LAST GOOD SECURITY RULES

Today October 17, 2025 • 1:01 PM

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // ---------- Helpers ----------
    function isSignedIn() {
      return request.auth != null;
    }

    function userDoc() {
      return get(/databases/$(database)/documents/users/$(request.auth.uid)).data;
    }

    function userExists() {
      return isSignedIn() && exists(/databases/$(database)/documents/users/$(request.auth.uid));
    }

    // Admins are ONLY admin/developer (moderators are NOT admins)
    function isAdmin() {
      return userExists() && (userDoc().role in ['admin', 'developer']);
    }

    function isModerator() {
      return userExists() && (userDoc().role == 'moderator');
    }

    // Works with either:
    //   { permissions: ["manageCompletions", ...] } (list)
    //   { permissions: { manageCompletions: true, ... } } (map)
    function permAllows(permission) {
      return userExists() && (
        userDoc().role == 'developer' || (
          exists(/databases/$(database)/documents/rolePermissions/$(userDoc().role)) &&
          (
            (
              get(/databases/$(database)/documents/rolePermissions/$(userDoc().role)).data.permissions is list &&
              (permission in get(/databases/$(database)/documents/rolePermissions/$(userDoc().role)).data.permissions)
            )
            ||
            (
              get(/databases/$(database)/documents/rolePermissions/$(userDoc().role)).data.permissions is map &&
              get(/databases/$(database)/documents/rolePermissions/$(userDoc().role)).data.permissions[permission] == true
            )
          )
        )
      );
    }

    function isOwner(userId) {
      return isSignedIn() && request.auth.uid == userId;
    }

    // Convenience: expected enrollment doc id
    function enrollmentDocIdFor(userId, courseId) {
      return userId + '_' + courseId;
    }

    // ---------- Quizzes ----------
    match /quizzes/{quizId} {
      allow read: if true;
      allow write: if isSignedIn() && permAllows('manageQuizzes');
    }

    // ---------- Quiz Results ----------
    match /userQuizResults/{resultId} {
      allow read: if isSignedIn() && (
        isAdmin()
        || (resource.data.userId == request.auth.uid)
        || resultId.matches('^' + request.auth.uid + '_.*$')
      );
      allow create: if isSignedIn() && request.resource.data.userId == request.auth.uid;
      allow delete: if isAdmin();
    }

    // ---------- Users ----------
    match /users/{userId} {
      allow read:   if isSignedIn();
      allow create: if true;
      allow write:  if isOwner(userId) || isAdmin();
    }

    // ---------- Courses ----------
    match /courses/{courseId} {
      allow read:   if true;
      allow create: if isAdmin();
      allow delete: if isAdmin();

      // Non-admins may ONLY change enrollmentCount and ONLY by ±1
      // (Admins may update anything.)
      allow update: if isSignedIn()
        || (
          isSignedIn()
          && request.resource.data.diff(resource.data).changedKeys().hasOnly(['enrollmentCount'])
          && (
            request.resource.data.enrollmentCount == resource.data.enrollmentCount + 1
            || request.resource.data.enrollmentCount == resource.data.enrollmentCount - 1
          )
        );
    }

    match /courseGroups/{groupId} {
      allow read, write: if isAdmin();
    }

    // ---------- Ladders (top-level) ----------
    match /ladders/{ladderId} {
      allow read:  if true;
      allow write: if isAdmin();
    }

    // ---------- Enrollments (docId MUST be "<uid>_<courseId>") ----------
    match /enrollments/{enrollmentId} {
      function idMatchesPayload() {
        return request.resource.data.userId is string
               && request.resource.data.courseId is string
               && enrollmentId == enrollmentDocIdFor(request.resource.data.userId, request.resource.data.courseId);
      }

      // Read if admin, owner, or caller is checking their own prefix id
      allow read: if isSignedIn() && (
        isAdmin()
        || resource.data.userId == request.auth.uid
        || enrollmentId.matches('^' + request.auth.uid + '_.*$')
      );

      // Create if signed-in user is enrolling THEMSELVES and id matches "<uid>_<courseId>"
      allow create: if isSignedIn()
        && (isAdmin() || request.resource.data.userId == request.auth.uid)
        && idMatchesPayload();

      // Update if admin or owner; userId/courseId must NOT change
      allow update: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid)
        && request.resource.data.userId == resource.data.userId
        && request.resource.data.courseId == resource.data.courseId;

      // Delete if admin or owner
```

```
  allow delete: if isSignedIn()
    && (isAdmin() || resource.data.userId == request.auth.uid);
}

// ---------- User Video Progress (docId = "<uid>_<courseId>") ----------
match /userVideoProgress/{progressId} {
  function progressIdMatchesPayload() {
    return request.resource.data.userId is string
           && request.resource.data.courseId is string
           && progressId == (request.resource.data.userId + '_' + request.resource.data.courseId);
  }

  allow read: if isSignedIn() && (
    isAdmin()
    || resource.data.userId == request.auth.uid
    || progressId.matches('^' + request.auth.uid + '_.*$')
  );

  allow create: if isSignedIn()
    && (isAdmin() || request.resource.data.userId == request.auth.uid)
    && progressIdMatchesPayload();

  allow update: if isSignedIn()
    && (isAdmin() || resource.data.userId == request.auth.uid)
    && request.resource.data.userId == resource.data.userId
    && request.resource.data.courseId == resource.data.courseId;

  allow delete: if isSignedIn()
    && (isAdmin() || resource.data.userId == request.auth.uid);
}

// ---------- Achievements / Badges ----------
match /userBadges/{badgeId} {
  allow read:  if isSignedIn() && (isAdmin() || resource.data.userId == request.auth.uid);
            allow write: if isAdmin();
}

// ---------- Onsite Completions ----------
match /onsiteCompletions/{completionId} {
  function actorCampus() {
    return userExists() ? userDoc().get('campus', null) : null;
  }

  // READ
  allow read: if isAdmin()
    || (
      isSignedIn()
      && (isModerator() || permAllows('manageCompletions'))
      && (
        actorCampus() == null
        || resource.data.get('userCampus', null) == actorCampus()
        || resource.data.get('userCampus', null) == null
      )
    );

  // CREATE
  allow create: if isAdmin()
    || (
      isSignedIn()
      && (isModerator() || permAllows('manageCompletions'))
      && (
        actorCampus() == null
        || request.resource.data.userCampus == actorCampus()
      )
    );

  // UPDATE & DELETE
  allow update, delete: if isAdmin()
    || (
      isSignedIn()
      && (isModerator() || permAllows('manageCompletions'))
      && (
        actorCampus() == null
        || resource.data.get('userCampus', null) == actorCampus()
        || resource.data.get('userCampus', null) == null
      )
    );
}

// ---------- Forms (public read; narrow public increment) ----------
match /forms/{formId} {
  function isPublicForm() {
    return resource.data.public == true;
  }

  allow get:  if isPublicForm()
    || (isSignedIn() && (permAllows('viewForms') || permAllows('manageForms') || isAdmin()));
  allow list: if isSignedIn() && (permAllows('viewForms') || permAllows('manageForms') || isAdmin());

  allow create, delete: if isSignedIn() && (permAllows('manageForms') || isAdmin());

  allow update: if
    (isSignedIn() && (permAllows('manageForms') || isAdmin()))
    ||
    (
      isPublicForm()
      && request.resource.data.diff(resource.data).changedKeys().hasOnly(['submissionCount'])
      && request.resource.data.submissionCount == resource.data.submissionCount + 1
    );
}

// ---------- Contents ----------
match /Contents/{contentId} {
  allow read:  if resource.data.status == 'published' || isSignedIn();
  allow write: if isAdmin();

  match /comments/{commentId} { allow read: if true; allow write: if isSignedIn(); }
  match /likes/{likeId}     { allow read: if true; allow write: if isSignedIn(); }
  match /shares/{shareId}   { allow read: if true; allow write: if isSignedIn(); }
}

// ---------- Languages ----------
match /languages/{langId} {
  allow read:  if resource.data.status == "published" || isSignedIn();
  allow write: if isAdmin();
}

// ---------- Translations ----------
match /translations/{translationId} {
  allow read, write: if permAllows('manageLocalization');
}

// ---------- Taxonomy / Catalog ----------
match /courseCategories/{categoryId} { allow read: if true; allow write: if isAdmin(); }
match /courseLevels/{levelId}        { allow read: if true; allow write: if isAdmin(); }

// ---------- Public Speaker Profiles ----------
match /speakers/{speakerId} { allow read: if true; allow write: if isAdmin(); }

// ---------- Site Settings ----------
match /siteSettings/main { allow read: if true; allow write: if isAdmin(); }

// ---------- Role Permissions ----------
match /rolePermissions/{role} {
  allow read:  if isSignedIn();
  allow write: if isAdmin();
}

// ---------- Navigation Links ----------
match /navLinks/{linkId} { allow read: if true; allow write: if isAdmin(); }

// ---------- Live Events ----------
```

```
    match /liveEvents/{eventId} { allow read: if isSignedIn(); allow write: if isAdmin(); }

    // ---------- Community ----------
    match /communityPosts/{postId} {
      allow read:  if permAllows('viewCommunityPage');
      allow write: if isSignedIn();
      match /replies/{replyId} { allow read: if permAllows('viewCommunityPage'); allow write: if isSignedIn(); }
    }

    // ---------- Certificates ----------
    match /certificates/{certId} { allow read: if isSignedIn(); allow write: if isAdmin(); }

    // ---------- Promotion Requests ----------
    match /promotionRequests/{reqId} {
      allow get, list: if permAllows('managePromotions');
      allow update, delete: if permAllows('managePromotions');
      allow create: if isSignedIn() && request.resource.data.userId == request.auth.uid;
    }

    // ---------- Static Catalogs ----------
    match /Campus/{campusId}             { allow read: if true; allow write: if isAdmin(); }
    match /charges/{chargeId}            { allow read: if true; allow write: if isAdmin(); }
    match /maritalStatuses/{statusId}    { allow read: if true; allow write: if isAdmin(); }
    match /membershipStatuses/{statusId} { allow read: if true; allow write: if isAdmin(); }
    match /ministries/{ministryId}       { allow read: if true; allow write: if isAdmin(); }

    // ---------- Documentation ----------
    match /documentation/{docId} { allow read: if true; allow write: if isAdmin(); }

    // ---------- Announcements ----------
    match /announcements/{announcementId} { allow read: if true; allow write: if isAdmin(); }
  }
}
```

————————————— END OF RULES ——————————————
REFFERENCE: https://chatgpt.com/s/t_68b6c23ba6008191b02e7e3fc6a1e8b0