IBL2 3303 MOBILE APPLICATIONS DEVELOPMENT

PROJECT DESCRIPTION

Total: 70 points

The team project (3 students per team) for this class involves conceptualizing, designing, and developing a mobile application on the Android or iOS SDK. The application should meet the set of functional and non-functional requirements described below. Teams should follow a well-managed development process that results in well-designed, well-documented software.

High-Level Requirements

We discussed in class that a mobile device serves as two things:
(a) It serves as your gateway to everything else; and
(b) It represents ***you***. Thus, the app that you conceptualize must be ***you***, represent ***you***, and be unique to ***you***.
In addition, the app must meet the following requirements:

- Must ***uniquely*** meet a clear need;
- Must be well-designed (by following the process outlined in the course);
- Must have a UI (i.e., it cannot ***only*** be a background service);
- Must have at least three domain objects;
- Must have data that persists throughout multiple user sessions;
- Must use at least one Internet-based service (e.g., Facebook, Firebase, or Google Maps);
- Must use at least one device sensor (e.g., GPS, accelerometer, or light sensor).

Note: All student teams in a class section ***must*** design ***original distinct apps***. There should ***not*** be multiple team apps that meet the same need (e.g., multiple apps that "find events near me" using GPS, Google Maps, and Facebook). I expect that each team designs and develops an original app that is distinct from apps already available on the market as well as other teams' apps in class.
Since the project entails designing and implementing an original app that represents **you**, use this opportunity to showcase your creativity in your design. Your app may be a game that meets the above requirements and originality considerations.

Stages of Evaluation

Checkpoint Grading Policies: Project Checkpoint 1 is graded ***individually***; Checkpoints 2–5 are graded ***as a group***. If team members do ***not*** show up to their meetings with the grader, they receive 90% of their group's grade (for their /individual/ grades). (For example, if Alice and Bob are in group that receives 20/20 for Checkpoint 5 but Alice does not show up to the meeting, then she receives 18/20; Bob receives 20/20.)

Checkpoint (1): Android Environment Setup and Running Sample App
Due Date:
Points: 5/70
Submit: Screenshots of Tic-Tac-Toe app running from the Android Studio IDE (on an Android mobile device or your emulator).
Place screenshots in a .zip file that you send to my e-mail.
Present to Grader: N/A
Evaluation Criteria: The app executes via the Android emulator (or on an Android device).

Checkpoint (2): App Requirements and Design
This checkpoint entails envisioning and designing a mobile app that meets the requirements specified in the "High-Level Requirements" section.
Due Date: Due Date:
Points: 5/70
Submit: ***A SINGLE design document*** that shows:
– Notes of your app conception, specifically:
–Narratives (one or two paragraphs describing how someone uses your app and why ***your*** app is necessary to achieve certain goals);
–A UML class diagram of your app's domain model designed following the object-oriented approach discussed in class;
– A ***categorized*** list of use cases describing users' interactions with the app (example categories can include "Account Management," "Mapping," and so on);
– Your app's ***relational model*** that describes tables and relations in your relational database schema (you need to include the relational model in your document, not *just* an ER diagram);

–Sketches (drawings of people using the app as you envision, how the app connects to external services, and so on);
–Screen layout mockups along with screen flows showing how clicking different UI elements navigate the user among mocked-up screen layouts;
– The external (Internet) service your app will use; and
–The device sensor your app will use.
Note: Your screen layout mockups, screen flows, and sketches may be drawn by hand, but they ***MUST be legible.***
1As a ***domain model***, your classes should be based on candidate nouns from your narrative.
Present to grader: Your app design document with the information above. You should explain your app to the grader during your group's meeting with him/her.
Evaluation Criteria: Clear evidence of the object-oriented design process.

Checkpoint (3): Installing, Debugging, Profiling, App Lifecycle Management, Logging
This checkpoint involves demonstrating your ability to do the following:
– Installing your app on a device;
– Debugging your app (e.g., stepping through code and setting breakpoints);
– Profiling your app using one or more profilers (e.g., Android Studio's CPU, GPU, or memory profilers or Instruments for iOS);
– The app lifecycle and logging. Here you must implement a simple part of your app and demonstrate using the Android log that you can trigger the app's lifecycle methods in Activities (e.g., onPause() and onResume()) and Fragments (e.g., onCreateView()).
Due Date: Due Date:
Points: 10/70
Submit:
– A screenshot of at least one type of profiler (e.g., CPU or memory footprint in Android Studio or Xcode);
– Screenshots of the app's lifecycle methods being triggered;

– Screenshots of debugging your app in Android Studio or Xcode (specifically, setting a breakpoint, viewing variables' states, and stepping through code).
– Your app project as a .zip file.
Present to grader:
– A mobile device running a recent version of Android or iOS (i.e., Android 6+ or iOS 10+);
– Installing your app on a device;
– Debugging your app (setting breakpoints and stepping through code);
– Profiling your app;
– Invocation and logging of lifecycle methods.
Evaluation Criteria:
– Students demonstrate understanding of IDE use;
– Students demonstrate comprehension of the Android/iOS app lifecycle.

Checkpoint (4): Review of Your App Architecture and Data Persistence
For this checkpoint, you need to show that you have setup your app's persistent data storage (either on the mobile device or using a cloud service such as Firebase).
Your app should be able to perform data creation, retrieval, update, and deletion (CRUD) operations with your data store.
Note that if you use a NoSQL external service like Firebase, you ***need*** to denormalize the relational data model that you developed for Checkpoint 2.
2. Your team should have setup a source code version control system for your project (e.g., a GitHub repository).
Though incomplete, your code should adhere to object-oriented design principles such as the single responsibility principle and separation of concerns.
3 You ***do not*** have to implement all use cases for your functional requirements, but you need to show evidence of progress toward this goal.
Due: Due Date:
Points: 10/70
Submit:
– Screenshots of your app's data storage schema (such as your app's SQLiteOpenHelper class for on-device storage or your schema for an external NoSQL service like Firebase);

– Screenshots of your app's class organization (e.g., Android Studio's "Project" pane);
– Screenshots of a version control repository for your app's source code; and
– All your code (just export the project as a .zip file).
Show to Grader:
– Your app's data storage schema (as described previously);
– Your app's ability to perform creation, retrieval, updates, and deletions (CRUD) with your data store (an on-device relational database or an external service such as Firebase);
– The version control repository for your app; and
– The layout of your app's classes.
Evaluation Criteria:
– Students demonstrate implementation of their app's data storage schema (using a relational or NoSQL database);
– Students demonstrate that their app can perform CRUD operations on stored data;
– Students demonstrate use of a version control system; and
– Students demonstrate that their app implementation of cohesive, loosely-coupled classes.

Checkpoint (5): Functional Demonstration of Your App
For this checkpoint, you have to demonstrate a functioning app. The app should work from beginning to end, although it ***does not*** need to be optimized for performance or resilient to failures such as loss of network connectivity, GPS signal reception, and so on.
Due:
Points: 20/70
Submit:
– A list of use cases (*only* a list, no descriptions)
– Screenshots of your app running
– All code (simply export the project as a .zip file)
Show to Grader: Your app working on a device. (If it works only on an emulator, you will only receive partial credit.)
Evaluation Criteria: TA will check off working use cases.

Checkpoint (6): Demonstration of Your App's Non-Functional Aspects
This checkpoint entails showing your app's non-functional capabilities such as usability, performance, availability, maintainability, modifiability, and scalability. The app should work from beginning to end, should be optimized for performance, and be resilient to failures (such as loss of network connectivity and GPS signal reception, screen rotation, and termination by the OS).
Non-functional requirements (NFRs) should be app-specific and (ideally) quantified.
The minimum requirements are as follows:
– Address at least one performance NFR and demonstrate improvements using "before and after" profiler snapshots;
– Address at least one other NFR (e.g., increased app security, design enhancements, accessibility support for users with disabilities, or localization in another language)
– Perform unit testing using an Android or iOS test framework (such as JUnit/Espresso or Xcode tests)
Due: Due Date:
Points: 20/70
Submit:
– List of use cases and NON-functional requirements met
– Screenshots of the app
– Profiler screenshots showing areas where performance was improved as well as the "baseline" performance before improvement (i.e., "before" and "after" screenshots of CPU, GPU, or memory consumption showing decreased consumption)
– All code (simply export the project as a .zip file).
Show to Grader: The working app on a device demonstrated to work under failures (network connectivity, GPS, screen rotation)
Evaluation Criteria: Number and quality of working use cases and NFRs met.

Final Report
Due: Due Date:
The final report for this class will be a team report that analyzes and critiques the design process described in class.

The report should consist of the following three sections:
– Describe the design process and its intended goals in your own words.
– Describe how you translated the design into the implementation of the app. In this section, describe what worked well, and where there were gaps in the process. For these gaps, explain how you bridged them.
– Suggest changes or improvements in the design process that you believe would make the design more complete and more easily translatable into the implementation.
The report should be 4–5 pages (single-spaced) written in 10–12 point font. Sample report templates are provided.

Report Expectations
– References:
You are responsible for citing any third-party intellectual property you use in the project.
If you build an Android app, you should include an Android reference such as:
[1] Android Open Source Project, Android, http://developers.android.com with an inline citation such as [1].
If your group uses Firebase for cloud storage, you should include a Firebase reference:
[2] Firebase, https://firebase.google.com/ with an inline citation such as [2].
Repeat this process for each third-party work you use, including external (code) libraries. Reports without references will receive lower grades than those with references.
– Clear Writing: Your writing should be simple, concise, and easy to understand; if not, revise it! Your writing should demonstrate a logical sequence of thought; spelling and grammar errors should be absent. Most word processing tools (including online ones such as Google Docs) have built-in spelling and grammar checkers; ***please use them!***5
Prospective employers and graduate schools expect clear writing.
– Figures: You are welcome to include figures in your report. Please cite them as “Figure 1,”

“Figure 2,” and so on. (The report should contain 4–5 pages of text *independent* of your figures.) Hand-drawn figures are ***not* allowed** in the final report; please draw them with software such as MS Visio, OmniGraffle, Dia, or http://draw.io (online).

1 As a ***domain model***, your classes should be based on candidate nouns from your narrative.

2 See https://firebase.googleblog.com/2013/04/denormalizing-your-data-is-normal.html for an example with Firebase. Firebase programming entails concurrent programming in Java (e.g., Cloud Firestore requires asynchronous Futures). *Java Concurrency in Practice* and Jenkov’s Java concurrency tutorial are excellent resources.

3 See https://en.wikipedia.org/wiki/Single_responsibility_principle, https://en.wikipedia.org/wiki/SOLID, and https://en.wikipedia.org/wiki/Separation_of_concerns for respective details.

4 If you have trouble formulating NFRs, recall that users expect low app latencies (under 100 msec for most tasks).

Consider storing fetched data via caches (in-memory, disk, or both). Cached data are much faster to retrieve than data stored online; see https://medium.com/@jorgemf/making-your-android-app-faster-735328eaba25.

5 You may find free online checkers (such as LanguageTool and Sapling) helpful