CS31620 Assignment Report: Native Mobile Quiz App

by Matthew Hibbin
mjh32@aber.ac.uk
10/01/2022

CONTENTS

1 Introduction

The problem presented in this assignment was the need for a mobile application in a school setting that would allow teachers to create quizzes for students revising for exams. Staff of the school would be able to create question banks containing multiple questions that students could use to test their knowledge. This assignment required a prototype of this application to be created that would omit any security features and use a locally stored SQLite database. The prototype therefore would only need to demonstrate the creation and deletion of question banks, adding and deleting questions within a question bank and attempting questions within a question bank with the following result being displayed.

The sections in this report will go over the different aspects the application's construction. The UI Design section will compare a non-functional UI prototype to the application's final design. The Software Design Section will compare a UML Class Diagram, constructed before the application's classes were complete, to the final software design. The Testing section will go over a series of tests done to show the operating condition of the final application. Finally, the Reflection section will go over the successes and downfalls of the application's construction as well as additional features, learnt skills that will aid with the major project and a conclusion.

2 UI Design

In order to create the application's UI, a non-functional UI prototype was made to be a rough model of what the final application will look like (see reference 0). The prototype was constructed in PowerPoint as a set of linked screens that demonstrated how a user would interact with the application. The design of the UI elements within the prototype were based off Android Material Design Components. This included the navigation drawer, text edit, text view, button, floating action button, list view, menu dialog, radio buttons, progress bar and bottom sheet.

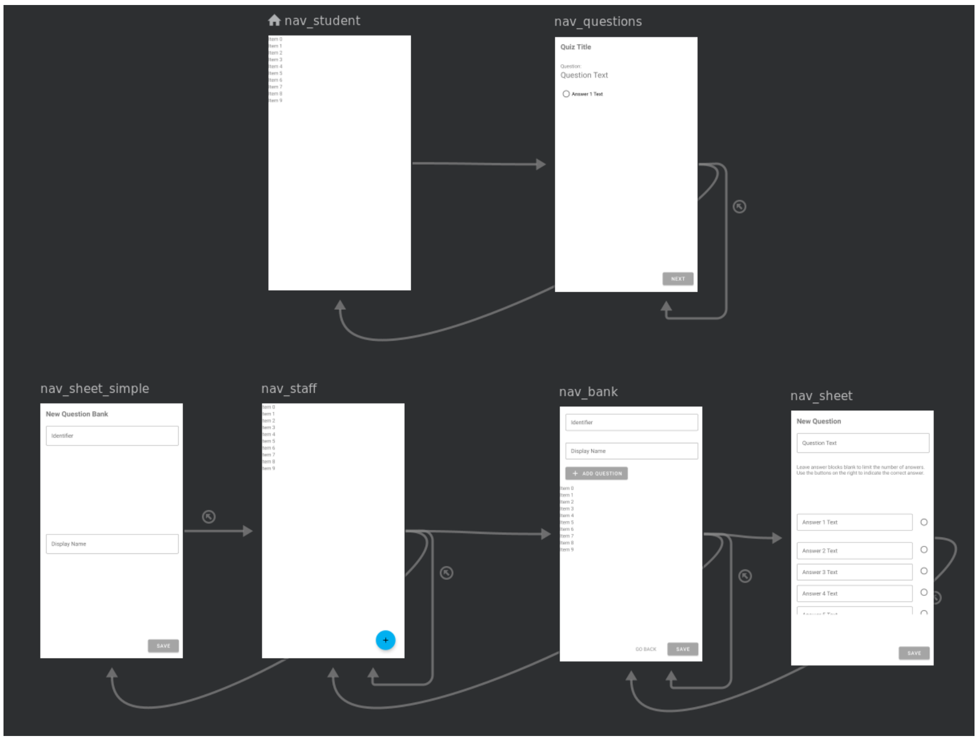
The prototype has an introduction screen that allows the user to enter either as a student or staff member. The user could return to the introduction screen by using the navigation drawer and selecting to log-out. When logged in as a student, the user is able to select a quiz from the list view in order to complete the questions it contains. A quiz represents a question bank that contains one or more questions. When taking a quiz, the user will be presented with a question and multiple radio buttons representing the possible answers. Only one answer can be selected as the user's chosen answer and will be saved once the next button is clicked for the next question to be displayed. Once all questions are complete, the user will be shown the resulting score of all correct answers both in text form and within a progress bar.

If the user were to select the staff login instead, they would again be presented with a list view containing question banks but instead of being clickable, they have a menu dialog. Also present is a floating action button in the bottom right corner which shows a bottom sheet dialog to create a new question bank. Interacting with the menu dialog for a question bank, will present an option to either edit or delete the question bank. If edit is selected then a new screen is presented with areas to edit the identifier, display name and manage the questions. A new question can be added by clicking the corresponding button, showing a bottom sheet to enter the question's details. Already created questions can be edited in the list view by clicking on them, revealing all of its editable contents. The prototype also contains possible flair features, such as the search button and menu dialog to change the theme, both located in the action bar.

Comparing the UI design within the prototype to the final application, the UI remains largely the same. Any changes made were due to either ease of use or for increased efficiency. The main change was omitting the introduction screen and instead having two options in the navigation drawer, quiz mode and question banks. All list views in the prototype were replaced by recycler views that used interactive cards to represent their data. The cards in quiz mode are completely clickable while the cards in question banks contain an edit and delete button. Another change from the prototype, the bottom sheet used for adding a new question is now used to edit a question as well in order to be more consistent and re-use resources. Due to the functional requirements of the application, the ability to select different question types was removed. The application was also given a secondary UI colour so elements like the floating action button are more easily visible. The two action bar flair features were also omitted due to redundancy (the user can change the theme on their phone system wide). For screenshots of the final UI, see reference 1.

3 Software Design

In order to quickly start the application's development, Android Studio's navigation drawer template was used. This template included a main activity with a working navigation drawer that navigated to three empty fragments. One fragment was removed and the other two were refactored as the student and staff fragment. With the aid from resources in the references section, a UML class diagram was constructed in order represent the Kotlin classes within the application (see reference 2). The diagram shows that the student, staff, questions and bank fragment all represent the main part of the program. The student fragment is the home fragment and only contains a recycler view that allows you to select a quiz to take. Once a quiz is selected, the user is navigated to the question fragment which allows them to take a quiz and displays the result. The staff fragment is accessed via the navigation drawer and contains a recycler view listing all the question banks. If the user chooses to edit a question bank, the application navigates to the bank fragment where all the attributes of the bank can be changed. The below image shows the navigation graph of the application.



In the navigation graph there is also the bottom sheet and simple bottom sheet dialogues. The bottom sheet is used with the bank fragment to either add or edit a question while the simple bottom sheet is used with the staff fragment to create a new question bank. The UML class diagram shows just one class for the bottom sheet but this was changed to two classes for each dialog as they differ enough to warrant their own classes.

The recycler adaptor class is used by the student, staff and bank fragment to display a list of data the user can interact with. The SQL manager class however, plays a more vital role as it allows each fragment, dialog and the recycler adaptor to interact with the local SQLite database. It contains methods to add, edit, delete and get both question banks and questions. The values tracker class also handles data as it's used to safely manage the application's global variables. These variables allow fragments to know which question bank or question they should be getting data from. The class is also able to tell whether a question is being added or edited within a bank and also keeps track of the correct answers when taking a quiz. The two final classes worth mentioning are the question view model and bank view model. These are data classes used by the SQL manager to return multiple variables of different types from one function.

4 Testing

The following test table was created to test multiple aspects of the application. Each test covers a functional requirement outlined in the assignment brief. There are also extra tests for flair features. See reference 3 for screenshots of each test.

Test Name	Test Description	Result	Passed
FR1: Create new bank	Create a new question bank	Question bank created	PASS
FR2: Add question	Add a question to a bank	Question added	PASS
FR3: List and delete question	Show and delete question	Question deleted	PASS
FR4: List all banks	Show question banks	Question banks displayed	PASS
FR5: Delete bank	Delete a bank	Question bank deleted	PASS
FR6a: List quizzes	Create question banks and show the quizzes	Banks displayed as quizzes	PASS
FR6b: Select quiz	Select the example quiz	Example quiz opened	PASS
FR7: Display result	Complete the example quiz	Results displayed	PASS
FR8: Random question	Is the first question different when quiz is reset	Question is different after two tries	PASS
FR9: Next question	Current question is skipped	Skipped question counts as incorrect	PASS
FR10: Select answer	Correct answers selected	Result shows perfect score	PASS
Flair: Edit bank	Change a bank's details	Details changed	PASS
Flair: Edit question	Change a question's details	Question details written to wrong question	FAIL

5 Reflection

5.1 Successful Elements

I believe one of the most successful elements of this assignment was the UI design. The final UI closely resembles the non-functional prototype as the elements within that prototype were already modelled off Android Material Design Components. However, the final UI differs itself with a modern appearance by using elements such as card views and borderless text layouts. I believe the application's UI design is well thought out and easy for the user to navigate.
Another success of the program is how it meets all the functional requirements and even provides extra functionality to edit already existing data. I also believe the Kotlin classes within the application

are very well organised. The SQL manager and values tracker classes in particular, are examples of success as they provide methods that allow separate classes to safely share data.

5.2 Potential Improvements

There are two improvements that could be made to the application that would have a notable effect on the final build. The first is the full functionality of the question editor as currently it won't always write data to the correct question. Data being written in the wrong place causes unintentional data loss, making the question editor currently dangerous to use. If this were resolved, it would save the need to delete and re-create a question to change its data.
The other improvement would be the use of JUnit tests and programmatic tests for the UI, improving the debugging of the program. Automated testing would be more efficient than a manual test table and possibly show errors not apparent with manual testing. However, the manual test table created in their place, does test both functionality and UI in each test.

5.3 Attempted Flair

Two attributes of the application were intended as flair to improve upon the design in the assignment specification. The first was the ability to edit details of an already existing question bank to omit the need to delete and re-create it. The functional requirements state that the questions within a question bank should be changeable but does not mention the details of the bank itself. This flair feature allows the identifier and display name of the bank to be changed when managing the questions within it.
The other flair attempt was the ability to edit the details of an already existing question to once again omit the need to delete and re-create it. Their is no mention in the functional requirements of a question's details being changeable so this flair feature intends to add that functionality. Unfortunately, as explained in the potential improvements subsection, this flair feature is not currently fully functional.

5.4 Lessons for the Major Project

This assignment had me design a working application's UI for the first time, showing me the process of it's creation and how functionality is added. For my major project, I am likely going to create UI that manipulates data in a similar manner to this application. From undertaking this assignment, I now know the process involved creating an intuitive UI and how it can interact with a SQLite database. These are now both skills that I am confident in and can apply to my major project.

5.5 Conclusion

I believe this assignment was very successful and the final application, not only meets but, exceeds the functionality outlined in the assignment brief. The UI design is intuitive and modern, the software design is organised and efficient, the testing shows all function requirements have been met and the application includes some flair features. When considering all of those elements and this report, I believe a suitable mark for this assignment would be roughly 65%.

6 References

In document references:
Reference 0: Non-functional UI Prototype: found at
(mjh32_cs31620_assignment/nonfunction_prototype.pdf)

Reference 1: Final UI Screenshots: found at (mjh32_cs31620_assignment/ui_screenshots)
Reference 2: UML Class Diagram: found at (mjh32_cs31620_assignment/uml_class_diagram.png)
Reference 3: Testing Screenshots: found at (mjh32_cs31620_assignment/test_screenshots)

Other references (used for the construction of the application):
Reference 4: Visibility Animations
Reference 5: Divider Decorator for Recycler View
Reference 6: Hide Floating Action Button on Scroll
Reference 7: Jetpack Navigation Tutorial
Reference 8: Recycler View and Card View Tutorial
Reference 9: Radio Button Demo
Reference 10: Text Input Edit Text Tutorial
Reference 11: Modal Bottom Sheet Tutorial
Reference 12: SQLite Tutorial