**A multiplayer game**

The game is in different states in the two browsers. For a multi-player game, we need the two browsers to have the game in the same state at the same time. Everything in the two browsers should be synchronous. Currently, the games are asynchronous and independent from each other.

The ball in the two browsers moves **independently**. Their movements are **asynchronous**.

This happens because the ball's position in each browser is independent of the other's position. But what if we could store the ball's position in a remote common database and our application reads the ball's position from the database and updates it when it changes.

**Database** **servers** are computers, which are remotely connected through the internet and maintain your data, which you can use in your applications. Multiplayer games use a remote database to work. Multiplayer games store the position of the state of the game at all times in a remote database.

All the players' consoles /browsers read the game from this remote database and write to it when they make any change in the game. Therefore, in our game, the two browsers will read the position of the ball from the common remote database and the balls on two different browsers, on two different computers, will always be **synchronized**.

Let's create a remote database on the cloud for our simple application. This remote database will store the state (positions) of the ball and will allow us to read or write to it at any time.

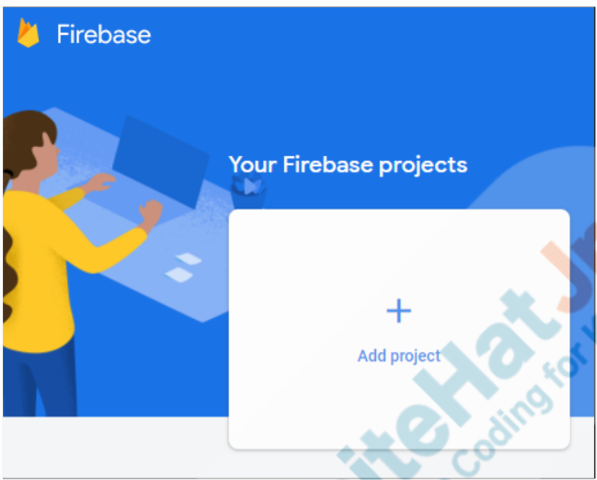
We will be using Google’s **Firebase Realtime Database** for this purpose.

The database can be compared to a JSON data structure format.

ball = { x: 250, y: 250 }

**Step** 1: Go to https://console.firebase.google.com and log in with your Gmail ID.

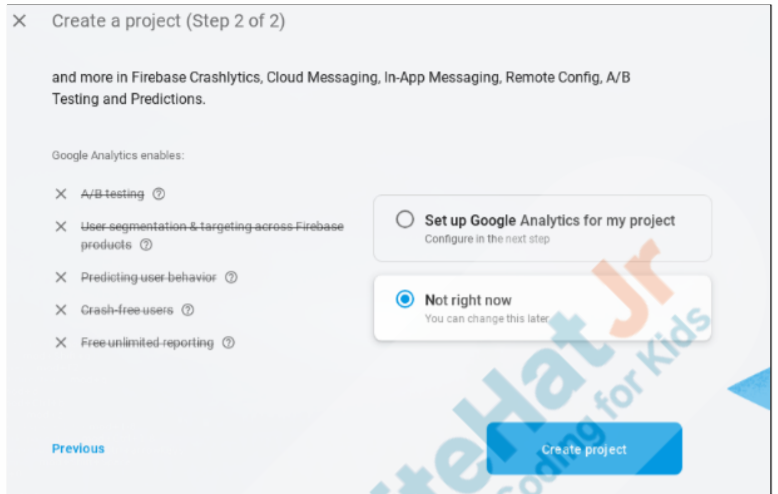
**Step** 2: Click on ‘Add Project’.



**Step** 3: Enter the name of your project. Accept the Firebase terms and click the Continue button.

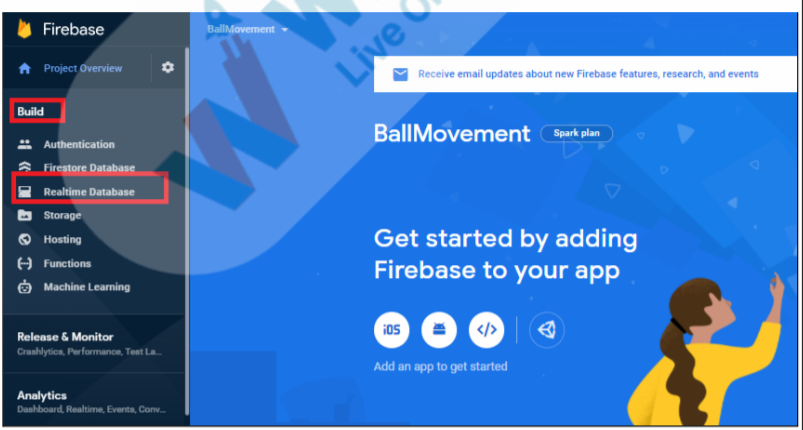


**Step** 4: Disable Google Analytics for this project. (We don’t need it)

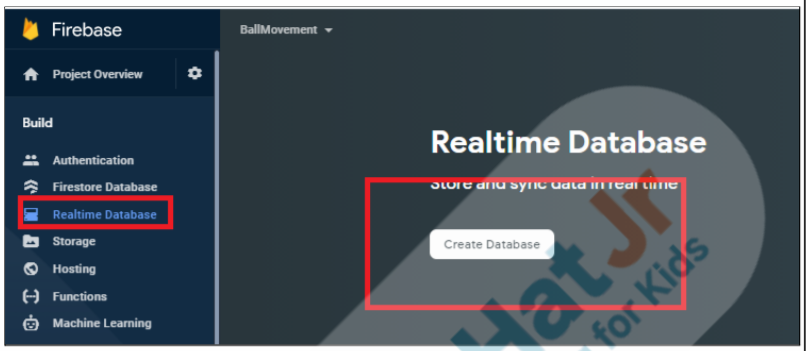


\* It takes a while to create the project; Once done, click the Continue button.

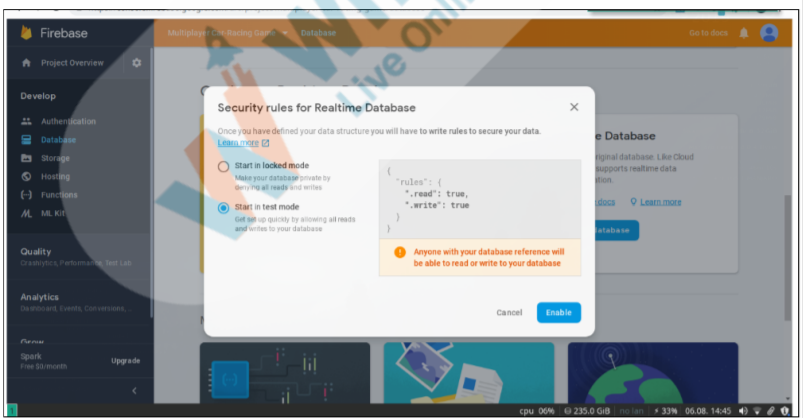
**Step** 5: Click Build and then on Realtime Database on the left-hand navigation panel.



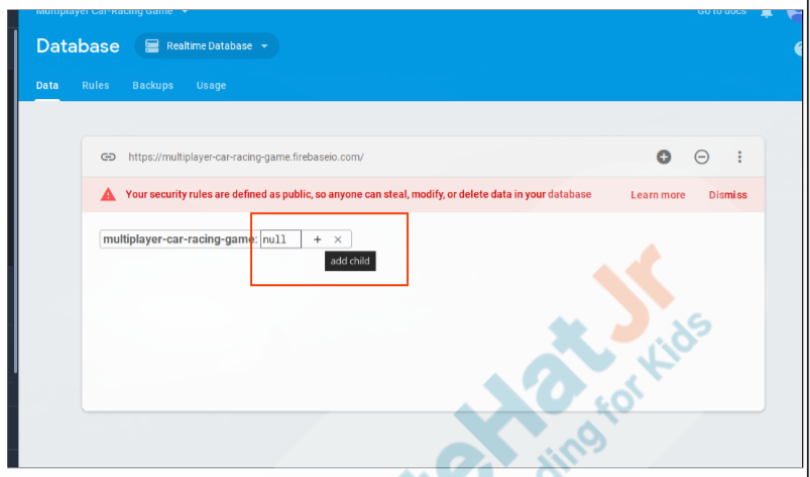
**Step** 6: Click ‘Create Database’.



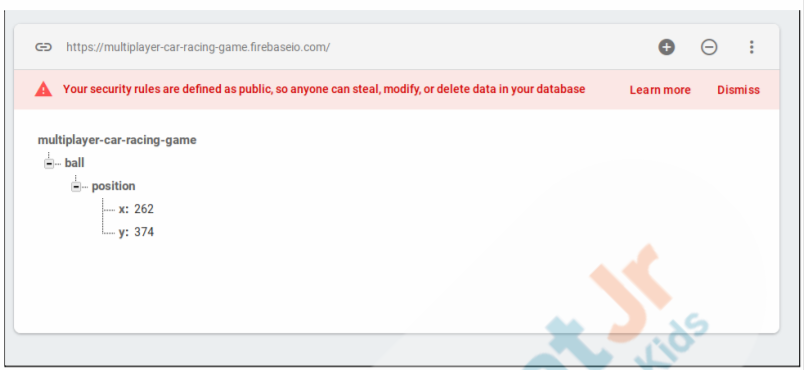
**Step** 7: Create the database in the test mode for now. Note: Test mode will help us quickly get started. It will be less secure in this mode, though later, we will learn how to make the database more secure.



Step 8: Add a child to create nodes, which can hold the ball’s x and y positions.







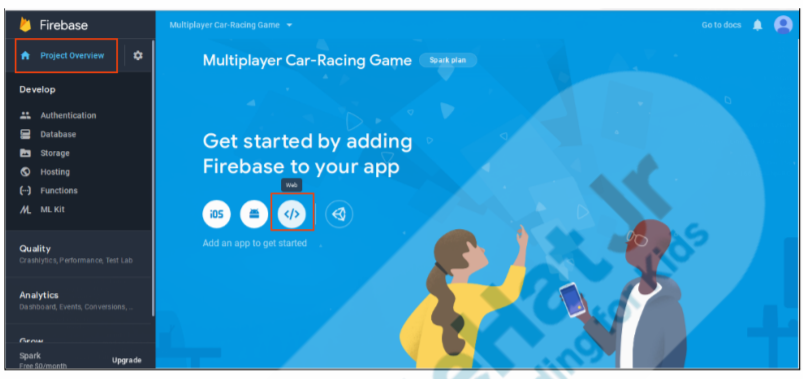
**Database** is ready, to access it from our code,

We need to **connect the Firebase** Realtime Database that we created **to** our **application**.

We will have to **import** a few **Firebase** **libraries**, which will allow us to read and write to our database.

We will also **need** some **configuration** **settings**, such as the API key, database URL, and so on, which can authenticate our application and allow it to write to the database. It is not wise to make this public - because then anyone will be able to write to your application. But our application does not contain any critical information so we can **ad**d this directly **to** our **HTML** **file**.

**Step** 1: Click “Project Overview” and Project Settings. Click on the icon of “add to the Web” to get started.



**Step** 2: Get the Firebase config key



**Step** 3: Add the SDK code to the index.html file along with the src library for Firebase database.



**Functions for database:**

.**ref**() is used to **refer** to the location of the database value we care about.

**.on**() creates a **listener**, which keeps listening to the changes in the database. Every time a change in the database values of position (reference) happens, the **readPosition**() function is called. If there is an error in reading the values in the database, the **showError**() function is called.

In the readPosition() function, we can read the position of the value in the database.

We will read them and assign the x and y values of the ball position in the database to the ball sprite.

We will also write a showError() function which will be called in case there is an error in fetching the data from the database. The showError() function is an optional function that can be avoided too.

When we move the ball in one browser, it is still not reflecting in another browser

to resolve that We need some function to **write values** in the database

.**set**() is used to set the value in the database. Essentially the arrow presses are changing the position values of the ball only in the database. In our application, we are reading those values and displaying the ball at that position.

There are a few bugs:

If the arrow key is pressed immediately when the app starts, the app shows an error.

Also, when the app is started, the initial position is 250, 250 before it synchronizes to the database values of the ball's position.

We are declaring the 'position' variable in line 2 but it takes some time for the application to read the values from the database and assign it to this value. Till then, the position value is undefined and the ball sprite is displayed at the default value of 250, 250 that we had used to create it. When we press the arrow key immediately at the start of the application, we are trying to write the 'undefined' position values into the database. We can fix this by drawing the ball or writing to the database only when 'position' is NOT EQUAL to undefined.