CS 240: Phase 6 Chess Gameplay Transcript

[00:00:00] In this video, we'll discuss the phase six portion of the chess project we've discussed in previous videos, the design and architecture for uh the phase six client and server. But uh in this video, we want to talk about the actual requirements for this phase.

Start visual description. The professor demonstrates the design and architecture for the phase six client and server, specifying the actual requirements for this phase. End visual description.

[00:00:20] Now, it does turn out that this phase is probably the hardest part of the project.

[00:00:26] So encourage you to get started early and take advantage of all the time you're given to do the assignment.

[00:00:34] So let's just uh peruse through this, the specification here. So, uh first we see the architecture diagram that we discussed in a previous video.

Start visual description. The professor shows the architecture diagram discussed in a previous video, emphasizing the need to implement this architecture in the project. End visual description.

[00:00:44] So you're going to want to implement that architecture in your project.

[00:00:51] Look to the pet shop example, for more concrete code, examples of what um an implementation of this architecture looks like.

[00:01:05] All right.

[00:01:05] So the overall goal of phase six is to implement all the gameplay functionality.

[00:01:11] So of course, when you're done with phase six, your chess client and server should be totally complete.

Start visual description. The professor explains that the goal of phase six is to implement all the gameplay functionality, ensuring the chess client and server are complete. End visual description.

[00:01:17] And so the only piece that's missing on the on the client after phase five is the gameplay.

[00:01:23] So, the goal is to implement all the gameplay um functionality.

[00:01:29] So conceptually there, there's kind of three phases the user goes through.

[00:01:33] They, when they use the client, right? They run the client.

[00:01:38] The first thing they do is register or log in. That's the first part.

[00:01:40] And the second thing they do is they can create games or list the games, um things like that, they can join games and that's the second phase. But once a player has joined a game as either an observer or as a player, then their client needs to enter the gameplay mode.

[00:02:00] And once they're in gameplay mode, all the communication is done between client servers done with web sockets.

[00:02:10] Now, if you um look here in the, in the spec, we can see that the commands that the user can execute once they've entered the game are, are these here so they can get help.

Start visual description. The professor highlights the commands that the user can execute once they have entered the game, such as getting help, redrawing the chessboard, leaving a game, making a move, resigning, and highlighting legal moves. End visual description.

[00:02:23] And if they don't know what the commands are, they can request that the chessboard be redrawn.

[00:02:29] And the reason we have that is sometimes the chess board might scroll off the screen depending on how many notifications the client is receiving from the server. And so, if the chessboard scrolls off the screen or something, then this is just an easy command that lets the user ask to have the board redrawn, uh the user can leave a game that removes the user from the game.

[00:02:56] And so that means that when the user um leaves the game that somebody else could come along and take their place at some point uh make move.

[00:03:09] And of course, there has to be a way in your chess client for the player whose turn is to make a move.

[00:03:16] And there has to be a way to resign.

[00:03:20] And there also has to be a way for the, the player whose turn is to highlight the legal moves.

[00:03:24] So if they're wondering where a particular piece can move, then they should be able to request that the legal moves for a particular piece um be highlighted.

[00:03:36] Now, that's not necessarily limited to the player whose turn it is.

[00:03:40] It could be an observer. Anybody could really ask to see the legal moves for a piece.

[00:03:45] It doesn't have to be the player who's turn it is.

[00:03:50] OK? So those are the high-level commands that any user can execute.

[00:03:55] Now, of course, if it's not their turn, they shouldn't be allowed to make a move.

Start visual description. The professor discusses the notifications sent from the server to clients whenever various events occur, and the importance of ensuring *the server-side code sends out these notifications appropriately. End visual description.*

[00:03:59] Um Things like that.

[00:04:01] So, uh yeah, making a move to be restricted to whoever's turn it is.

[00:04:09] OK? Now, the next section here talks about all the notifications that get sent from server to clients whenever various events occur.

[00:04:15] So make sure you read through that and make sure that your server-side code sends out these notifications to the clients as appropriate.

[00:04:24] I won't drag you through all the details you can just read through that and then we get into the web socket section which, which talks about how web sockets must be used in, in the project.

[00:04:38] And we've covered a lot of this in previous videos, but you can read through that to get a refresher and make sure you know exactly what's expected.

[00:04:46] Um It talks about in here the, the different kinds of web socket messages.

[00:04:50] So there's, there's basically two kinds, there's user game commands and server messages.

[00:04:58] Now, these are just two classes that are in the starter code.

[00:05:00] So if you look in the starter code for phase six, you'll find um both of these classes.

[00:05:07] And so the idea would be that these are our base classes.

[00:05:12] So you need to create subclasses of these classes for each specific kind of message.

[00:05:18] So these base classes would include the fields that are needed for every message.

[00:05:23] But then you'll need to make subclasses of these to include the other data that's needed by uh these specific kinds of messages.

[00:05:32] For example, make move is going to need to have information about what move the user wants to make.

[00:05:37] So you'll need a subclass of, of user game command named make move command or something like that, that includes the move information.

[00:05:46] So I would expect that um for user game command, you'd have four subclasses potentially of, of that class. And then on the server side, we have a similar situation, we have a base class named server message it's got all the common fields to the different message types and then you'll create subclasses for these three different kinds of messages that contain the data specific to them.

[00:06:15] And so this uh the spec goes into more detail on what that looks like.

[00:06:22] Now, perhaps most interestingly in in this specification is this web socket interactions section.

[00:06:30] This section explains in full detail, all the messages that are being passed back and forth between the, the client and the server.

Start visual description. The professor explains the web socket interactions, detailing all the messages passed between the client and the server, including the order and purpose of these messages. End visual description.

[00:06:38] Now, we've already talked about what the messages are, but this talks about which messages get sent when and in what order and, and for what purpose.

[00:06:46] And so you want to read through that and make sure your, your client and your server um satisfy all those requirements.

[00:06:53] And so basically at a high level, it’s just implement gameplay, make it work according to uh the requirements listed in here.

[00:07:05] Now, one thing to be aware of on phase six as well is that the web test page.

[00:07:15] So if we go to the web test page on the server on your server, now this is familiar from phase three and other previous phases where we’ve tested our web endpoints, our web API end points.

Start visual description. The professor demonstrates the web test page on the server, showing how to test web socket interactions by connecting the client to the web socket and sending different kinds of messages to the server. End visual description.

[00:07:34] But there’s also a section on this page that allows you to test your web socket interactions.

[00:07:41] And so the first thing you want to do here is connect your client to the web socket.

[00:07:48] So initially, it just has a connect button.

[00:07:51] But once you connect and open a web socket to your server, then you can use these other buttons to send all the different kinds of, of messages to the server.

[00:08:03] The only difference being these are web socket messages instead of http messages.

[00:08:07] But otherwise it, it works a lot like it, it did for the web API part.

[00:08:12] And so take advantage of this and, and test your, your server and your client, well, your server, the client, this is the client in this scenario.

[00:08:22] So this is for testing your servers, web socket logic.

[00:08:26] And so take advantage of that.

[00:08:29] And so in here, you can enter the messages that are being sent to the server and down here at the bottom, it shows all the messages that come back from your server.

[00:08:36] So that’s really handy for, for testing and debugging.

[00:08:40] Of course, if we go back to the specification, um you can see that if we look at the rubric, um there are test cases in the starter code for your web socket functionality as well.

[00:08:58] So we have web socket tests.

[00:08:59] So you need to make sure that um your server passes all of the web socket tests.

[00:09:07] And we are grading the code quality for this phase. Again, there aren’t any unit tests that you need to write for this phase. So that’s, that’s nice.

[00:09:18] And so if you um when you want to pass off your project. This is, this is similar to phase five where the project is passed off partially with the auto greater and partially by uh A T A doing manual tests on your, on your project.

[00:09:34] And uh so you’ll just need to uh go through the auto grater and meet with the T A and then you can look down here at the bottom to see um how many points everything’s worth.

[00:09:45] One thing to be aware of on phase six is the partial credit is available on previous phases. That wasn’t really the case.

[00:09:54] I mean, you could be late and get partial credit that way.

[00:09:56] But really you had to finish all the functionality at least to move on with the project and to pass it off.

[00:10:03] But for phase six, because it is the last part we do um want to give partial credit for the functionality you do complete in case you don't complete at all.

[00:10:13] And so you can look at this rubric and see how many points each fun functionality area is worth.

[00:10:19] And then of course, we have our code quality um check at the bottom. So, pass the test cases, implement the interactive functionality, receive partial credit if needed and then your code quality will be tested as well.

[00:10:32] And once you get all that done and passed off, you are done.

[00:10:37] Congratulations.