**Traffic Sign Test Application Overview**

Michael Maye

Colorado State University Global

CSC475: Platform-Based Development

Dr. Muhammad A Khan

07/08/2023

**Traffic Sign Test Application Overview**

The submitted application is a simple test relating to traffic signs in the United States. This overview will explore the components this system implements and their interactions, including the test itself, question generation, the user interface layout, and results tracking methods. Additionally, it will highlight key features like user data persistence, navigation, and the application's overall functionality and user experience.

**Overview**

Upon opening the application, the user is prompted to enter their name and begin the test. They are then presented with eight different road signs, each with its correct answer and three random wrong answers in radio buttons. When the user submits an answer with the submit button the system moves them to the next question until the test ends. Upon submitting the last answer, the system displays a screen congratulating them and displaying their score. This paper is a detailed overview of this application and the different components it implements.

**Test Component**

The core functionality of the application is the Test class. This class represents a collection of 'Question' objects containing all essential elements for each question presented to the user. The 'Test' class generates a randomized question and adds it to its collection of questions, which it stores in an ArrayList within the object. The length of the test is set by calling the 'Test' constructor. This constructor requires an int as an argument, representing the test's number of questions. The 'Test' class exposes upcoming questions in the test by utilizing the 'getQuestion()' method. This method performs a pseudo-pop operation using '.remove(0)' to remove and return the first question in the array for each call. The 'Test' class also includes a get method to retrieve the current length of the array

**Question Component**

The 'Question' class represents a collection of information used to build the question UI displayed to the user. Each question is built off of an ArrayList of Integer[], and each Integer in the Integer[] contains an image as an id reference to a raw .jpg resource as well as the correct name of the sign in the image as an id reference to a string resource. Upon each call to the constructor, the 'Test' class passes in a unique random integer which determines what index to pull from in the ArrayList. The 'Question' class then takes the image id reference, sets the image into the Question object, chooses a random integer from 0 through 3 that represents the index of the correct answer, sets the correct answer to that index, and saves the index number for future reference and loads three other decoy answers into the array of answers. Every 'Question' object comes with getters to retrieve the image of the question, the array of answers, and the index of the correct answer.

**User Component**

The 'User' class contains the constructor to build a User object. Contained within the User object are the values for the user's name and score. The application uses this information to save user information to the application SharedPreferences file. The User contains methods to retrieve the user's name and score and a setter for the user's score to allow updating on subsequent tests.

**Test Layout Component**

The 'fragment\_test.xml' file comes with a Welcome layout, a Test Layout, and an End Test Layout, each contained in the same view under separate ConstraintLayouts. Upon opening the application, the system greets the user with the Welcome layout and sets the Test and End Test layouts to View.GONE and hidden from the user. The Welcome Layout prompts the user for their name, which the system will use to store their score in SharedPreferences. Once the user submits their name, the system presents the user with the test layout that contains a TextView that displays the question number they are on, an ImageView that displays the current questions sign, and a RadioGroup containing four RadioButton that represent the answers the user can choose. Once the user submits the final question, the system displays the End Test Layout congratulating the user for completing the test and their score.

**TestFragment.java**

The majority of computation occurs in the TestFragment class, which handles all user interactions and the logic for setting up and completing the test. When TestFragment initializes, it sets the Test and End Test layouts to View.GONE, hiding them from the user, prompting the user for their name. After entering their name and clicking the start button, the system calls the getUser() method. This method takes in the user's name from the getUserName() function and checks to see if it is currently in SharedPrefrences. If the user is already in SharePreferences it returns that User object, and if they are not, it creates a new user and saves them to SharedPreferences.

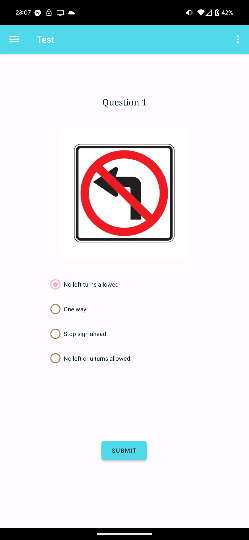
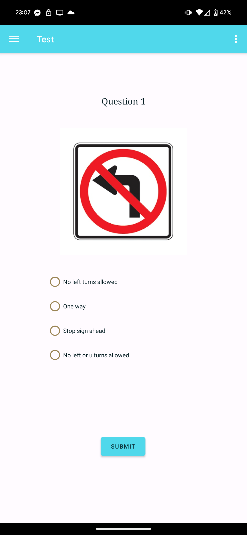
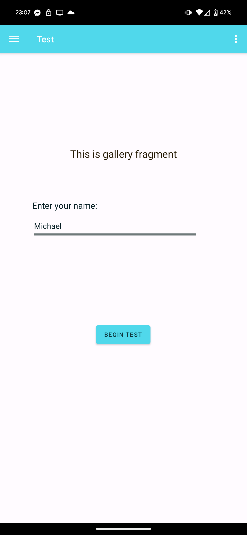
Once the user creation is complete, the system calls setupTest(), which sets the layouts to show only the Test View and calls the setQuestion() and startTest() methods. The method setQuestion() pulls the first question in the test array and builds the UI with the image and answers within the question. Start Test sets the onClickListner to the submit button so that when clicked, it calls the checkSubmission() method to see if the user chose the correct answer and increment the score if needed, and sets the following question with a subsequent call to setQuestion().

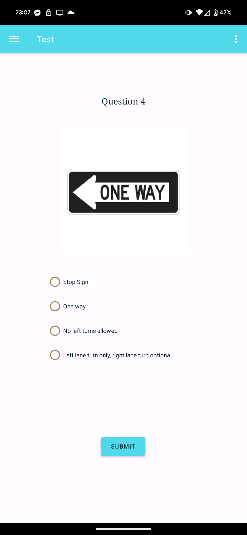
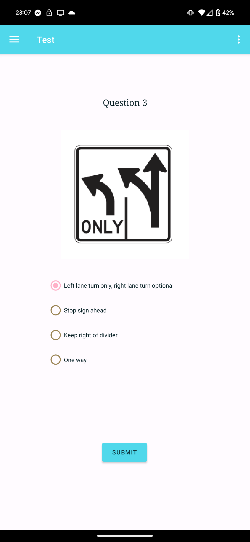
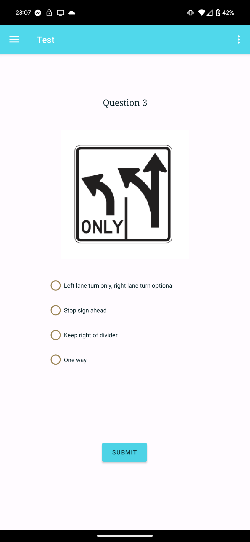
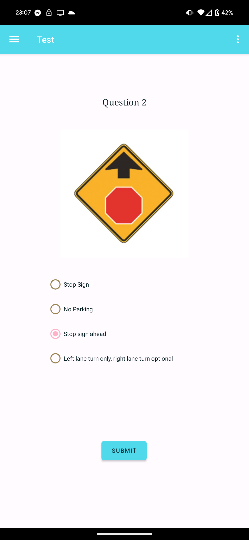
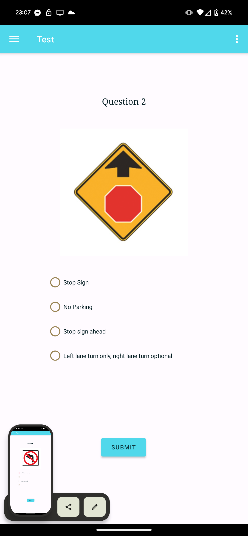
As the user clicks the submit question, the system checks the length of the test array to see how many questions are left. Once the length reaches 0, the system calls the endTest() function. This function changes the visibility on the layouts so that only the End Test Layout is displayed and sets the TextView representing the user score with the number of correct answers the user submitted.

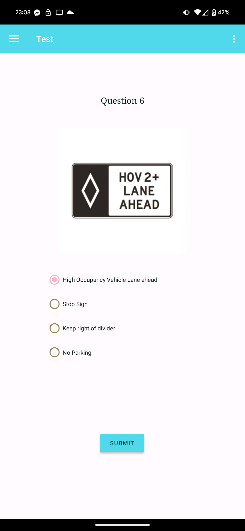
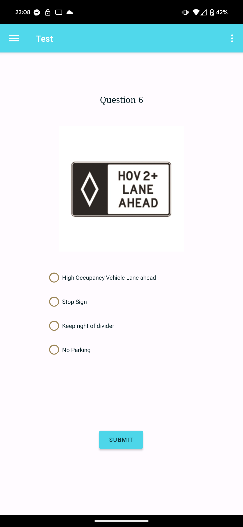
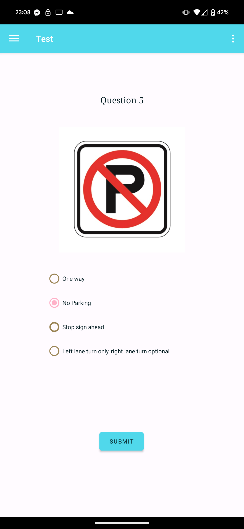
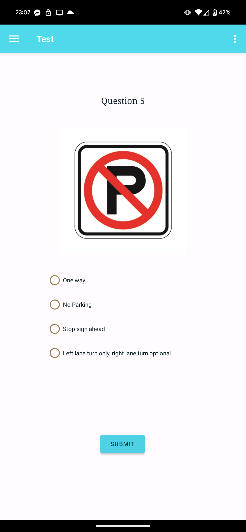
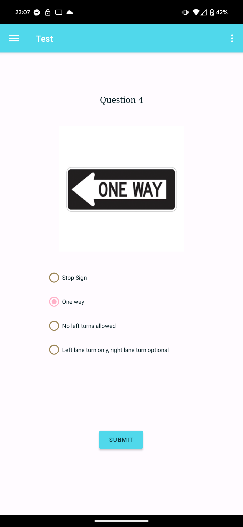
**Figure 1**

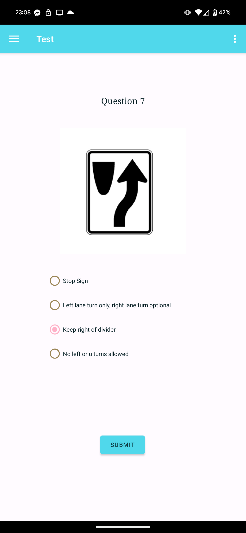
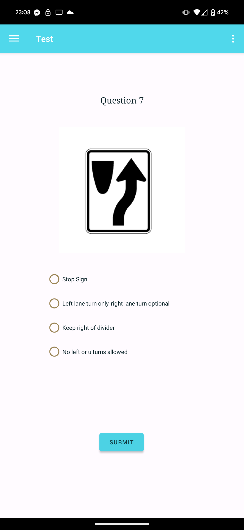
*Screenshots of application:*

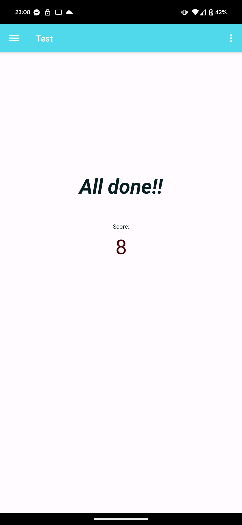
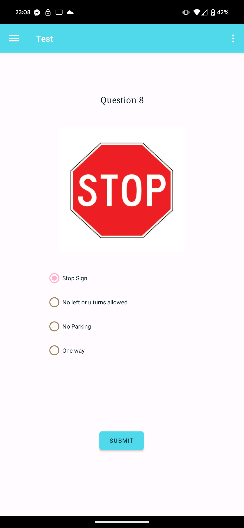
A screenshot of a computer

Description automatically generated





A red stop sign with white text

Description automatically generated

**Conclusion**

Throughout this overview, the various components that make up the application have been explored, including the test component, question component, user component, and test layout component. Each component is critical in creating a seamless and interactive user experience. The application successfully educates users on traffic signs by leveraging the Test, Question, and User classes along with the well-structured test layout..

**References**

*Android fragments - javatpoint*. www.javatpoint.com. (n.d.). https://www.javatpoint.com/android-fragments

*Fragments  :  android developers*. Android Developers. (n.d.-a). https://developer.android.com/guide/fragments

GeeksforGeeks. (2023, February 9). *Shared preferences in Android with example*. GeeksforGeeks. https://www.geeksforgeeks.org/shared-preferences-in-android-with-examples/

McCown, F. (2022). 5. In *CSC 475: Platform Based Development*. essay, ZyBook.

*Save simple data with sharedpreferences  :  android developers*. Android Developers. (n.d.-b). https://developer.android.com/training/data-storage/shared-preferences