**🎱 Bingo Game   
  
 Software Requirements Specification (SRS)   
&  
 Roadmap**

Date: 15/7/2025

Author: Abdallah Divkar

**1. Project Overview**

**Objective**

Develop an interactive Bingo game with a phased approach, beginning with a simple random number generator and evolving into a full-scale multiplayer game with bot players, Bingo cards, scoreboards, and modern UI/UX design.

**Target Users**

* Casual users interested in playing solo Bingo
* Friends/family playing online multiplayer
* Educational, recreational, or stream-integrated use cases

**2. Functional Roadmap by Version**

**Version 1.0: Bingo Ball Drawer (MVP)**

* **Key Features**
  + Python-based backend generates Bingo numbers (e.g., B5, O74) **without repetition**.
  + Frontend includes:
    - * A visual **Bingo ball or cage spinner** showing the currently drawn number.
      * A **display of all previously drawn numbers**.

**Suggested Tech Stack**

* **Backend**: Python + Flask or FastAPI
* **Frontend**: HTML, CSS, JavaScript (or React if desired)
* **UI Animation**: Lottie, SVG, or Canvas for Bingo cage/ball effects
* **Communication**: REST API (GET /draw)
* **State Management** (Frontend): JavaScript object/array (for now)

**Tools**

* Postman (API testing)
* GitHub (Version control)
* Render/Heroku (Backend deployment)
* Netlify/Vercel (Frontend hosting)

**Version 2.0: Interactive Single-Player Game**

* **Key Features**
* Display a **5x5 Bingo card** with numbers distributed according to standard Bingo rules.
* Automatically **mark off numbers** as they’re drawn.
* Detect Bingo conditions:
  + Horizontal, vertical, diagonal
* Basic UI updates when user wins.
* **Additions**
* **Frontend Framework**: React.js or Vue.js (for dynamic UI updates)
* **Card Logic**: Custom utility or Bingo card generation library
* **State Management**: React Context API or Redux (if scaling)
* **Design Libraries**: MUI, TailwindCSS, or Chakra UI

**Version 3.0: Multiplayer Mode**

* **Key Features**
* **Real-time multiplayer** game with:
  + Host/player model
  + Unique Bingo cards per user
  + Shared draw state
  + Win detection synced across all clients
* Lobby or room creation (with room codes or invite links)
* Optional chat system for interaction
* **Additions**
* **WebSocket Support**:
  + Flask-SocketIO (Python) or
  + Node.js with socket.io
* **User Sync**: Broadcasting draw events to all clients
* **Game Room Management**:
  + Create, join, and leave functionality
* **Persistent Game State**: Redis or in-memory cache for real-time sync

**Version 4.0: Robot Mode (AI Opponent)**

* **Key Features**
* Add **automated bot players** to games.
* Bots simulate drawing and matching numbers.
* Bots can auto-call Bingo based on logic or difficulty level.
* **Bot Logic Suggestions**
* React to draw events and simulate thinking delay.
* Call Bingo only when a win pattern is met.
* Difficulty setting can control bot speed, aggressiveness, accuracy.
* **Additions**
* Finite State Machine or custom rule-based logic
* Scheduled execution via setTimeout or asyncio
* Optional: Use AI logic with decision trees (not needed for basic bots)

**Version 5.0: User System & Scoreboard**

* **Key Features**
* **User accounts** (registration/login)
* Track and display:
  + Games played
  + Wins/losses
  + Fastest wins
* Global and room-level leaderboards
* Game history tracking
* **Additions**
* **Authentication**:
  + Firebase Auth
  + Clerk/Auth0
  + Custom JWT-based system
* **Database**:
  + Firebase Realtime DB
  + MongoDB or PostgreSQL (for relational tracking)
* **Backend Enhancements**:
  + CRUD APIs for game history and stats

**Version 6.0: UX & Performance Enhancements**

* **Key Features**
  + Mobile-first responsive design
  + Theme options (Dark Mode, Holiday Themes)
  + Audio effects (e.g., “B...12!” voice callouts)
  + In-game chat
  + Game replays and summary screens
  + Sharing/Invite links for rooms
* **Tools**
* **Styling**: TailwindCSS, MUI, SCSS
* **Audio**: HTML5 Audio API, Howler.js
* **Routing/UI Management**: React Router
* **Performance Optimizations**:
  + Code splitting, lazy loading
  + Service Workers (for PWA support)