CS378: Framework-Based Software Development for Hand-Held Devices

Project #4

Due time: 11:59 pm on 11/25/2025
Submit using Blackboard web site
Total points: 100
Instructors: Ugo Buy and Siddharth Ganesh

Rough Spec. You are to design and code a game of *gopher hunting* as a Flutter app. Imagine you are in a field occupied by a gopher. The gopher could be hiding in one of any number of holes in the field's ground. Two bots that you will design play against each other in an effort to find the hole that contains the gopher. (The game does not say what to do with the gopher, once it is found.)

Two players are represented by instances of two classes contained in your app, running different algorithms and playing against each other. (There is no human input once the game is started.) There are exactly 64 holes in the field; the holes are arranged as a 8×8 matrix of squares equally-spaced with respect to each other. The gopher is in one of the holes. The first player to find the hole with the gopher wins the game.

The app supports only a continuous-play mode whereby the two players play without interruption taking turns until one player wins the game. Each time a player makes a guess at the location of the gopher, it receives feedback indicating how close the guess was to the gopher's location, including whether the gopher's location was discovered correctly.

The main class is responsible for setting up the game, instantiating the two player instances, asking each player to produce a move when it is its turn, providing feedback to the players, and showing the guesses of the two players on the display.

Whenever a player makes a guess, the main code provides one of four possible responses:

1. Success—The player guesses the hole containing the gopher and wins the game. A message is displayed in a snack bar indicating the game winner.

2. Near miss—The player guesses one of the 8 holes adjacent to the gopher's hole. The display is updated to reflect the new guess and the game continues with the other player making a guess.

3. Complete miss—The player gets this response in all other cases. The display is updated to reflect the new guess and the game continues with the other player making a guess.

Here are some additional requirements on the app.

1. Your app should contain three main classes appearing in three different Dart files, namely the main class in charge of managing the game and two classes modeling the players. Feel free to use additional classes as needed.

2. At the beginning of the game, the main class should choose a random location for the gopher in the 8x8 matrix. This location is not accessible by the players.

3. The app's display should show a button for starting the game, a button for stopping a game and a button for exiting the app.

4. The app display should show two 8x8 tables clearly indicating the progress of each player. Use color coding to distinguish the moves of the two players. Each guess should be clearly marked with a number showing the order of the guesses, with the first guess being marked 1, the second guess being marked 2, etc. In addition, the display should show the (identical) location of the gopher in both tables.

5. Whenever a player makes a guess, the main class pauses (sleeps) for two seconds in order to show its new move on the display. After the pause, the main class will resume by asking the player whose turn is next to produce a guess.

6. Each player defines two **async** methods, one for making the next move, the other to reset the game. The main class will call these methods when the player must make a move or reset itself. The main class should wait on the completion of player's move, but proceed without waiting when asking a player to reset itself.

7. The two players should use different heuristics to win the game. Make sure that the heuristics are sensible.

8. The moves of the players should use suitable animations in order to make a move appear smooth.

Project deliverables. You are responsible for submitting the following 4 artifacts.

1. A refined spec of the rough spec above. In the refined spec, make sure to spell out tthe precise format of list items and the app bar.

2. A zip archive containing the Flutter project implementing your refined spec

3. A brief prerecorded video in which you explain how your code works. The maximum duration of the video is 180 seconds.

4. A document answering the following questions on your use of LLMs. The document should not exceed one page.

   (a) List the LLM or LLMs that you used for this project.

   (b) Did you use an LLM for refining the spec or project code or both?

   (c) What kind of prompting did you use for LLM? (E.g., one of zero-shot prompting; few-shot prompting; chain of thought prompting).

   (d) Rank on a Likert scale the results return by each LLM that you used (5–LLM produced exact results; 4–Minor corrections applied to LLM results; 3–Major corrections applied; 2–Most LLM output was unhelpful; 1–LLM output did not help at all.)

You are required to use at least one LLM for this assignment. Your submission will be graded according to the following criteria:

1. The completeness of the specs.

2. Code compliance with your specs.

3. Accuracy of your video

4. Ability to use LLMs effectively and assess LLM results

You must work alone on this project. Your project code should be in a special zip archive called xxx_yyy.zip, where xxx and yyy denote your first and last names. The archive will contain four items: (1) your revised spec, (2) zipped Flutter project, (3) short video, and (4) answers to the questions on LLM use. Submit the archive using the submit link in the assignment page of the Blackboard course web site. No late submissions will be accepted.