Software Requirements Specification

Project Information

Project: ACAP-BICOL (ACAP 1.0) – Rice Crop Manager Advisory Service (RCMAS) Collaboration

Release Number: 1.0

Date Updated: September 12, 2023

Attached worksheets:

ACAP-RCMAS Collaboration

- ACAP-RCMAS – Use Case Suite v1.0
  (acap_rcmas_use_case_suite.v1.0.pdf)
- ACAP-RCMAS – Features
  (acap_rcmas_feature_set.v1.0.pdf)

ACAP 1.0

- ACAP 1.0 – Software Requirements Specification v4.0
  (acap_1.0_software_requirements_specification_v4.0.pdf)
- ACAP 1.0 – Use Case Suite v4.0 (Lite version)
  (acap_1.0_use_case_suite_v4.0_lite.pdf)

Introduction

The collaboration project between ACAP-Bicol (ACAP 1.0) and IRRI aims to bridge the PAGASA weather forecast data, being utilized by ACAP for generating crop recommendations and bulletin PDFs with IRRI's RCMAS application for sending timely SMS crop recommendation advisories to select farmers in the Bicol region.

ACAP created several new well-documented REST API endpoints to allow the sharing of its internal PAGASA weather forecast data in a structured and organized format, with little to fewer modifications to what ACAP uses with IRRI and other trusted clients securely through the use of hard-coded tokens. Furthermore, ACAP started storing and archiving past weather forecast data for historical purposes, also made available on the REST APIs to use as a reference for backtracking and lookup.

Since this is a prototype project that uses existing ACAP Bicol components, the team had ACAP re-created on a separate infrastructure that mirrors the official ACAP Bicol system along with its standard-pricing plan cloud infrastructure for development purposes, where the new updates and features are made and tested on top of the existing system.

This document discusses the collaboration project's high-level components, use cases, and features with its ACAP Bicol (ACAP 1.0) predecessor.

Use Cases

While the ACAP-RCMAS collaboration project focuses on new REST API development to enable the sharing of its internal PAGASA weather forecast data in organized, structured formatting to trusted clients, it also uses several ACAP Bicol (ACAP 1.0) components for managing and maintaining its internal PAGASA weather forecast data.

The project organizes its use cases into four categories with subcategories.

I. Use Cases by Functional Area

The prototype project divides use cases according to their functional specification, identified into (7) subcategories, omitting the other ACAP 1.0 components which are not actively used (but which are available for reference in the attached ACAP 1.0 Software Requirement Specification document *(acap_1.0_software_requirements_specification_v4.0.pdf):*

- *PAGASA Weather Forecast REST API endpoints*, which have use cases for serving the latest-synced "active" PAGASA weather forecast data for:
    - Seasonal Weather Forecast
    - 10-Day Weather Forecast
    - Special Weather Forecast
- *PAGASA Historical Weather Forecast REST API endpoints*, which have use cases for serving past (historical) PAGASA weather forecast data for:
    - Seasonal Weather Forecast
    - 10-Day Weather Forecast
    - Special Weather Forecast
- *PAGASA Historical Weather Forecast Data Management,* which has uses cases for storing (archiving) past PAGASA weather forecast data and for deleting outdated historical weather forecast data
    - Archiving and storage of Historical Seasonal Weather Forecast
    - Archiving and storage of Historical 10 Day Weather Outlook Forecast
    - Archiving and storage of Historical Special Weather Forecast

    - Automatic deletion of outdated Historical Seasonal Weather Forecast
    - Automatic deletion of outdated Historical 10-Day Weather Outlook Forecast
    - Automatic deletion of outdated Historical Special Weather Forecast
- *Login and User authentication component*, which has the use cases for user login authentication;
- *Admin (private) web page components* comprising privileged admin access to web pages for managing the ACAP website, such as the:
    - *Seasonal weather forecast component*, which has use cases for administering the Seasonal Weather data by uploading PAGASA's seasonal weather forecast Excel file and displaying the uploaded data on-site
    - *ACAP settings component*, which has use cases for manually updating the site-wide seasonal weather forecast, 10-day weather forecast, and severe weather advisory data through manual encoded data input concerning PAGASA data and semi-automatic site data updates via Excel files upload for other relevant weather data that are not yet possible to have automatic updates
- *Client Authorization component*, which has use cases for creating API keys and allowing API access to clients who have these keys in their HTTP requests,
- *Cron Jobs for Automatic Weather Forecast Data Syncing to ACAP's Database,* which has use cases for regularly and automatically syncing PASAGA's 10-day weather forecast Excel files, El Nino/La Nina page, and Severe Tropical Cyclone Bulletins page contents to ACAP's database on scheduled time intervals.
- *Representational State Transfer Application Programming Interface (REST API) Endpoints*, which have use cases for allowing signed-in admins to mutate ACAP data over HTTPS for bulletins, phonebook, reports, users, and *PAGASA weather forecast* (seasonal, 10-day, and special), sending SMS, and for generating bulletin PDFs

II. Use Cases by Stakeholder

a. Two (2) types of stakeholders can access the new REST APIs: the *system administrator*, and the *collaborator.* Use cases are divided according to the stakeholders' roles and key needs.

III. Use Cases by Priority

A. Use cases are partitioned into three priorities; essential, expected and desired.

IV. Use Cases by Business Object and Actors

A. There are eleven (3) business objects defined that interact with the actors; user records, weather forecast records, weather archive records, and administrator-accessible web pages.

Details

Actors are described as follows:

- All
    - Stakeholders that can access the the PAGASA weather forecast REST APIs
    - Key Needs:
        - Convenient access to the PAGASA REST APIs content over the network
        - Secure, trusted access to the REST APIs
- System Administrator
    - System administrators are responsible for initializing and configuring ACAP to work with their region. They will also be responsible for deploying the frontend and backend applications to the recommended cloud services, other Platform-as-a-Service (PaaS), or their internal infrastructure.
    - System administrators are also responsible for generating API keys for sharing with Collaborators, and configuring the PAGASA REST APIs to allow access to HTTP requests having these API keys.
    - System administrators are ideally highly-experienced software developers familiar with ACAP's overall architecture.
    - Key needs:
        - Account security
        - Convenient network access, especially during deployment and troubleshooting logs from the cloud services dashboard
- Collaborator
    - Collaborators are trusted 3rd party clients or partners given access to ACAP's PAGASA Weather Forecast REST APIs through the use of hard-coded random-generated API keys.
    - Key Needs:
        - Convenient access to the PAGASA REST APIs content over the network
        - Secure, trusted access to the REST APIs
        - Up to date data retrieved from the REST APIs

Functional Requirements

The software uses the base set of use cases and features as ACAP 1.0, described in more detail in the attached PDF file *(acap_1.0_software_requirements_specification_v4.0.pdf)* with additional new functional requirements for the ACAP-RCMAS collaboration.

- With the ACAP-RCMAS collaboration, System Administrators can create hard-coded API keys to share with Collaborators and allow access to designated API endpoints.
- When Collaborators make an HTTP request to the PAGASA weather forecast (seasonal, 10-day, or severe cyclone weather outlook) REST API endpoints, the system should return the corresponding latest, "active" (seasonal, 10-day, or severe cyclone weather outlook) API response.
- When Collaborators make an HTTP request to the Historical PAGASA weather forecast (seasonal, 10-day, or severe cyclone weather outlook) REST API endpoints, the system should return the corresponding "past" (seasonal, 10-day, or severe cyclone weather outlook) API response.
- Before the Cron job uploads today's fetched, parsed, and validated 10-day weather forecast data to the database, it should store yesterday's 10-day weather forecast data in the historical database collection.
- Before the Cron job uploads the latest web-scraped severe cyclone data to the database, it should store the last active cyclone weather forecast data in the historical database collection.
- Before the system stores the latest Administrator-uploaded seasonal weather forecast data to the database, it should keep the last active seasonal weather forecast data in the historical database collection.
- The system should delete outdated archived historical weather forecast data older than (3) three months for the 10-day and severe cyclone weather forecasts and (6) months for the seasonal weather forecast.

Non-Functional Requirements

Usability requirements

- Collaborators should see the latest ACAP-synced PAGASA 10-day, seasonal, or severe cyclone weather forecast data from the PAGASA weather forecast REST APIs. Since we are currently unsure of when PAGASA exactly updates its weather forecast data from its side (as mentioned in the project planning risks), ACAP should include the date and time when it last synced the weather forecast data with PAGASA in the API responses (either through Cron jobs or manual update by Administrators) for reference based on observations.

- Collaborators should have a means to assess and internalize ACAP's internal process flow and guidelines for synching PAGASA's weather forecast data, such as the consolidated date and time of synching or other internal arrangements, as temporary solutions to the mentioned project planning risks through software documentation.

- Collaborators should have access to well-written API documentation on querying the PAGASA weather forecast API endpoints and information on what data to expect in the API responses. The API documentation should be hosted and accessible online for easy and flexible access.

- Collaborators should have fast API response times of 5 – 12 seconds from the new PAGASA weather forecast REST API endpoints.

- The front-end Administrator or Super Administrator accessible website should follow a responsive web design layout and patterns. Users should be able to view the website in a user-friendly manner on most major desktop and mobile web browsers and best viewed on the Google Chrome web browser.

- The front-end website should load fast. It should not lag, freeze, or become unresponsive while loading or when users view or use the various web page features.

Reliability and up-time requirements

*The system should be able to host a moderate amount of Collaborator users (~50+) and medium size of weather and user/site data not exceeding Firestore's (Spark plan) 1GB at a stable phase since we are still operating using standard cloud service accounts for both ACAP 1.0 and ACAP-RCMAS. *(We can increase the max data storage threshold should the DA RFO 5 decide the need to subscribe to paid pricing plans.)*

The server and REST APIs should have a 99.9% up-time with at most 45 minutes of downtime. We expect the website, server, and ACAP services to be accessible, especially if severe cyclones are in the Philippines.

Safety requirements

ACAP utilizes stable technologies to build the system. The application, currently deployed on the standard plan (free-tier) cloud services, expects minimum human safety hazards during the maintenance and operation of the system since cloud-based applications allow flexible access to its components over the cloud.

Security requirements

- Sensitive data like API keys, phone numbers, and user-generated reports should be securely stored and encrypted.
- The new PAGASA REST API endpoints should allow Cross-Origin Resource Sharing (CORS) from any domain or origin. It should, however, only allow access to HTTP requests from clients with valid API keys.
- The other REST API endpoints should reject HTTP requests that do not have or have invalid Bearer Authorization tokens from the front end. Signing into the ACAP admin pages automatically generates fresh, valid Bearer Authorization tokens using the Firebase Authentication service.
- HTTP requests accessing the NodeJS REST APIs should have minimal to thorough validation.
- Only System Administrators can create new API keys and allow access for them in the PAGASA REST API endpoints.
- Only Super Administrators can create authentic new Administrator users and edit or delete existing Administrator accounts.
- Only Administrators can manually configure the site's weather forecast data if they need a manual update.
- The website prompts users to input their super admin-provided username and password to control Admin site access.

Performance and scalability requirements

The system should be able to perform well, given Firebase's generous Spark plan (free) pricing tier, similar to ACAP 1.0. The website should be accessible on Firebase Hosting, and the Individual (free tier) plan Render account for the NodeJS backend server should respond promptly to HTTP requests within the standard Render plan's limits. Firebase Authentication can support 100+ user accounts, while a pay-as-you-go pricing subscription for the Firestore database and Cloud Firebase Storage may be required if site data exceeds Firebase's Spark plan limits.

Additionally, the collaboration project also hosted the REST APIs, including the new PAGASA Weather Forecast REST API endpoints in a Vercel (Hobby plan) serverless account to allow faster API responses since the Vercel (also using a standard, Hobby plan account) does not "sleep" after 15 minutes of inactivity, unlike the Render (Individual plan).

ACAP's infrastructure, and therefore the collaboration project's comprised of several cloud services (Firebase, Render, GitHub, and Vercel) currently using standard (free-tier) pricing plans, can scale pretty well by upgrading to paid plans.

The system can forego REST API hosting in Vercel should the Render (Individual plan) be upgraded to a paid subscription since the Render app will not "sleep" after 15 minutes of inactivity.

Maintainability and upgradability requirements

Maintainability is our ability to make changes to the product over time. We will address this by anticipating several types of change and by carefully documenting our design and implementation.

Like ACAP 1.0, the system has plans for future releases, which will happen when new system features are requested. It will require updating the initial design and architecture of the current stable running system and implementing the code changes.

Should a future release be requested, we must be able to deploy and upgrade with minimal downtime and service disruption.

This project has plans to integrate with ACAP 1.0 should the DA RFO 5 decide to use its new PAGASA weather forecast APIs and Historical Weather Forecast archiving features in the future. If this happens, we should update the ACAP 1.0 software documentation with the information from this documentation.

Business life-cycle requirements

The business life-cycle of a product includes everything that happens to that product over several years, from the initial release, through important but infrequent use cases, until product retirement. The product's main life-cycle requirements are listed below.

Details:

- The product should be stable. It should use systems and software architectural design patterns for web applications that will live until the technologies and tools used to build it are considered obsolete. It will be scheduled for updates until an upgrade is critically needed.

System hardware requirements

ACAP 1.0 and the collaboration project rely on the standard plan (free-tier) hardware specifications of the following cloud services discussed in more detail in the *Cloud Infrastructure References Pricing Estimates* table of the attached ACAP 1.0 Software Requirements Specification document *(acap_1.0_software_requirements_specification_v4.0.pdf)* since the DA RFO 5 has not yet expressed their need to upgrade to paid cloud services subscriptions up to this point. These services are upgradable to paid subscription plans when deemed necessary.

As a recap, these cloud services include:

- **GitHub** – for storing project source codes and revision tracking
- **GitHub Pages** – for website UI hosting
- **GitHub Actions** – for deploying to development and production environments and for running Cron jobs
- **Firebase**
    - Firebase Authentication – User authentication, registration, login
    - Firebase Cloud Storage – for storing and hosting PDF files
    - Firestore Database – NoSQL database for storing user and site data
- **Render** – Web Services for hosting a NodeJS backend with custom-created API endpoints for managing miscellaneous background processes.

Additionally, the collaboration project used a new cloud service to enable fast API response times for Collaborators, given that we are using standard plan (free) cloud accounts for development (and production) with limitations.

- **Vercel** – A cloud platform for hosting websites, web services, and serverless functions. It also allows hosting traditional APIs running in full server implementation (standalone Express apps) within the allowable limits of its serverless environment.
    - We like to note that Vercel is only a temporary replacement to the standard (Individual) plan Render hosting, which offers a slow API response time of 1 – 2 minutes after waking up from a cold start if no one uses it after 15 minutes of inactivity.
    - Since Vercel's serverless function size exceeds the 50 MB (zipped) limit when installing the dependencies for generating bulleting PDFs, we can forego using Vercel, should the Render hosting account be upgraded to a paid subscription, to eliminate its 1 – 2 minute cold start.

New Cloud Infrastructure References Pricing Estimates

The Cloud Infrastructure Reference Pricing Estimates from ACAP 1.0 also apply to the collaboration project, which is covered in more detail in the attached document (acap_1.0_software_requirements_specification_v4.0.pdf) for reference.

This table only includes **Vercel**, the new addition to the collaboration project.

	Standard Plan	Upgraded Plan	Recommendations
Vercel	**HOBBY** (standard) Price: $0 • **Bandwidth**: 100 GB / mo • **Serverless Function Execution**: 100 GB-Hrs • **Serverless Function Execution Timeout**: 10 s • **Serverless Function Size**: 50 MB (zipped), 250 MB (max unzipped) • **Serverless Function Memory**: 1024 MB • **Serverless Function Concurrency**: 1000 • **Serverless Function Payload Size Limit**: 4.5 MB • **Disk Size:** 13 GB • **Cron Jobs:** (2) two triggered per day • **Team:** n/a • **Build Execution:** 100 hrs • **Deployments/day:** 100 • **Deployments (CLI)/week:** 2000 • **Image optimization:** 1000 images • **Runtime logs:** stored for 1 hr	**PRO** Price: $20 per user / month • **Bandwidth:** 1 TB / mo, or $40 per 100 GB increment • **Serverless Function Execution:** 1000 GB-Hrs, or $40 pero 100 GB-Hrs increment • **Serverless Function Execution Timeout:** 60 s • **Serverless Function Size**: 50 MB (zipped), 250 MB (max unzipped) • **Serverless Function Memory**: 3008 MB • **Serverless Function Concurrency**: 1000 • **Serverless Function Payload Size Limit**: 4.5 MB • **Disk Size:** 13 GB • **Cron Jobs:** 40 • **Team:** 10 members per team • **Build Execution:** 400 hrs • **Deployments/day:** 6000 • **Deployments (CLI)/week:** 2000 • **Image optimization:** 5000 images, or $5 per 1000 increment • **Runtime logs:** stored for 1 day More information on https://vercel.com/docs/limits/overview	For development purposes of REST APIs for sharing with Collaborators, we recommend using the Vercel HOBBY plan over the Render (Individual) plan since Vercel has faster cold starts and API response times. Vercel also offers a straightforward approach for CI/CD deployment, which makes it one of the great options for automating deployments. However, we cannot fully commit to switching to Vercel over Render because Vercel's serverless function size exceeds the 50 MB (zipped) limit when installing ACAP's bulletin PDF dependencies, even in its upgraded PRO plan.

System software requirements

The web server must have the following:

- A web server that can host static HTML, CSS, and JavaScript for hosting the website's UI.

The backend server cloud hosting, usually a Platform-as-a-Service (PaaS), must be capable of running the following:

- A NodeJS web server for running background processes and hosting ACAP's APIs.
- NPM scripts as Cron jobs in case GitHub Actions for running the Cron jobs used for automatic weather forecast data syncing to the database are unavailable

The database, storage, and user authentication uses a Firebase spark plan project.

*Please view ACAP 1.0's **Cloud Infrastructure References Pricing Estimates** table (in the attached acap_1.0_software_requirements_specification_v4.0.pdf PDF file) for more details

Application program interfaces (APIs)

Carrying on from ACAP 1.0, the collaboration project's frontend website also uses the Firebase Web SDK and standard JavaScript APIs for NextJS (React) for building its UI. These are open-source and can easily be acquired. NodeJS, also a JavaScript runtime for servers, is used to create and run the backend REST APIs while using the Firebase Admin SDK for NodeJS.

The custom REST APIs are accessible over a secure SSL connection and require a signed-in admin's authentication.

The Firebase APIs are accessible as the client (website or NodeJS backend) uses the correct Firebase-generated API keys and service account JSONs.

Data import and export requirements

The system will store all data in new collections and documents in the Firestore NoSQL database, where other trusted client programs over the web can access it with secure authentication or authorization. Custom NPM scripts run through the command line will facilitate raw data import and export from the Firestore database.