Product Backlog

Team 11

Austin Wirth, Emma Wynne, Evan Walsh, Harris Christiansen, Miranda Mott

---

‘A-Maze-Balls’

Problem Statement

The team seeks to create a physics based multi-platform game where players shoot a ball, similar to billiards, but with varying obstacles that must be navigated and a goal that the ball must reach. This app will appeal to the casual gaming market on mobile app stores.

Background Information

Gaming is a very large market, especially among mobile app stores. This game will appeal to users who enjoy puzzle and physics based games. There will be a large number of levels so that the user can play for a long period of time. Our team decided to create this application because it is an interesting and challenging problem to create a game that is fun to play many times.

Environment

The app will be developed using the Unity Game Engine for mobile (iOS and Android). It will be written in Javascript/C# for Unity and use a SQL database for score tracking.

Functional Requirements

1. As a user, I would like to choose which level to play from a list of available choices.
2. As a user, I would like to fire the ball toward the target.
3. As a user, I would like to unlock the next level upon completion of the current one.

4. As a user, I would like to view the leaderboard.
5. As a user, I would like to upload my score to a leader board.
6. As a user, I would like to manipulate the ball through the level into a target goal.
7. As a user, I would like the game to keep track of my wall bounce count.
8. As a user, I would like to track my lowest (best) score for each level.
9. As a user, I would like to track my total score for the game.
10. As a user, I would like to be able to create an account.
11. As a user, I would like to be able to log into an account and remain logged in.
12. As a user, I would like to be able to log out of my account.
13. As a developer, I would like the balls to interact with obstacles, which must be navigated to reach the goal.

--IF TIME ALLOWS--

14. As a user, I would like to be able to create levels.
15. As a user, I would like to store the levels that I have created.
16. As a user, I would like to be able to share levels.
17. As a user, I would like to be able to browse shared levels.
18. As a user, I would like to be able to play shared levels.

Non-Functional Requirements

1. As a user, I would like to access this product on the web and mobile.
2. As a user, I would like the app to have fast response times.
3. As a user, I would like the app to use as little device memory as possible.
4. As a sysadmin, I would like to the site be able to handle at least 100 users at once.
5. As a developer, I would like to make the levels increasingly difficult.
6. As a developer, I would like to create at least 10 levels.
7. As a user, I would like game physics to respond as expected.
8. As an admin, I would like to be able to delete leaderboard records.
9. As a sysadmin, I would like the app to have strong security to protect user passwords
10. As a sysadmin, I would like user authentication to be reliable

Use Cases

Case: Launch a new game	**System Responses**
1. Choose “Play” from the main menu	2. Level menu opens
3. Select a level	4. Confirm level dialog appears

5. Confirm selection

6. Dialog disappears

Case: Fire the ball

1. Press left and right arrows to aim

System Responses

2. Ball firing line will move accordingly

3. Tap on the ball to fire

4. Ball will move though level

Case: Unlock next level

1. The user gets the ball in the target

System Responses

2. Congratulations dialog appears

3. User taps to dismiss dialog

4. Dialog with option to go to next level or exit

5. Select option

6. Dialog disappears

Case: View the leaderboard

1. Choose “High Scores” from the main menu

System Responses

2. List of levels is displayed

3. User selects a level to view scores

4. List of high scores for that level is displayed

5. User clicks “Back” to return to the previous menu

6. Main menu is displayed again

Case: Upload new high score

1. User enters a display name

System Responses

2. Send name and score to the database

Case: Manipulate ball to target

1. Tap ball to fire

System Responses

2. Ball will move until stopped by friction

3. Repeat until ball reaches goal

4. Backend records score

Case: Keep bounce count

1. Ball strikes wall.

System Responses

2. Game increases bounce count.

Case: Track best score per level

1. User completes level.

System Responses

2. Game stores level score.

Case: Track overall best score

1. User completes level.

System Responses

2. Game calculates new overall score.

Case: Create an account

1. User clicks ‘Sign Up’

System Responses

2. Loads form for signing up

3. User fills out form

4. User clicks ‘Create’

5. Game validates form

Case: Log into an account

1. User clicks ‘Log In’

System Responses

2. Shows fields for username & password

3. User enters credentials

4. Verify log-in information is correct

5a. If correct, go to choose level screen

	5b. If incorrect, prompt to enter again
Case: Remain logged into app	**System Responses**
1. User opens app	2. Checks if logged in, loads accordingly
Case: Hit obstacles in levels	**System Responses**
1. User launches ball toward obstacle	2. Ball strikes obstacle and bounces off

--IF TIME ALLOWS--

Case: Create a level	**System Responses**
1. User selects “Create Level” from menu	2. User is presented with a level creator with tools to add obstacles to a blueprint
3. User selects a new obstacle to add or an existing obstacle to edit	4. User is prompted to rotate or reposition
5. User chooses a location/orientation for the obstacle and confirms it	6. The obstacle location is saved 7. The user is prompted to either add or move more obstacles, or to finish and save the level.
Case: View created levels	**System Responses**
1. User selects ‘View My Levels’	2. Populate a view with user’s created levels
Case: Share a level	**System Responses**
1. Publish completed level	2. Level is added to the shared level DB
3. Name level and difficulty	4. Level is made available to the general public
Case: Browse shared levels	**System Responses**
1. User selects ‘View Shared Levels’ levels	2. Populate a view with all user-created
Case: Play a shared level	**System Responses**
1. User selects a level	2. Confirm level dialog appears
3. User confirms selection	4. Dialog disappears