Operation Summer – Authoritative Architecture & Design Portfolio

Version 0.9 – May 22 2025 (maintain this file as the single source of truth; update version + date on every major change)

---

0 Purpose & Scope

This portfolio assembles **all scaffolding, decisions, and design assets** required for an AI-agent (or human team) to take Operation Summer from green-field to working production. It intentionally omits the n8n/education-generation pipelines per directive, and focuses on:

1. **Flutter Front-End** – UI/UX, widget hierarchy, state, theming
2. **Firebase Back-End** – Auth, Firestore, Storage, Cloud Functions
3. **Calendar + Chore Board** – domain models, sync logic, Google APIs
4. **Print Pipeline** – Firestore queue, Cloud Functions, local CUPS daemon
5. **Dev & Ops** – repo layout, CI/CD, deployment, environment configs

The document is fully self-contained so an autonomous agent can bootstrap the repo, configure cloud resources, generate boiler-plate code, and compile on first run.

1 Technology Stack Overview

Layer	Chosen Tech	Rationale
UI	Flutter 3.22 (Dart 3)	Single code-base for iOS / Android / PWAs; strong widget ecosystem
State Mgmt	`riverpod` 2.x	Compile-safe DI, testable, no context gymnastics
UI Widgets	Syncfusion `sfcalendar`, BoardView (`kanban_board`)	Rich calendar DnD; out-of-box Kanban
Design System	Tailored tokens (see §4.5) with Material 3 theming	Consistent brand across mobile & web
Auth	Firebase Auth (Email/Pass + Google OAuth 2)	Native provider in FlutterFire
DB	Cloud Firestore (regional: `us-central1`)	Realtime sync, offline cache
Storage	Firebase Storage	Secure, signed URLs for PDFs/images
Serverless	Cloud Functions v2 (Node 20, TypeScript)	Scale-to-zero, Google SDKs builtin
Scheduling	Cloud Scheduler (+ Functions)	Run daily auto-print at 07:00 AM CT
Google APIs	Calendar v3, Sheets v4 (REST via Auth token)	Native integration; quotas generous
Print Daemon	Node.js script + CUPS on Raspberry Pi 4	Local network printing; secure outbound-only
CI/CD	GitHub Actions with Firebase CLI & Flutter	Automated test + deploy on main merge
Hosting	Firebase Hosting (web PWA)	Free SSL, CDN

Key decision points resolved:

- *State mgmt*: Riverpod chosen over Provider/Bloc for compile-time safety and easier DI.
- *Calendar widget*: Syncfusion is commercial → use **community licence** (free for indie/individual) which covers our use-case.
- *Local print* vs cloud: Google Cloud Print is EOL → adopt LAN CUPS proxy pattern.
- *Single-account vs multi-account*: v1 uses **single parent account** with multiple child profiles; roles extensible later.

2 Repository Layout (Monorepo)

```
operationsummer/
├── app/                          # Flutter root
│   ├── lib/
│   │   ├── core/                # constants, theming, utils
│   │   ├── features/
│   │   │   ├── auth/
│   │   │   ├── children/
│   │   │   ├── chores/
│   │   │   ├── calendar/
│   │   │   ├── lessons/
│   │   │   └── printing/
│   │   └── main.dart
│   └── pubspec.yaml
├── functions/                    # Cloud Functions (TypeScript)
│   ├── src/
│   │   ├── chores.sync.ts
│   │   ├── sheets.sync.ts
│   │   ├── print.enqueue.ts
│   │   ├── print.scheduler.ts
│   │   └── index.ts             # exports
│   ├── package.json
│   └── tsconfig.json
├── print-daemon/                 # Local CUPS bridge
│   ├── index.js
│   └── service.install.md
├── infra/                        # IaC & config
│   ├── firestore.rules
│   ├── firestore.indexes.json
│   ├── github/                  # GH Actions workflows
│   │   └── flutter-ci.yml
│   └── env.example
└── docs/                         # This file plus diagrams
```

Instruction to agent: create directories exactly as above before code-gen.

3 Development Environment Setup

1. **Clone repo** → `git clone ...`
2. **Flutter**: `fvm install 3.22.0 && fvm use 3.22.0` (optional fvm)
3. **Firebase CLI**: `npm i -g firebase-tools` (≥12)
4. **Google Cloud** SDK if deploying functions locally
5. **Secrets**: copy `infra/env.example` → `.env.local`; fill:

```
FIREBASE_PROJECT_ID=
GOOGLE_CLIENT_ID_IOS=
GOOGLE_CLIENT_ID_ANDROID=
GOOGLE_CLIENT_ID_WEB=
SHEETS_SPREADSHEET_ID=
DEFAULT_PRINTER=HomePrinter
```

6. **Print daemon host**: install Node 20 & CUPS. Follow `print-daemon/service.install.md`.
7. **Emulator Suite**: `firebase emulators:start` (Firestore, Auth, Functions) for local test.

4 Flutter Application Architecture

4.1 Layered Structure

- **Presentation Layer** – Widgets + Riverpod `StateNotifier`s
- **Domain Layer** – Pure Dart models & business logic (eg. `Chore`, `Lesson`, `Event`)
- **Data Layer** – Firestore, Calendar, Sheets repositories implementing interfaces

Agent: generate code using *Clean Architecture template*; each feature folder mirrors the three layers.

4.2 State Management

- Use `** + **`.
- Global providers: `authProvider`, `currentChildProvider`, `firestoreProvider`, `calendarProvider`.
- Feature providers: `choreListProvider(childId)`, `lessonProvider(date, childId)`, etc.

4.3 Navigation

- Use `go_router` with typed routes; nested navigation for tabs.

4.4 Core Dependencies (add to pubspec)

```
flutter:
  sdk: flutter
firebase_core: ^3.0.0
firebase_auth: ^5.0.0
cloud_firestore: ^5.0.0
firebase_storage: ^12.0.0
google_sign_in: ^7.0.0
flutter_riverpod: ^3.0.0
riverpod_annotation: ^3.0.0
syncfusion_flutter_calendar: ^23.1.40
kanban_board: ^1.3.0
flutter_pdfview: ^2.1.0
printing: ^6.2.0
go_router: ^10.0.0
intl: ^0.19.0
```

Run `flutter pub run build_runner build --delete-conflicting-outputs` for riverpod codegen.

4.5 Design System Tokens

```
// core/theme.dart
const seedColor = Color(0xFF4E8C68);  // Farm green
const primary = MaterialColor(0xFF4E8C68, {...});
// Typography scale (Google Fonts "Rubik")
// Sizes: displayLarge 32, headlineMedium 24, bodyLarge 16, labelLarge 14
```

Component library: use Material 3 `FilledButton`, `Card`, `NavigationBar`. For kid avatars, use CircleAvatar with emoji fallback.

4.6 Widget Skeletons

```
DashboardScreen
 ├── ProfileHeader
 ├── TodaySummaryCard
 ├── QuickNavGrid
ChoreBoardScreen
 ├── KanbanBoard (3 columns)
CalendarScreen
 ├── SfCalendar (month/week)
LearningHubScreen
 ├── DateSelectorBar
 ├── LessonList
     └── LessonCard → LessonDetailPage (PDF viewer)
```

5 Firebase Back-End Architecture

5.1 Firestore Collections

```
users/{uid}
  ├── children/{childId}
  │     ├── chores/{choreId}
  │     └── lessons/{lessonId}
  └── printJobs/{jobId}
```

See §1 table for fields.

5.2 Security Rules (excerpt)

```
match /users/{userId} {
  allow read, write: if request.auth.uid == userId;
  match /children/{childId}/{sub=**} {
    allow read, write: if request.auth.uid == userId;
  }
  match /printJobs/{jobId} {
    allow read: if request.auth.uid == userId;
    allow write: if request.auth.uid == userId;
  }
}
```

5.3 Cloud Functions Modules

File	Trigger	Purpose
`chores.sync.ts`	Firestore onWrite `children/*/chores/*`	Award points; push change to Sheets
`sheets.sync.ts`	HTTPS `/sheetWebhook`	Receive Apps Script updates, mutate Firestore
`print.enqueue.ts`	Callable `printAllMaterials`	Build file list → create `printJobs` doc
`print.scheduler.ts`	Pub/Sub schedule daily 07:00	Auto-enqueue print jobs per child

Each function authenticates Sheets API using service-account JSON via `googleapis`.

6 Google API Integration Layer

6.1 OAuth Flow

- Use `google_sign_in` requesting scopes:
  - `email`, `https://www.googleapis.com/auth/calendar`, `https://www.googleapis.com/auth/spreadsheets`
- Store access token in memory; refresh via plugin.

6.2 Dart Service Classes

```
class CalendarService {
  final calendarApi = calendar.CalendarApi(client);
  Future<List<Event>> fetchEvents(DateTime from, DateTime to) {...}
  Future<void> updateEvent(String id, DateTime start, DateTime end) {...}
}
```

Sheets is handled server-side; no client sheet code.

7 Print Pipeline Subsystem

7.1 PrintJobs Doc

```
{
  "files": ["gs://bucket/aj_worksheet.pdf", ...],
  "printer": "HomePrinter",
  "status": "queued",
  "timestamp": 2025-05-22T12:00:00Z
}
```

7.2 Local Print Daemon (Node 20)

`print-daemon/index.js` skeleton:

```
import {initializeApp, cert} from 'firebase-admin/app';
import {getFirestore} from 'firebase-admin/firestore';
import {exec} from 'child_process';
// init admin
const db = getFirestore();
const PRINTER = process.env.DEFAULT_PRINTER;

function watch() {
  db.collection('users').doc(process.env.USER_ID)
    .collection('printJobs').where('status','==','queued')
    .onSnapshot(snap => snap.docChanges().forEach(async ch => {
      const job = ch.doc;
      await job.ref.update({status:'printing'});
      for (const url of job.data().files) {
        const file = await download(url);
        exec(`lp -d ${PRINTER} "${file}"`);
      }
      await job.ref.update({status:'completed'});
    }));
}
```

Set as systemd service (see `service.install.md`).

8 Continuous Integration / Delivery

8.1 GitHub Actions Workflow (`infra/github/flutter-ci.yml`)

```
name: Flutter CI
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - uses: subosito/flutter-action@v2
      with:
        flutter-version: '3.22.0'
    - run: flutter pub get
    - run: flutter analyze
    - run: flutter test
  deploy-functions:
    needs: build
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
    - uses: actions/checkout@v4
    - run: npm i -g firebase-tools
    - run: firebase deploy --only functions --token ${{ secrets.FIREBASE_TOKEN }}
```

Add similar job for `flutter build web` + `firebase deploy --only hosting`.

9 Deployment Topology

1. **Firebase project** in `us-central1` (low latency to Wisconsin).
2. **App platforms**: Android (Play private release), iOS TestFlight, Web (https://opsummer.web.app).
3. **Local network**: Raspberry Pi (192.168.1.x) runs CUPS + print-daemon; firewall blocks inbound; outbound 443 allowed.
4. **Service account** for Sheets API; spreadsheet shared with it.

10 Environment Configuration Matrix

Key	Location	Used By
FIREBASE_PROJECT_ID	.env, CI secret	Flutter, Functions, Daemon
GOOGLE_CLIENT_ID_*	.env, iOS/Android plist	Flutter OAuth
SHEETS_SPREADSHEET_ID	Firestore user doc or env	Functions
DEFAULT_PRINTER	.env on Pi	Print daemon
SERVICE_ACCOUNT_JSON	Secret Manager	Functions

11 Testing Strategy

- **Unit Tests**: models, providers logic (`flutter_test`).
- **Widget Tests**: ChoreCard drag behaviour.
- **Integration**: Use `integration_test` package to simulate add → complete chore → assert points.
- **Backend Tests**: Use Firebase Emulator to test Cloud Functions triggers.
- **Print Pipeline**: In CI, mock `lp` command to assert it is called.

12 Key Decision Points & Rationale

Decision	Options Considered	Selected	Reasoning
State mgmt	Provider, Bloc, Riverpod	**Riverpod**	Compile-time safe, minimal boiler-plate
UI Calendar	Syncfusion, TableCalendar, custom	**Syncfusion**	Built-in DnD, week/day views
Kanban	Homemade DnD, Trello-clone pkg	**kanban_board**	Fast, reliable, MIT licence
Print	Browser print, Cloud Print, CUPS daemon	**CUPS**	Cloud Print sunset, LAN reliability
Auth model	Child logins vs single parent	**Single parent**	Reduces complexity v1, can extend
Sheets sync location	Client-side vs server	**Cloud Functions**	Avoids exposing sheet creds

13 Risk & Mitigation Matrix

Risk	Impact	Likelihood	Mitigation
Syncfusion licence changes	UI stops working	Low	Indie licence archived in repo; fallback to TableCalendar
Local print daemon offline	Materials not printed	Med	UI shows job error; manual retry; daily summary email
Google API quota	Calls fail	Low	Use incremental sync; request higher quota if needed
Firestore billing spikes	Cost	Low	Enable budget alert; index sparingly

14 Future Extensions

- Child-facing login mode
- Rewards store redeeming points
- Push notifications (FCM) for chores
- Offline (PWA) PDF caching

Appendix A – Baseline `pubspec.yaml`

```
name: operation_summer
version: 0.1.0
environment:
  sdk: ">=3.2.0 <4.0.0"
dependencies:
  flutter:
    sdk: flutter
  # core packages (see §4.4)
  firebase_core: ^3.0.0
  ...
```

Appendix B – `functions/package.json` (excerpt)

```
{
  "dependencies": {
    "firebase-admin": "^12.0.0",
    "firebase-functions": "^4.0.0",
    "googleapis": "^133.0.0",
    "pdfkit": "^0.14.0"
  },
  "devDependencies": {
    "typescript": "^5.4.0"
  },
  "engines": { "node": "20" }
}
```

Appendix C – full Firestore Rules

(see infra/firestore.rules)

Appendix D – GitHub Actions `flutter-ci.yml`

(see §8.1 for snippet; full file under infra/github)

Appendix E – Print daemon install script

Refer to `print-daemon/service.install.md` – includes:

```
sudo apt-get install cups
sudo lpadmin -p HomePrinter -E -v usb://...
# Service setup
sudo cp print-daemon/ops-print.service /etc/systemd/system/
sudo systemctl enable --now ops-print.service
```

Appendix F – Design Tokens JSON (sample)

```
{
  "color": {
    "primary": "#4E8C68",
    "secondary": "#FFC857",
    "background": "#F9F9F9"
  },
  "spacing": {
```

```
    "xs": 4,
    "sm": 8,
    "md": 16,
    "lg": 24,
    "xl": 32
  }
}
```

End of Document