Bingo Mobile App

1. There are actually two different applications:

Bingo Player

Bingo Manager

1. Bingo Player Mobile App
   1. Background

The App used for player to play in a bingo session

* 1. Description

Player with some player-id, requests to join a specific session of specific game {session-id, game-id}

If the result is “ok”, then player receives the board dimentions (width, height), and the specific text to display in each cell (there are width\*height cells).

The App should display now the board to the user

Note: App should be able to support user to participate in several sessions.   
 This not for first phase, but take it into design.

App enable user to press on the cells.

When user press on a cell, a “bingo answer” should be sent, with the information:  
player: {game-id, session-id, player-id} + answer: {x,y,cell-text}

The app will receive response: “ok” or “fail”

If “ok”, mark the cell as “answered” (color it or some other way).

If “fail” – display the failure somehow.

In the “ok” response user can get to more notifications:

“Winner”: true/false (nothing mean false)

“Finished”: true/false

If “winner=true”, mean the user is the winner of the session! do some nice image/music etc

If “finish=true”, mean user finished the session, although he is not the winner, because some other player finished earlier.

Note: There are several kinds of wins, which can be even when “finished=false”, and user can continue play and win another kinds of wins

Example: see presentation

* 1. API

API is REST Request/Response toward a server

Methods are GET / POST

Content data is with json format

The exact format will be added later

1. Bingo Manager App:

Background: used to create and manage bingo session (“manager” role)

Description: TBD

1. Terminology:

Bingo: The known game we create

Game, game-id: A definition

Session, session-id

Player, player-id: identification of end user player that participate in a specific session of specific game.

The player messages are identigied by {game-id, session-id, player-id}

1. API
   1. General response

All responses should be 200 OK, and will contain json document in the body

Success response will contain the root node: {“result” : “ok”}

"result": "ok"

Error response will contain “error” node in the json root with error string as value.

Error response will contain "result": with message that is NOT "ok"

For example:

"result": " session not found : ses2"

* 1. REST API
     1. API: ManagePlayer

**OPERATION**: JOIN (Other operations TBD)

**Purpose**: Player JOIN to existing session

Player supply: game0id, session-id, nick-name

REST Method: POST

https://<Server>/api/ManagePlayer?game-id=<user-game-id>&session-id=<user-session-id>&player-id=<user nick>&op=JOIN&ver=1

Content: currently empty json - “{}”

Success response:

{

"board": {

"width": 2,

"height": 2,

"cells": [

{

"x": 0,

"y": 0,

"answer": "15",

"answered": false,

"statistics": {}

},

---- REST OF CELLS ---

]

},

"result": "ok"

}

Notes:

* number of cells ind x,y inxdexes is according to board.width and board.height
* board.width and board.height define the board shape
* should be able to receive “answered”:true and in such case, the cell should be displayed as answered

MOCK SERVER: <https://6048939bb801a40017cce09c.mockapi.io/>

Phase 2

1. mock server new address:   
   htps://[606b158af8678400172e58d5.mockapi.io](http://606b158af8678400172e58d5.mockapi.io/)

With NO “api” prefix. For example:

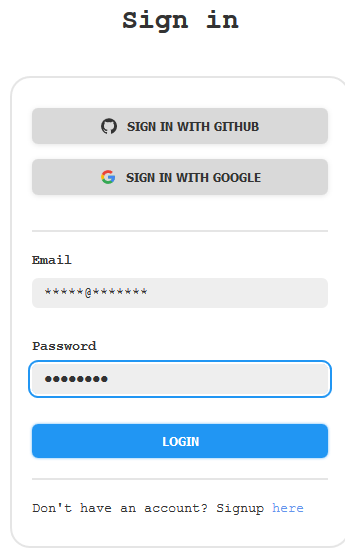
https://606b158af8678400172e58d5.mockapi.io/ManagePlayer

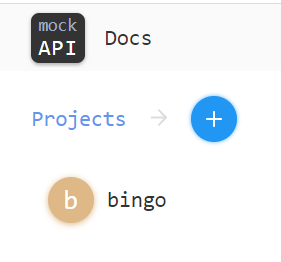
1. Mock server editing:

To edit the mock server responses  
browse to: <https://www.mockapi.io/login>

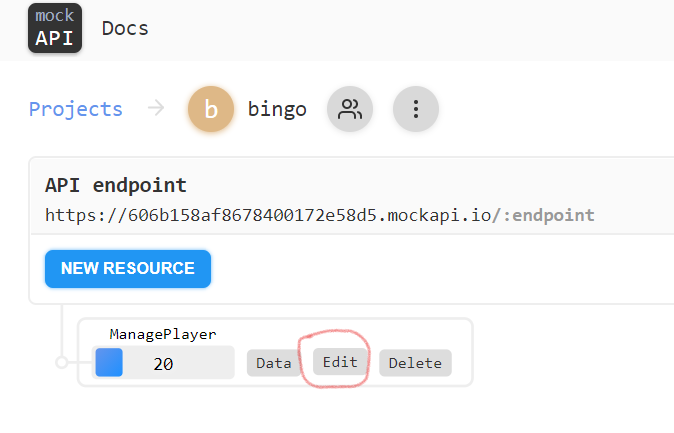
Email: -- I will send separatelly--

Password: -- I will send separatelly --  
and press “LOGIN”

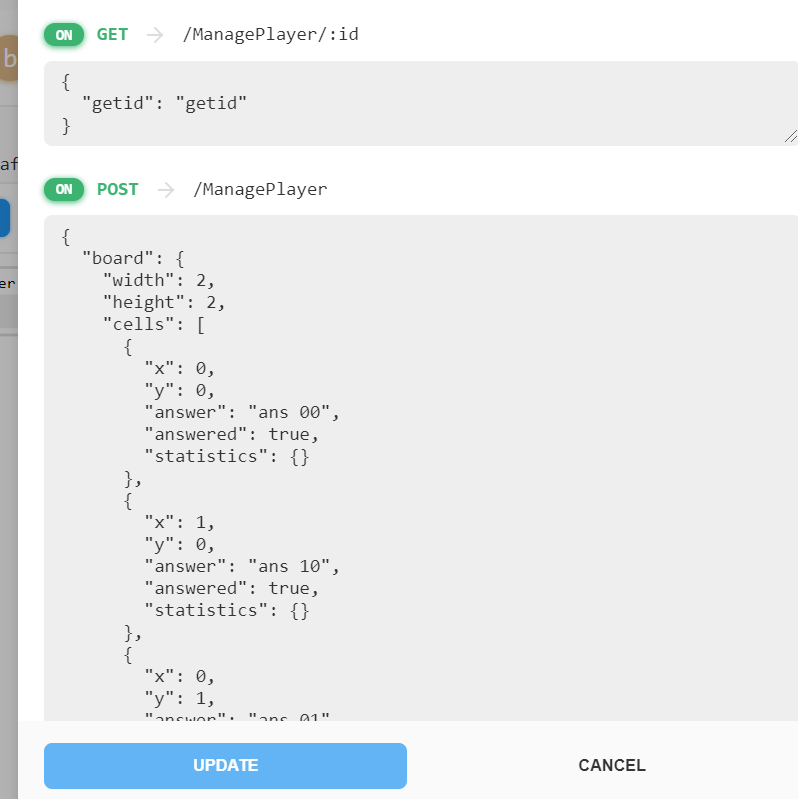


You get:  


Press “bingo” + hoover above “ManagePlayer” and press “edit”



And you get the response for the ManagePlayer API:

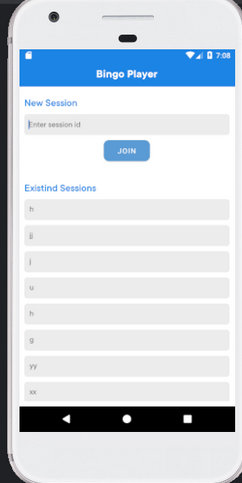


You can change the response and press “UPDATE” to activate it.

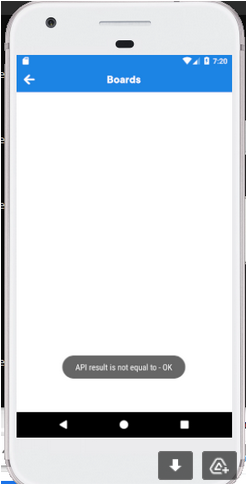
1. UI – The cells should be as bigger as possible – use whole screen  
   If cell text too long, it should be shown by hoovering on it  
   some examples:  
   



1. Cells that have “answered=true”, should be colored, and be disabled  
   for example (you can color whole cell, no need for circle):  
   
2. UI – Entry page:



1. “Join session” instead “New Session”
2. User should fill: game id, session id and nick name, to compose the request:  
   https://<Server>/api/ManagePlayer?game-id=<user-game-id>&session-id=<user-session-id>&player-id=<user nick>&op=JOIN&ver=1
3. No need for the history sessions
4. If API not “ok”, display the content of the result, not only “API not equal to – OK”



Phase 3 – Support “PlayerAnswer” API

User press active cell (that didn’t have answered=true).  
A “BingoAnswer” POST request is sent as follows:

[<Server>/BingoAnswer?game-id=numbers-bingo&session-id=game1&player-id=Doron&ver=1](http://localhost:7071/api/BingoAnswer?game-id=2&session-id=0&ver=1&player-id=Michal@bingo.com&debug=C:%5CUsers%5C33751%5Cbingo%5Ctemplates%5Cnumbers-game.json)

{

"x":0,

"y":0,

"answer":"18"

}

game-id, session-id and player-id – are those used for the “ManagePlayer” “JOIN” request

Request Content

X,Y – are cell location

Answer – is cell text

The json part is the POST message content

Response:  
{

"result":"ok", (or other text in case of failure)

"finished":”false”/”true”, (optional. If not exist, consider false)

"winner":”false”/”true” (optional. If not exist, consider false)

}

If “result” is “ok”,

Cell should be colored and disabled, like “answered=true of the above ManagePlayer

If winner is “true” – should display some “WINNER” splash

If winner is “false”, but finished is “true” – display “FINISHED” splash