Cat Math

We are the Tic-Tacs:
Abit Kumar Mahato - MCS
Ben Hawk - CEG
Jessica Venema - BS in CS
Ethan Schultz - CS
Liliane Owens - CS

Presented by Tic–Tacs on Dec, 03 2024

Elevator Pitch

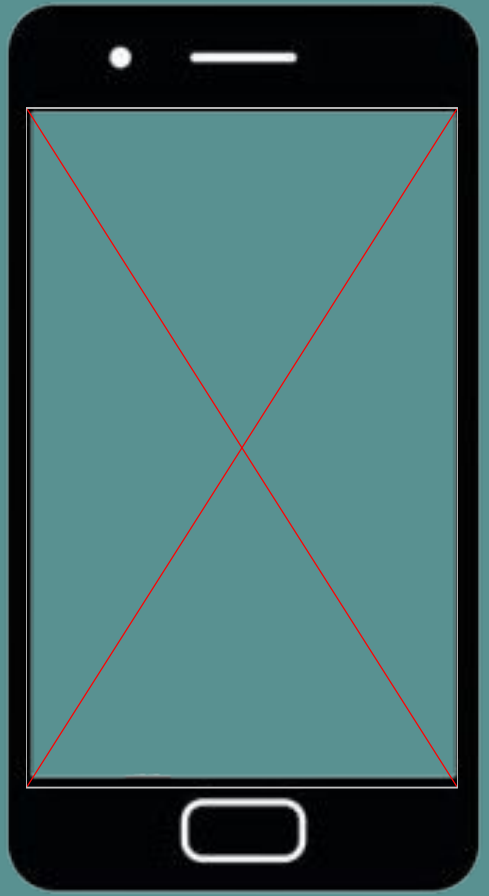
- We want to create an app for kids to improve their math skills and have fun while playing it
- There exists many apps to fulfill this task, such as Smart Tales, Math Learner, Math Games, and Todo Math.
- Our proposed solution is Cat Math. We chose this name because even hearing it will intrigue you to investigate the app.
- Our app features engagement through the use of experience systems, fun game modes, user customization, and appealing design.

Overview

- The App is made up of five main components:
  - Main Menu
    - Grants easy access to all components
  - Experience System
    - Provides incentive and keeps users engaged
  - Calculator Mode
    - Functions as a useful tool to perform quick calculations
  - Math Problems Mode
    - Allows users to practice their ability to solve simple equations
  - Math Drills Mode
    - A timed mode for users to test their quick thinking skills

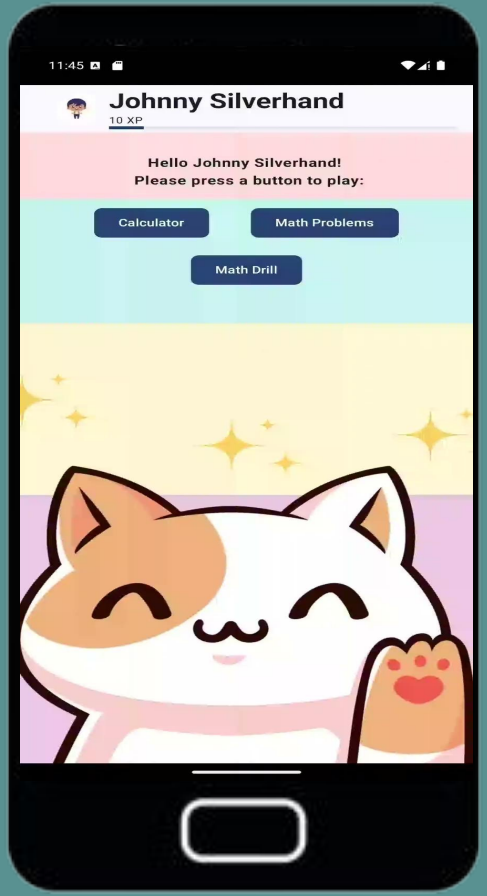
Main Menu

- As a User, I want to open the app and feel excited to play the games.
- The Main Menu shall display the User's current XP as well as give quick access to all modes.
- The background and buttons of the Main Menu shall be colorful while not being distracting.
- The Main Menu has options for the Calculator, Math Drill, and Math Problems modes.



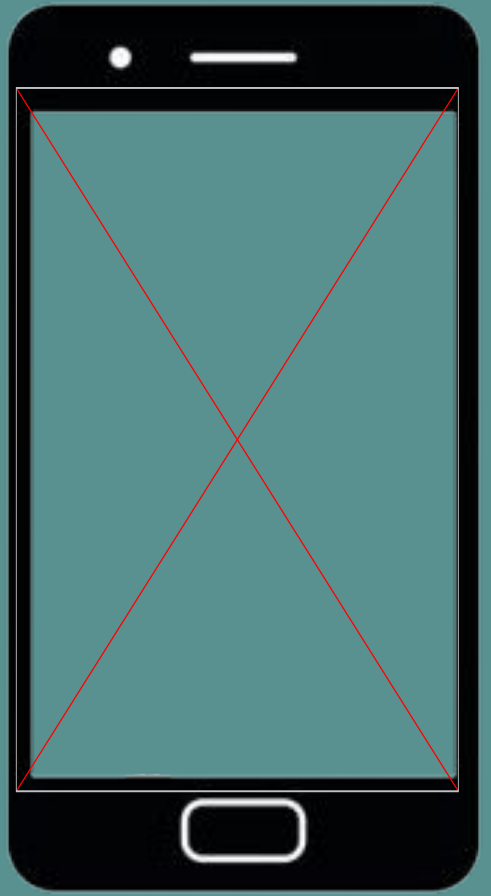
User Experience

- As a user I want to earn experience points (XP) to experience both and achievement and receive an avatar.
- Users shall receive XP for using the calculator, as well as for correctly answering problems in the Math drills and Math problems modes.
- The user's level will increase by a flat amount for the calculator mode, and will increase for every question they answer correctly in the Math Problem and Math Drill modes



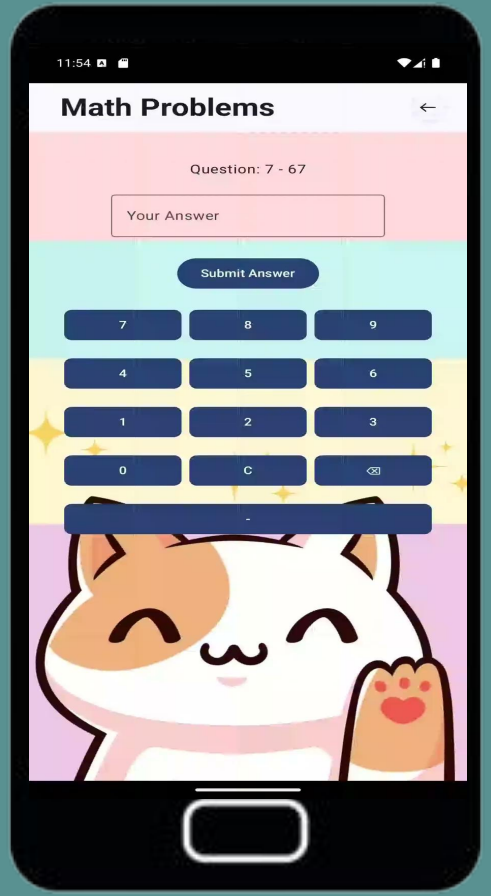
Calculator Mode

- As a User, I want to use the calculator mode to help me check my homework questions.
- The calculator shall support two variable addition, subtraction, multiplication, and division for integer variables as well as decimals.
- The calculator will display all of the numbers and operation entered by the user, as well as the final result.



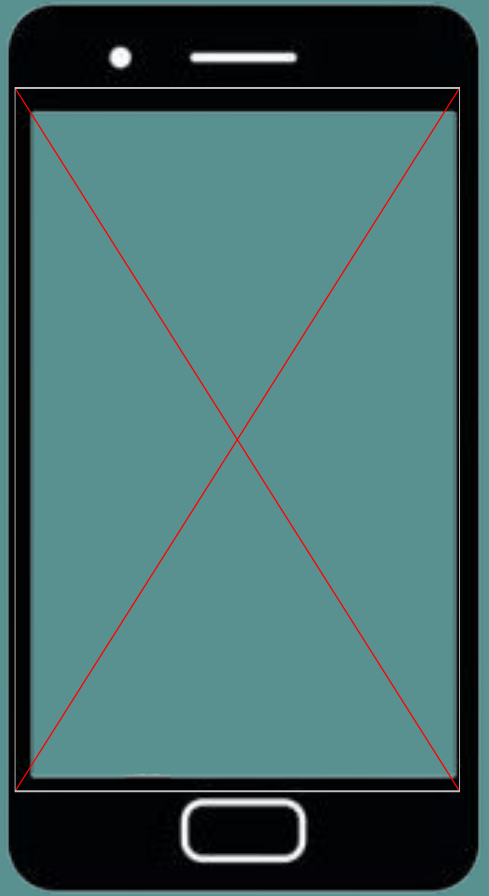
Math Problem Mode

- As a user, I want to use the math problems mode to teach me basic math operations.
- Questions shall be two variable equations which will include addition and subtraction.
- Questions shall be randomly generated, and will be sanity checked to ensure that there are no invalid equations.
- Upon answering incorrectly, a popup will display a cat fact as well as the correct answer, giving the user an opportunity to learn.
- Upon answering correctly, the user will gain some XP, and a popup will display a random fun fact about cats.



Math Drills Mode

- As a User, I want to use the Math Drill mode to test my ability to quickly solve simple math questions.
- The Math Drills mode shall operate on a 60 second timer.
- Upon answering incorrectly, the game mode will end, and the user will receive some XP and a fun fact.
- When the timer runs out, the user will receive XP based on how many correct answers they had, as well as a fun fact.



Test Plan

Testing Plan:

As we are building our project in java, Unit Testing is the best automated and robust testing method that can be used. Unit tests validate the correctness of code by checking if individual components work as intended. This helps catch bugs early and improves the reliability of your application. Some of the elements that I will be testing are:

- Unit testing for individual components
- Method-level validation
- Error handling verification
- Mathematical Logic Validation
- XP Point Calculation Tests
- Question Generation Accuracy
- User Interface Component Tests
- Error Handling
- Calculator Functionality Test
- Catfact Generation Test

Testing Framework:

The framework I choose for unit testing is JUnit4, which is a well-established testing framework for Java and Kotlin-based projects and comes built-in in android studio.

Testing Plan Justification:

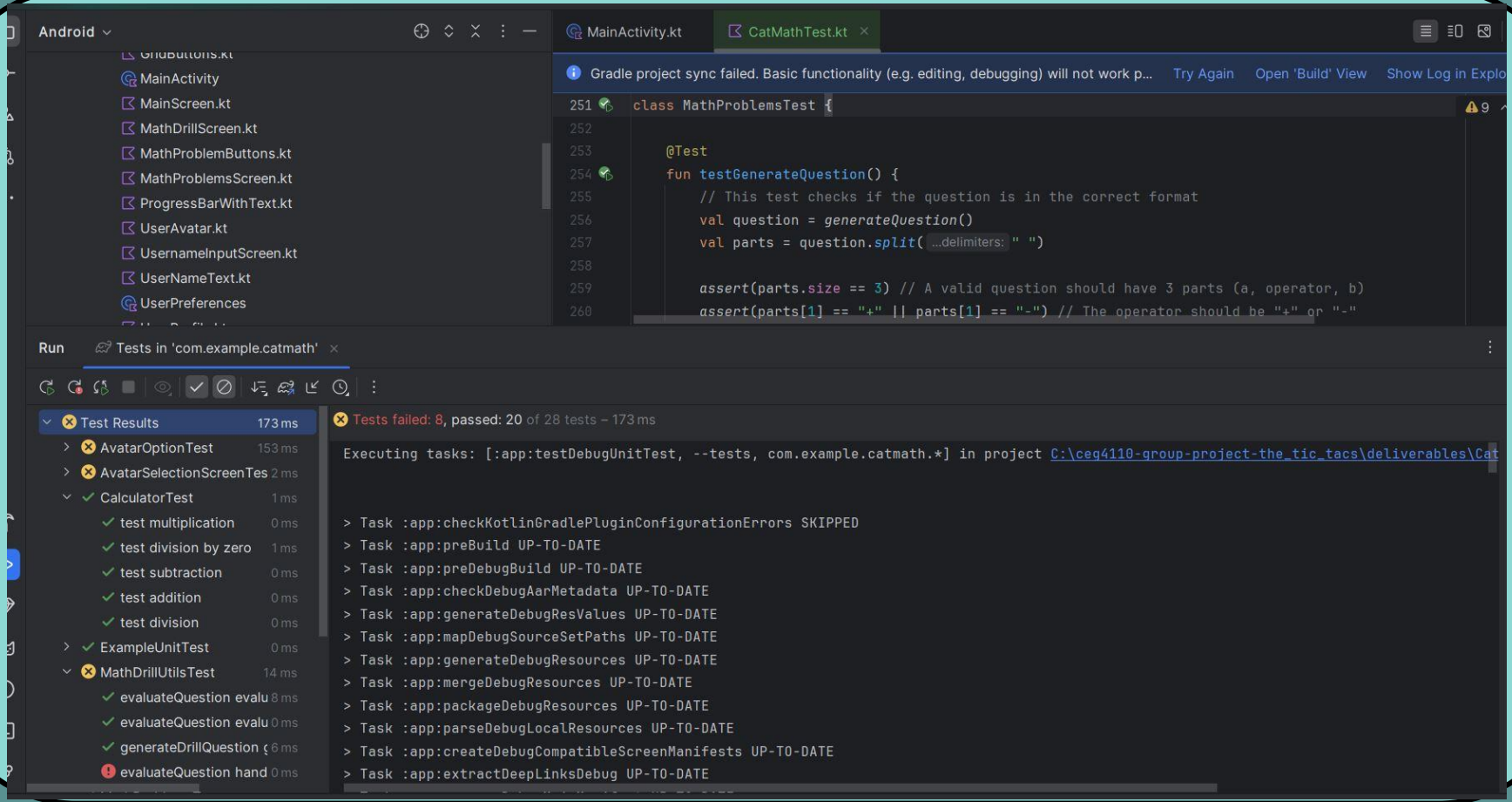
Unit tetsing is already part of the team's development plan, making it easy to implement without disrupting workflows. It integrates well with Android's build system , allowing tests to run automatically and ensuring early detection of bugs. Unit testing helps to test each small feature of the app . So, we can keep testing our code along with the development of the project. It will be easy for us to debug each feature compare to the whole app. We can write Independent, repeatable test cases.

Justification for Framework:

JUnit4 is standard Java testing framework and already widely used in Android development, and it's well-suited for testing Kotlin code. Since we are already using JUnit4 for unit testing, it's easy to integrate with the Jetpack Compose testing tools for UI testing, without introducing complex new frameworks or dependencies.

JUnit4 combined with Compose UI Testing will adequately test both the logic (XP validation, avatar selection) and the UI (element visibility, interaction). This combination ensures that both the backend logic and frontend UI are thoroughly tested. I also had used Junit in my undergrad level once for one of my java project. So, I choose Junit for the unit testing of this project.

Test Sample


Android
MainActivity
MainScreen.kt
MathDrillScreen.kt
MathProblemButtons.kt
MathProblemsScreen.kt
ProgressBarWithText.kt
UserAvatar.kt
UsernameInputScreen.kt
UserNameText.kt
UserPreferences
MainActivity.kt
CatMathTest.kt
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work p...
Try Again
Open 'Build' View
Show Log in Explo
251 class MathProblemsTest {
252
253 @Test
254 fun testGenerateQuestion() {
255 // This test checks if the question is in the correct format
256 val question = generateQuestion()
257 val parts = question.split(...delimiters: " ")
258
259 assert(parts.size == 3) // A valid question should have 3 parts (a, operator, b)
260 assert(parts[1] == "+" || parts[1] == "-") // The operator should be "+" or "-"
Run
Tests in 'com.example.catmath'
Test Results 173 ms
AvatarOptionTest 153 ms
AvatarSelectionScreenTes 2 ms
CalculatorTest 1 ms
test multiplication 0 ms
test division by zero 1 ms
test subtraction 0 ms
test addition 0 ms
test division 0 ms
ExampleUnitTest 0 ms
MathDrillUtilsTest 14 ms
evaluateQuestion evalu 8 ms
evaluateQuestion evalu 0 ms
generateDrillQuestion 6 ms
evaluateQuestion hand 0 ms
Tests failed: 8, passed: 20 of 28 tests – 173 ms
Executing tasks: [:app:testDebugUnitTest, --tests, com.example.catmath.*] in project C:\ceg4110-group-project-the_tic_tacs\deliverables\Cat
> Task :app:checkKotlinGradlePluginConfigurationErrors SKIPPED
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:generateDebugResources UP-TO-DATE
> Task :app:mergeDebugResources UP-TO-DATE
> Task :app:packageDebugResources UP-TO-DATE
> Task :app:parseDebugLocalResources UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinksDebug UP-TO-DATE

Conclusion

- The application fulfills the project requirements and a few extra additions.
- The project meets the needs of the users.
- Overall, we we were able to implement our plans for this project. Our app is working as well as expected, we hope that users will be thrilled by it.

Thank You

Any Questions??