Software Architecture Specification Report

Team MGN

October 17, 2017

Breakdown of Individual Contributions

Joseph Olin

- Responsible for communication with American Printing House for the Blind
- Responsible for report proofing and submissions
- Will work on the server side of the app
- Will work on the client side of the app
- Will work on integration testing the app

Ben Pister

- Responsible for maintenance of the project website
- Will work on unit testing the server side of the app

Michael Roark

- Responsible for compiling the reports prior to proofing
- Responsible for general documentation of the project
- Will work on the server side of the app
- Will work on the client side of the app

Section 1.1

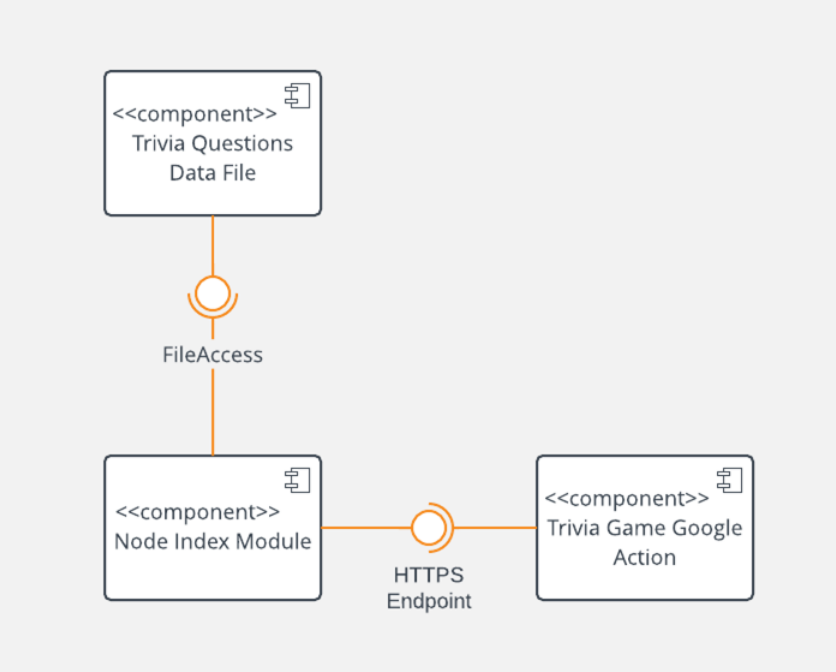
The system will be composed of three modules – a client module, a server module, and a data module. Relatively little coupling will be present between each module, as a change in implementation or location of any one module will require at most an endpoint change. Data persistence, as of right now, will be maintained in JSON files, which will contain question and answer data related to Orientation and Mobility. Flow will be directed from the client to the server, which will access the data module, and then reply back to the client. This cycle will continue as many times as needed for users to play the trivia game.

Section 1.2

There are three subsystems composing the Google Home Trivia App system. The first system is the "client side" of the app. This is the code that runs inside Google Assistant, and which Google Assistant uses to trigger our app. This subsystem is primarily responsible for defining the utterance that triggers the launch of our application.

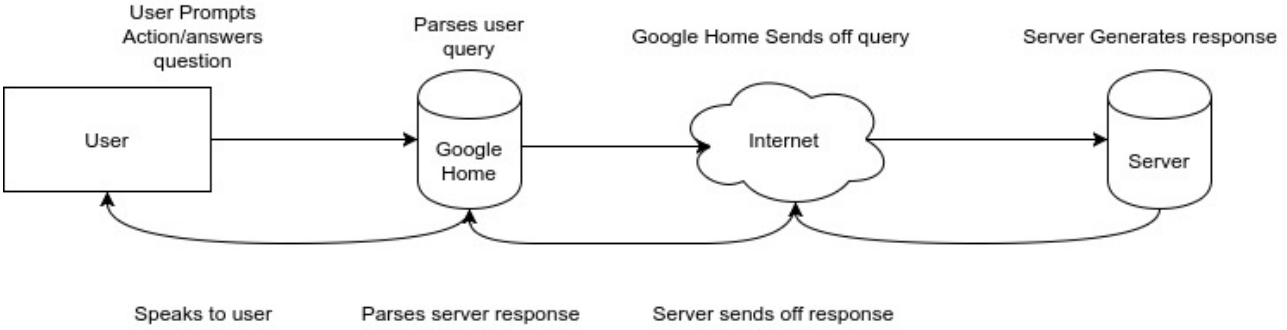
The second subsystem is the "server side" of the app. This subsystem handles the conversation with the user(s). It processes the questions data, passing it back to the client (Google Assistant) to be spoken to the user(s). It also handles the users' responses to questions, and notifies the user if their response was correct or incorrect.

The third subsystem is the data store containing the questions data. This may be a NoSQL database kept on a separate server, which the server side of the app will access and query. However, it's likely the data will be kept in a JSON file on the same server as the server side implementation.



Section 1.3

As is the design of the Google Home infrastructure, the hardware mapping can be defined on two systems. The Google Home device of the user, and the server that handles actions.



The above diagram illustrates the flow of data over the two devices. It shows that a users activates the Google Home with spoken words, and the Actions SDK directs the request to the appropriate server. The server is where requests will be processed, and return a response that the Google Home will then process. Almost the entirety of development will be done on the server, as the Actions SDK only defines entry points into the system.

Section 1.4

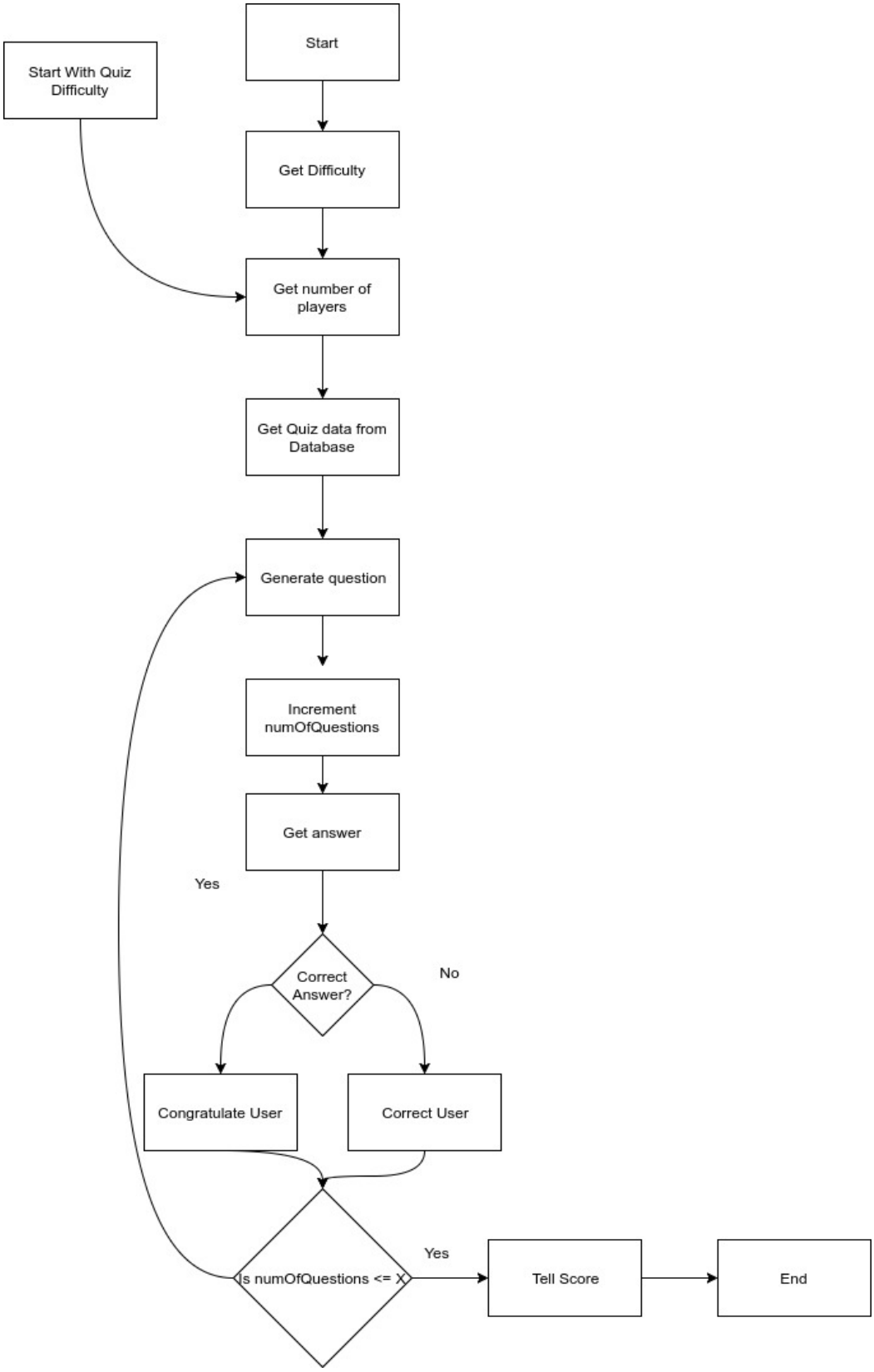
The persistent data for our application will be the questions that are to be served up to the users along with the accompanying answers, which will persist for the purpose of providing the users their options and checking their answer against the correct answer.
The Questions and Answers can be easily stored in a file system as JSON files. The file system will be best segmented by Quiz type and then further by Quiz. Since the Google Actions SDK functions on JSON, these JSON file quizzes will be easily workable with Google integration.

Answers will be saved for further examination through APH's system. This will allow them to migrate things as needed in the future, giving them control of the data to use as they need and takes it out of our hands and our liability.

Section 1.5

System Administrator: Will have full access and be able to create new quizzes, edit existing quizzes and allow/disallow access for those individuals with lesser previledge levels.
Teachers: Will have access to the necessary quizzes to teach the indivuals or groups of individuals they are assigned to teach.
System Maintenance: Will have access to most of the application for troubleshooting purposes. Will not have access to user authentication granting assignments.

Section 1.6



The chart to the left displays the entire system working in tandem. There are currently two entry points of the system when designing for a single user. An additional entry point might be "Start with X players".
The design keeps speed of start up in mind as we want to maximize teachers efforts while using the software. To help with this a short overhead is necessary.

As was discussed with our meeting with the American Printing House contacts, we need to consider multiplayer. However a hard requirement was that we have difficulty levels. The chart mirrors this, and assumes that the multiplayer is not already a part of the system, but we will design in such a way that multiplayer is something that we can later develop and easily integrate the new code..

Section 1.7

The system will be started up when the user utters one of the trigger phrases defined in the client side of the system. When a trigger phrase is uttered, Google Assistant activates our app, which calls the server side of the app for instructions on what prompt to give the user.

There are three ways in which the system may shut down. First, the system will shut down if the user prompts Google Assistant via voice to stop the app. Second, the system will shut down if the user neglects to respond within a space of time determined by Google Assistant (e.g. on mobile devices, Google Assistant may stop the app if it doesn't detect a response in five seconds, or similar). The third way the system will shut down is by the server after the game finishes. In this case, the user(s) will have answered all game questions, the app will have reported to the users their scores, and then will be commanded by the server side to stop.

Any errors/exceptions in the system will be handled on the server side. It's likely that users will attempt to respond to questions using incorrect answer formats (e.g. "The answer is car horn" when it should be "The answer is d"). In such cases, the server side will pass a prompt back to the client side requesting the user try answering again using a different answer.

Relatively little would be required to migrate the server module or data subsystems to other servers. Since the server side is composed of Node.js modules, it would be as simple as moving the files to the new server and re-building the server modules. The server module https end point(s) would likely change, in such a case. These endpoint changes would have to be reflected in an update to the client module. Moving the data submodule will as simple as moving or copying a data file or set of data files to the new server. The connection for reading data from the data subsystem might have to change if such were to occur. This connection would require an endpoint change so that it could be used within the server module subsystem for retrieving questions data.