PROJECT PLAN

Project Information

Project: Agro-Climatic Advisory Portal - Bicol (ACAP-BICOL) / ACAP 1.0

Release Number: 4.0

Time Frame: March – December 2022

Date Updated: August 24, 2023

Summary of Project

The Agro-Climatic Advisory Portal Bicol (ACAP-BICOL) is an online web application for disseminating weather and climate information of the Adaptation and Mitigation Initiative in Agriculture (AMIA) Villages in the Bicol Region. It follows a hybrid type of three-tier web application architecture consisting of a website for the client layer, the Firestore database for the database layer, REST APIs running on a NodeJS server, and several client-side logic for the business layer.

The portal aims to guide extension workers in creating and disseminating relevant and tailored advisories and recommendations for farmers and fisherfolk that will help them address the impacts of climate change thru the use of its online-accessible tools for linking smart automatic bulletins PDF creation linking with PAGASA weather forecast data, the centralized process of making information available for general viewing or download and by relaying PAGASA weather-related data for laymanized public viewing in a readily-accessible and timely manner.

This document discusses ACAP's software development plans, technology stacks, and general software development approaches.

The software consists of the following core components:

- **Login and User Authentication Component**
- **User Accounts Management Component**
- **PDF Bulletins Generator Component**
- **Site Search**
- **Public Website Pages**
    - Home Page

    - ACAP Services
        - 10-Day Weather Forecast
        - Seasonal Weather Forecast
        - Special Weather Forecast
    - Cropping Calendar
    - Public Crop Recommendations
        - Regional Seasonal Climate Outlook and Advisory Recommendations Generator
        - 10-Day Farm Weather Outlook and Advisory Recommendations Generator
    - Public Bulletins (PDF Downloads)
        - 10-Day Farm Weather Outlook Bulletins
        - Seasonal Outlook Bulletins
        - Special Weather Forecast Bulletins
- **Admin (Private) Web Pages**
    - Crop Recommendations Management
        - Regional Seasonal Climate Outlook and Advisory Bulletins and Reports Creator
        - 10-Day Farm Outlook and Weather Advisory Bulletins and Reports Creator
        - Special Weather Forecast Bulletins and Reports Creator
    - SMS Management
        - Phonebook
        - Send SMS feature
    - ACAP Settings
        - Seasonal Weather Forecast Data Management
            - PAGASA seasonal weather forecast Excel file uploader
            - Manual weather systems that may affect the region input
        - 10-Day Weather Forecast Data Management
            - PAGASA 10-Day weather forecast Excel files uploader
            - Manual moon phases input
        - Special Typhoon Advisory Data Management
            - Manual data syncing feature of PAGASA's Severe Tropical Cyclone web page contents to ACAP's Special Weather Forecast section
            - Manual wind speed list and affected municipalities editor
    - User Profile Management
        - Account Password Updater
- **Representational State Transfer Application Programming Interface (REST API) Endpoints**
    - A set of secure and authenticated REST API endpoints 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
- **Cron Jobs for Automatic Weather Forecast Data Syncing to ACAP's Database**
    - Daily 10-Day Weather Forecast Excel Files from PAGASA data syncing

    - Weekly El Nino/La Nina PAGASA web page contents syncing
    - Daily (every 2 hrs interval or so) syncing of PAGASA's Severe Tropical Cyclone Bulletin web page
- **Data Processing and Upload Scripts**
    - NPM scripts used for pre-processing, formatting and uploading raw cropping calendar data (CSV file) and crop recommendations (Excel file) to the Firestore database
    - NPM scripts for seeding the Firestore database with default data

Hosting Summary

The components of the portal's website pages are constructed as static web pages using React/NextJS' Static-Site Generation (SSG) to enhance Search Engine Optimization (SEO) and simplify the deployment of the website's front-end on GitHub Pages.

The publicly viewable web pages are accessible on GitHub Pages through the https://amia-cis.github.io domain. Meanwhile, the NodeJS backend running the REST APIs is available as a Render Web Service (Blueprint) app running on an Individual (free tier) plan found at https://amia-cis.onrender.com.

ACAP utilizes various cloud services with generous amounts of standard (free) tier pricing plan limits to manage dynamic background processes for its backend components. It employs Firebase components under a standard Spark plan for database functionality, user log-in and authentication, and cloud storage for file storage. An Individual (standard) pricing plan Render Web Service (Blueprint) app hosts an external backend for REST APIs that runs miscellaneous background data processing, including tasks such as climate/weather data processing, PDF creation, SMS sending, and manual updates to several weather forecast parameters.

ACAP can cater to a moderate audience size, approximately 10 to 100 administrators, with an average daily traffic of around 200 or more public viewers. These capacities are well-maintained within its standard (free) plan cloud infrastructure limitations. However, ACAP faces restrictions in supporting the site's backend operations for large numbers of concurrent admin user processes, and the Firestore database has limitations on the amount of data it can store beyond the free-tier limits. If ACAP's database data were to exceed these limits, an option to upgrade to a pay-as-you-go Firebase pricing model is available.

Hosting on the DA AMIA Server

We've halted the plans to fully migrate the front-end and cloud-based backend components locally to the DA AMIA bare OS server. This decision aims to concentrate on and give priority to the development

of the core features and components. Furthermore, estimates after some time proved that a comprehensive local migration to DA AMIA's bare-OS server would extend beyond the project timeline because replacing the existing cloud components with local alternatives proved to have a greater scope, requiring a total re-write of the finished project components which were already about 40% finished at the time. All parties have agreed to adopt the proposed cloud services approach for ACAP 1.0 after contemplating the benefits, costs, and potential risks of using cloud services.

Summary of Methodology

General Development Approach

The team follows the three generic phases of software engineering on the four major aspects of the software project. Analysis, design and implantation of:

- User Interface (UI) - data visualizations, UI design and layout for various web pages, optimized website loading and and responsive design
- Functionality - data (weather forecast and crop recommendations) management, sorting, querying and display on the UI, site-wide PDF bulletins viewing and download, user registration and login
- Data acquisition, preparation and cleaning
- PDF bulletin creation and automatic upload to storage
- Database – Firestore collections schema and design
- REST API development
- Cron Jobs for PAGASA Weather Forecast data syncing to ACAP’s database
- Deployment to live, production environments
- Testing

Project team organization

The project team consists of supervised sub-teams:

Team	Assigned
Project conceptualization, ideas and technical workflow process	Jojo, Gaye, Leo
Coordination, meeting and presentation with stakeholders	Jojo, Gaye, Angel, Leo
Data preparation and acquisition	Jojo, Gaye
Technical software project planning, development and deployment	Angel
Phonebook/contacts management	Dom
SMS sending and viewing/logging features	Dom
Testing (using and preview of core features)	(All)

Development and collaboration tools

ACAP uses the following tools and technology choices considering several factors:

- **HTML5, CSS and JavaScript (React/NextJS)** – for the website UI
  - React offers an intuitive, modular way of creating declarative reusable components for flexibly building web pages and user interfaces (UI)
  - Redux (redux-toolkit), a state management library for React, can intuitively manage state and async data across multiple disconnected components.
  - NextJS is one of the latest, stable, and most popular React frameworks officially endorsed on React's "react.dev" website
  - The Static Side Generation (SSG) feature of NextJS, combined with Incremental Static Generation (ISG), creates SEO-optimized websites that work well for uploading on static hosting services such as GitHub Pages and Firebase Hosting.

- **NodeJS** – for the backend server with custom-created REST API endpoints to manage authenticated background processes (such as creating PDF files, managing Admin accounts and sending SMS)
  - ACAP uses a server to ensure using authenticated transactions when communicating data to and from the database to serve reliable, authentic data to the client website.
  - We chose a server to leverage running most of the heavier background processes, such as the secure generation of highly-customized bulletin PDFs, which can prove taxing when done in the front end (especially for low-spec devices)
  - NodeJS backend are easier to scale, and its light-weight modular nature fits the app's simple to average requirements
  - NodeJS uses JavaScript, which is also used to build the website client (React/NextJS). ACAP's helper and automation scripts (NPM scripts) are also built with JavaScript and run with NodeJS

- **GitHub** – for project progress, code revisions tracking
  - The project git repository, accessible in a private GitHub repository https://github.com/ciatph/climate-services-webportal is available for access or collaboration upon request.

- **GitHub Actions**
  - We use GitHub Actions, also available in the GitHub repository, for deploying the ACAP client and server apps to development and production environments through customized Continuous Integration and Deployment (CI/CD) targets for respective cloud services.

    - We also use the GitHub Actions scheduled workflow feature to run the Cron job NPM scripts for regularly syncing PAGASA's 10-day, El Nino / La Nina, and Severe Tropical Cyclone Bulletin data to ACAP's Firestore database.

- **Github Pages** – for hosting the website's UI (web pages)
    - The public-viewable web pages are currently hosted on Github Pages under the https://amia-cis.github.io domain.
    - We chose GitHub Pages for hosting the front-end website as requested by our then Technical Lead Consultant from Alliance. Meanwhile, the NodeJS backend server, is hosted on a remote Render app at https://amia-cis.onrender.com because GitHub Pages only allow hosting static websites (HTML, CSS, JavaScript). It does not allow hosting server scripts like NodeJS, PHP, or Python.
    - The GitHub Pages ".github.io" domain name is quite popular, and easy to remember by. There are also options of replacing this with a custom (paid) domain name.
    - GitHub Pages have flexible website hosting plans, starting with a standard (free) pricing plan, which is a great option for displaying publicly-accessible prototype websites online with no cost

- **Firebase** – A set of cloud-based Backend-as-a-service (BaaS) development tools by Google that helps developers build, deploy and scale custom-built mobile or web applications. It offers several pre-made functioning services, allowing developers to focus on building the business logic of their apps. Firebase has a standard (free) pricing plan called the Spark plan, allowing developers to build, ship, and prototype web applications at no cost, with an option to upgrade to a paid subscription (Blaze pricing plan) should the prototype exceed the limits of the standard plan. ACAP uses (3) three of these services:
    - Firebase Authentication (Spark plan) – for user authentication and for managing user registration and login
    - Firestore Database (Spark plan) – Firebase NoSQL database for storing user and site data
    - Firebase Cloud Storage (Spark plan) – Firebase storage for storing and hosting PDF files

- **Render** (Individual plan) – a cloud Platform-as-a-Service (PaaS) for hosting the NodeJS backend
    - The backend is hosted on an Individual plan (free tier) Render Web Service app on https://amia-cis.onrender.com
    - We initially used Heroku, then switched to Render for hosting the backend after Heroku stopped its standard (free) pricing plan last October 2022. Render and Heroku (when Heroku's standard pricing plan was active) offer a flexible and customizable deployment and hosting environment for NodeJS apps for free, which is a good option for testing prototype apps at no cost. Both offer a paid pricing plan should the NodeJS app exceed the limits of the standard pricing plan.

- We chose Render (and Heroku) for hosting the NodeJS backend because both have methods for installing custom OS-level package software dependencies required by ACAP's PDF bulletin creator component.
- While we chose Render as a quick swap for Heroku, we recommend using Heroku (or Firebase Cloud Functions, described below) if the budget permits. Heroku, Firebase Cloud Functions, or other more popular cloud services (such as Amazon Web Services AWS, Digital Ocean, and others) look more mature and stable than Render.
- (We can optionally use Firebase Cloud Functions, a NodeJS server in the same Firebase project package as Firestore, Auth and Cloud Storage upon upgrading the Firebase project to Blaze plan (needs billing information))
- We like to note that NodeJS apps running in a Render standard (Individual plan) ***"sleep" after 15 minutes of inactivity*.** This causes a delay of 1 – 2 minutes "waking up" (starting the server) if anyone uses it after it sleeps, for example, when generating a bulletin PDF preview or accessing the REST API.
    - ACAP only uses mutative *"write"* REST API requests (such as creating bulletins, manual weather data updates, and sending SMS) from Render to take advantage of more secure, authenticated "write" transactions since Render apps sleep after 15 minutes of inactivity
    - ACAP's website directly fetches data from its Firestore database for *"displaying"* only in the client website using the Firebase (web) Firestore APIs to eliminate the lag time waiting for the Render app to boot up.

The mentioned (limited) standard-plan cloud infrastructure services currently host the production website user interface and backend. A Windows or Linux PC/laptop can act as a local development environment for the web app by downloading the project repository and following its set-up instructions.

Control of changes

New functionalities, updates, modifications, and enhancements written in the codes will be properly documented and committed to the ciatph GitHub repository at https://github.com/ciatph/climate-services-webportal. All documents will have assigned revision numbers.

Project Plan manner of update

This project plan will be updated as needed throughout the project and sent to team members thru email or shared from Google Docs.

Software Development Work Breakdown Structure and Estimates

The following work breakdown estimates are tentative and subject to change according to *development risks*, described in more detail in the Risks section. We use hourly estimates to gauge if the tasks and objectives are doable within the confines of the project timeline while *considering the number of developers working in parallel*.

*It is worth noting that a web developer does each task one after another and not in parallel.

Step	Description	Estimate
1.	Preparation	
1.1.	Developer Training/Research	
2.	Inception	
2.1.	Requirements Gathering	
2.2.	Requirements Specification	
2.3.	Requirements Validation	
3.	Elaboration	
3.1.	High-level Design	
3.1.A.	User Interface Design (Draft)	
3.1.B.	Identification of Main Objects and Relationships	
3.2	Low-level Design	
3.3	Object Design	
3.3.A.	Database Design	
3.2.	Design Review and Evaluation	
4.	Construction	
4.1.	System Implementation	
4.1.A.	Database Implementation	
4.1.B.	Functionality Implementation	

4.1.B.1.	PDF Management (Creation and Upload)	
4.1.C.	User Interface with Functionality Implementation	
4.1.C.1.	**Public Home Page**	
4.1.C.2.	**Public ACAP Services Page**	
4.1.C.2.A.	Regional Seasonal Climate Outlook and Advisory	
4.1.C.2.B.	10-Day Farm Weather Outlook Summary	
4.1.C.2.C.	Special Weather Forecast	
4.1.C.3.	**Public Cropping Calendar**	
4.1.C.4.	**Public Crop Recommendations Page**	
4.1.C.4.A.	Regional Seasonal Climate Outlook and Advisory Generator/Preview	
4.1.C.4.B.	10-Day Farm Outlook and Weather Advisory Generator/Preview	
4.1.C.5.	**Public Bulletins PDF (Downloads) Page**	
4.1.C.5.A.	10-Day Farm Weather Outlook Bulletins	
4.1.C.5.B.	Seasonal Outlook Bulletins	
4.1.C.5.C.	Special Weather Forecast Bulletins	
4.1.C.6.	**Accounts Management**	
4.1.C.7.	**User Login**	
4.1.C.8.	**Admin Crop Recommendations Report Management**	
4.1.C.8.A.	Seasonal Crop Recommendations/PDF Generator, Viewer and List	
4.1.C.8.B.	10-Day Farm Outlook and Weather Advisory Recommendations/PDF Generator, Viewer and List	
4.1.C.9.	**Admin SMS Management (% Dom)**	
4.1.C.9.A.	Phonebook	
4.1.C.9.B.	Send SMS	

4.1.C.9.C.	**Admin ACAP Settings (Weather Data Management)**	
4.1.C.9.C.1.	Seasonal Weather Rainfall Forecast Data Management	
4.1.C.9.C.1.A.	PAGASA seasonal weather forecast excel file uploader	
4.1.C.9.C.1.B.	Manual tropical cyclone input	
4.1.C.9.C.1.C.	Manual weather systems that may affect the region input	
4.1.C.9.C.2.	10-Day Weather Forecast Data Management	
4.1.C.9.C.2.A.	PAGASA 10-Day weather forecast excel file uploader	
4.1.C.9.C.2.B.	Manual moon phases input	
4.1.C.9.C.3.	Special Typhoon Advisory Data Management	
4.1.C.9.C.3.A.	Manual data syncing feature of PAGASA's Severe Tropical Cyclone web page contents to ACAP's Special Weather Forecast section	
4.1.C.9.C.3.B.	Manual wind speed list and affected municipalities editor	
4.1.C.9.	Integrate Components	
4.1.D.	**Cron Jobs for Syncing PAGASA Weather Forecast**	
4.1.D.1.	Daily 10-Day Weather Forecast Excel Files from PAGASA data syncing	
4.1.D.2.	Weekly El Nino/La Nina PAGASA web page contents syncing	
4.1.D.3.	Daily (every 2 hrs interval or so) syncing of PAGASA's Severe Tropical Cyclone Bulletin web page	
4.1.E.	**Representational State Transfer Application Programming Interface (REST API) Endpoints**	
4.1.F.	**Data Processing and Upload Scripts**	
4.1.F.1.	NPM scripts used for pre-processing, formatting and uploading raw cropping calendar data (CSV file) and crop recommendations (Excel file) to the Firestore database	
4.1.F.2.	NPM scripts for seeding the Firestore database with default data	
4.1.G.	**Site Search**	

4.1.H.	**Technical Documentation**	
4.1.H.1.	Software Documentation	
4.1.H.2.	End-User Guides and Manuals	
4.2.	**Testing**	
4.2.A.1.	Testing Planning	
4.2.A.2.	Test Execution	
4.3.	Implementation Review and Evaluation	
5.0	Transition	
5.1.	Release Packaging	
6.0	Reflection	
6.1.	Postmortem Report	
	Total	

*Please note that the total time estimate is tentative, and may be subject to change as we further along during the development process

Deliverables in this Release

Deliverable Name	Description	Delivery Date
Project Plan	Updates on project evaluation and management	September 6, 2023
Final SRS for ACAP 1.0	Updates on Software Requirements Specification ACAP System Requirements	September 6, 2023
Final Release	Full system functionality ready for official usage	December 30 - 31, 2022

*Please note that the time estimates are tentative, and may be subject to change as we further along during the development because of the items mentioned in the **Risks Management** section and other unforeseen circumstances.

Risk Management

The main risks of this project are:

1. It may take time to get acquainted with the raw data format, structure, manual process, and natural user flow of generating reports. An understanding of the process, variables, and computations involved are necessary to have a firm grasp of automating the steps and processes for bulletin generation.

2. New data and functionality may need to be integrated into the middle of the project. If this happens, we may need to adjust the estimated schedules depending on their complexity, especially if the new requirements will need some time to integrate into the existing system, or if it requires new technology that needs preparation/study time.

3. We may need advice for visual PDF layout themes or designs from the creative and user-experience side, including knowing what data is best and expected to be displayed on each.

4. We may need to subscribe to Firebase's Blaze (pay-as-you-go) plan, if the accumulated data usage and storage over time exceeds Firestore's free-tier limits.

5. We may need to subscribe to Render's paid subscription plan (or check out Heroku, Firebase Functions, or similar premium, more stable cloud service alternatives) if ACAP admins will feel the need to generate PDF bulletin previews fast without being restricted by the Render Individual plan's apps "sleeping" after 15 minutes of inactivity.

6. We must request a file copy (or at least agree on a mock file) of new data as soon as possible if there are revision plans for the existing file formatting and structure to capture every detail and quirks from ***actual files*** at an early stage. Raw data cleaning, pre-processing, uploading and syncing with existing features take a lot of time, especially if the expected new data format, layout, and structure differ too much from the first batch of working and pre-processed data sets. Working with actual files at an earlier stage removes ambiguity and also allows a well-informed and better grasp of how to adjust the affected website/pdf layouts accordingly.

7. The cloud services used by ACAP may change their service usage policies, access methods, and other usage aspects which their respective companies deem relevant after a while, for example,

3, 5, or several years from now. ACAP may cease to function as expected should these changes happen. ACAP developers should stay tuned for such updates and update ACAP accordingly.

8. Similar to the above, the technologies, software, and tools that ACAP uses may be considered obsolete in the next few years, or more suitable, new technologies may appear. ACAP 1.0 will continue to work as expected until these technologies are considered obsolete.

9. Cloud services may experience partial outage every now and then, leading to ACAP service downtimes.

10. The PAGASA (seasonal outlook, El Nino/La Nina, and severe tropical cyclone) web pages may change in structure and content shortly. If this happens, ACAP's Cron job NPM scripts for web scraping and syncing the contents of these pages to ACAP's Firestore database will fail. The NPM scripts should be re-programmed to handle new formatting and content of these web pages.

11. The PAGASA 10-Day weather forecast Excel files may change in structure, format, and content shortly. Or, the PAGASA URL where they are accessed may change. If this happens, ACAP's Cron job NPM scripts for downloading, extracting, and parsing the Excel file contents to ACAP's Firestore database will fail. The NPM scripts should be re-programmed to handle the new formatting and contents of the PAGASA 10-Day weather forecast Excel files.

12. We could lose resources. E.g., team members could get sick or spend time on other projects. We should consider proper task management and scheduling to avoid lapses in the software project.

13. ACAP 1.0 currently runs on cloud services, i.e., Firebase, a Backend-as-Service (BaaS), and Render, a Platform-as-a-Service (PaaS), under standard (free-tier) pricing plans. Its paid subscription plans depend on the region, as illustrated in *"Figure 2.0: ACAP 1.0 Scaling factors"*. The project expects the currently assigned ACAP 1.0 web developer to manage the paid subscription options should the region concerned decide to upgrade its cloud stack.

    With regards to **Risk #12**, *proper software project turnover and transfer of cloud resources or subscriptions* must be observed should there be a change of web developers overseeing the ACAP 1.0 operations at any given time.

14. We may need *more similarly experienced web developers/programmers* to help with the growing list of new features and existing feature enhancement requests on the fly to keep up with the project's scheduled timeline.

Ideal Software Development Team Organization

An ideal software development team for an average-sized yet complex, feature-packed web application and system like ACAP works best with a tag team of (2) two or more experienced Web Developers – one for working on backend development and the other to concentrate on frontend development. The need for more than (2) two web developers also depends on the extent and scope of objectives. Some complex tasks, like the SMS sending feature, require dedicated focus from another web developer.

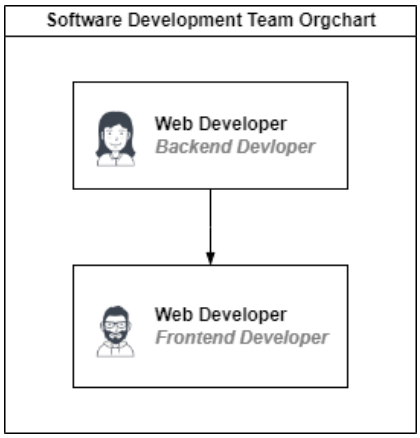

Figure 1.0 Software Development Team Orgchart

Ideally, both Web Developers should have full-stack web development knowledge, so they may work in sync and switch roles interchangeably if needed.

Web Developer – Backend (1)	
Requirements	NodeJS, Express, Firebase (for NodeJS), Git, Platform-as-a-Service (PaaS) cloud services like Render and Heroku, or similar
Skills	- Creating and debugging NPM scripts - Creating and maintaining Express web servers - Creating REST APIs that interface with the Firestore database and remote external APIs - Web scraping - Security implementation in the backend - Continuous Integration / Continuous Delivery (CI/CD) for automatic deployments; or alternate (manual) deployment strategies to target cloud services or other PaaS - Vanilla HTML, CSS and JavaScript
Expected Output	- Maintain and troubleshoot ACAP's REST APIs

	- Development of new REST API endpoints as required by the project, while coordinating its functionality and implementation to the Frontend developer - Maintain the bulletin PDF creator component, including updating its layout and text if necessary - Create NPM scripts for automation and helper scripts for various data processing and development purposes - Maintenance and troubleshooting of existing Cron jobs automation scripts (NPM scripts) - Oversee the deployment (CI/CD) of the frontend and backend to their respective cloud PaaS or cloud service (current: Render, Firebase and GitHub Pages) - Monitor PaaS logs for system errors and apply fixes if possible - Coordinate cloud services payment subscriptions (billing) with project sponsors and team, in case a decision was made to avail paid subscription plans - Coordinate raw data creation and acquisition from other team members. Deploy data to the Firestore database using NPM scripts. - Regularly report progress and roadblocks, and coordinate with other team members in a timely manner - Technical Software Development Documentation

Web Developer – Frontend (1)	
Requirements	NodeJS, React, Firebase (for Web), Git
Skills	- Web user interface (UI) development using React - Responsive web design and layout - Using REST APIs on the frontend - Using Firebase on the frontend
Expected Output	- Maintain ACAP's website user interfaces (UI) - Create responsive and interactive new user interfaces (UI) and web pages as required by the project using React - Display data from the Firestore database as required in the UI using Firebase (for Web) - Consume the REST API endpoints from the backend. This task requires coordination with the Backend developer.

	- Regularly report progress and roadblocks, and coordinate with other team members in a timely manner - Software Documentation – creation of end user guides and manuals with reference to the Technical Software Development documentation.

Pros of hiring more developers

- Parallelism. Sub-tasks and items from complex "general" objectives, whose "specific" requirements may change or whose technical details may become more known anytime during the project duration, are worked on simultaneously without waiting to finish the first tasks. Parallelism can contribute to the timely submission and release of deliverables and output.
- Flexibility to implement more suitable feature requests and enhancements - With regards to parallelism, a team of developers can flexibly implement new simple to complex features and product enhancements, which users and stakeholders may deem necessary anytime during the project duration.
- Timely response on existing complex project upkeep, maintenance, unexpected external factors (like mentioned in the previous risks items).
- If (1) one developer is unavailable and the system crashes, encounters errors, or experiences downtime, other knowledgeable developers (at most 2) can troubleshoot, diagnose and restore the system.
- Maintenance, housekeeping and upgrade of existing technologies -- The lead software project developer can have more flexible time to research, canvas, experiment and test new technologies and approaches to further improve and boost the system.
- By delegating and owning specific tasks, complex goals feel more manageable, and participating developers/programmers can produce cleaner, quality, and thoroughly reviewed output despite tight deadlines, especially in case of unexpected scenarios or new required features or revisions.

Cons of hiring more developers

- New developers may need a long onboarding time to familiarize themselves with the existing system and codebase, which may impact the start of the project's software development.
- The project may not have enough budget to hire more developers.
- Lack of web developer applicants who possess the required technical knowledge and experience needed by the project

Cons of <u>not</u> hiring more developers

- Assigning all simple and complex programming tasks and objectives, especially the broader "general" ones, to only (1) programmer lacks parallelism. It will eventually lead to burnout for the solo developer/programmer while striving to produce quality output and meet schedules, and the *<u>whole project being late and behind the timetable</u>*.
- Assigning programming tasks to those inexperienced with ACAP's technical software development requirements is less productive. It will slow the development or have little to negligible impact on the programming work.

It is also worth noting that not all programmers/developers are willing to take extra programming sidelines, especially if they feel that the current project that they are working on needs their focused, undivided attention to deliver top-quality output; or if the target sidelines have equally large scopes which may compete with their prioritized work.

Project Planning Dependencies

Does this project conflict or compete for resources with any other project?

No, this project has its own resources (database, hosting and data), which can be readily set-up and acquired.

Are the same human or machine resources allocated to maintenance of past versions and/or planning of future versions during this release time period?

Yes and no. This release is the first official release (ACAP 1.0) after beta and a series of internal (unofficial) alpha releases. All releases up to this point used the same machine resources, comprising standard (free) plan pricing cloud services.

The human resources (team) remained partly the same, with some members leaving and new ones joining during the planning and maintenance of past versions up to this release period. We cannot say if all or some members of the current team will be available on future versions from this release period.

Does this project depend on the success of any other project?

No. This project is already functional, but its seasonal weather forecast component will improve more after setting up automatic syncing with the seasonal weather forecast API from PAGASA. The seasonal weather forecast API from PAGASA is not yet available for now.

Does any other project depend on this project?

Yes. (2) two new projects now depend on this project.

1. Around June 2022, UPLBFI started another project that uses ACAP 1.0 as a template for an ACAP for Region 6 (Central Visayas). There are also plans to use ACAP 1.0 as a base template for other regions. We forked the original ACAP 1.0 repository to another private GitHub repository for sharing purposes to assigned regional developers on https://github.com/amia-cis/climate-services-webportal-v1/. The developers will fork this template repository and add new features and updates to their forked repositories specific to their regions. Guidelines for emulating the process for local development to online deployment used by ACAP 1.0 are made available at https://github.com/acaptutorials/assets-cms/wiki/ACAP-1.0-Duplication-Video-Tutorials-and-References, as decided by the factors (see figure). While we encourage each regional developer to follow the deployment and hosting process used for ACAP 1.0, each region is free to adjust according to its available resources and infrastructure.

ACAP 1.0 Scaling

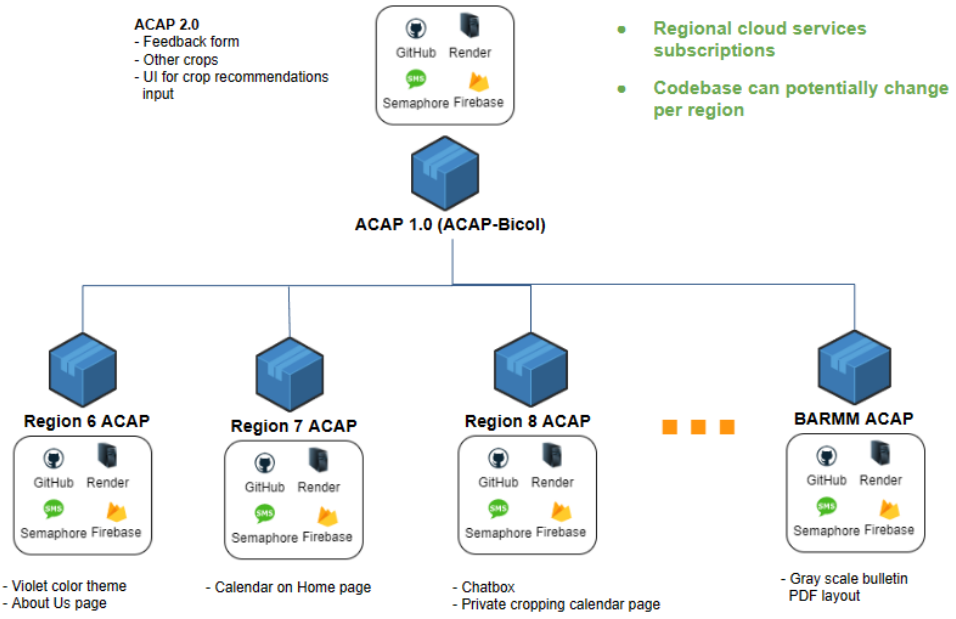

Figure 2.0: ACAP 1.0 Scaling factors

2. We started a collaboration with IRRI in June 2022 for IRRI's Rice Crop Manager Advisory Service (RCMAS) app. IRRI is creating a prototype project, where it needs to schedule sending an SMS containing crop recommendations to selected farmers in the Bicol region during forecasted

"heavy rainfall" periods. We created new API endpoints on ACAP 1.0's existing set of endpoints to allow secure, trusted sharing of formatted weather forecast data (sourced from PAGASA) for IRRI.

We forked the original ACAP 1.0 repository to another private GitHub repository for development purposes specific to IRRI on https://github.com/ciatph-dev/acap-rcmas. And we hosted the new API endpoints in another hosting environment that mirrors ACAP 1.0, separate from the official and stable ACAP 1.0, while the collaboration project is in a prototype development phase. The new development APIs available at https://acap-rcmas.vercel.app/ have plans for integration back with ACAP 1.0 should DA AMIA Region Field Office 5 (DA-RFO 5) decide to use it for future purposes.

Are there any other important dependencies that will affect this project?

Yes. ACAP 1.0 depends on external files, cloud services and project collaboration agreements.

- ACAP 1.0 has external dependencies whose availability and structure/formatting will affect the established ACAP processes and operations.
  - PAGASA 10 Day Climate Forecast Excel files (day1.xlsx – day10.xlsx) [link]
  - PAGSA El Nino / La Nina Monitoring web page [link]
  - PAGASA Tropical Cyclone Bulletin web page [link]
  - PAGASA seasonal outlook forecast Excel file (shared thru email by PAGASA)

- ACAP's uptime also depends on the continuous uptime and availability of the cloud services:
  - Firebase
  - Render
  - GitHub Pages
  - GitHub Actions (schedule workflows)
  - Semaphore SMS
  - MapBox

- ACAP 1.0 depends on collaborations and memorandum of agreements (MOA) between UPLB Foundation, Inc. (UPLBFI), Alliance Bioversity International, the Department of Agriculture (DA), AMIA DA Regional Field Office 5 (RFO 5), and the Philippine Atmospheric, Geophysical and Astronomical Services Administration (PAGASA).

- ACAP's paid cloud services subscription plans depend on the region, as illustrated in *"Figure 2.0: ACAP 1.0 Scaling factors"*. The project expects the currently assigned ACAP 1.0 web developer to manage the paid subscription options should the region concerned decide to upgrade its cloud stack.