CEN4010 Principles of Software engineering, Spring 2018

Milestone 1 Project Proposal and High-level Description

Team Witch Hunters

Team #4

Ivan Maykov – Project Owner  
Kyle Prince – SCRUM Master  
John Floyd – Team Member  
Shaquana Jones – Team Member

February 19, 2018

**Witch Hunt**

Executive Summary:

Witch Hunt. It’s a web-based game played with friends. Since the phone-capable market is booming currently, there is no better time to make this game. Quite often you hear stories of people on their phones next to each other, now you get to do it while voting each other off. It doesn’t require any fancy graphics, just a web connection and a phone.

The game works like this. 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. Players have a public chat that everyone can see, and they can create private chats that only the people involved can see. This allows the game to be played in long and short distance.

2) Competitive Analysis

|  |  |  |
| --- | --- | --- |
| Features | Summary | Competitors |
| Login | This is a basic login. Each player will need to create an account to play the game. If possible, we may implement a Facebook social aspect, where people can login from Facebook. In addition, if time allows, we can make it so you can add friends. | This social concept will allow plays to more easily connect and play with friends. It is similar to the social features most commonly seen today. The ability to login with Facebook, is seen in almost all apps and websites, such as chess.com, hulu.com among others. |
| Game | This is the above game. It will play out described above, in a number of rounds. Every round will be a few minutes long, and the game will end. | A simple concept where the players do most of the work. Our product is dependent on the concept of simplicity. Big advantages over similar mobile games is low RAM and processor usage. This is the big selling point. It is fast, it is simple, and it can be played on any smartphone. Similar mobile games would be online chess or Akinator (the 20 question guessing game). However, most of those games have a large amound of overhead, and require a more adept smartphone. |
| Game Matchmaking | A person can setup a game to have random players, or they can set it up to play with friends. You can set it up with a name (and players can enter the game name to join.) | Our product will work about as well as any competitor matchmaking. Similar web-games that use matchmaking are chess games, such as chess.com. Our game has far simpler matchmaking as it only focuses on language. |
| Private Chat | This chat allows people to talk in secret. This can hide information from a potential witch. | This will be a simple chat to allow players to communicate privately. It does not need to be complex. This private chat is most similar to the chat windows in Facebook or Omegle. Ours will lack some of the features like emoji, and will focus solely on text. |
| Public Chat | This is the main way for players to communicate and try to find the witch. Its main function is to remove having to switch between apps to focus on the game. | This will be a simple chat to allow players to communicate privately. It does not need to be complex. Our chat is very simple, similar to the chat of Omegle. You will see the usernames of the people leaving messages. |
| Ads | This is the game’s main method of generating profit. | As the game plays out in rounds, and finding out who dies, putting ads here would come from the google AdSense. This is similar to the ads on Akinator or Youtube. |

3) Data Definition

* Game: One match of Witch Hunt game, from the start till the end, when the witch or the remaining villagers win.
* Player: Users playing the game. Each user will need to sign up and be logged in to play the game. Each username will be a primary key for the game.
* Witch: User who other players try to vote out. There is always only 1 witch per game. This user is chosen at random. The witch wins if he can eliminate all or all but 1 of the players.
* Villager: Players who aren’t the witch. These players vote on who is the witch.
* Remaining Villagers: villagers who have not been killed. These players win if they vote the witch to death.
* Round: One chunk of chatting, then voting on who is the witch.
* Voting: Villagers choose someone to kill. Witch chooses someone to kill. The witches vote will kill 1 person. The person who gets the majority of the villagers vote dies.
* Killed/Lost: The state a player is in if they are voted out or killed by the witch.
* Matchmaking: The pre-game setup to group players for a game. Players can create a unique name for their game, and other people can join that game by using that name. This unique name will be a primary key in the sql code.
* Private Chat: A chat seen only by players allowed in. This is a simple text only chat. Usernames will be listed next to what the individuals say. The player who creates the chat will control who can see and participate in that chat. Each user can create 1 private chat.
* Public Chat: A chat seen by all players. This is a simple text only chat. Usernames will be listed next to what the individuals say.
* Account: User account from which they play the game. The player will need to sign up, otherwise they cannot play the game.

4) Overview, Scenarios and Use cases:

        Jeremie and his friends, (Aelita, Ulrich, Yumi, Odd, William, and Elizabeth) decide to play Witch Hunt. He goes on the site, and creates a custom game. He tells his friends the entrance password and they all join and begin the game.

        Aelita is chosen by the game to be the witch. The players start chatting in the public chat. Ulrich and Yumi create a private chat, and figure if Elizabeth is the witch, she’ll kill Yumi first. If it’s William, he’ll kill Ulrich. Jeremie would kill Odd. They figure Aelita would kill Yumi. She doesn’t tell him that he’d kill William just in case he’s the witch. They also figure it would be best to vote out odd first, as he’s the least predictable one since he’d just mess with them for fun. At the end of Round 1, Odd is killed by a vote, and Elizabeth is killed by the witch.

        At the start of round 2, everyone is suspicious of Yumi since they figure that’s who she’d kill first. Jeremie, Aelita, Ulrich, and William form a private chat. Jeremie and Aelita try to convince William and Ulrich to vote off Yumi, since they need 3 votes. Yumi privately messages Ulrich telling him if she dies and the witch is not found its Aelita. She figures Aelita is smart enough to hide and use the group to kill her off. Just in case, she also sends the message to William. The group votes in round 2, and Yumi is killed off by both the group and the witch.

        At the start of round 3, Ulrich is suspicious of Aelita since she or Jeremie would have the most reason to kill Yumi. She’s the smartest after those 2. William messages Ulrich and says they should kill Aelita, since it did match what Yumi said. They both vote Aelita despite Jeremie and Aelita’s protests of her innocence. Aelita is voted off and Jeremie, William, and Ulrich win the game.

Tie-breakers will be broken by someone chosen at random. If the person doesn’t choose, that person is killed.

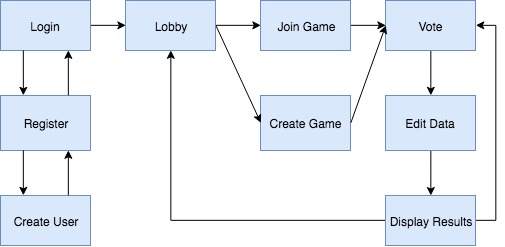
5) High-level function requirements

Our goal is to develop a scalable, mobile-first user interface using Bootstrap for our project. Users will be able to create a login and play locally or over the web. Users should be able to form teams with registered friends or with strangers. A team will consist of no more than 9 players with no fewer than 5 players in a game. Graphics can be provided using JavaScript, with player data being hosted in PHP on the LAMP server. As the game progresses players will be eliminated which will induce changes to the properties in SQL table, which maintains the data for the match. The user interface will provide players the option to vote on other players, whom they believe is the witch. Once either all the players are eliminated, or the witch is found and killed, the game will display a closing screen that informs players whether the witch eliminated all of the citizens, or if the citizens eliminated the witch. The game then returns to the opening screen, where the player is able to start another game.

6) Non-functional requirements

1. Performance - Response time - maximum load time, 1 sec.
2. Usability / Portability – Game must be properly formatted and functional on all iOS and Android mobile devices
3. Accessibility – Users must have an account and login to access the game
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.
5. Security Requirements – Any personal data needs to be encrypted when stored on database
6. Storage – Web server storage requirements 100MB max, Database max size initially 1GB. Individual pages displayed should not exceed 1MB for faster load times.
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.
8. User Signup – Simple and quick signup with limited user personal information, Name, email only.
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.

7) High Level System Architecture and Database Stucture:



The primary languages used to develop the application will be HTML, CSS, Javascript, and PHP. Bootstrap will be the framework used to design the user interface. User data will be hosted on the LAMP server and be stored and accessed using MySQL queries. Supported browsers include Chrome, Firefox, and Internet Explorer. For the user interface, the team plans on using Bootstrap to create a scalable design. The game will be available on major supported browsers such as Google Chrome, Apple Safari, Mozilla Firefox, and Microsoft Edge.

The data associated for the game will be held in two tables on the team’s database. Data for logging in and sign up will be held in a table named *users* with two text fields: username and password. Each time someone attempts to login, the data inputted in the form will be cross referenced with the existing data in the table. If a user exists with the given username and password, the player is signed in. Likewise, when a user attempts to register with a new account, their player data will be saved to *users*.

When a player joins or creates a match and decides they are ready to play, player data will be saved in a table *game*. This table will hold the username of those playing in the match, a Boolean variable that determines whether the player is the witch, a variable that determines if the player is dead, an integer variable that measures the amount of votes the player received, and a Boolean variable that tracks if the Witch voted on the player.

No images, videos, or special file types will be kept on the server, only the items in the tables declared above.

No API’s will need to be created for successful implementation.

No significant non-trivial algorithms or processes are planned to be implemented into the game.

8) High-Level UML diagrams



Player



register.php



login.php



Users



newgame.php



Games



Chat.php



Vote.php

Use Case:

Website

Start New Game

newgame.php

Web Server

Choose Witch

gameUpdate.php

Website

Display if user is witch or villager

gameUpdate.php

Website

Display Chat

chatUpdate()

Timer(chatTime)

Website

Display Vote Choices

Website

Deaths

Website

End Game

If (isDead()) or count<=2

Timer(voteTime)

Sequence Diagram:

9) Identify actual key risks for your project at this time