Lab 9

Firebase Database

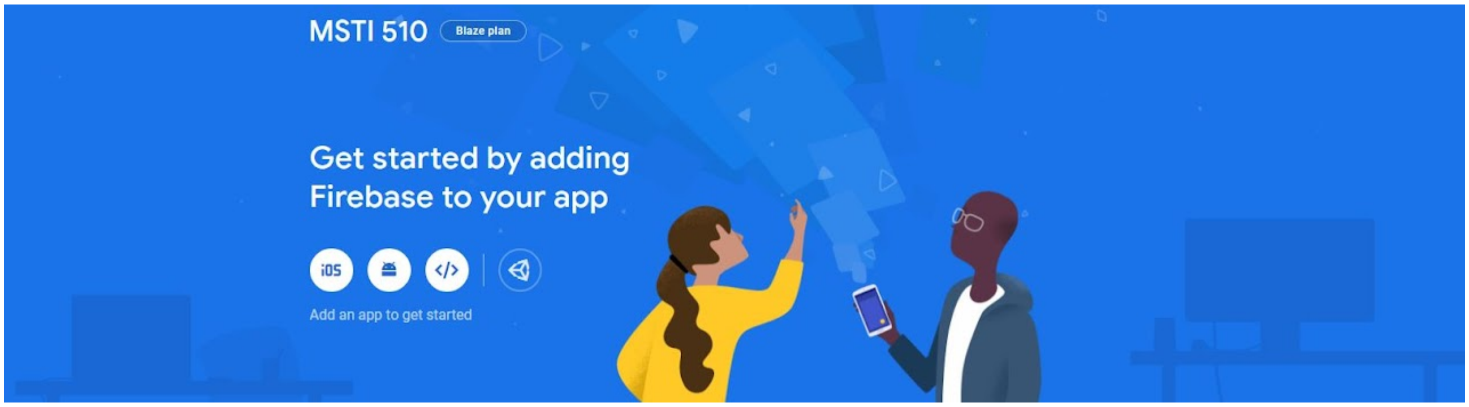
In this lab you will create a database backend for your robot app and use that to allow multiple front end interfaces to communicate with one another.

Step 1: Get starter code and set up Google Firebase

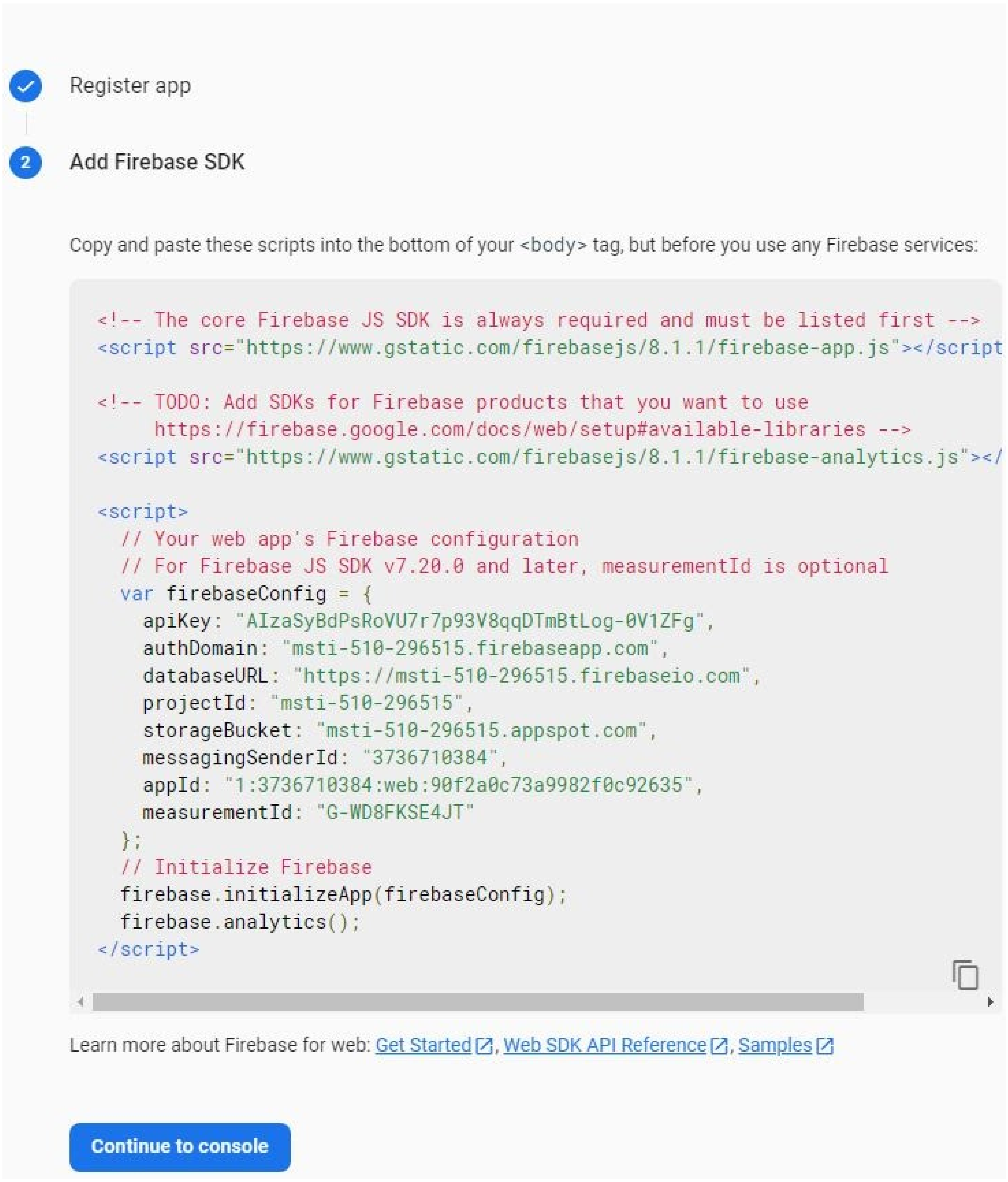
In this lab you will make a robot which can be controlled from a different front-end app. In particular, you will make your robot speak a sentence sent from a separate UI. We provide an example control UI for you to modify in this lab. This UI sends the text entered into a text box to a Firebase database. Fork this Codepen pen into your own account so you can modify it.

Next, you will need to set up the database backend for your web applications. Go to the Google Firebase Console and add a new project. You will need to sign into a Google account (you can use your UW address). You can name the project anything you like (note: if you pick the same project you setup as part of the Google API services in lab 8 you will be forced into a pay-as-you go billing scheme).

Once created, you should see a page like the following.


MSTI 510
Blaze plan
Get started by adding Firebase to your app
iOS
</>
Add an app to get started

Click on the web icon to obtain the code snippet you will need to add to your apps that will communicate with the database you are about to create. Press copy, then paste the snippet into the Javascript of your robot control UI.

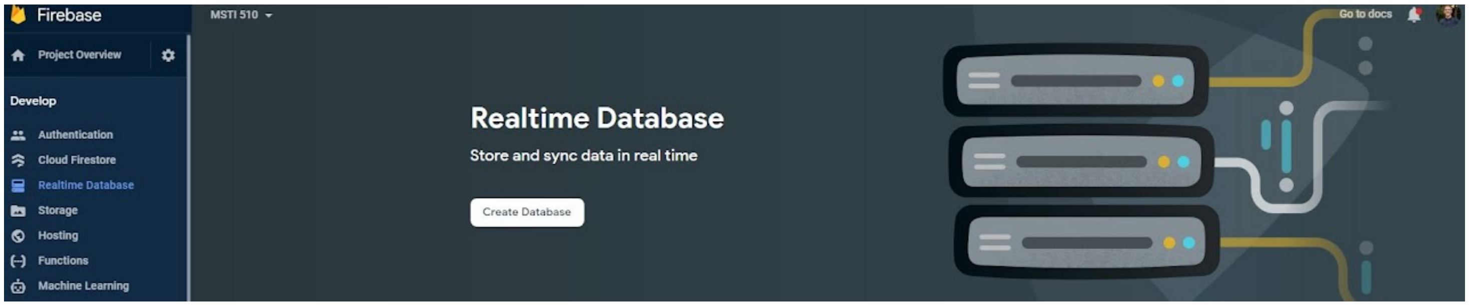


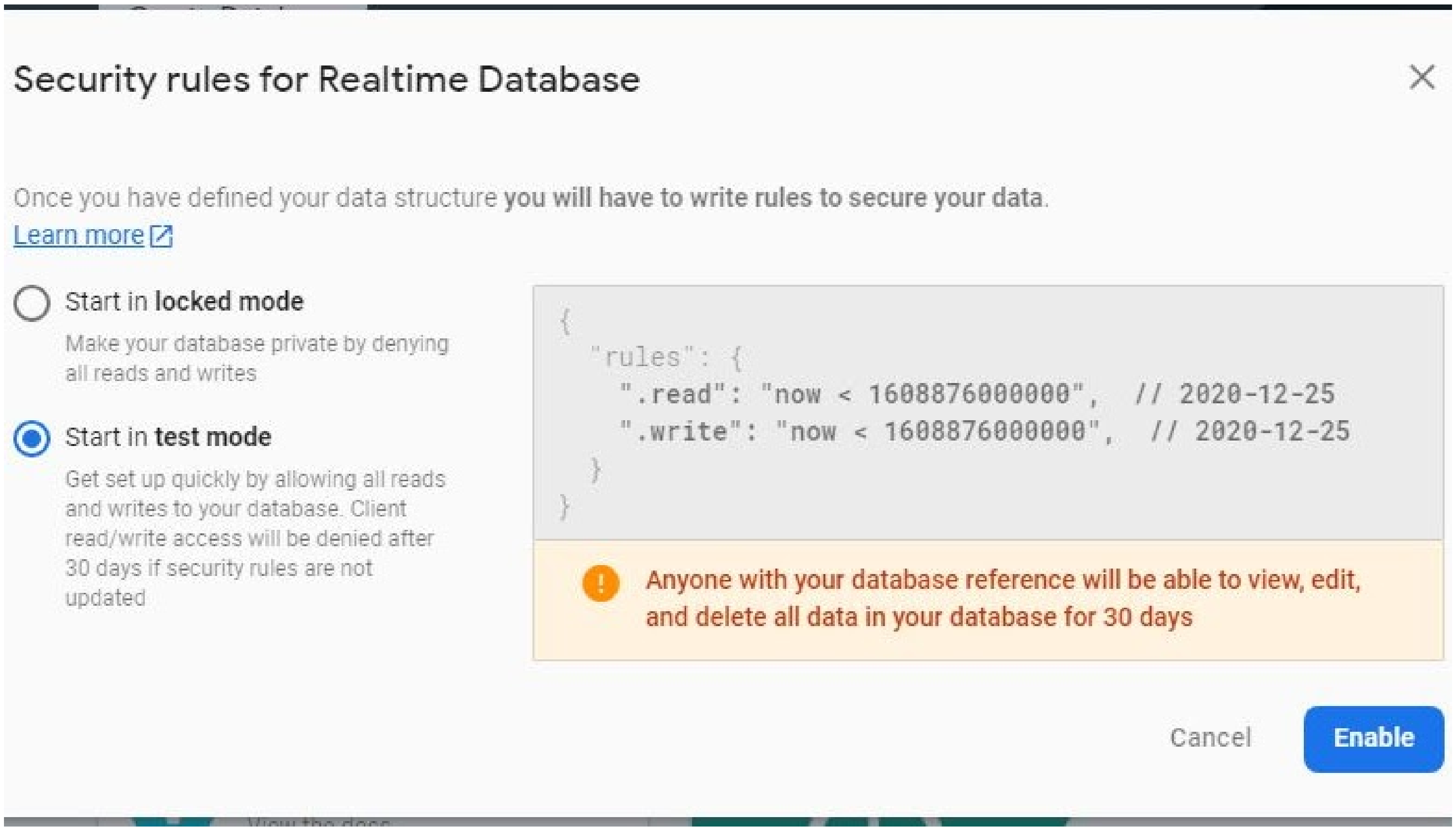
Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.1.1/firebase-app.js"></script

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/8.1.1/firebase-analytics.js"></

<script>
  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  var firebaseConfig = {
    apiKey: "AIzaSyBdPsRoVU7r7p93V8qqDTmBtLog-0V1ZFg",
    authDomain: "msti-510-296515.firebaseapp.com",
    databaseURL: "https://msti-510-296515.firebaseio.com",
    projectId: "msti-510-296515",
    storageBucket: "msti-510-296515.appspot.com",
    messagingSenderId: "3736710384",
    appId: "1:3736710384:web:90f2a0c73a9982f0c92635",
    measurementId: "G-WD8FKSE4JT"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

Learn more about Firebase for web: Get Started, Web SDK API Reference, Samples

Next go to Database and scroll down to find Realtime Database. Click on Create database. Select Start in test mode and enable.





Security rules for Realtime Database

Once you have defined your data structure **you will have to write rules to secure your data.**
Learn more

Start in **locked mode**
Make your database private by denying all reads and writes

[✓ Start in **test mode**]
Get set up quickly by allowing all reads and writes to your database. Client read/write access will be denied after 30 days if security rules are not updated

```
{
  "rules": {
    ".read": "now < 1608876000000",  // 2020-12-25
    ".write": "now < 1608876000000",  // 2020-12-25
  }
}
```

Anyone with your database reference will be able to view, edit, and delete all data in your database for 30 days replacing the rules with the following, and publishing the new rules:

```
{
  "rules": {
    ".read": true,
    ".write": "auth != null"
  }
}
```

Note that these rules still allow any authenticated user to read and write data of other users. If you keep your rules this way, you might later get emails from Google warning you about this security issue.

At this point your control UI should start talking to the database. When you enter text on the control UI and and press Send, the data in the database should change. You can inspect the data by going to Database > Data.

Step 2: Add firebase to your robot face

Next fork one of your robot faces to use in this lab. If needed, add a way for displaying text on the robot face (e.g. speech bubble or subtitle). Set up the robot face to receive updates from Firebase making sure you import all necessary JS libraries. Use the information of your database just like when you set up the control UI in Step 1, so that both your UI and your robot face are talking to the same database.

To make the robot face respond to changes in the database you will need to register a callback for the "value" event (which happens every time a value changes). Revisit the example covered in class if you are unsure how to do that. When an update is received, your robot face should get the current snapshot of the database extract the sentence to be spoken by the robot and then say the sentence using text to speech functionality of the browser.

Step 3: Make your web app talk to Python through Firebase

Next you will make your robot respond to camera events, using Python to process the camera image. You will achieve this by making your Python code receive updates from Firebase, just like in Javascript in Step 2, and then send back the results of image processing to Firebase, so your robot has access to it. Make your robot react to something detected in the image based on the Python image processing functionality you choose. For example, you can make the robot eyes follow the detected human face or make the robot smile when a person appears.

You can heavily reuse the examples from class for this part of the lab: (1) Javascript code for sending camera images to the Firebase database (Codepen link) and (2) Python code for receiving images, processing it, and sending back the results to Firebase (backend.ipynb).

Using the same framework you can control other actions of your robot such as its facial expression or which direction it is looking. Update both your control interface and robot to add another way of controlling your robot.

(Optional): Implement different authentication

Currently your database requires authentication for writing onto the database. Instead of anonymous authentication, implement OAuth to let the control UI log in with a Google account (or something else you prefer) before letting the user control the robot. You can browse Firebase sample code provided by Google to see examples.

(Optional): Fix security of the database

As mentioned earlier, with your current database rules, any user can read and write data of other users. Change your database rules to create separate data for different users and only allow certain users to control the robot. Check Firebase documentation for more information on security.

Step 4: Submit your code on Canvas

Complete this lab by submitting a public link to your CodePen pens on Canvas, by Dec 8 Tuesday, 11:59pm. We will test your application by opening your robot face and control UI making sure the robot can be commanded to say two different sentences. We will inspect code as needed. See Canvas for a grading rubric.