0 — Project Bootstrap (1 hour)

1. **Create two Git repos**
    - operation-summer-web (React)
    - operation-summer-mobile (Flutter)
2. **Create a Google Cloud project** named Operation-Summer-Prod.
3. **Enable billing** on that project.
4. **Install global tooling** on the dev box / Codespace

bash

CopyEdit

npm i -g firebase-tools

dart --version   # ensure Flutter is installed

1 — Firebase Foundation (30 min)

Inputs you must provide
• GCP project ID
• Firebase CLI login (Google account)

1. firebase init in an empty folder → choose **Firestore, Storage, Cloud Functions, Hosting**.
2. Pick **JavaScript + ESLint** for Functions.
3. In Firestore rules, paste the starter rules below (will tighten later):

hcl

CopyEdit

```
rules_version = '2';
service cloud.firestore {
  match /databases/{db}/documents {
    match /families/{famId}/{document=**} {
      allow read, write: if request.auth.token.familyId == famId;
    }
    match /users/{userId} {
      allow read: if request.auth.uid == userId;
      allow write: if false;
    }
  }
}
```

4. Deploy: firebase deploy --only firestore:rules.

2 — Google API Credentials (45 min)

Inputs you must provide
• Credit-card enabled GCP project
• OAuth consent-screen info (app name, logo, support email)

1. **OAuth client**
    - In **Google Cloud → APIs & Services → Credentials** create an *OAuth 2.0 Client ID* (type "Web application").
    - Add http://localhost:3000 and https://<your-firebase-app>.web.app to **Authorized JS origins**.
    - Add https://<your-firebase-app>.web.app/__/auth/handler to **Authorized redirect URIs**.
    - Download the client_id and client_secret.
2. **Enable APIs** (left nav → "Library")
    - **Google Calendar API**
    - **Google Drive API**
    - **Google Sheets API**
    - **YouTube Data API v3**
3. **Create a YouTube API key** (for public search only).
4. **Store secrets**
    - Put the OAuth client JSON into Firebase Functions config:

bash

CopyEdit

firebase functions:config:set google.oauth.client_id="..." google.oauth.client_secret="..."

    - Store the YouTube API key likewise:

bash

CopyEdit

firebase functions:config:set youtube.key="..."

3 — OpenAI Setup (10 min)

Inputs you must provide
• Paid OpenAI account (with billing)
• Secret key

1. Generate an **API key** under *platform.openai.com*.
2. In Functions config:

bash

CopyEdit

firebase functions:config:set openai.key="sk-..."

4 — Print Service (Optional 20 min)

Inputs you must provide
• PrintNode (or ezeep) account
• Printer already paired with the PrintNode desktop agent

1. Copy the PrintNode **API key**.
2. Store in Functions config:

bash

CopyEdit

firebase functions:config:set printnode.key="..."

3. Note your **printerId** from the PrintNode dashboard; record it in a Firestore doc /families/<famId>/settings/print {enabled:true, printerId:12345}

5 — n8n Orchestration (60 min)

Inputs you must provide
• Server/VM credentials (or Cloud Run)
• .env values collected above

1. Deploy n8n (Docker easiest).

bash

CopyEdit

```
docker run -d --name n8n \
  -p 5678:5678 \
  -e WEBHOOK_TUNNEL_URL=https://n8n.yourdomain.com \
  -e TZ=America/Chicago \
  --restart unless-stopped \
  n8nio/n8n
```

2. In n8n UI:
    - Add **OpenAI** creds (openai.key).
    - Add **Google** OAuth credentials (client_id / secret) with scopes:

arduino

CopyEdit

```
https://www.googleapis.com/auth/calendar.readonly
https://www.googleapis.com/auth/drive.file
https://www.googleapis.com/auth/spreadsheets.readonly
```

Select "offline" access so n8n stores a refresh token.

3. Build **"Daily-Content" workflow** with these nodes:

1. Cron (05:00)
2. Firestore → fetch family & kid docs
3. Google Calendar → today's events
4. GPT-4o → generate script / worksheet / project / exercise
5. DALLE → coloring PNG
6. Function node → make PDF (use pdfkit in Code step)
7. Drive → upload files
8. Firestore → write /dailyContent/<date>
9. HTTP → call Cloud Function /sendPush
10. IF(settings.print.enabled) → HTTP to PrintNode /print

6 — Cloud Functions Skeleton (30 min)

Run firebase functions:config:get > .runtimeconfig.json locally to expose your secrets.

Create functions ⬇

ts

CopyEdit

```
// /functions/src/index.ts
import * as functions from 'firebase-functions';
import * as admin from 'firebase-admin';
admin.initializeApp();

export const sendPush = functions.https.onRequest(async (req, res) => {
  const {familyId, title, body} = req.body;
  const tokensSnap = await admin.firestore()
    .collection('families').doc(familyId)
    .collection('deviceTokens').get();
  const tokens = tokensSnap.docs.map(d => d.id);
  await admin.messaging().sendMulticast({tokens, notification:{title, body}});
  res.sendStatus(204);
});

export const choreHook = functions.firestore
  .document('families/{famId}/chores/{choreId}')
  .onUpdate(async (change, ctx) => {
    const after = change.after.data();
    const before = change.before.data();
    if (!before.completed && after.completed) {
      await admin.firestore().doc(`users/${after.assignedTo}`)
        .update({points: admin.firestore.FieldValue.increment(after.points)});
    }
  });
```

Deploy: firebase deploy --only functions.

7 — React Parent App (1 day sprint)

Inputs you must provide
• firebaseConfig snippet for web
• Google OAuth client ID

1. Scaffold:

bash

CopyEdit

npx create-react-app operation-summer-web --template typescript

cd operation-summer-web

npm i firebase @mui/material react-router-dom

2. Set up **Firebase Auth** with Google provider; on login store familyId in local storage.
3. Implement pages:
    - Dashboard (calendar agenda + chores table)
    - Chore editor modal (writes to Firestore)
    - Daily content viewer (reads /dailyContent/today)
    - Settings (save PrintNode toggle / device token)
4. Build & deploy to Firebase Hosting:

bash

CopyEdit

npm run build

firebase deploy --only hosting

8 — Flutter Kids App (1 day sprint)

Inputs you must provide
• google-services.json / GoogleService-Info.plist from Firebase
• Firebase project IDs

1. Scaffold:

bash

CopyEdit

flutter create operation_summer_mobile

cd operation_summer_mobile

flutter pub add firebase_core firebase_auth cloud_firestore firebase_messaging youtube_player_flutter printing

2. Init Firebase in main.dart and sign in anonymously—or via custom token if parent logs in first and creates child tokens.
3. Core screens:
    - **TodayScreen** (events + chores list)
    - **ActivitiesScreen** (video player, coloring page download + print, worksheet viewer)
    - **RewardsScreen** (points bar)
4. Listen to FCM; on "daily_ready" open Activities screen.
5. Implement Printing.layoutPdf to print coloring + worksheet upon tap.
6. Build to Android/iOS; test FCM & Firestore updates.

9 — Device Token Registration (10 min)

Add to both apps:

js

CopyEdit

```
import { getToken } from 'firebase/messaging';
const token = await getToken(messaging, { vapidKey: '<public-VAPID>' });
await setDoc(doc(db,'families',familyId,'deviceTokens',token), {created:Date.now()});
```

10 — Test Day

1. Manually trigger n8n workflow (“Execute Once”).
2. Confirm:
    - Firestore dailyContent/<today> populated
    - PDFs & video in Storage/Drive
    - Push arrives on mobile
    - Printout spits from printer (if enabled)
3. Complete a chore in mobile, verify points increment & push to parent.

External Accounts & Keys Checklist

Service	Key / Token	Where to store
Google OAuth	client_id, client_secret	Functions config, n8n creds
Google API	YouTube API key	Functions config
OpenAI	sk-...	Functions config, n8n creds
Firebase	Service account (for n8n)	n8n .env or credential file
PrintNode	API key, printerId	Functions config; printerId in Firestore
VAPID (FCM web)	public VAPID key	React app env

Final Word

Hand this document to your AI coding agent **exactly in order**.
If a step requires a secret, supply it through environment variables or Firebase Functions config *before* the agent proceeds.
Once all ten blocks are complete, Operation Summer will be operational and ready for real-family testing.