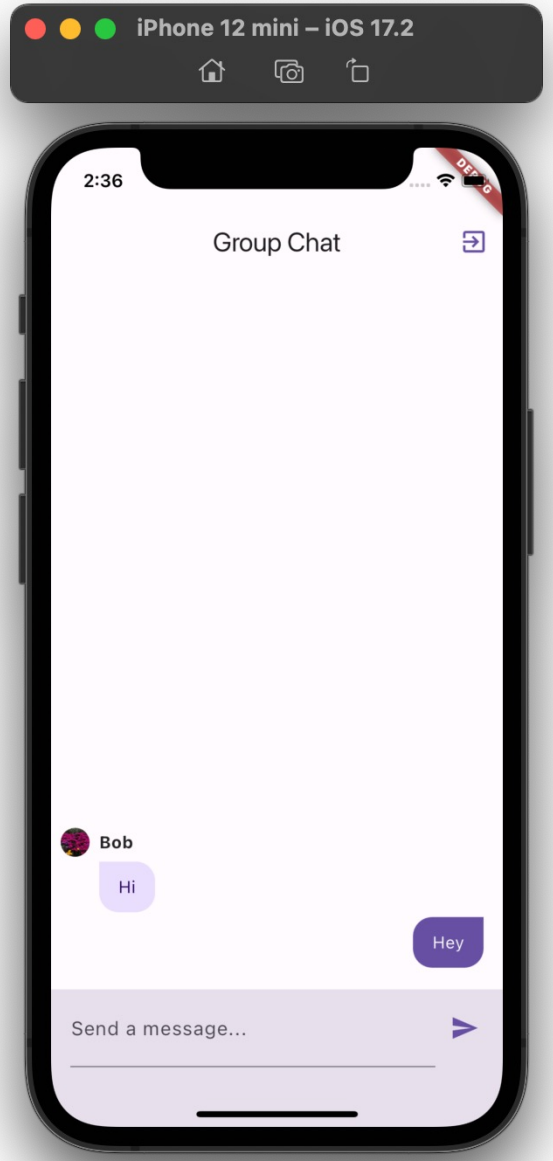
Security Rules & Push Messaging

Shan-Hung Wu
CS, NTHU

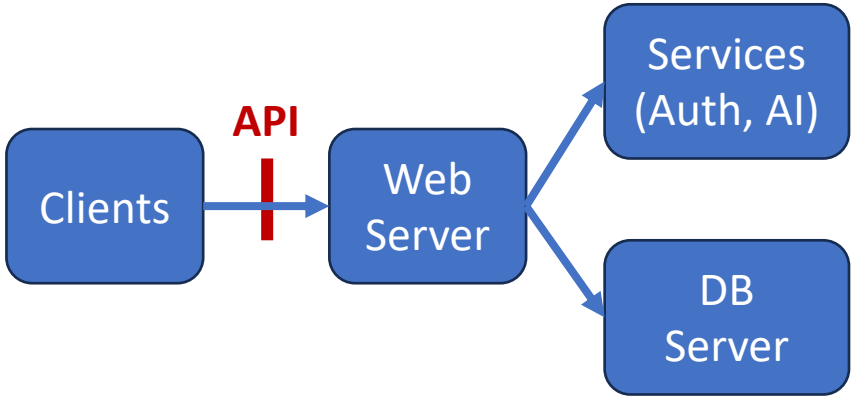
Let’s Chat!

- Real-time messaging
- Authentication
  - Sign up & log in
  - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications

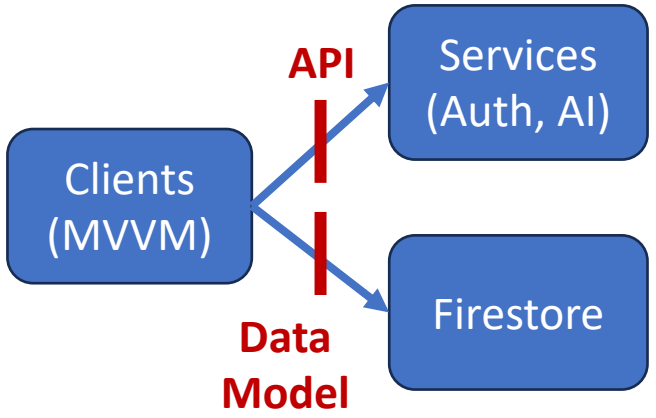


Why Security Rules Matter?

Traditional App Architecture



Architecture w. Firestore



- With Firestore, data model exposed to clients
- Never trust your clients!
- ***Security rules*** protect your data from abuse

Security Rules for Firestore

- Run `firebase init firestore`
- Optionally move “firestore.rules” and “firestore.indexes.json” to “/firestore”
    - Edit pointers in “/firebase.json”
- Edit the firestore.rules
    - Now, we can use auth claim to verify if user has access to a resource
    - E.g., each user should only be able to modify her own to-do items
- Deploy:

```
firebase deploy --only firestore:rules
```

Security Rules for Cloud Storage

- Run `firebase init storage`
  - Select “Storage” and create “storage.rules” file
- Optionally, move “storage.rules” to “/storage” folder and update the path in “/firebase.json”
- Edit the rule
  - E.g., only authenticated users can access data

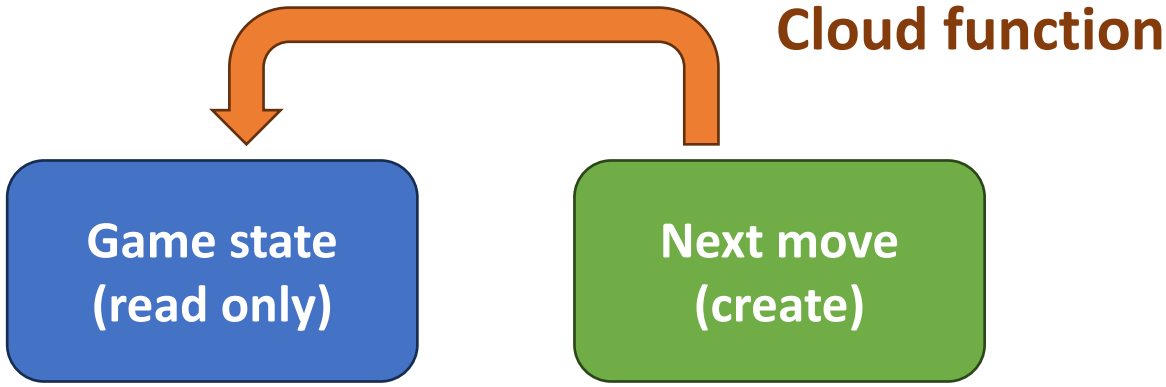
- Deploy:
  `firebase deploy --only storage`

Complex Rules

- Security rules may be complex in some apps

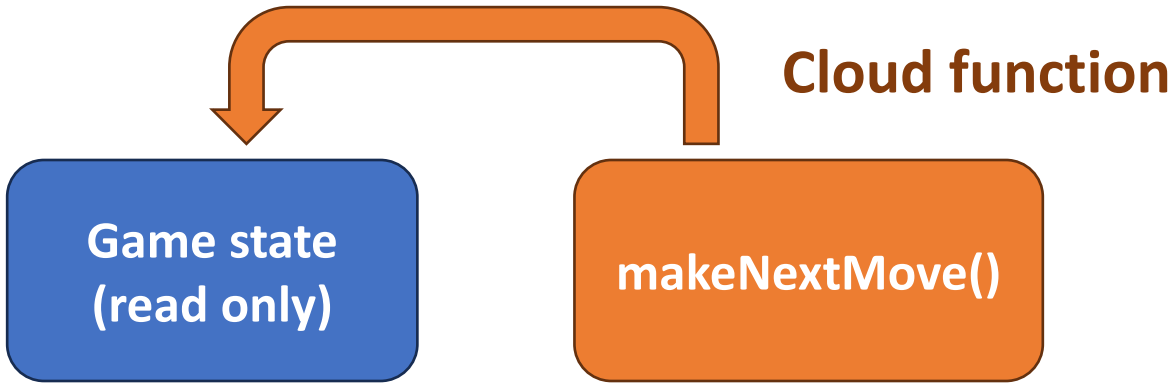
- E.g., in a multiplayer game, you check if a “game state” doc submitted by a user is valid for making a move
  - Is the user participating in the game?
  - Is it the user’s turn?
  - Is the move legal?
  - Did the user skip some previous moves?

Solution 1: Data Denormalization



- Make “game state” read-only
- Allow creation of a “next move” doc
- Then, use Cloud Function to modify “game state” safely
- Costs:
  - Latency
  - No offline updates for game sate

Solution 2: Callable Functions

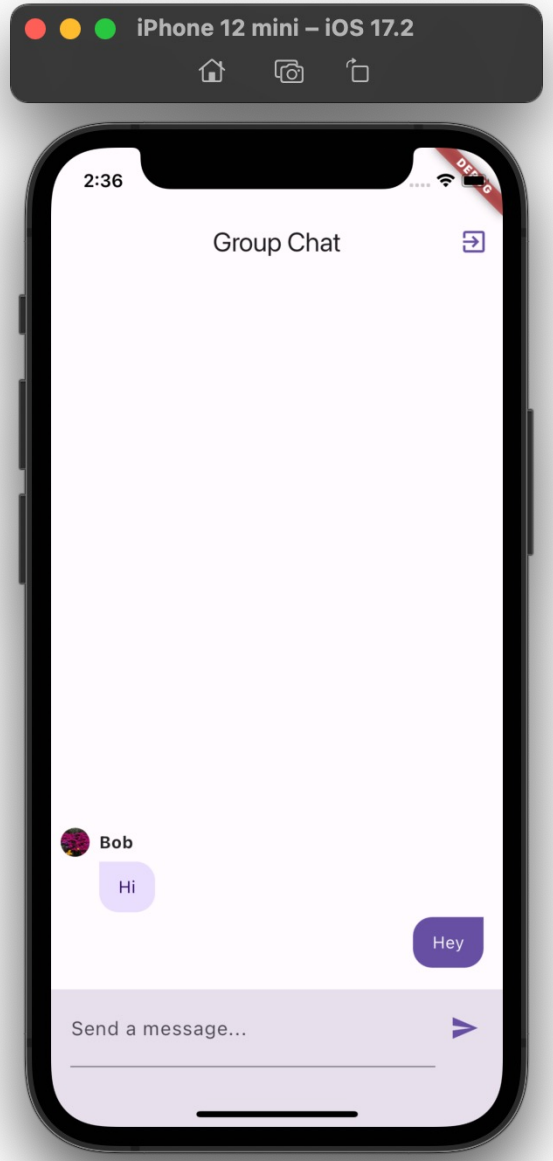


- Make “game state” read-only
- Call the “makeNextMove()”, an HTTP callable

```
const {https } = require("firebase-functions/v2");
exports.makeNextMove = https.onCall(async (request) => {
  const { userId, from, to } = request.data; // params
  ...
}
```

Let’s Chat!

- Real-time messaging
- Authentication
  - Sign up & log in
  - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



Solution 3: Custom JWT Claims

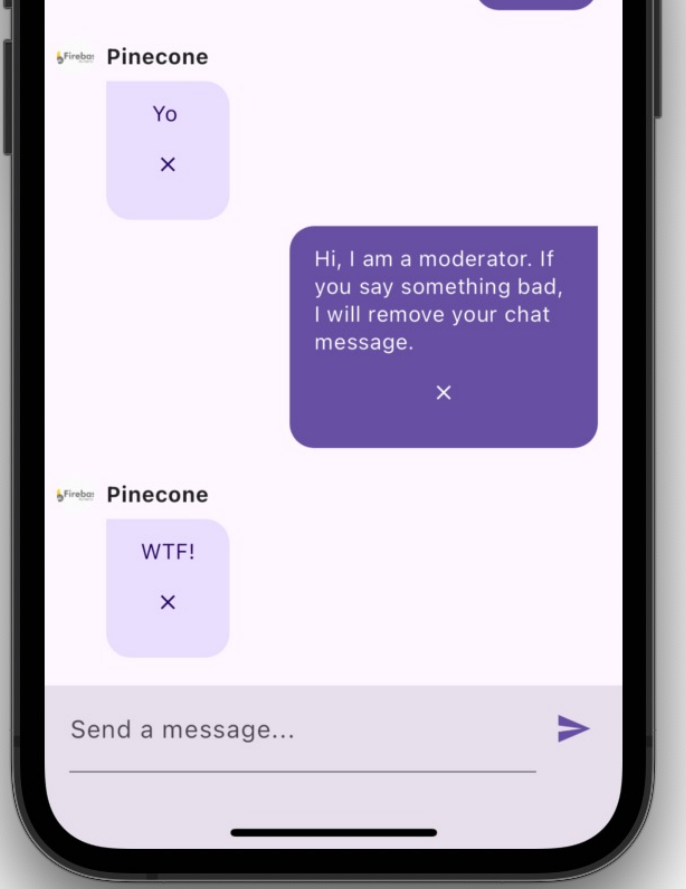
```
(uid, expdate, custom claim
    sha256(uid, expdate, custom claim, secret))
```

- Can be set using Admin SDK in Cloud Functions

```
admin.auth().setCustomUserClaims(
    userId,
    { role: 'admin' },
);
```

- Benefit: save extra doc read

Example: Moderated Chat

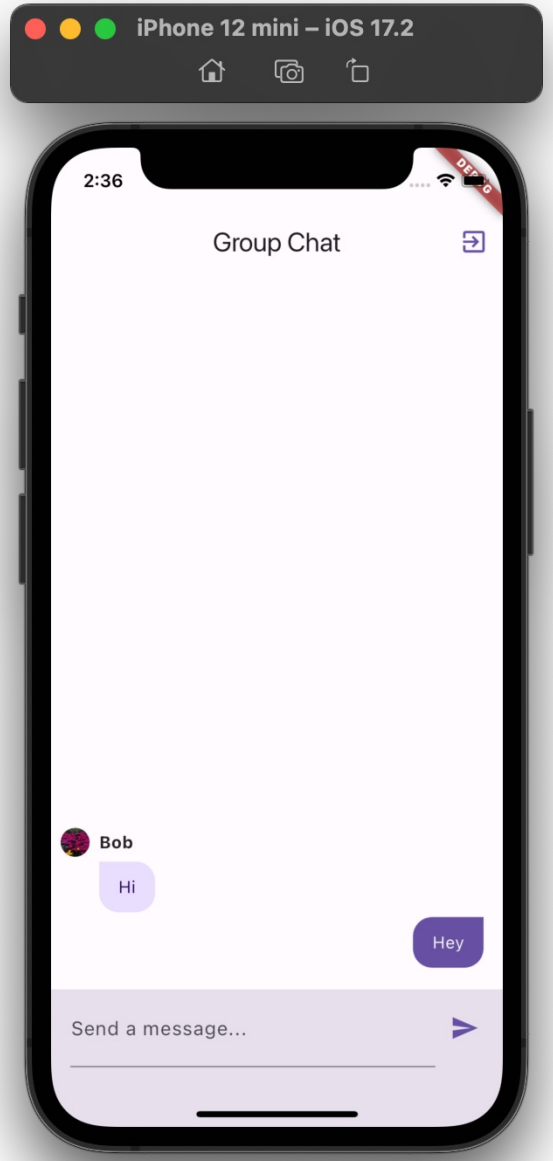


- Limitations:
- ***Not*** applied to clients immediately
  - Ask user to log in again to force token refresh:
- <1000 bytes

```
await FirebaseAuth.instance
                  .currentUser?
                  .getIdToken(true);
```

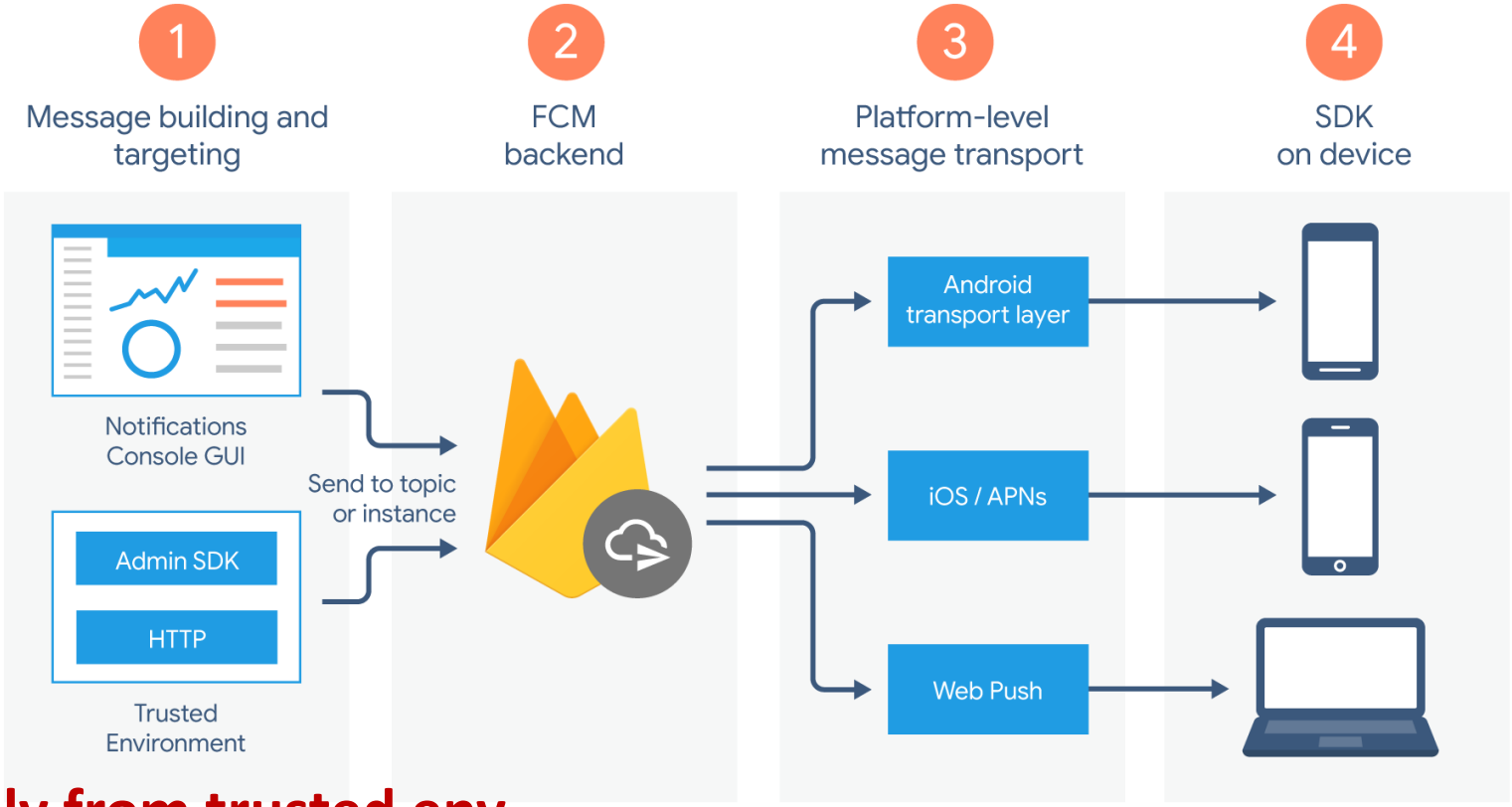
Let’s Chat!

- Real-time messaging
- Authentication
  - Sign up & log in
  - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



Push Messaging

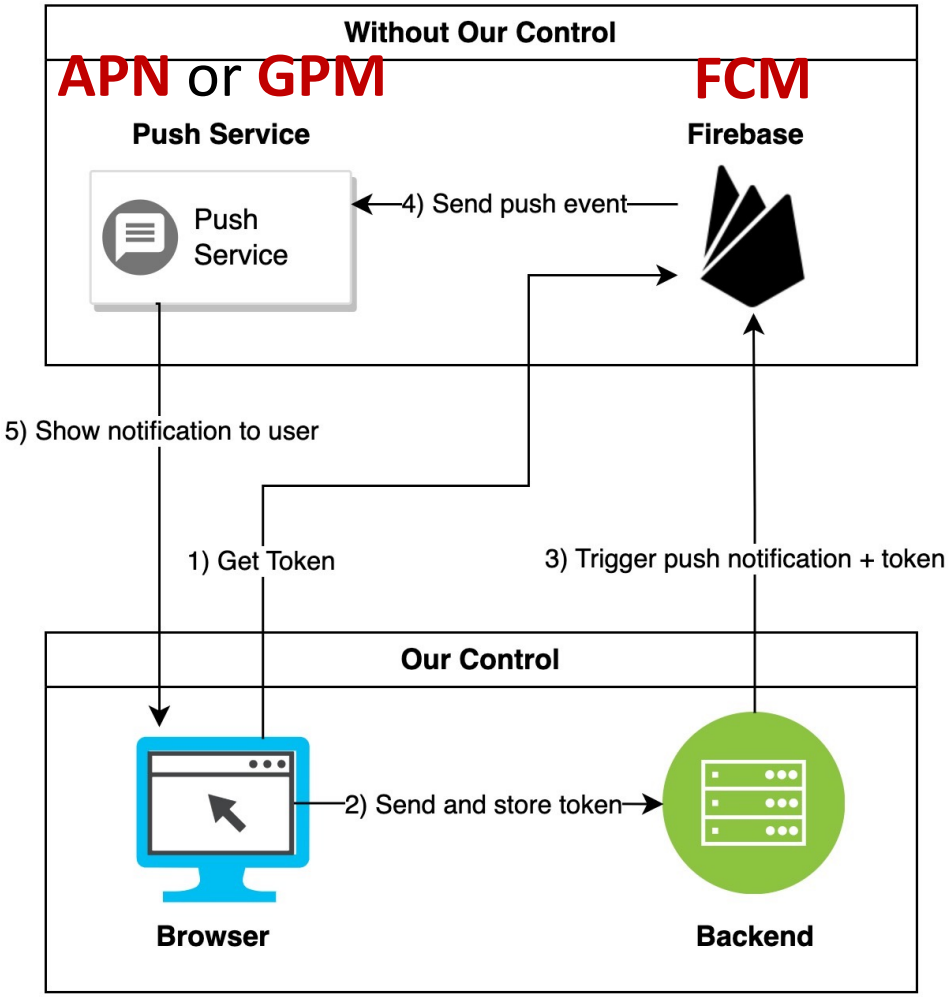
- via Firebase Cloud Messaging (FCM)



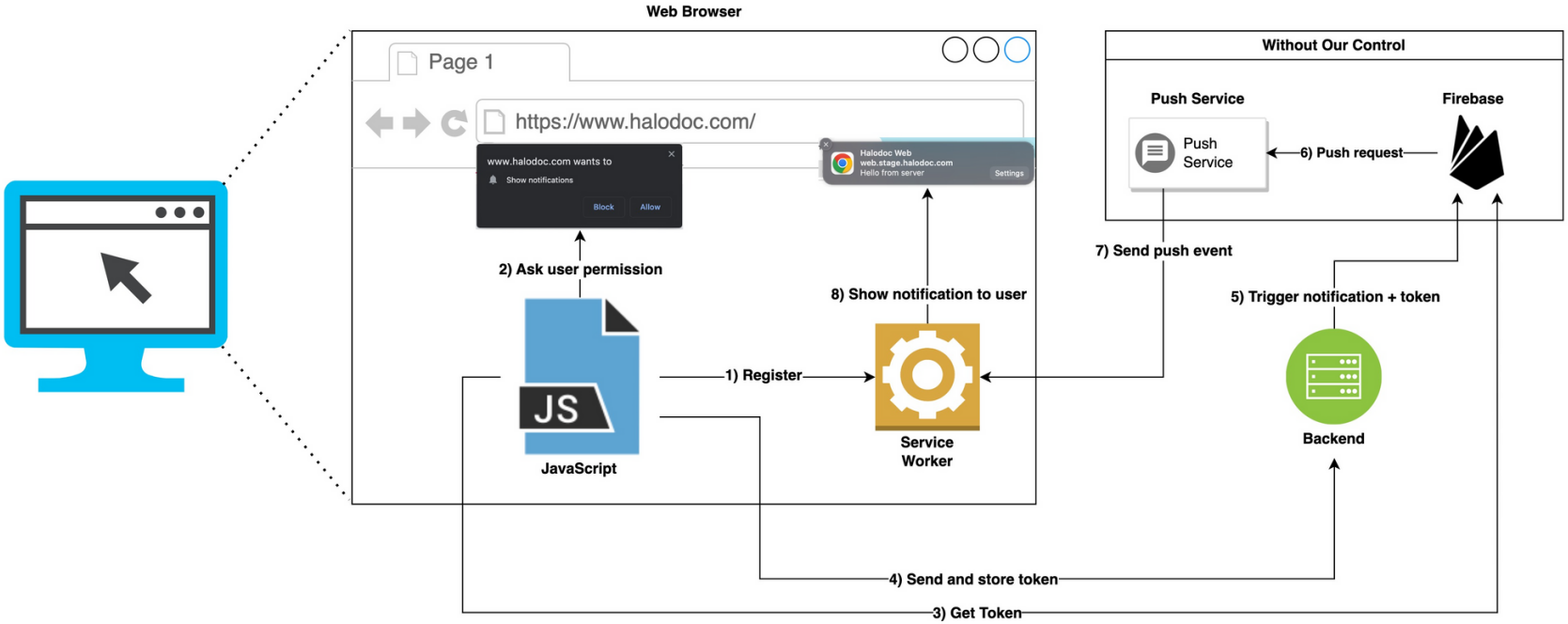
Only from trusted env.

Flow for Mobile Apps

- Sending:
- To a device via ***device token***
- To multiple devices via ***topic subscription***



Flow for Web Apps



- Service worker runs in background
  - See “/web/firebase-messaging-sw.js”
- No `subscribeToTopic()` API
  - Use an HTTP callable instead

Other Message Types & Handling

App \ Message	Notification	Data
Foreground	Not shown by OS; call onMessage()	onMessage()
Background	Shown by OS	onBackgroundMessage()
Terminated	Shown by OS	No handler

- Open callback: `onMessageOpenedApp()`
- If message has both notification and data payloads:
  - Handled by OS first
  - Then by open callback when message opened

References

- Authentication using custom claims + security Rules
- FCM messages
- HTTP callables
- Handling foreground and background messages