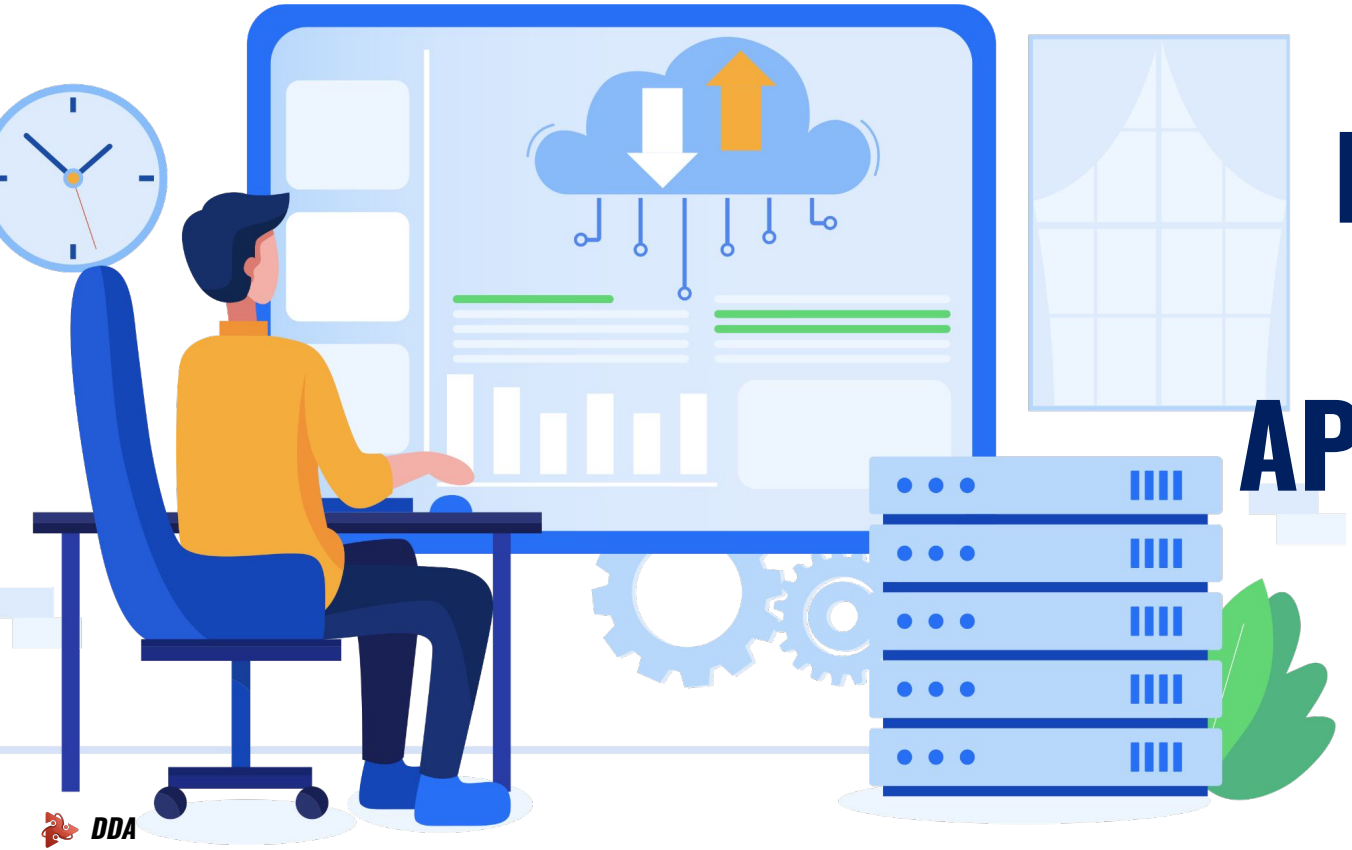


(DDA) 2.2

DEVELOPING DYNAMIC APPLICATIONS

2021

DDA

COVERAGE

Week 8

Learning Objectives

1. Recap Firebase Web & Setup
2. Dashboarding 101
3. CRUD in action
4. Dashboarding with Web Templates & Charts

*Note: We are **using Firebase v9.***

We are NOT using Firebase v8

Do not use Firestore unless you are familiar with it. Because your Unity will need to use it too.

USE AS YOU WILL

You can download dashboard templates in our DDA MS Teams channel > Files

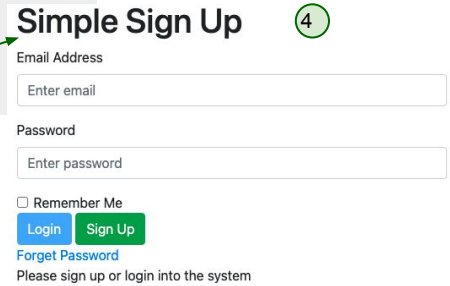
Firebase Web
#Recap

Recap Firebase

How to Setup your Firebase & Web + Forms

1. Start with the Firebase config settings and place into a "**config.js**" JS script that can be found in your Firebase console -> project settings (Web)
2. **Initialize your app** based on the **firebase config**
3. Create another **"main.js" JS file that contains your necessary Firebase imports**
4. Create your UI using HTML
5. Add on logic to your "main.js" to **retrieve the auth object or database object from Firebase**
6. Add Event Listeners that listen to your form
   Remember: to place in necessary error handlers

```
/project
--/js
   |-- config.js
   |-- main.js
signup.html
```




```
const firebaseConfig = {
 apiKey: "",
 authDomain: "",
 databaseURL: "",
 projectId: "",
 storageBucket: "",
 messagingSenderId: "",
 appId: "",
 measurementId: "",
 };
```

```
//[STEP 1] Base firebase imports
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.5.0/firebase-app.js";

import {getDatabase, ref, child, get, set, onValue, orderByChild, orderByVal, query, equalTo,startAt,startAfter,
 endAt,endBefore,limitToFirst,limitToLast} from "https://www.gstatic.com/firebasejs/9.5.0/firebase-database.js";

import {getAuth, createUserWithEmailAndPassword,signOut,onAuthStateChanged,updateProfile,updateEmail,
 updatePassword, signInWithEmailAndPassword} from "https://www.gstatic.com/firebasejs/9.5.0/firebase-auth.js";

//[STEP 2] Setup our base global references to the services
//================================================
//Must initialize Firebase app w/ config to start
const app = initializeApp(firebaseConfig);
const db = getDatabase(); //db reference
const auth = getAuth(); //auth service reference
```

You must initialize the app, db

```
let btnSignup = document.getElementById("btn-signup"); //signup btn
btnSignup.addEventListener("click", function (e) {
 e.preventDefault();
 let email = document.getElementById("email").value;
 let password = document.getElementById("password").value;

 //let email = $("#email").val();
 //let password = $("#password").val();
 console.log(`Sign-ing up user with ${email} and password ${password}`);
 //[STEP 4: Signup our user]
 signUpUserWithEmailAndPassword(email, password);
});
```

What do I need to Import?

V9 of Firebase loads via a modular format. Hence we only load the functions that we need.

It is important to declare the script as a **module type** when we are including it in the HTML

```
<script src="../lib/snippets/crud-player.js"
type="module"></script>
```

```
import {
 getDatabase,
 ref,
 child,
 get,
 set,
 onValue,
 orderByChild,
 orderByValue,
 query,
 equalTo,
 startAt,
 startAfter,
 endAt,
 endBefore,
 limitToFirst,
 limitToLast,
} from
"https://www.gstatic.com/firebasejs/9.5.0/firebase-database.js";
```

What you are importing depends on your needs of the service.

A full list of documentation can be found below
Database:
https://firebase.google.com/docs/reference/js/database.md#database_package

Authentication:
https://firebase.google.com/docs/reference/js/auth

Creating Dynamic Keys

```
import { getDatabase, ref, push, set } from
"firebase/database";

// Create a new post reference with an auto-generated
id
const db = getDatabase();
const postListRef = ref(db, 'posts');
const newPostRef = push(postListRef);
set(newPostRef, {
    // ...
});
```

To create a randomly generated key which is unique, we can tap onto the `push()` function.

It works the same way as the C# variant, what we need to have is a reference path (1)

Then we have our **push**. This "*Push to my reference*" (2) depends on where the reference path is. Once the path is allocation, the key will be generated inside as a new node using push.

WHY do we need such?
When data is spontaneous
Not so concerned about key values

https://firebase.google.com/docs/database/web/lists-of-data#append_to_a_list_of_data

What is an Index (Optimisation)

An index is a powerful tool. Say for example, our NRICs are unique and we know that it is unique. So it is treated like a key in our database. So once we know exactly the key we can retrieve the data

In databases, an index works by “compiling” that data nicely. So that we can sort our data efficiently. In Firebase terms we use **.OrderByChild(“somechildproperty”) or .OrderByKey(“somekey”)**

When we index, the database will query and find the data much more efficiently. However, having said so, firebase is pretty efficient. So it depends on how much data you have, and how you want to manipulate the data.

```
{
  "rules": {
    ".read": true,  // 2021-11-11
    ".write": true,//"now < 1636560000000",  // 2021-11-11
     "playerStats": {
      ".indexOn": ["highScore"]
    },
    "leaderboards": {
      ".indexOn": ["highScore"]
    }
  }
}
```

Using OrderByChild hence we index the child properties

Additional Reading

https://firebase.google.com/docs/database/security/indexing-data

Sorting Data (READING)

Method	Usage
orderByChild()	Order results by the value of a specified child key or nested child path.
orderByKey()	Order results by child keys.
orderByValue()	Order results by child values.

*You can only use **ONE** order-by method at a time.*
Calling an order-by method multiple times in the same query throws an error.

To retrieve sorted data, start by specifying one of the order-by methods to determine how results are ordered:

```
const latestPlayerRef = query(
   ref(db, "players/"),
   orderByChild("createdOn"),
   limitToLast(1)
 );
 let result = await get(query(latestPlayerRef));
```

Encountering similar errors?

```
▸ Uncaught (in promise) Error: Index not defined, add ".indexOn": "level", for path "/playerStats", to the rules
    at Repo.ts:482
```

When you are using **orderByChild**, do ensure you have placed the **appropriate indexes** in your Firebase Database rules

```
{
  "rules": {
    ".read": true,  // 2021-11-11
    ".write": true,//"now < 1636560000000",  // 2021-11-11
    "players": {
      ".indexOn": ["createdOn"]
    },
    "playerStats": {
      ".indexOn": ["highScore", "level"]
    },
    "leaderboards": {
      ".indexOn": ["highScore"]
    }
  }
}
```

o/lists-
t data

Filtering Data (READING)

Method	Usage
limitToFirst()	Sets the maximum number of items to return from the beginning of the ordered list of results.
limitToLast()	Sets the maximum number of items to return from the end of the ordered list of results.
startAt()	Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
startAfter()	Return items greater than the specified key or value depending on the order-by method chosen.
endAt()	Return items less than or equal to the specified key or value, depending on the order-by method chosen.
endBefore()	Return items less than the specified key or value depending on the order-by method chosen.
equalTo()	Return items equal to the specified key or value, depending on the order-by method chosen.

Unlike the order-by methods, you can combine multiple limit or range functions. For example, you can combine the startAt() and endAt() methods to limit the results to a specified range of values.

Getting the latest player entry

```
const latestPlayerRef = query(
   ref(db, "players/"),
   orderByChild("createdOn"),
   limitToLast(1)
 );
 let result = await
get(query(latestPlayerRef));
```

Filtering Data (Getting a Range using startAt endAt)

Method	Usage
startAt()	Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
startAfter()	Return items greater than the specified key or value depending on the order-by method chosen.
endAt()	Return items less than or equal to the specified key or value, depending on the order-by method chosen.
endBefore()	Return items less than the specified key or value depending on the order-by method chosen.

Working with Imports

Imports used in Firebase are meant to keep things as modular as possible. In order to load faster, and bring about better efficiency

```
import {
 getAuth,
 setPersistence,
 signInWithEmailAndPassword,
 browserSessionPersistence,
 inMemoryPersistence,
 browserLocalPersistence, //default
} from
"https://www.gstatic.com/firebasejs/9.5.0/firebase-auth.js";
```

```
import {
   getDatabase,
   ref,
   child,
   get,
   set,
   onValue,
   orderByChild,
 } from
"https://www.gstatic.com/firebasejs/9.5.0/firebase-database.js";
```

```
//base firebase config
import { initializeApp } from
"https://www.gstatic.com/firebasejs/9.5.0/firebase-app.js";

//config settings derived from firebase console
const firebaseConfig = {
 apiKey: "A ",
 authDomain: " ",
 databaseURL: " ",
 projectId: " ",
 storageBucket: " ",
 messagingSenderId: " ",
 appId: " ",
 measurementId: " "
 };
  //Must initialize Firebase app w/ config to start
 const app = initializeApp(firebaseConfig);
```

Lost? Watch this

How to import Firebase with JavaScript modules - Firecasts
https://www.youtube.com/watch?v=lGqKYpvLkhE

Additional Reading

e.com/docs/database/security/indexing-data

Dashboarding #Games

What kind of Data?????

No. Of new users
-> comparing against last wk? Yesterday?

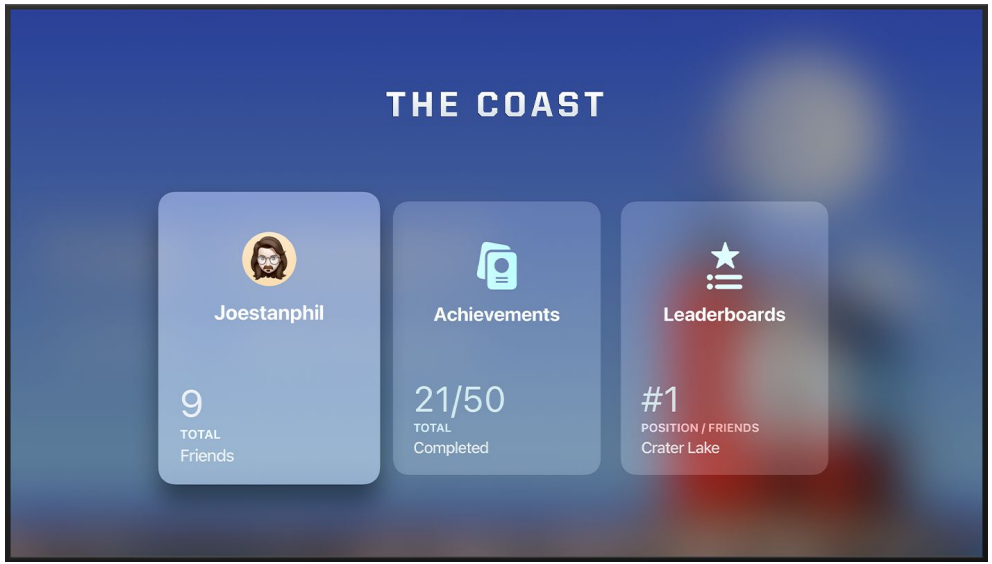
Total Time Played
-> tally of session logs

Shots fired
-> for each shot -> save to db

Headshot percentage
Num headshots/ Number of shots

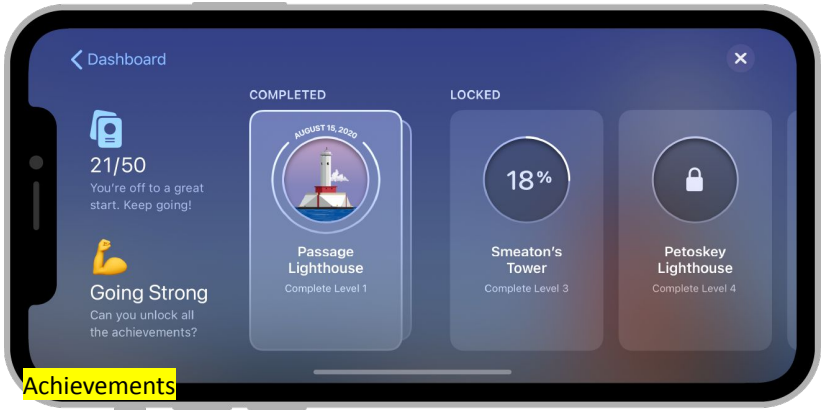
https://miro.com/app/board/o9J_lhdzg1g=/

Dashboarding 101



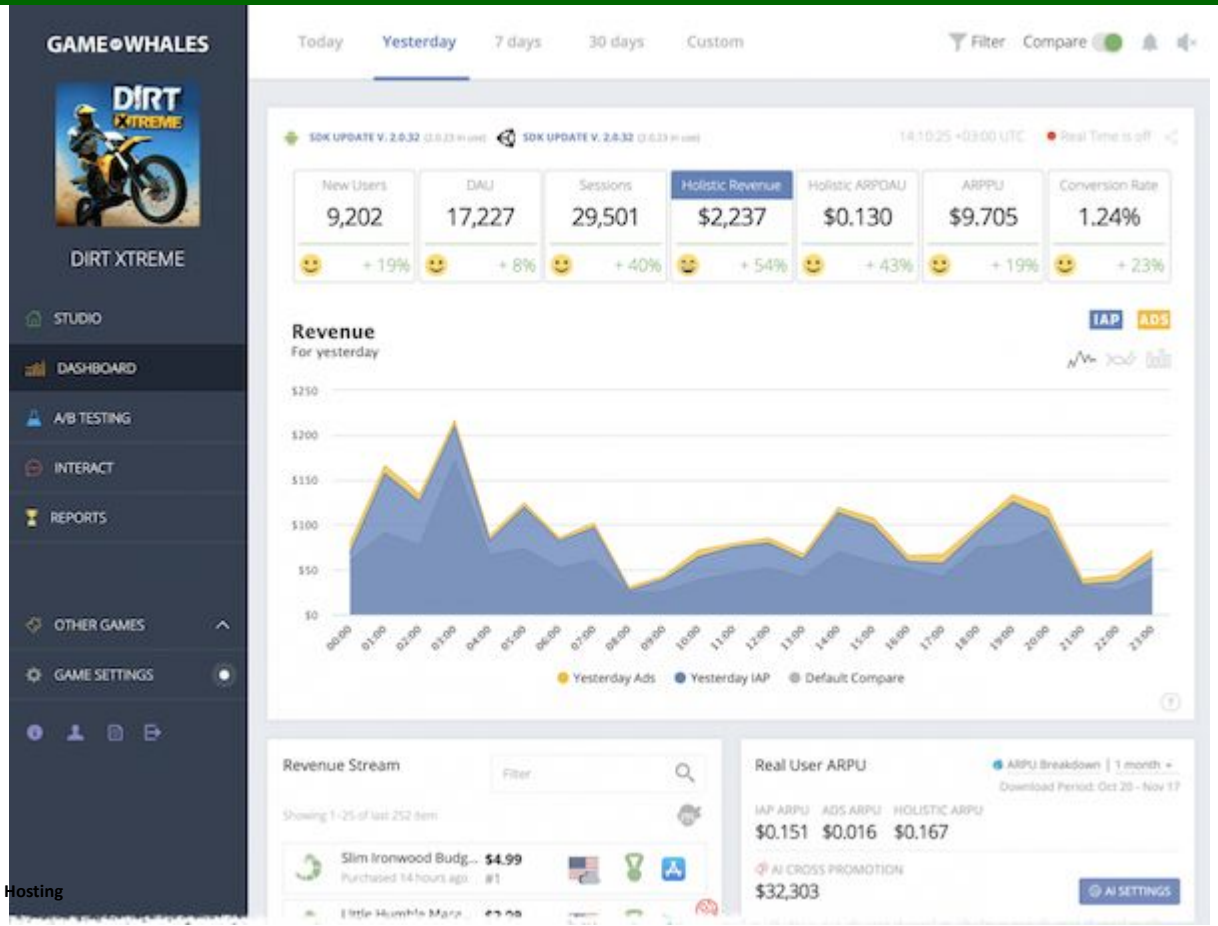
Apple Game Center
https://developer.apple.com/design/human-interface-guidelines/game-center/overview/introduction/

Leaderboards



Achievements

Dashboarding 101



Dashboarding 101

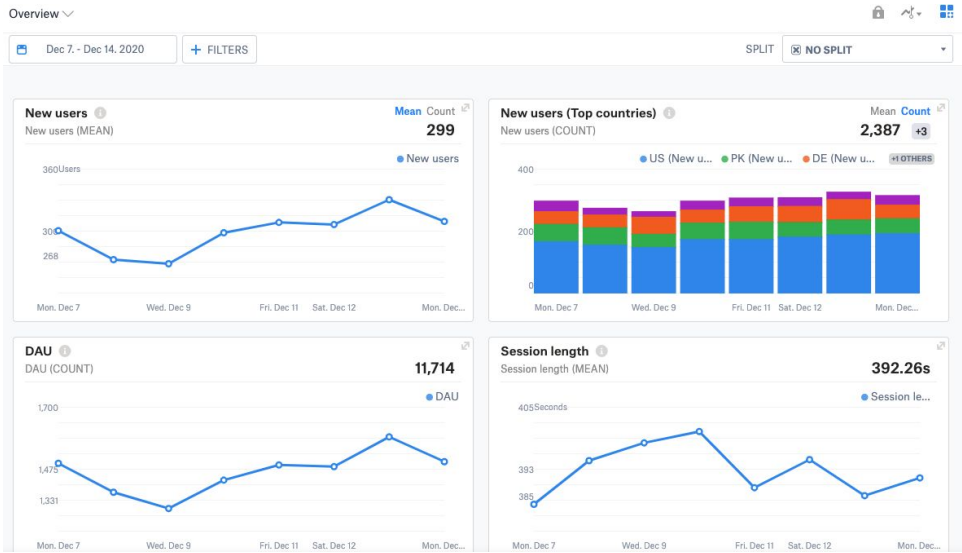
Thoughts: Behavior vs Performance?

Metrics are at the basis of solid decision making. When you focus on what your users want to accomplish with these metrics it opens up the design space, allowing for creative problem solving and a goal-oriented development process.

For Whom?

- Customers (Managers, etc)
  - Track habits
  - $$
- Development Team
  - Track errors, hacks
  - Track player interactions
  - Data driven game design (identify game optimisation)
- Players
  - For extensive insights into their game

- Daily Active Users/Monthly Active Users
- Day 1/7/30 Retention
- First Time Users Experience (FTUE)
  - What was one on-boarding experience you like



https://deltadna.com/blog/analyzing-day-1-retention-ftue/
https://grendelgames.com/dashboards-for-serious-games-2/
https://en.wikipedia.org/wiki/Active_users

Dashboarding 101 (Session Time metric)

Average Session Time metric can be used to gauge the bounce rate along with the number of times the same player keeps coming returning to the game each day.

What makes a good session time?
You have to determine what makes a session.

eg.
User click on Start?
User spend X amount of time?
User trigger some events ?

Your definition of an action that turns a user into an 'active user' depends on your business model and goals.

Average session length - an average time spent in the app per user. Defined as the sum of the length of all sessions divided by the number of sessions within a given period.

https://www.blog.udonis.co/mobile-marketing/mobile-games/session-length

Dashboarding 101 (Session Time metric)

DAU (Daily Active Users) – the number of unique users per day;

WAU (Weekly Active Users) – the number of unique users per week;

MAU (Monthly Active Users) – the number of unique users per month.

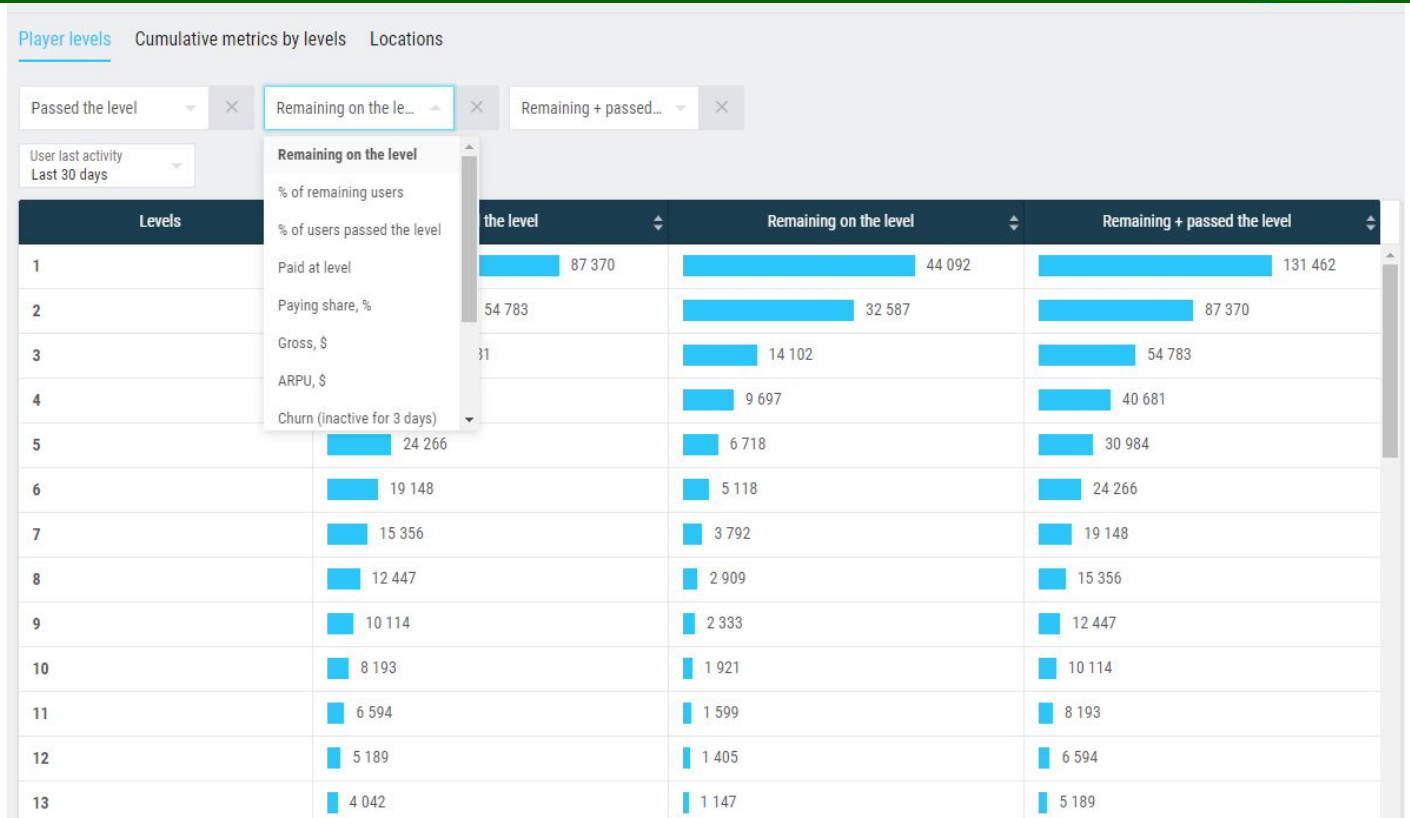
Session length is a metric that shows the amount of time a user spends playing a mobile game. It is also one of the most important mobile game KPIs in terms of user experience and engagement.

A session starts at the moment when a user opens an app and lasts until the user closes it or stops actively using it. In other words, it measures the duration of a single continuous app experience, or in this case, **gameplay experience**. When the game gets sent to the background, this typically marks the end of a session.

If you want to **calculate session length**, there is a simple formula. Just subtract the time when the user became inactive from the time the app was opened.

https://medium.com/udonis/mobile-game-session-length-how-to-track-increase-it-b8244a5fc94f

Dashboarding 101 (Level Completion metric)



Dashboarding 101: Player Behavior

Onboarding — are players making it through your onboarding mechanics such as tutorials or starting levels?

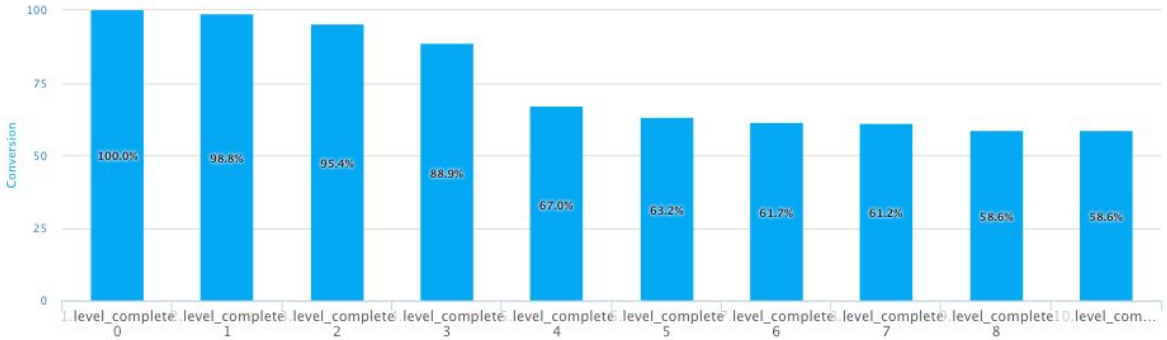
Progression — are players progressing through your levels?

Economy — are your game economies working out as expected?

Design validation — are your game design choices working out as you thought they would?

Application validation — are all areas of your application being utilized as you expect? Are there parts that players ignore or don't notice?

Monetization — are your monetization strategies optimal? Are there impediments to players carrying out purchases?



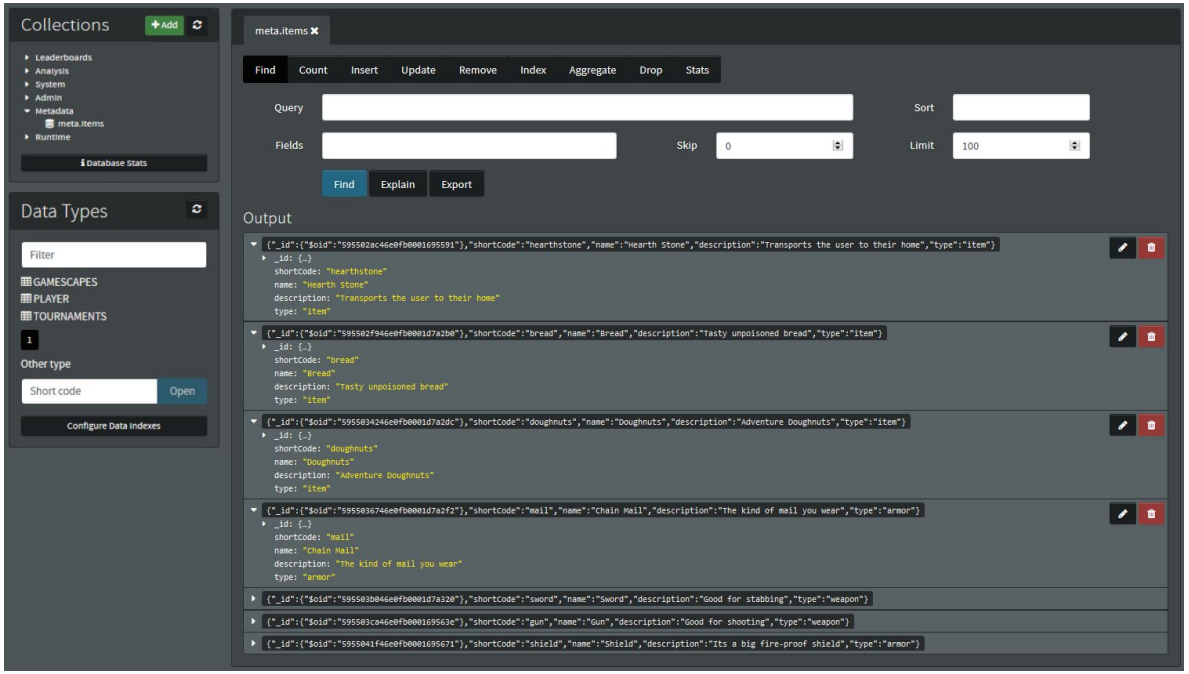
Tutorial conversion - the share of new users who have successfully completed the tutorial.

https://docs.unity3d.com/Manual/UnityAnalyticsTerminology.html

Dashboarding 101: Items

You might have items in game and to store them in the database for more efficiency.

This will allow some additional dynamics to the game flow



https://docs.unity3d.com/Manual/UnityAnalyticsTerminology.html

Working with #Auth.CurrentUser

Auth.CurrentUser

The Authentication in Firebase is very powerful and packed with features.

Once we are logged in, we can use our authentication reference to retrieve the current user session and get user's details (userId, DisplayName, ProfilePic, etc)

```
const auth = getAuth();

//Auth refers to our auth object that is derived from the Firebase auth
service
//onAuthStateChanged is an observer that provides a Promise return
//currentUser is our Promise (the naming doesn't matter)
onAuthStateChanged(auth, (currentUser) => {
   if (currentUser) {
     // User is signed in, see docs for a list of available properties
     // https://firebase.google.com/docs/reference/js/firebase.User
     const uid = currentUser.uid;
     statusMsg.innerHTML = `(OnAuthStateChanged) Welcome back:
${currentUser.email} :: ${currentUser.uid}`;
     console.log(`(OnAuthStateChanged) Current user is logged in:
${currentUser.email} ::

   } else {
     statusMsg.innerHTML = `Please sign up or login into the system`;

   }
 });
```

Reading Reference https://firebase.google.com/docs/reference/js/firebase.User

Additional Reading

https://firebase.google.com/docs/database/security/indexing-data

Firebase Hosting

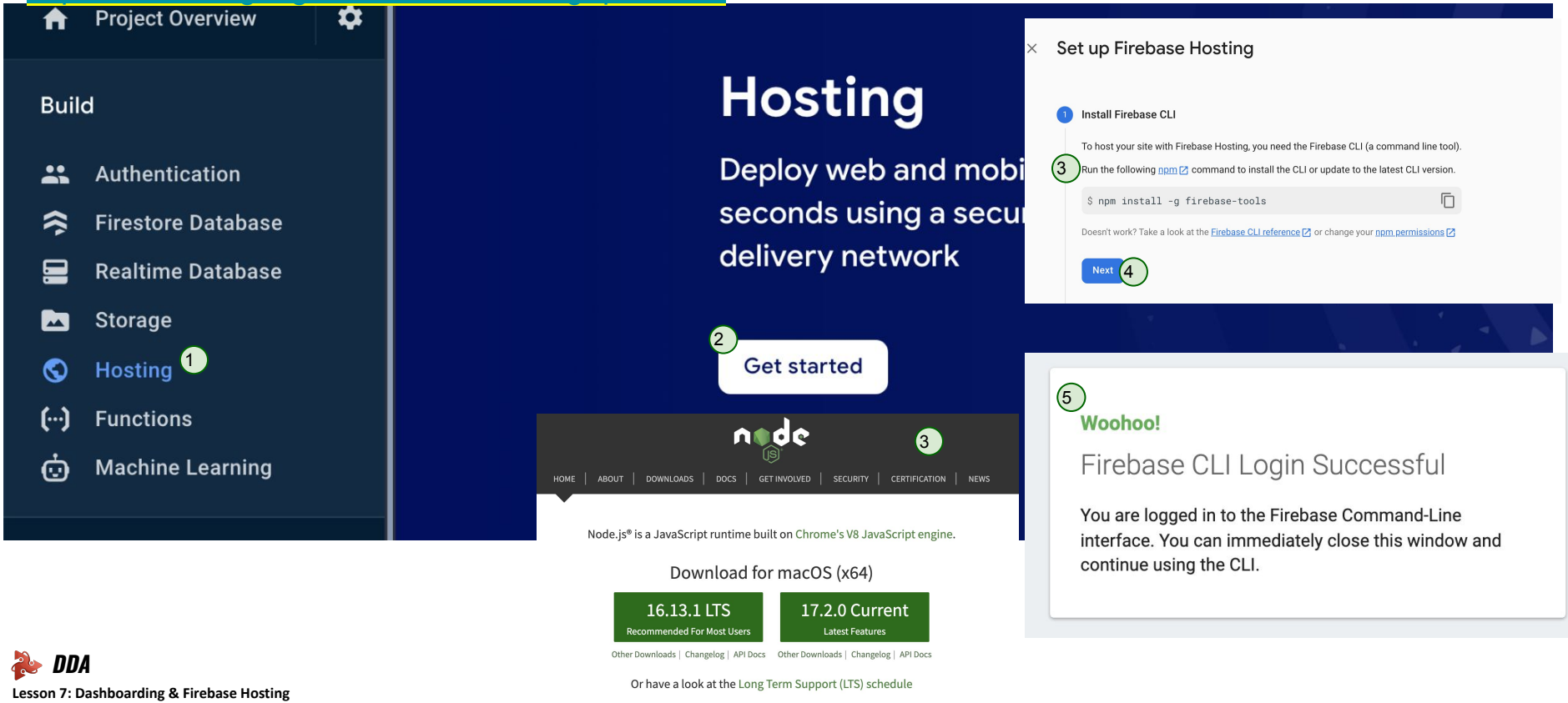
#Firehosting

Firebase Hosting - Capabilities

Serve content over a secure connection	The modern web is secure. Zero-configuration SSL is built into Firebase Hosting, so content is always delivered securely.
Host static and dynamic content plus microservices	Firebase Hosting supports all kinds of content for hosting, from your CSS and HTML files to your Express.js microservices or APIs.
Deliver content fast	Each file that you upload is cached on SSDs at CDN edges around the world and served as gzip or Brotli. We auto-select the best compression method for your content. No matter where your users are, the content is delivered fast.
Emulate and even share your changes before going live	View and test your changes on a locally hosted URL and interact with an emulated backend. Share your changes with teammates using temporary preview URLs. Hosting also provides a GitHub integration for easy iterations of your previewed content.
Deploy new versions with one command	Using the Firebase CLI, you can get your app up and running in seconds. Command line tools make it easy to add deployment targets into your build process. And if you need to undo the deploy, Hosting provides one-click rollbacks.

Firebase Hosting (Step 1)

https://firebase.google.com/docs/hosting/quickstart



Firebase Hosting (Step 2)

https://firebase.google.com/docs/hosting/quickstart

```
     ######    ##  ########  ######   ########  #########  ######  ######
     ##        ##  ##     ##  ##       ##     ## ##     ##       ## ##
     ##        ##  ##     ##  ##       ##     ## ##     ##       ## ##
     ##       #### ##     ## ######## ########  ##     ##  ######  ########

You're about to initialize a Firebase project in this directory:

  /Users/champ/Documents/DDA/DDA-Github-Desktop/IMYear2.2/Modules/DDA/firebase-web

? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices.
ace> to select, <a> to toggle all, <i> to invert selection)
❯◯ Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instance
 ◯ Firestore: Configure security rules and indexes files for Firestore
 ◯ Functions: Configure a Cloud Functions directory and its files
 ◯ Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
 ◯ Hosting: Set up GitHub Action deploys
 ◯ Storage: Configure a security rules file for Cloud Storage
 ◯ Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

6

```
     ######## #### ########  ######## ########     ###    ######  ########
     ##        ##  ##     ## ##       ##     ##  ##   ##  ##      ##
     ######    ##  ########  ######   ########  #########  ######  ######
     ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
     ##       #### ##     ## ######## ########  ##     ##  ######  ########

  You're about to initialize a Firebase project in this directory:

    /Users/champ/Documents/DDA/DDA-Github-Desktop/IMYear2.2/Modules/DDA/firebase-web

  ? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices.
  ◉ Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
  ◯ Hosting: Set up GitHub Action deploys
  ◯ Storage: Configure a security rules file for Cloud Storage
 ❯◯ Emulators: Set up local emulators for Firebase products
  ◯ Remote Config: Configure a template file for Remote Config
  ◉ Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instance
  ◯ Firestore: Configure security rules and indexes files for Firestore
  (Move up and down to reveal more choices)
```

7

8

```
=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: Use an existing project
? Select a default Firebase project for this directory: malcolm-firebase-playground (Malcolm Firebase Playground)
i  Using project malcolm-firebase-playground (Malcolm Firebase Playground)

=== Database Setup
i  database: ensuring required API firebasedatabase.googleapis.com is enabled...
✔  database: required API firebasedatabase.googleapis.com is enabled


Firebase Realtime Database Security Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Realtime Database Security Rules? (database.rules.json)
```

9

Firebase Hosting (Step 3 - Initialization)

https://firebase.google.com/docs/hosting/quickstart

```
firebase deploy.

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

[? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? (y/N) n
```

10

```
? What file should be used for Realtime Database Security Rules? database.rules.json
✔  Database Rules for malcolm-firebase-playground-default-rtdb have been written to database.rules.json.
Future modifications to database.rules.json will update Realtime Database Security Rules when you run
firebase deploy.

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? No
? Set up automatic builds and deploys with GitHub? No
✔  Wrote public/404.html
✔  Wrote public/index.html

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...
i  Writing gitignore file to .gitignore...

✔  Firebase initialization complete!
champ@ictadmins-MBP firebase-web %
```

10

```
=== Deploying to 'malcolm-firebase-playground'...

i  deploying database, hosting
i  database: checking rules syntax...
✔  database: rules syntax for database malcolm-firebase-playground-default-rtdb is valid
i  hosting[malcolm-firebase-playground]: beginning deploy...
i  hosting[malcolm-firebase-playground]: found 2 files in public
✔  hosting[malcolm-firebase-playground]: file upload complete
i  database: releasing rules...
✔  database: rules for database malcolm-firebase-playground-default-rtdb released successfully
i  hosting[malcolm-firebase-playground]: finalizing version...
✔  hosting[malcolm-firebase-playground]: version finalized
i  hosting[malcolm-firebase-playground]: releasing new version...
✔  hosting[malcolm-firebase-playground]: release complete

✔  Deploy complete!

Project Console: https://console.firebase.google.com/project/malcolm-firebase-playground/overview
Hosting URL: https://malcolm-firebase-playground.web.app
champ@ictadmins-MBP firebase-web %
```

firebase deploy --only hosting