**Product backlog:**

### **Epic 1: Bingo Card Generation -> Priority: 1**

* **Story: As a player, I want to generate a Bingo card so that I can have a board to play on.**
  + **Task 1.1:** Design Bingo card UI (5x5 grid layout).
  + **Task 1.2:** Implement logic to generate random, non-repeated numbers.
  + **Task 1.3:** Ensure center space (if free) is automatically marked.
  + **Task 1.4:** Test card generation for uniqueness and correctness.
* **Story: As a player, I want to receive a clean 5x5 Bingo grid with random, non-repeated numbers so that I have a valid card to play with.**
  + **Task 1.5:** Reset functionality to clear and regenerate a new card.
  + **Task 1.6:** Validate no duplicate numbers within the same card.
  + **Task 1.7:** Write unit tests for card generation.
  + **Task 1.8:** Verify card generation, randomness, no duplicates, center marking, and reset via unit ,and integration.

### **Epic 2: Number Calling and Validation -> Priority 2**

* **Story: As a developer, I want to draw numbers randomly without repetition and broadcast them to all players at the same time so that everyone plays fairly.**
  + **Task 2.1:** Implement number pool (1–75 or game-defined range).
  + **Task 2.2:** Ensure random draw without repetition.
  + **Task 2.3:** Create server-side broadcast to players.
  + **Task 2.4:** Build terminal listener to display called numbers in real time.

* **Story: As a player, I want to confirm when a called number appears on my Bingo card so that the system can automatically validate and mark it.**
  + **Task 2.5:** Allow player to mark a number on card.
  + **Task 2.6:** Auto-validate against the list of drawn numbers.
  + **Task 2.7:** Highlight marked numbers visually on the board.
  + **Task 2.8:** Store validated state in database/session.
  + **Task 2.9:** Validate number pool, random draws, real-time broadcasting, marking, and storage via unit, integration, and E2E tests.

### **Epic 3: Bingo Claim & Validation -> Priority 3**

* **Story: As a player, I want to announce when I achieve a (row/column/diagonal or Bingo) so that the game engine can check and confirm my claim.**
  + **Task 3.1:** Add "Claim Bingo" command.
  + **Task 3.2:** Implement algorithm to check rows, columns, and diagonals for completion.
  + **Task 3.3:** Trigger server validation on claim.
  + **Task 3.4:** Provide feedback (success/failure) to the player.
* **Story: As a developer, I want to reject invalid Bingo claims so that players cannot win without meeting the winning condition.**
  + **Task 3.5:** Add logic for detecting false claims.
  + **Task 3.6:** Display warning/error message for invalid claims.
  + **Task 3.7:** Log rejected claims for audit/debugging.
  + **Task 3.8:** Test claim validation, rejection logic, feedback, and logging via unit, integration, and E2E.

### **Epic 4: Multiplayer & Game Flow -> Priority 4**

* **Story: As a player, I want to join a multiplayer Bingo game so that I can play alongside others in real time.**
  + **Task 4.1:** Implement player authentication (username/session).
  + **Task 4.2:** Allow multiple players to join a lobby.
  + **Task 4.3:** Sync Bingo cards across players.
  + **Task 4.4:** Enable real-time updates via WebSocket or similar.
  + **Task 4.4.1:** If the session is continued, points get assigned per user
  + **Task 4.4.2:** Assure high standard of an UX while playing multiplayer
* **Story: As a developer, I want to declare the first player with a valid Bingo as the winner and end the round so that the game progresses fairly and efficiently.**
  + **Task 4.5:** Monitor first valid claim.
  + **Task 4.6:** Broadcast winner announcement to all players.
  + **Task 4.7:** Lock game board once winner is declared.
  + **Task 4.8:** Provide option to start a new round.
* **Story: As a player, I want a point system that provides excitement to online lobbies.**
  + **Task 4.9:** Gain different quantities of points based on win type (little/big bingo)
  + **Task 4.10:** Save points round to round in a continued lobby
  + **Task 4.11:** Potentially have the ability to purchase a second card with points.
  + **Task 4.12:** Verify authentication, lobby, synchronization, real-time updates, winner logic, and points via unit, integration, and E2E.

**Feedback analysis:**

* Presentation of the multiplayer feature could be conveyed in a more user-friendly manner.

**Additions:**

New task added to the product backlog, more specifically to Epic 4 ( Multiplayer ) to assure that the feature is conveyed in a clear manner. This was implemented because of the feedback, where multiple groups seemed to be confused about the multiplayer option our game implements. Furthermore, we have implemented an extra task in each Epic in order to test and validate the correct functioning of the game features.

In terms of the prioritization of our Epics, we felt that giving more importance to the necessary ones for the basic functioning of the game was more crucial than ones that enhanced the experience of the users. Hence why Multiplayer and game flow (not essential for functioning unlike others) was given the least importance. Furthermore, we found that the natural progression of our backlogs was logical, given that in order to be able to implement the next feature, the previous one had to be functioning.

**Sprint Planning**

**Sprint 1 (3/10-17/10)**

**Features/Tasks:**

* 1.1: Design Bingo card UI (5x5 grid layout).
* 1.2: Implement logic to generate random, non-repeated numbers.
* 1.3: Ensure center space (if free) is automatically marked.
* 1.4: Test card generation for uniqueness and correctness.
* 1.5: Reset functionality to clear and regenerate a new card.
* 1.6: Validate no duplicate numbers within the same card.
* 1.7: Write unit tests for card generation.
* 1.8: Verify card generation, randomness, no duplicates, center marking, and reset via unit and integration.

**Acceptance criteria for the sprint:**

* Users can successfully generate a new 5x5 Bingo card.
* The generated card displays numbers that are random and do not repeat within the same card.
* The center space of the Bingo card is automatically marked (if applicable to the game rules).
* Users can reset and regenerate a new Bingo card, clearing the previous one.
* The system validates that there are no duplicate numbers on a single Bingo card.
* Unit tests for the card generation logic are implemented and pass successfully.

**Sprint 2 (17/10 - 31/10)**

**Features/Tasks:**

* 2.1: Implement number pool (1–75 or game-defined range).  
  2.2: Ensure random draw without repetition.
* 2.3: Create server-side broadcast to players.
* 2.4: Build terminal listener to display called numbers in real time.
* 2.5: Allow player to mark a number on card.
* 2.6: Auto-validate against the list of drawn numbers.
* 2.7: Highlight marked numbers visually on the board.
* 2.8: Store validated state in database/session.
* 2.9: Validate number pool, random draws, broadcasting, marking, and storage via unit, integration, and E2E tests.

**Acceptance Criteria for the Sprint:**

* **Number Calling:**
  + The system can implement a number pool (1-75 or a game-defined range).
  + The system can ensure random drawing of numbers without repetition.
  + The system can broadcast called numbers to all players simultaneously.
  + Players can view called numbers in real-time.
* **Number Marking and Validation:**
  + Players can mark a called number on their Bingo card.
  + The system automatically validates marked numbers against the list of drawn numbers.
  + Marked numbers are visually highlighted on the Bingo board.
  + The validated state of marked numbers is stored in a database or session.

**Sprint 3 (31/10 - 14/11):**

**Features/Tasks:**

* 3.1: Add "Claim Bingo" command.
* 3.2: Implement algorithm to check rows, columns, diagonals.
* 3.3: Trigger server validation on claim.
* 3.4: Provide feedback (success/failure) to the player.
* 3.5: Add logic for detecting false claims.
* 3.6: Display warning/error message for invalid claims.
* 3.7: Log rejected claims for audit/debugging.
* 3.8: Test claim validation, rejection, feedback, and logging via unit, integration, and E2E.

**Acceptance Criteria for the Sprint:**

* **Bingo Claim Announcement & Validation:**
  + Players can initiate a "Claim Bingo" command.
  + The system can implement an algorithm to accurately check for completed rows, columns, and diagonals.
  + Server-side validation is triggered upon a player's claim.
  + Players receive immediate feedback (success or failure) regarding their claim.
* **Invalid Claim Rejection:**
  + The system includes logic to detect and reject false Bingo claims.
  + A clear warning or error message is displayed to players for invalid claims.
  + Rejected claims are logged for audit and debugging purposes.

**Sprint 4 (14/11 - 28/11):**

**Features/Tasks:**

* 4.1: Implement player authentication (username/session).
* 4.2: Allow multiple players to join a lobby.
* 4.3: Sync Bingo cards across players.
* 4.4: Enable real-time updates (WebSocket or similar).
* 4.4.1: Assign points per user when sessions are continued.
* 4.4.2: Ensure high-standard UX for multiplayer gameplay.
* 4.5: Monitor first valid claim.
* 4.6: Broadcast winner announcement to all players.
* 4.7: Lock game board once winner is declared.
* 4.8: Provide option to start a new round.
* 4.9: Award different points for different win types.
* 4.10: Save points across rounds in a continued lobby.
* 4.11: Optionally allow purchase of a second card with points.
* 4.12: Verify authentication, lobby, synchronization, real-time updates, winner logic, and points via unit, integration, and E2E.

**Acceptance Criteria for the Sprint:**

* **Multiplayer Game Joining:**
  + Players can authenticate and join a multiplayer Bingo game using a username or session.
  + Multiple players can successfully join a game lobby.
  + Bingo cards are synchronized across all participating players.
  + Real-time updates (e.g., called numbers, marked cards) are enabled via WebSocket or similar technology.
  + If a session is continued, points are assigned per user.
  + The multiplayer experience maintains a high standard of user experience (UX).
* **Winner Declaration and Round Management:**
  + The system monitors and identifies the first player with a valid Bingo claim.
  + A winner announcement is broadcast to all players.
  + The game board is locked once a winner is declared.
  + Players have the option to start a new round.
* **Point System:**
  + Players gain different quantities of points based on the win type (e.g., little/big bingo).
  + Points are saved round to round in a continued lobby.
  + Players potentially have the ability to purchase a second card with points.

**Project architecture**

