**Technical Documentation**

Project Title: Meet and Media Sync (CloudCapture)

Date: April 2025

Developers: Ridam and Bhavya

Repository: <https://github.com/phaserunner03/meetAndMediaSync>

# Table of Content:

1. Project Overview
2. Key features
3. Proposed Architecture
4. Scope
5. Deliverables
6. Folder Structure
7. API endpoints
8. Environment Variables
9. Setup and Installation
10. Deployment
11. Success Criteria

# 1. Project Overview

**Objective:**

The **Meet and Media Sync** project is a web-based application designed to manage and synchronize google meetings, manage attendees and media files(screenshots). It includes a frontend built with React, a backend powered by Node.js and Express, and integration with Google Drive and GCP for media storage and transfer. It also has a separate tool for capturing screenshots and storing them in your personal drive.

**Frontend Overview:**

The frontend is built using React and TypeScript. It uses Vite for fast builds and Tailwind CSS for styling. The application is modular, with reusable components and context-based state management.

**Key Libraries:**

Axios: For API requests.

@tanstack/react-table: For table rendering and management.

Lucide-react: For icons.

ShadCN: For tailwind components

React-DatePicker: For date selection.

JsPdf: For exporting PDF

Zod: For hookforms validations

Tailwind: Styling CSS library

**Backend Overview**

The backend is built using Node.js and Express. It uses MongoDB as the database and integrates with Google Drive and GCP for media storage and transfer.

**Key Libraries**

Mongoose: For MongoDB object modeling.

Nodemailer: For sending emails.

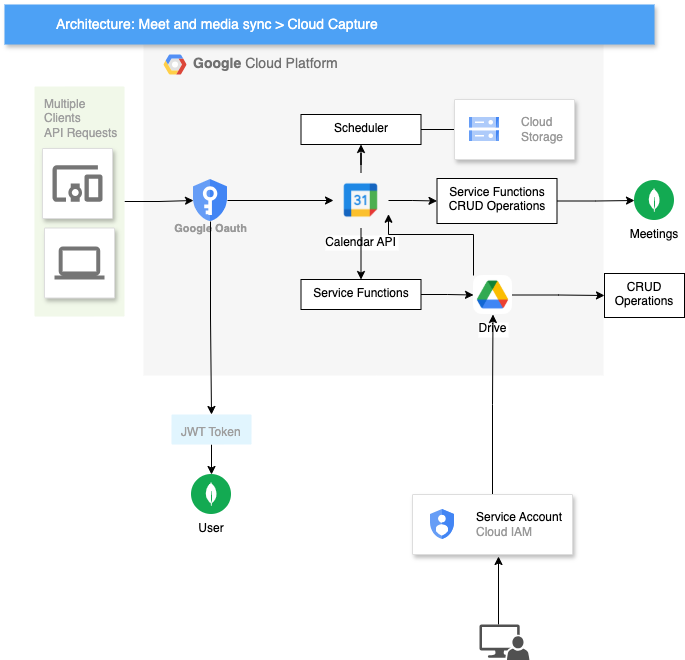
Swagger-jsdoc: For API documentation.

Express: For routing and middleware.

# 2. Key Features

* Google SSO Authentication
* Meeting scheduling with Google Calendar API
* Screenshot capture via browser extension
* Media organization in user’s Google Drive
* Periodic sync from Drive to GCP bucket
* Role-based access (Scheduler / Employee)

# 3. Proposed Architecture



**Frontend**: React (TSX + TailwindCSS)

**Backend**: Node.js with Express (TypeScript)

**Database**: MongoDB

**Authentication:** OAuth2.0

**Cloud APIs:**

* Google Calendar
* Google Drive
* Google Cloud Storage

# 4. Scope

Build an application where:

* Admins can create meetings.
* Employees can view/join meetings.
* Browser extension captures screenshots during meetings.
* Media is transferred to a structured GCP bucket periodically.

# 5. Deliverables

| S.No | Deliverable Description | Status |
| --- | --- | --- |
| 1 | Working web portal (frontend/backend) | ✅ |
| 2 | Google SSO integration | ✅ |
| 3 | Meeting scheduling with Calendar API | ✅ |
| 4 | Screenshot extension linked to Drive | ✅ |
| 5 | Drive-to-GCP Bucket sync script | ✅ |
| 6 | Structured GCP bucket organization | ✅ |
| 7 | Extension installation prompt | ✅ |
| 8 | Deployment on Google Cloud Run | ✅ |

# 6. Folder Structure

Frontend Folder structure for the MeetAndMediaSync:

Dockerfile

components.json

index.html

package.json

postcss.config.js

tailwind.config.js

tsconfig.app.json

tsconfig.json

tsconfig.node.json

public/

config.json

image.png

vite.svg

src/

App.css

App.tsx

dummy.ts

index.css

main.tsx

context/

authContext.tsx

driveContext.tsx

meetingContext.tsx

reportContext.tsx

data/

permissions.json

hooks/

use-mobile.tsx

lib/

utils.ts

providers/

Providers.tsx

constants/

api.constants.ts

index.ts

message.constants.ts

permissions.constants.ts

routes.constants.ts

routes/

PrivateRoute.tsx

Router.tsx

utils/

axiosConfig.tsx

driveApi.tsx

notificationHandler.tsx

page/

Dashboard.tsx

Home.tsx

error/

NotFound.tsx

Unauthorized.tsx

components/

common/

AuthNavbar.tsx

Loader.tsx

Logo.tsx

Navbar.tsx

PageWrapper.tsx

auth/

Divider.tsx

Login.tsx

Signup.tsx

signupForm.tsx

ui/files of shadcn

layout/

AppSidebar.tsx

handleScrollToSection.tsx

Herosection/

Footer.tsx

Hero.tsx

Tabs.tsx

Dashboard/

Create/

Create.tsx

MeetingForm.tsx

Home/

DashboardHome.tsx

EventCard.tsx

EventTab.tsx

MeetingActions.tsx

Drive/

Drive.tsx

Meeting/

MeetingCardView.tsx

MeetingFilter.tsx

Meetings.tsx

Roles/

AddNewRole.tsx

AddUsers.tsx

Report/

ExportData.tsx

Report.tsx

ReportModal.tsx



Backend Folder structure for the MeetAndMediaSync:

 src/

server.ts

config/

database.ts

gcp-service-key.json

gcpConfig.ts

permission.json

seedConfig.ts

constants/

collections.constants.ts

cron-jobs.constants.ts

environments.constants.ts

index.ts

mails.constants.ts

permissions.constants.ts

scopes.constants.ts

seeds.constants.ts

service-messages.constants.ts

status-codes.constants.ts

tokens.constants.ts

types.constants.ts

controllers/

authController.ts

driveController.ts

meetingController.ts

reportController.ts

roleController.ts

transferController.ts

userController.ts

models/

Media.ts

Meeting.ts

MeetingDetails.ts

Notification.ts

Role.ts

StorageLog.ts

User.ts

middleware/

authMiddleware.ts

requestValidationMiddleware.ts

schemas/

drive.schemas.ts

meetings.schema.ts

roles.schemas.ts

transfer.schema.ts

user.schema.ts

routes/

authRoutes.ts

driveRoutes.ts

index.ts

meetingRoutes.ts

reportRoutes.ts

roleRoutes.ts

transferRoutes.ts

userRoutes.ts

services/

authService.ts

driveServices.ts

gcsService.ts

meetingService.ts

reportService.ts

roleService.ts

transferService.ts

userService.ts

utils/

cronJob.ts

emailTemplate.ts

googleCalendar.ts

logger.ts

meetingValidation.ts

Extension Folder structure:



background.js

content.js

icon.png

manifest.json

popup.html

popup.js

# 7. APIs & Services

**Authentication APIs**

* GET /auth/v1/google: Redirect to Google OAuth.
* GET /auth/v1/google/callback: Handle Google OAuth callback.
* POST /auth/v1/logout: Logout the user.
* GET /auth/v1/check-refresh: Check and refresh JWT token.

**User Management APIs**

* GET /users/v1/user
* Get authenticated user details.
* GET /users/v1/allUsers
* Fetch all users.
* POST /users/v1/addUser
* Add a new user.
* POST /users/v1/notify
* Notify admin about a new user.
* DELETE /users/v1/deleteUser/{id}
* Delete a user.
* PUT /users/v1/editUserRole
* Edit a user's role.
* Role Management APIs
* GET /roles/v1/allRoles
* Fetch all roles.
* POST /roles/v1/addRole
* Add a new role.
* PUT /roles/v1/editRole/{id}
* Edit an existing role.
* DELETE /roles/v1/deleteRole/{id}
* Delete a role.

**Meeting Management APIs**

* GET /meetings/v1/all: Get all meetings.
* POST /meetings/v1/schedule: Schedule a new meeting.
* PUT /meetings/v1/update/{eventId}: Update an existing meeting.
* DELETE /meetings/v1/delete/{eventId}: Delete a meeting.

**Google Drive APIs**

* GET /drive/v1/folders: Get all folders in CloudCapture.
* GET /drive/v1/folders/{folderId}: Get files in a specific folder.
* DELETE /drive/v1/files/{fileId}:Delete a file.
* POST /drive/v1/mediaLogs: Create a media log.

**Transfer APIs**

POST /transfer/v1/gcp: Trigger transfer of screenshots to GCP.

**Report Generation APIs**

GET /reports/v1/report: Generate a report.

**Health Check API**

GET /healthcheck: Check if the server is running.

# 8. Success Criteria

| Module | Success Criteria |
| --- | --- |
| Google SSO | Authenticated login, user info stored in DB |
| Meeting Scheduling | Google Meet link created, attendees invited |
| Screenshot Extension | Captures screenshots with timestamp, auto-saves to Drive |
| Media Sync Script | Transfers folders from Drive to GCP bucket every 2 hours |
| Folder Structure | GCP Bucket → /YYYY-MM-DD/organizerName/meetingId/ |

# 