Product Backlog - The Hogwarts Hobo game

Team 29
Adam Sorrenti, #500903848
Kritarth Shah, #500907217
Kateryna Shkinova, #500911953
Keith Jose, #500816528
Raunak Bajaj #500912844

Table of Contents:

Overview:

This agile Product Backlog clearly lists the stories developed on the basis of the specs; the priority for each story on the scale 1 to 3, with 3 being the highest priority; and the stories / tasks finished in this iteration. This Product Backlog also lists the estimated effort (in perfect person-weeks) for each story / task, as well as the actual effort for each of the finished stories / tasks. Also shown, is a velocity diagram overviewing how much work has been delivered for each sprint along with burndown charts that show how quickly our team burned through our customer's user stories.

Product/Sprint Backlog Items

Original Requirements	Story	Initial Priority (1-3)	Engineering Tasks	Effort Estimate	Sprint Details
Basic user interface	A *user* will be output a certain set of standardized **instructions based on current game parameters**(such as *current track, health status, and hobo information*) to interact in the game.	3	Implement console-based UI for game startup and exit. Consider standardized output messages for user interaction and synchronization with current game parameters.	0.8 perfect person weeks	Initial Sprint The user interface of the game was the first story to be completed. The health counter was created displaying the current health of the user starting at 100. A current track tracker was made to remind the user what track they were on. Information received from the hobos was also displayed for the user. The story was completed much later than expected due to COVID-19. Actual Effort: the effort to implement all tasks in this story was according to the estimated effort

Train Probability Distribution	The Hogwarts Hobo game must have an algorithm to simulate trains passing through the tunnels in random intervals of time. These are the trains that the user must avoid as part of the objective of the game.	3	Use a negative exponential distribution and Poisson processes to model the time it takes for a train to pass through the tunnel and the time it takes for the next train to arrive.	1.2 perfect person weeks	Creating the train probability distribution was the second story to be completed.Using the negative exponential distribution and Poisson processes, the probability distribution was successfully added. Actual Effort: it took more effort than expected because of lack of knowledge on probability.
Hobos messages	Hobos will deliver information to the user based on the *train probability distribution* using a paper airplane. There will be a percentage the user will have of catching the airplane due to the **darkness** of the tunnel.	3	Implement multiple states that use a random number generator that determines whether or not the user had caught the paper airplane. If caught, output a message relaying the time until the train arrives.	0.2 perfect person weeks	Generating the hobos messages was the third story to be completed. Using the information on the train probability distribution, it was outputted to the paper airplane and sent to the user as a prompt. The user would have a random percent of possibly catching the airplane.

					The development team completed this story within the expected timeframe. Actual effort was the same as estimated.
Health Status	The user will start the game with a 100% health status. Every time the player hits or runs into an oncoming train, the user will lose 10% of their health. The goal is to keep the highest health possible by the end of the game.	3	Implement a status variable that would be initialized at a 100 and at each collision with a train object, the variable would decrease by a constant of 10 health points.	0.2 perfect person weeks	Creating the health status was the last story to be completed for the first sprint. The development team completed this story before the expected timeframe. Actual effort was less than expected, since some of the tasks of this story were already implemented in the "Basic User Interface" story.
Syncing game time with real time	Each second of the game represents each second in real time.	1	Implement a time function that keeps track of the duration of the game and that each second of gameplay represents a real second rather than one time unit	0.4 perfect person weeks	The story was not completed. Actual effort: 0

			(unknown duration).		
Move tracks on collision	A *user* upon collision with a train *on the current track* should be **moved to the next higher- or lower-numbered track.**	2	Interfacing with our UI and health system, the user should receive a standardized output based on the number of tracks in the current game. Users will be moved to the next highest- or lower-numbered track depending on location and current trains based on a simple algorithm. Interfacing with optimization programs should take place here also.	0.4 perfect person weeks	Second Sprint Moving the user to a higher- or lower-numbered track upon collision was the first story to be completed in the second sprint. The development team was able to finish this within the expected timeframe. Actual effort was as expected by our estimate.
Hobo bands	Multiple users playing the game may choose to form bands, in order to move together in synchronization. Only the leader of the group will be able to communicate with the rest and give orders as to which track to jump to	1	Through an airplane message, a user will receive an invitation to join a band. If the user accepts it, he/she will now be receiving orders from the captain and will be considered a part of that group until he/she chooses to leave the group.	0.6 perfect person weeks	Not implemented. Actual effort: 0

	next.				
Machine learning model for train position	A machine learning model will be implemented to predict the likelihood of a train passing through the tunnel on a particular track. This will help in minimizing the number of collisions as we will be able to better determine the most suitable track to jump to.	1	Collect data on individual train patterns to use in a machine learning model. Implement the machine learning model in the simulation algorithm to minimize the number of collisions experienced by the player.	1.2 perfect person weeks	Created a machine learning model to study the pattern of the trains and the train probability distribution. This was the third story to be completed. The development team completed this later due to the difficulty of machine learning. Actual effort was about 1.5 perfect person weeks.
Take in game parameters	The *tester* should be inputting game parameters(such as, *number of tracks, the mean values of train duration and inter-train distance, the probability distribution of train duration and inter-arrival time, the number of other hobos at that same tunnel exit*).	1	Special case UI should be developed for testing purposes providing more specific analytics and input areas for game parameters. Error handling should be implemented to catch fatal game parameters(e.g. Zero number of tracks, etc.)	0.3 perfect person weeks	The UI for testing and error handling was the next story to be completed. The development team completed this story in the expected timeframe. This was the last story to be completed in the second sprint. Actual effort: as

					expected by our estimate
Predicting the behaviour of other Hobos	The machine learning model will try to predict whether the hobos are sending false messages. This will help users to make a decision on whether he/she should follow other hobos' instructions.	1	Collecting data on other users and their decision making the model will use machine learning techniques and predict if the message the user receives is false. If so, the user will be prompted to ignore other hobo's messages.	0.8 perfect person weeks	The story is not completed. Actual effort: 0
Output results report	A results report to indicate experimental verification: parameter values, duration of test runs, and percentage of successes and failures (collisions) for the benchmark algorithm.	1	Interface with current test case parameters and game analytics to output results via a log file. Log file should have standardized outputs to be easily considered for errors and success/failure	0.5 perfect person weeks	Created a log file to output the results of each test. This was the second story to be completed. The development team completed this story later than the expected time, due to multiple other assignments. But the actual effort was the same as estimated.
Project Documentation	Includes the README.txt file and comments along	1	Create a file that lists the members of the project and	0.5 perfect person	Creating the documentation of the

	with the source code of the project.		the contents of the package.	weeks	project was the last story to be completed. The development team completed this in the expected timeframe. There were no more tasks to be completed. Actual effort: we were able to complete it in less than our estimated 0.5 perfect-person weeks.
Lying Hobos	There will be times when the messages received from hobos may not always be truthful. This will add a bit more thinking whether or not the player should act accordingly to the information.	1	Generate a random track number that displays in the hobo messages that may or may not be the track in which the train will pass on.	0.1 perfect person weeks	Created the function which randomizes the track that appears in the hobos messages. The development team completed this story within the expected timeframe. Actual effort was the same as estimated.
Animation Gameplay	The game will contain animation such as the player with hobos chasing, incoming trains and tracks to move between,	1	Using Unity to create the game player. Visual objects such as trains, players, hobos and tracks need to be created and referenced to the	1.2 perfect person weeks	Not completed. Actual effort: 0

			backend of source code.		
Jump in synchronization	The band leader of the hobos would shout which track to jump to before or after a jump. This would move the entire band where to go.	1	Outputs a higher or lower-numbered track in which the leader informs the rest of the band where to jump.	0.8 perfect person weeks	Not implemented. Actual effort: 0

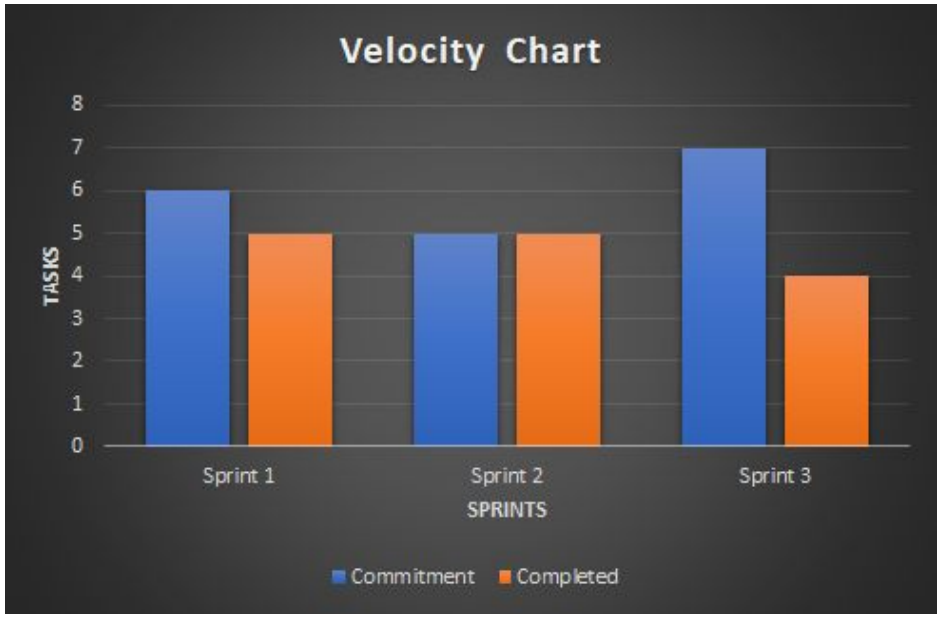


Figure 1: Overall velocity diagram showing how much work has been delivered for each sprint.

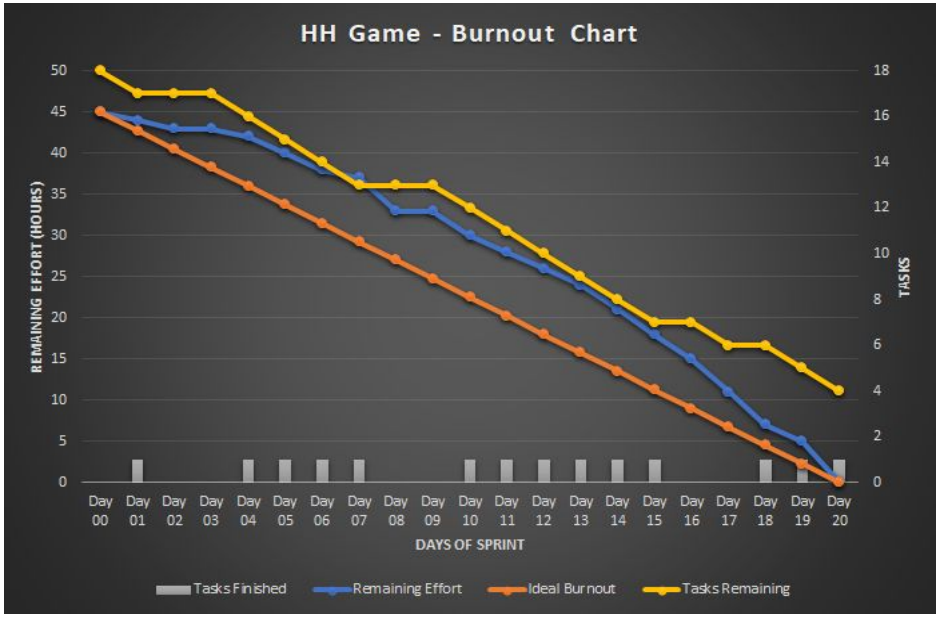


Figure 2: Overall burndown chart showing how quickly our team burned through our customer's user stories. Note the flat period

where work halted due to the forced transition to online work environments due to the COVID-19 pandemic.

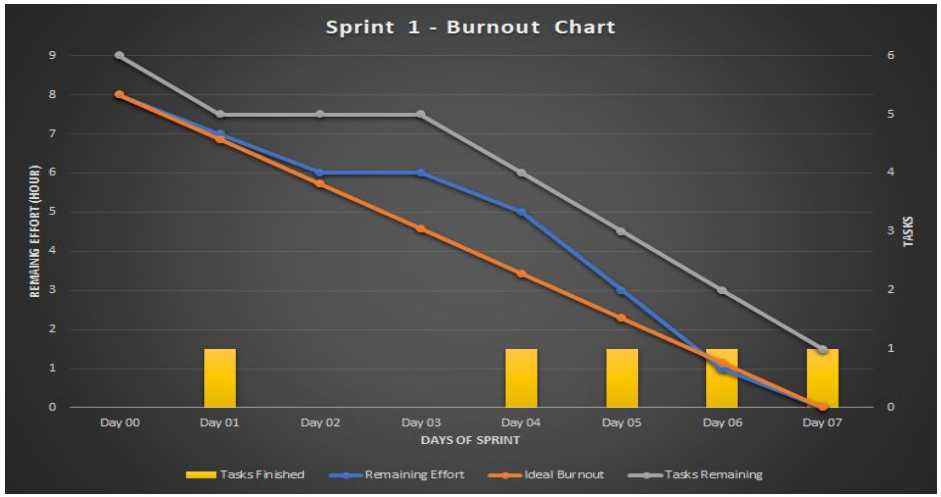


Figure 3: Sprint 1 Burnout chart showing teams burnout rate through user stories.

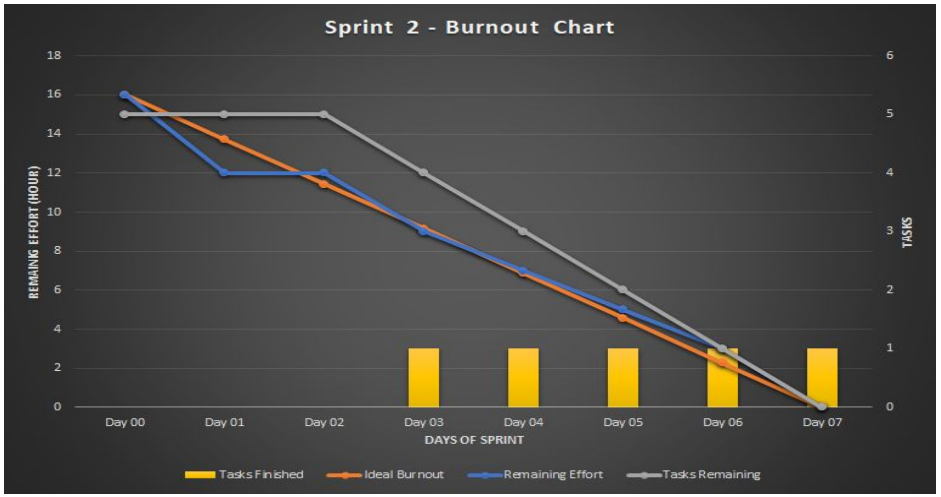


Figure 4: Sprint 2 Burnout chart showing teams burnout rate through user stories.

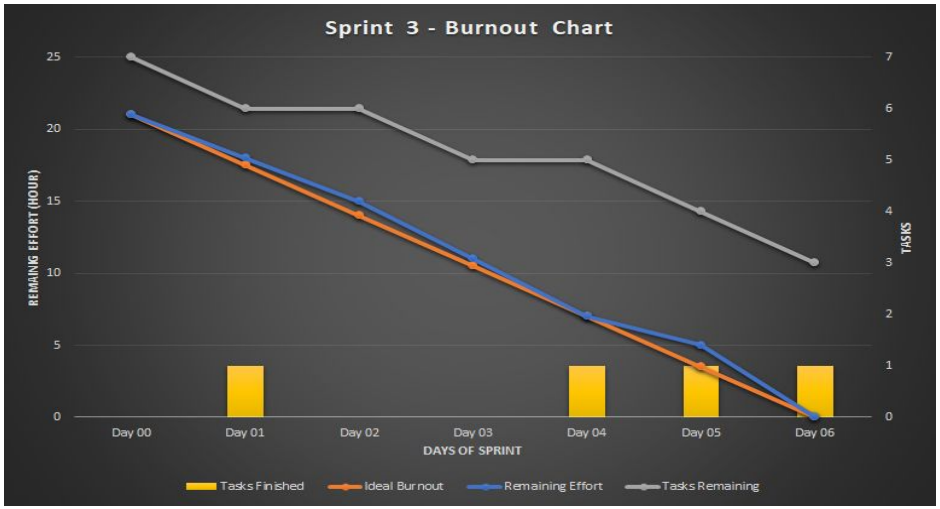
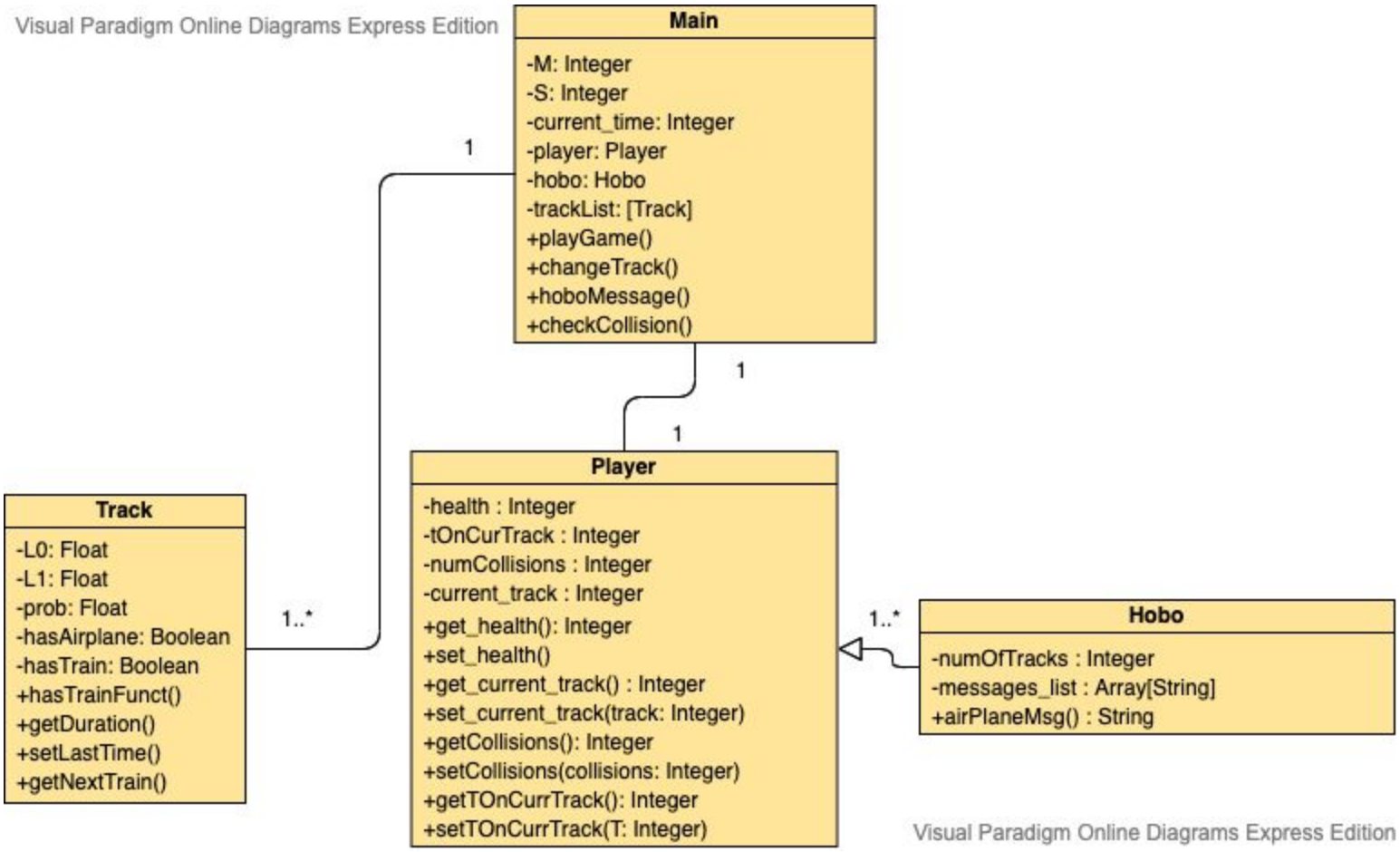


Figure 5: Sprint 3 Burnout chart showing teams burnout rate through user stories.

Hogwarts Hobo Game Class Diagram


Visual Paradigm Online Diagrams Express Edition
Main
-M: Integer
-S: Integer
-current_time: Integer
-player: Player
-hobo: Hobo
-trackList: [Track]
+playGame()
+changeTrack()
+hoboMessage()
+checkCollision()
1
1
1
Track
-L0: Float
-L1: Float
-prob: Float
-hasAirplane: Boolean
-hasTrain: Boolean
+hasTrainFunct()
+getDuration()
+setLastTime()
+getNextTrain()
1..*
Player
-health : Integer
-tOnCurTrack : Integer
-numCollisions : Integer
-current_track : Integer
+get_health(): Integer
+set_health()
+get_current_track() : Integer
+set_current_track(track: Integer)
+getCollisions(): Integer
+setCollisions(collisions: Integer)
+getTOnCurrTrack(): Integer
+setTOnCurrTrack(T: Integer)
1..*
Hobo
-numOfTracks : Integer
-messages_list : Array[String]
+airPlaneMsg() : String
Visual Paradigm Online Diagrams Express Edition

playGame() Function Flowchart