Updated Product Backlog
Sprint 2
Note: Changes and additions will be in red

Christian McKain, Jacob Riggs, Julian Ng, Cole Compton, Adrian Taubner

Problem Statement:

In 1971, a revolutionary game emerged to the market that became immensely popular called the Oregon Trail. This game was text-based and took advantage of the limited technology that was available at the time. To this day the Oregon Trail is a beloved game to many, however, in the past few decades, despite skyrocketing standards for technology and video games, few developers have been able to produce text-based games anywhere near as widespread or original as this one.

Background Information:

We are in an age where nostalgia is a key aspect of our culture. Reboots of television shows, remakes of movies, and old fashion becoming high fashion are all examples of popular trends that define our culture. Video games are an extreme example of the impact of nostalgia on consumer desires. Games released ten years ago are constantly being remastered and are selling today as well as they did originally. The Oregon Trail is a video game from the past that was loved by many, but has yet to see a worthy predecessor and needs more than a simple re-release. Our project is going to hit that niche and wishes to capture the love of the Oregon Trail and its game mechanics, but adapted to a more modern audience in its content.

Environment:

Backend:

For our backend, we plan on using a MySQL database alongside Java code to handle requests from the server.

Frontend:

For our frontend, we plan on using the Vue javascript framework.

Functional Requirements:

Backlog Id	Functional Requirement	Hours	Status
1	As a user, I would like to be able to create an account.	5	Done
2	As a user, I would like to be able to change my email.	2	Done
3	As a user, I would like to be able to change my password.	2	Done
4	As a user, I would like to be able to continue saved games.	8	Done
5	As a user, I would like to be able to quit a game.	3	Done
6	As a user, I would like to be able to start over.	2	Done
7	As a user, I would like to be able to see my score on the leaderboard.	4	Done
8	As a user, I would like to be able to log in.	5	Done
9	As a user, I would like to be able to log out.	2	Done
10	As a user, I would like to be able to buy items.	4	Done
11	As a user, I would like to be able to view my in-game inventory.	10	Done
12	As a developer, I would like stored user information to be hashed to prevent security breaches.	6	Done
13	As a developer, I would like to prevent SQL injection.	2	Done
14	As a developer, I would like to be able to prevent duplicate users.	2	Done
15	As a developer, I would like the application to require an account to play.	2	Done
16	As a developer, I would like to be able to keep track of users with individual login tokens.	3	Done

17	As a user, I would like it to be functional on chrome, firefox and edge.	1	Done
18	As a user, I would like to be able to play through at least 10 different events.	15	Done
19	As a user, I would like to be able to win or lose the game.	15	Done
20	As a user, I would like to be able to be notified about events that affect my game state.	10	Done
21	As a user, I would like to be able to view other users' scores on the leaderboard.	4	Done
22	As a user, I would like to be able to view collected "trophies".	4	Done
23	As a user, I would like to view the trophies that I earned during a game	4	Done
24	As a user, I would like to be able to see my score at the end of the game	5	Done
25	As a user, I would like to be able to use items during the game	4	Done

Non-Functional Requirements:

Architecture:

This web application will feature a separate front-end and back-end. The frontend will be written in Vue.js and will send requests to pull from the back-end to handle player inventories and games. The back-end will use Java with a MySQL database that will be able to store player inventories, games, and a leaderboard. User queries will be relatively fast and give detailed errors if they occur.

Usability:

The interface for our game will be intuitive and straightforward. It will be clear how to access different features relating to user accounts and gameplay, and we expect to have at least 95% server uptime.

Security:

We will take measures in order to protect our users' accounts. The users will be able to securely login to their accounts with login API implemented in our application. Our application will also prevent vulnerabilities to corrupting our database by covering exploits such as SQL injections.

Use Cases:

Case 1:

Create an account

Action	System Response
1.On startpage, click on "Create Account" button	2.Shows Account Creation Page
3.Enter email, username and password and click "Create" button	4.Sends request to backend
	5.If valid credentials, store data in database
	6. Display home page

Case 2:

Change Email

Action	System Response
1. On home page, click on "Settings" button	2. Display settings page
3. Enter new email address and click "Change Email" button	4. Sends request to backend
	5. If valid email, change database entry accordingly
	6. Display successful change message

Case 3:

Change Password

Action	System Response
1. On home page, click on “Settings” button	2. Display settings page
3. Enter new password and click “Change Password” button	4. Sends request to backend
	5. If valid password, change database entry accordingly
	6. Display successful change message

Case 4:

Continue saved game

Action	System Response
1. On homepage, click “Continue Game” button	2. Sends request to backend
	3. Server fetches previous gamestate from database
	4. Changes display game page according to fetched data to continue game

Case 5:

As a user, I would like to be able to quit a game.

Action	System Response
1. User is in an active game	
2. User presses the exit button in the top right	3. Client posts to server telling it to save the game
4. User is returned to the start game tab	

Case 6:

As a user, I would like to be able to start over.

Action	System Response

1. While the user is in an active game	
2. User selects the reset button in the top right	3. Client post to server to reset game
	4. Server post fresh character and room
5. User is at the start of a new game and can see a fresh stat block	

Case 7:

As a user, I would like to be able to see my score on the leaderboard

Action	System Response
1. User has completed one game	
2. User selects the high score tab	3. Server responded with top 20 players
4. If user is within the top 20 they will be able to see their score along with the other 19 players	

Case 8:

As a user, I would like to be able to log in

Action	System Response
1. Users opens the website and is redirected to the login screen	
2. User inputs and username and password	3. Server checks if it is a valid username password combination
	4. If valid the user is redirected to the start game tab
	5. If invalid the user is informed the username or password are incorrect

Case 9:

As a user, I would like to be able to log out.

Action	System Response
1. User is currently logged into service	
2. User selects the logout button	3. Tokens are cleared and user is returned to the login screen, and all unsaved data is pushed to the server

Case 10:

Buy items

Action	System Response
1. Select item to be purchased	2. Item is highlighted in the UI
3. Click purchase button	4. Item is added to inventory, gold is removed from player currency
	5. UI values update to show the new item and reduction in gold

Case 11:

View my inventory

Action	System Response
1. Locate inventory sidebar and begin to scroll through it	2. Items are displayed in a grid and scrolled through
3. Click an item	4. Item information is displayed in item detail section next to inventory sidebar

Case 12:

Hash passwords

Action	System Response
1. Register a new user with a valid password	2. Application creates a hashed password using the user-provided password
	3. Hashed password is stored to the database

4. User sees message indicating successful account creation	

Case 13:

Prevent SQL injections

Action	System Response
1. Try to create an account with a username or password containing a SQL string	2. Application detects the attempted breach and responds with an error message

Case 14:

Prevent duplicate users

Action	System Response
1. On the login page select register	2. Display registration page
3. Enter Email/Username that is already being used	4. Send request to Database
	5. Database recognizes username/email exists in the database
	6. Notifies user that either username, email or both are already used

Case 15:

Require an account to play

Action	System Response
1. Login using account credentials	2. Confirm correctness of credentials and notify database
	3. Display the homepage on success

Case 16:

Individual login tokens

Action	System Response
1. Login using account credentials	2. Confirm successful login
	3. Assign randomized token to the user
	4. Display homepage for user
	5. Use token to keep track of the user and any information attached to that user during his or her session

Case 17:

Functional on chrome, firefox, and edge

Action	System Response
1. Uses Google Chrome to search our website	2. Recognize connection and successfully display login page
3. Uses Firefox to search our website	4. Recognize connection and successfully display login page
5. Uses Google Chrome to search our website	6. Recognize connection and successfully display login page

Case 18:

Possibility to play through at least 10 unique events

Action	System Response
1. Presses start game button	2. Generates event using database and displays for the user
3. Determines what they want to do, and then selects an option	4. Uses user decision to determine what the result is going to be using the database
	5. Displays event that represents result of user decision
6. Assuming game has not been won or lost, user observes results and then presses continue	7. Generates event that has not already been chosen and displays it for the user

Repeat steps 3-7 until game win or loss

Case 19:

Win or lose the game

Action	System Response
1. Player takes action that wins or loses the game	2. Recognizes action and records it
	3. Displays win or lose message

Case 20:

Notify about events that affect the game state

Action	System Response
1. Player takes action that triggers an event	2. Recognizes action and saves the effect as a new game state
	3. Display changes of game state
4. Player can now choose action based on new state	

Case 21:

View leaderboard

Action	System Response
1. Presses the leaderboard button	2. Server pulls the current top 20 scores for the game
3. User can view the current top 20 players names scores and gold accumulated	

Case 22:

Collect trophies

Action	System Response
1. Player takes action that triggers winning a new trophy	2. Recognizes action and save new trophy
	3. Display new trophy collected message

Case 23:

View earned trophies from game

Action	System Response
1. End game	2. Sends array of earned trophies from game database

Case 24:

See game score while playing

Action	System Response
1. Acquires or loses score while playing the game	2. Server updates the score in the database
	3. Game page display is updated with new score in field

Case 25:

Use item in game

Action	System Response
1. Select a special scenario option to use item	2. Server sends back scenario outcome from using item
	3. Item is removed from inventory