind that some kinds of drawables work much better than others for certain purposes. For example, bottom navigation buttons, navigation drawer items, and action bar items will generally need vector assets, rather than raster (image) assets.

Code quality: 6 points

For this portion of the grade, the instructors will assess the overall code quality—primarily in terms of conformance to the Google Style Guide and code clarity, but also examining encapsulation, degree of cohesion within classes and packages, degree of coupling between classes, occurrence of repeated code, organization of packages, classes, interfaces, etc.

Extra credit: 10 points

In addition to the extra credit opportunity for early delivery (see "Deadline"), up to 10 extra credit points may be awarded, at the instructors' discretion, for exemplary execution of some or all of the project components. These points may be awarded for virtually any aspect of the project—outstanding documentation, code quality, UI design, data design, etc.