The Survivors

Igor Ivanov, Nathan Robbins, Jane Tian

CS 4540

Abstract

Our project is a simplified survival simulation game. Users complete trivia style challenges in order to earn virtual currency, which they can then use to buy items to prepare them for a variety of disasters. For example, a user can buy a respirator to help prepare them for potential wildfires or volcanic eruptions. Users can then compare their scores against other users both on a total score level or a per-disaster score level, in order to see where they stand against the competition.

This project is designed to be a fun way for users to engage with various trivia and get some idea of the types of materials that might help them be prepared for certain disasters. While some disasters are not necessarily likely for any particular user to experience, like a volcanic eruption, disasters like floods or earthquakes can affect many people, and users will hopefully come away from our game with a little bit of awareness of some of the materials they might consider purchasing to prepare themselves for events like these.

URLs

Nathan: http://ec2-3-86-186-25.compute-1.amazonaws.com/
Igor: https://18.218.128.50:44333/
Jane: https://ec2-34-204-79-219.compute-1.amazonaws.com/

Introduction

The project we created is a Fallout-themed survival preparation game with a simple gameplay loop of completing trivia challenges, earning currency from said challenges, then buying items in the store with the earned currency to help prepare for disasters. The user must first create an account to access these pages as they require authorisation. The user can also view and compare their high scores against other users once their account has been created. Each page can be navigated to with the use of the top navbar, where the user is also able to register or sign-in to their account.

The challenges page consists of a variety of trivia questions, where the user has the ability to select between a range of difficulty options, or all of the questions. These different categories can be selected through a button group at the top of the page that has tabs the user can click for each difficulty. The questions are pulled from an excel file that contains a list of questions and answers. The question to be displayed is selected by rolling a random id then selecting the corresponding question from the file. The answer checking system utilises fuzzy string matching, or approximate string matching, to determine whether the entered answer is correct. Upon clicking a different category, the page displays a green overlay with a loader icon while the page renders and loads.

The store page pulls all the items stored in the database and displays their name, cost, score they give, disasters these scores count towards and an image of the item. Each item also has a detail button that the user can click to display a bootstrap modal which shows the item name, cost, score and amount currently owned by the user using Ajax. The quantity of items that can be purchased at once can be changed using ‘-’ and ‘+’ buttons, where the item(s) can then be purchased upon clicking the buy button. Clicking this button pops up a sweet alert confirming the number of items to be purchased and the total cost. If confirmed, the bought item will be added to the user’s owned items and the appropriate amount of money deducted from the user’s total. If the user does not have enough currency, the transaction will fail, displayed to the user through another sweet alert.

The high score page also features a number of different tabs that display a highscore table for each disaster, as well as a total score table. The scores are calculated by pulling the scores of each user’s owned item from the database and totalling them, then ranking them from highest to lowest. Each high score is displayed alongside the user’s name, which can be clicked to display the user’s individual details page. This page shows the user’s name, total score, inventory and disaster preparedness. The inventory shows each item owned by the user, the quantity of said item owned and the total score provided by the item. Disaster preparedness shows the score for each disaster, displayed as a percentage.

Feature Table

Feature Name	Scope	Primary Programmer	Time Spent	File/Function	LoC
Scaffolded in IdentityUsers and extended with our functionality in ApplicationUser	Back-end	Nathan	4 hrs	Identity/IdentityHostingStartup.cs, DBModels/ApplicationUser.cs, ItemInstance.cs	~70
Challenges db models, connection strings	Backend	Igor	5 hrs	DBModels/PrepContext.cs; Appsettings.json, Startup.cs	75
Challenges db seeding/import	Backend	Igor	3.5 hrs	Utils.cs	119
Challenges logic and styling	Backend/ Frontend	Igor	10 hrs	Controllers/ChallengeController; Views/Challenge/Index.cshtml; Views/Shared/_QuestionPartial.cshtml; wwwroot/js/challenge.js wwwroot/css/site.css	448

Overview page	Frontend	Igor	1 hr	Views/Home/Index.cshtml	62
Additional items seeding	Backend	Igor	0.5 hr	DBModels/UserDBInitializer	152
AWS deployment	DevOps	Team	8 hrs	AWS VM	N/A
DB Seeding code	DB	Nathan with others	4 hrs	DBModels/UserDBInitializer.cs	~110
Overall Highscores Page	Frontend	Nathan	4 hrs	Controllers/UserController.cs, Views/User/Highscores.cshtml	~70
Disaster Specific Highscore Pages	Frontend	Nathan	4 hrs	Controllers/UserController.cs, Views/User/DisasterScores.cshtml	~85
Highscore pages dynamically link to all other available score pages	Frontend	Nathan	1 hr	Views/Shared/DisasterLinks.cshtml	~30
User details page that displays user scores and items owned	Frontend	Nathan	4 hrs	Controllers/UserController.cs, Views/User/Details.cshtml	~130
Set-up models and relations for Items and Disasters	DB	Nathan	2 hr	DBModels/Item.cs, DBModels/Disaster.cs, DBModels/ItemDisaster.cs	~40
Store page that displays all items that can be bought	Frontend	Jane	8 hr	Controllers/ShopController.cs, Views/Shop/Index.cshtml, wwwroot/css/shop.css	~200
Change the quantity of items to be bought	Frontend	Jane	2 hr	Views/Shop/Index.cshtml, wwwroot/js/shop.js	~15
Buy items that deducts the appropriate amount of money from the user and adds item to their owned items using Ajax and SweetAlerts	Frontend/ Backend	Jane	3 hr	Views/Shop/Index.cshtml, wwwroot/js/shop.js, Controllers/ShopController.cs	~80
View details about the items in the store using Ajax	Frontend/ Backend	Jane	2 hr	Views/Shop/Index.cshtml, wwwroot/js/shop.js, Controllers/ShopController.cs	~60

Individual Contribution

Team Member	**Time Spent on Proj**	**Lines of Code Committed**
Igor	28 hrs	856
Nathan	23 hrs	535
Jane	17 hrs	735

Summary

Our team project is a survival themed game whose purpose is to help inform users about disasters and what items may be needed to prepare for them. The project was split into three sections: the challenges, store and high score pages, each assigned to a team member to work on. The challenges page presents the user with trivia questions which allow them to earn currency, the store page then allows the user to spend this currency to purchase items, and the high score page collates the items owned by each user to display their preparedness score for each disaster. While mostly keeping to our own sections, each team member stepped up to help others when necessary and also took care of other responsibilities, such as configuring required models and users or setting up the site's general styling.

As a team, we performed to a good standard, as each member contributed large amounts to the project to produce an overall worthwhile and substantial application. The team coordinated together smoothly and had good teamwork and communication. One issue we encountered was the setting up of the site on AWS. As a team, we stumbled across a large number of errors while attempting to deploy our site, though through research, advice and eventual fixes, we were able to resolve these issues and successfully deploy the application.

The application leverages usage of Ajax and SweetAlerts, both of which could be applied using the experience we gained from class, as well as other aspects of web design, such as file importing, CSS keyframe animation and usage of libraries to implement functions such as fuzzy string matching. The application was worthwhile as it allowed us to apply knowledge we gained throughout the semester but also learn new things and build on our existing foundations, and use it to create a fun and informative website with a large range of functionality.