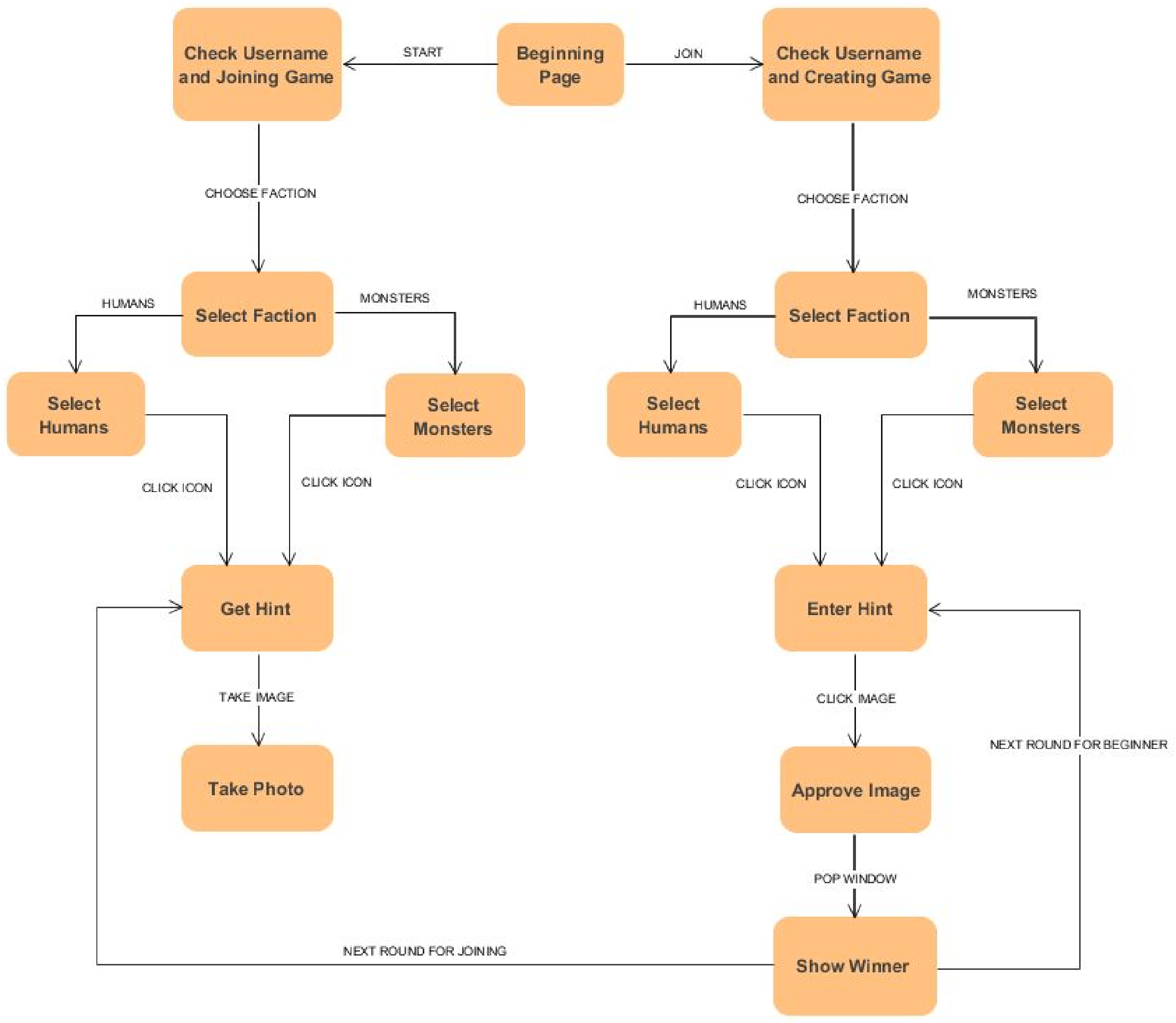
CS 408 Incremental Testing and Regression Testing

Yew Aili, Qian Zhang, Dennis Chia, Xiaotian Yu, Yuzhu Yang

1. Classification of Components

1.1 Define all Components



Start a game:

Input	Click “Start A Game” button
Output	Redirect to input username
Parent Dependency	Null
Child Dependency	Input Username

Join a game:

Input	Click “Join A Game” button
Output	Redirect to input username
Parent Dependency	Null
Child Dependency	Input Username

Input username:

Input	Username
Output	Success or failure message
Parent Dependency	Start A game, join a game
Child Dependency	Choose a faction

Select a faction:

Input	Click “Human” or “Monster” button
Output	Redirect to “Select an Avatar” page
Parent Dependency	Input Username
Child Dependency	Select an Avatar

Select an Avatar

Input	Click avatar image button to choose
Output	Redirect to “Set a Hint” or “Waiting” Page
Parent Dependency	Choose a faction
Child Dependency	Set a Hint, Waiting

Set a Hint

Input	Enter a hint for current round, click “Ready”
Output	Success or failure message
Parent Dependency	Select a Avatar
Child Dependency	Waiting page, Take an image, View score

Get a Hint

Input	Read hint for current round
Output	Hint displayed.

Parent Dependency	Waiting Page
Child Dependency	Result Page

Take an image

Input	Select “Take Image” button
Output	Camera application opens on mobile, allowing users to take a photo.
Parent Dependency	Waiting Activity
Child Dependency	Result Page

1. 2 Form of Incremental Testing

The form of incremental testing we followed is top-down, which was similar to the incremental testing we used in Sprint 1. It was easier to continue starting from top-down as we already had stubs in place that could be used to simulate behavior of lower modules. We finished most of the implementation of sprint 2 components described in product log, and merged the components together. The code we merged can run smoothly in series. Thus, we believe that using top-down incremental testing was the right choice.

2. Incremental and Regression Testing

2.1 Automation

We automated parts of regression testing by using unit test cases made specifically for our test plan. We kept using the same testing framework as what we used for regression testing in sprint 1. For front end, we used the Espresso Framework and for the server testing we used Mocha. Espresso testing framework provided by Android Testing Support Library provides APIs for writing UI tests to simulate user interactions within a single application. Mocha is a JavaScript test framework that tests the webserver running on Node.js.

2.2 Defect Log

Incremental Testing

Severity: 1(Severe) - 3(Not severe)

Module	**Timing Progress Bar**

Defect #	Description	Severity	How to Correct
1	After making adjustment on progress bar for countdown timer, the progress bar does not function until the last 10 seconds	2	There is no problem with countdown timer but the progress bar. We decide to synchronize the progress bar with countdown timer by sharing the setProgress function which countdown timer is using

Module	**Game Page**		
Defect #	**Description**	**Severity**	**How to Correct**
1	A new method was not in its enclosing class.	1	MainActivity.this changed to GamePage.this
2	Request-entity was too large.	1	Need to set a size limit for body-parser’s request-entity.
3	gameID was not being sent to the server.	1	Check for correct spelling, which is GameID instead of gameID.
4	Data was not being parsed.	1	Use a body-parser in server.

Module	**Player Score**		
Defect #	**Description**	**Severity**	**How to Correct**
1	The application can support several groups of players playing the game at the same time. Present player’s score is mis-deleted due to a player’s leaving in another group	1	Instead of using database to keep track of player score, we used server to do the same thing. While deleting player score in server, forgot to filter player score by gameId. It resulted in players

			information from different groups were all mixed together. To correct this, we use gameId to filter players, then delete player score from those filtered players.

Module	Result Page		
Defect #	**Description**	**Severity**	**How to Correct**
1	In the case that no player sends the correct photo, after the hint giver selects photo from Google Photo, the photo cannot be displayed on the result page	1	Instead of passing “INTERNAL_CONTENT_URL” to startActivityForResult, pass “EXTERNAL_CONTENT_URL”
2	In the case that no player sends the correct photo, the players can start next round without knowing the correct photo sent by the hint giver while the players should not be allowed to start next round until the hint giver sends the photo	2	The button of next round is disabled until the hint giver sends the photo.

Module	Image Page		
Defect #	**Description**	**Severity**	**How to Correct**
1	Image was not being stored properly.	1	Bytearray needs to be stored as strong.

Module	Hint Giver

Defect #	Description	Severity	How to Correct
1	1. The hint giver is first sent as an array with the order in the array. 2. The order needs to be handled by the client which adds computation time on the client	3	Hint giver is now saved in the web server and is sent to the client every time the client requests it.

Module	HintActivity		
Defect #	**Description**	**Severity**	**How to Correct**
1	Not setting time or hint causes an error sometimes in GamePage.	1	Disable Ready button if time or hint is not set.

Regression Testing

Module	GamePage		
Defect #	**Description**	**Severity**	**How to Correct**
1	Toast was causing an error.	2	Replace this with getApplicationContext. “This” may be different from the activity.
2	A portion of code in the GamePage.java was causing an error.	1	There was still some merging arrows left in the code. Remove the arrows.

3	Hard-coded text was displayed when there was no server activity.	3	Make it blank, but make sure that the position was fixed.

Module	**Progress Bar**		
Defect #	**Description**	**Severity**	**How to Correct**
1	The progress bar does not function until the last 10 seconds. This problem is fixed in incremental testing by using countdown timer's setProgress function. Details are described in incremental testing section.	2	After fixing it in incremental testing, we input several time limit to test the functionality of progress bar. There are three equivalence classes: 3 mins to 1 min 1 min to 11 secs 10 secs to 1 sec After fixing, progress bar works well with all testing chosen from equivalence classes

Module	**ImagePage**		
Defect #	**Description**	**Severity**	**How to Correct**
1	Image displayed after user confirmed was out of place.	1	Make sure image position is relative to static item's position.

Module	**Ready**		
Defect #	**Description**	**Severity**	**How to Correct**
	Player getting ready doesn't trigger the UI to disable the button.	1	A player tapping the ready button sends an event to the web server and

			disables the button. When all the players are ready, it sends a start game event