Milestone 4 Beta Launch and Reviews

CEN 4010 Spring 2018

Team #4

April 16,2018

*Witch Hunt*

Ivan Maykov: Product Owner

John Floyd: Scrum Master

Kyle Prince: Team Member

Sha’Quana Jones: Team Member

|  |  |
| --- | --- |
| Revision | Description |
| 1.0 – 04/16/2018 | Initial Document Release |

**2.2 Product Summary:**

Product Name: Witch Hunt

Witch Hunt is a web-based multiplayer game. It enables people to play with any person, as long as they are able to connect to the internet. As the mobile market is becoming increasingly popular, there is no better time to make this game. As the game only requires an internet connection and a web browser, it enables players to play with others regardless if they’re using a smartphone, desktop, or laptop computer. The game starts with one player being selected as the witch while everyone else is a villager. The game is played in rounds, with players using a chat that enables them to communicate with others during it. After a certain period of time, the other players vote on who is the witch. That person is removed from the game (loses/is killed). If they are correct, the surviving villagers win. If they are wrong, the witch kills a villager. The witch wins if they are the last man standing or it’s just them and one villager. As it is browser-based, the game is able to be played regardless of physical distance to other players.

All major committed functions:

There are a variety of major functional capabilities that the team will be implementing in the final release of the application.

* The application will be accessible on the LAMP server through its URL.
* It will enable users to register with a username and password
* It will enable users to login with a previously created username and password.
* Logged in users will be able to join a previously created public game
* Logged in users will be able to join a previously created private game, given an entrance password.
* Logged in users will be able to create a private game
* During a game, users will be able to choose a player to cast their vote on.
* Users will be notified of who was voted out, and whether they were a Witch or not.
* The person who was voted on by the Witch will be voted out.
* Users will be able to send messages during a game.

Unique Features:

Being able to play through the browser is a unique feature as it enables players to play regardless of the device used to access a web browser.

URL: http://lamp.cse.fau.edu/~CEN4010\_S2018g04/WitchHuntTest001/

**Milestone 4 Demo -** [**https://youtu.be/pJqScIxrYXk**](https://youtu.be/pJqScIxrYXk)

**2.3 Usability Test Plan**

Overview:

This usability test plan will test Witch Hunt’s voting during its development. Voting in the game works as such. You have a number of players on their phones join a game. One of the people will be selected to be a witch, everyone else is a villager. The game is played in rounds. Each round the players discuss who is the witch. After a certain period of time, the other players vote on who is the witch. That person is removed from the game (loses, is killed). If they are correct, the surviving villagers win. If they are wrong, the witch kills a villager. The witch wins if they are the last man standing or it’s just them and one villager. This test will establish user performance as well as test for any design inconsistencies to ensure a functional and pleasing game. Design errors will include testing for design (presentation), navigation errors (visual flow of game) and user requirements (ensuring user is satisfied with game functionality). The usability test will use multiple games to test the functions of the game itself and use the players as the user group to provide feedback on any inconsistencies that may arise through testing. The testing will begin with the players roles being set and the rounds to determine which player is the witch in play. We will check for any errors in design at this point and make sure user performance is successful in connection for communicating to determine the witch. We will then test for the code to count those who have been “killed” to be unable to play the game but see the end results of who won. After many rounds there will be a point where voting of who is the witch will take place and we will test the design it total of navigation and so on.

Plan:

The first step in planning will be to ensure that the URL in which the game takes place works and can be accessed by users of the game. Completion Criteria will include either the witch winning (By being the last man standing or face to face with one villager) or the villagers who are trying to take the witch out (guess) guessing right on which player is the witch. We as a group are the intended user to ensure the proper testing for a successful and satisfying game. Our goal is to provide a great design visually as well as maintaining a functional and free flowing game for all users to enjoy. The starting point of our game would be for either a user to log in or create a login in order to have access to witch hunt. These aspects will contribute to the testing to ensure the voting of the game is functional as it’s an important piece, without it there is no point to the game as the goal is to guess the witch and win!

Questionnaire Form:

Choose from the following questionnaire by clicking on the response that best fits your feelings on the following characterizations where 1=Strongly Disagree, 2=Disagree, 3=Neither Agree Nor Disagree, 4=Agree and 5=Strongly Agree.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Strongly Disagree | Disagree | Neither Agree Nor Disagree | Agree | Strongly Agree |
| 1. Voting on which player is the witch is done with ease. | 1 | 2 | 3 | 4 | 5 |
| 2. Creating private chats for voting is easy to do. | 1 | 2 | 3 | 4 | 5 |
| 3. Figuring out who’s the witch to vote for is thrilling. | 1 | 2 | 3 | 4 | 5 |

**2.4 QA test plan**

Test objectives:

The QA test plan will involve the Voting function of our Witch Hunt game. Our goal is to establish user performance as well as test for any design inconsistencies to ensure a functional and pleasing game. The motivation for this function is to creating a voting poll to determine which player is the witch as well as create private chats to decide which player might or is the witch in the game. Through testing we will be able to run through the voting section and get a grasp of the games performance in relation to the users as well as identify any errors or changed needed. The following will include key points to look after to ensure proper testing:

* Does voting only include live players excluding “killed ones?
* Does the voting poll display players name and a box to check off which player might or is the witch of the game?
* Does the coding terminate when the villagers vote correctly on who’s the witch?

Hardware and software setup:

The software for coding the voting function of the game is written in php. The process starts by asking the user(s) which player is the witch and either end the game with the villagers guessing the witch correctly or submits which villagers guessed wrong and are therefore “killed” from the game. We make a constraint to say that the voting will not take place unless there is a certain number of players I the game. This is the setup of the voting function in the software that later will correlate with other functions that contribute to the game as a whole that will be called in the main.

Feature to be tested:

The testing with include the voting function of the witch hunt game and will cover the bullet points noted in the objective of the QA test plan to ensure user performance and identify any inconsistencies with the game itself. We will look to see that the voting that comes after the initial voting will only include live players, ensure voting includes user names to be voted as the witch and follow conditions to terminate in the event that the witch is guessed correctly.

Actual test cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | Test title | Test Description | Expected Output | Test Results |
| 1 | Does voting only include live players excluding “killed ones? | After the initial voting to determine which player is the witch, the next voting poll should only include players that are still active and have not been “killed” | Voting poll displays live players to be included in polling | Voting poll displays live players only on screen |
| 2 | Does the voting poll display players name and a box to check off which player might be or is the witch of the game? | The voting poll will display live or active players to be included in the vote for who is the witch in Witch Hunt | Voting poll displays list of live players names on the screen in polling | Screen displays players with their user names accordingly |
| 3 | Does the coding terminate when the villagers vote correctly on who’s the witch? | There will exist a loop to terminate the game if the players guess the witch correctly resulting in a win! | A display will appear announcing that the villagers have won in guessing who the witch is in the game | Code terminates game when player guess is correct |

Testing was done onGoogle Chrome and Mozilla Firefox to ensure that the website for Witch Hunt functioned according to team requirements and has passed all three test cases for the voting function.

**2.5 Code Review**

For the code review, members were able to review the code submission from Ivan Maykov. With further analysis of the commit made to the master branch, some recommendations can be made.

* String variables that contain queries should be commented to inform reader what the statement should do.
* Add necessary spacing between segments of code.
* Code that is related to one functional implementation should be separated and be easily identifiable.
* Comments should be added to other declared variables that reveals their functionality
* Variables should be given names that enable a reader to identify their functionality.
* Potential for a SQL Injection in *castVote.php*.
* The username is passed as the parameter for the statement. (lines 16, 18)
* The username may contain code that negatively affects database.
* Should be changed to access by a user ID generated by the system.
* Functionality should be separated to make sure that users are in the same round
* Users should not be able to go to the vote page unless others are finished voting in the previous round.
* Could be broken into functions (ex. submitVote(), isGameActive(), getKilled()) in order to better control the state of the game.

**2.6 Self-check on best practices for security**

For the application, the only way to play is if the user has a username and password. The username will be visible to other players and therefore does not require extensive security measures. However, the player’s password needs to be protected to prevent unauthorized access. Therefore, the player’s password will be encrypted when sent to the server for storage when the player creates a login.

Password encryption has been verified and password inputs have been tested using the following input set:

{

12345678

Dj38hs9k

J8h83hsa

8djfn451

34mkd81q

30nhA73d

}

**2.7 Self-check on nonfunctional specs**

1. Performance - Response time - maximum load time, 1 sec. DONE

2. Usability / Portability – Game must be properly formatted and functional on all iOS and Android mobile devices ON TRACK

3. Accessibility – Users must have an account and login to access the game DONE

4. Expected Load – During game play, max 8 user concurrently. As popularity of game increase and multiple games are played concurrently, expected load will increase. Initially a max of 100 concurrent games with a max of 8 players, requires 800 concurrent users with a max pull of 250KB per user. DONE

5. Security Requirements – Any personal data needs to be encrypted when stored on database DONE

6. Storage – Web server storage requirements 100MB max, Database max size initially 1GB. Individual pages displayed should not exceed 1MB for faster load times. DONE

7. Fault Tolerance – Web site can be hosted on FAU Lamp server initially. Further growth will require hosting with a service provider with a min 99.98% uptime with daily site and database backups. Redundant server is unnecessary until 10,000 users are reached. DONE

8. User Signup – Simple and quick signup with limited user personal information: username and password only. DONE

9. Ease of Use – User should be able to navigate game and play without instructions. If user does need view instructions, they should be brief and no more than 7 paragraphs. ON TRACK