Requirements Specification Document

April 19, 2024

Project Sponsor
Dr. Eck Doerry

Team Mentor
Vahid Nikoonejad Fard

Capstone Instructor
Michael Leverington

Team Members
Dylan Anderson, Jennie Butch, Noah Gooby
Nathan Hill, Jade Meskill

Version
1.0

Team Lead Signature: _____ Date: _____

Client Signature: _____ Date: _____

Table of Contents

1 Introduction

Flooding is the single most common and destructive natural disaster, causing over $3.7 billion in damage and claiming more than 120 lives annually across the United States. Nationwide, the frequency of disastrous flood incidents has more than doubled since 2000 and is expected to more than triple by 2050. Here in Coconino County, a recent analysis of post-wildfire flood risks has shown that local water flows could soon reach between 3x – 16x normal levels. The growing frequency of these natural disasters demonstrates the urgent need for an efficient, accurate, and innovative solution.

Traditional flood detection systems are often cumbersome, expensive, and too impractical to be deployed on a large scale. Many current solutions rely heavily on substantial physical infrastructure, such as complex gauges and even entire meteorological stations in order to collect data on flood risks and local water levels. Furthermore, these approaches typically require a team of surveyors utilizing expensive, non-trivial equipment who must travel to remote areas in order to record accurate measurements between gauge points and a chosen zero point. This approach is expensive, requires a large amount of manpower, and is unable to provide real-time data or warnings of imminent flood risks.

The HydroCams project, sponsored by Dr. Doerry, seeks to mitigate these issues. Through the use of cheap, easily installable, solar-powered, cellular-connected smart cameras, our software will automatically detect gauging points and perform all necessary metrological calculations through the use of computer vision (CV) and Structure from Motion (SfM). This allows water levels to be measured automatically based on which gauge points are obscured, providing local entities with accurate, real-time information of water levels and flood risks. This approach will likely be foundational to all future flood detection systems, as it fully addresses the key issues that plague nearly all current solutions: cost, accuracy, access to real-time information, and efficiency, which will save lives and potentially millions of dollars in damages.

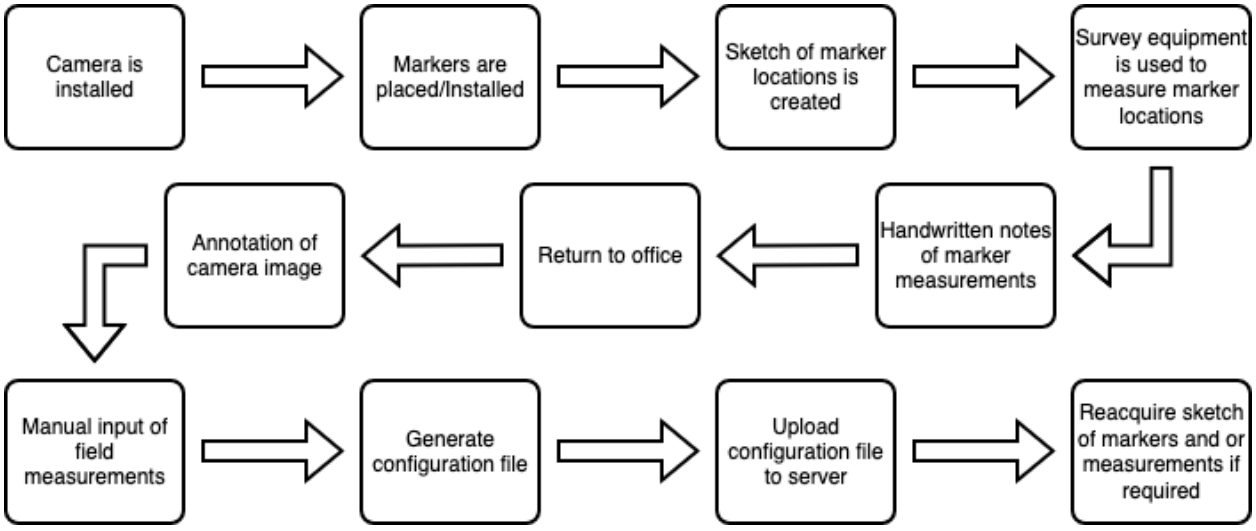
In the upcoming sections, we will discuss the problem at hand, our envisioned solution, and the specific requirements needed to achieve said solution. Additionally, we will explore the few potential risks that may appear during development, along with a detailed description of our expected development timeline. By closely following our outlined plan, we aim to revolutionize the world of flood detection.

2 Problem Statement

With the understanding that flood monitoring camera installation is a complex and time-consuming task, our client Dr. Doerry's goal is to streamline the process. This allows these cameras to be installed in a cost-effective and timely manner. The main focus of this project is to optimize the marker installation and camera configuration process. The current process, outlined in Figure 1, is very cumbersome, utilizing extremely expensive equipment from trained surveyors, and requiring multiple steps that are prone to human error. These steps include:

1. Sending a technician (or team of technicians) to a survey point
2. Technicians must hand-draw a map of the current markers
    a. Creates a large margin of error in terms of accuracy
3. Use of expensive, specialized survey equipment
    a. Technicians must be thoroughly trained in order to operate this equipment
    b. Operating this equipment on hazardous terrain introduces risk of incredibly costly damage
4. Must annotate hand-drawn map with markers and respective measurements between them
    a. Again leads to severe risk of inaccuracy from human error
5. Technicians then need to travel back to the office
6. Technicians must manually annotate camera images based on their hand-drawn field notes
7. Lastly, technicians must manually enter all necessary measurements for calibration, again introducing risk for error – then manually upload the calibration file to a server

Figure 1 | Current Configuration Workflow Overview



As one can see, there are multiple opportunities for errors and improvements. Such problems include mislabeling field notes, inaccuracies, loss of notes, requirements for specialized training, all requiring either assumptions to be made or additional trips back to the field. While these may appear to be minor issues, just one could add immense time and cost depending on camera location and accessibility.

3 Solution Vision

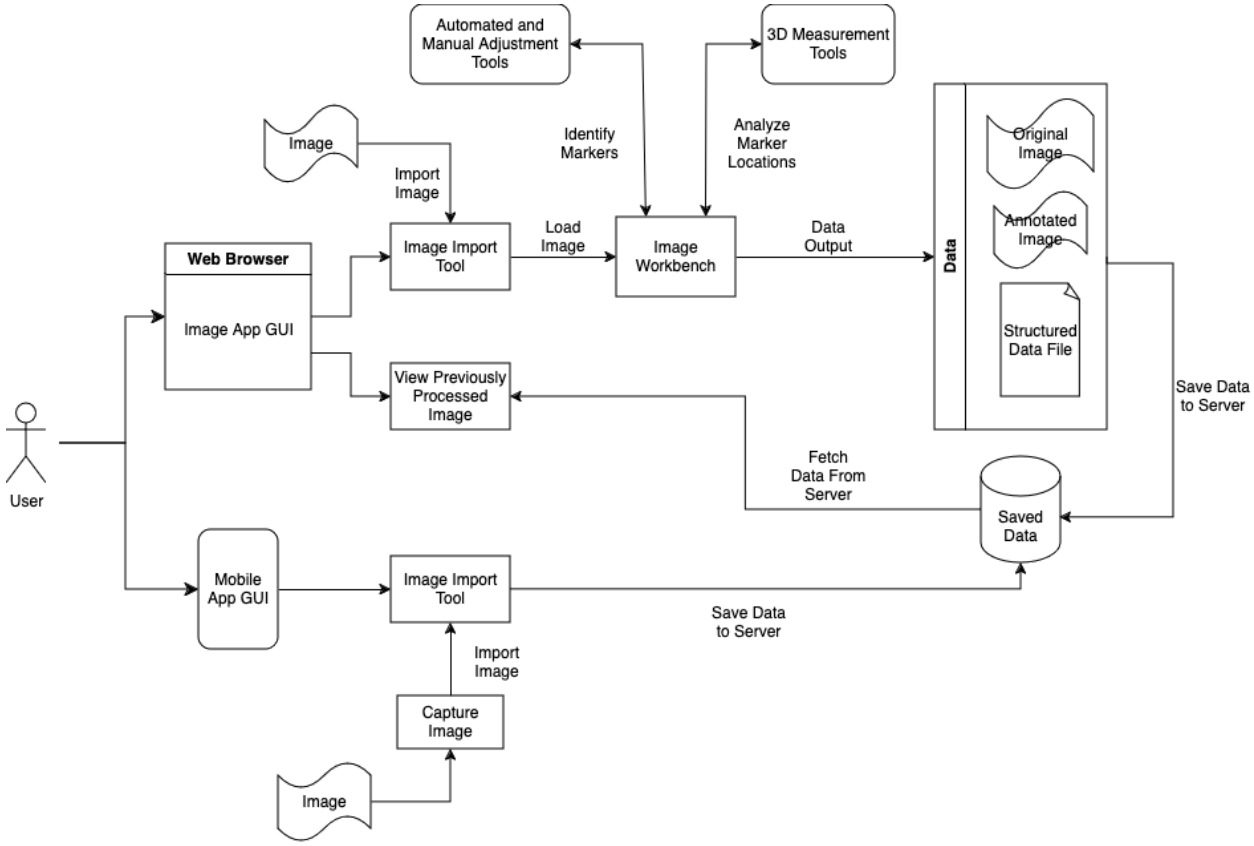
In order to address the problems presented above, we have outlined a solution that will greatly decrease the time and complexity required to set up and configure the current system. Our solution consists of: An image workbench to allow for image manipulation and marker identification, a computer vision program to identify markers automatically, a structure from motion component to calculate 3D distances, and potentially a mobile application to be used on-site to take the pictures required for structure from motion. In specific, our solution will offer:

- For the image workbench:
    - Local image upload functionality
    - Rectangular selection tool for marker identification
    - Modification / deletion of existing selections
    - Basic navigation (zoom, pan, etc.)
    - Local calibration file output
- For the computer vision program:
    - Automatic marker identification
    - Ability to tune for specific conditions (marker color, background, etc.)
- For the structure from motion component:
    - Accurate 3D measurement between markers
- For the mobile application:
    - Basic camera functionality
    - Photo export to remote server
    - Potentially, offline image workbench and CV functionality
- Overall:
    - Original and annotated image export to remote server
    - Calibration file export to remote server

For our solution to be effective, it will require multiple forms of input. First and foremost, the images from the HydroCam installations themselves are required for any and all further steps. Additionally, some form of human input (or at least observation) is required to ensure proper marker identification. Computer vision could simplify this further, automating the marker identification process. That marker identification data will be exported in some data serialization format, and stored on the remote server. The calibration file is then used to calibrate the camera to properly detect flooding, though this is outside our project's scope. This alone simplifies the current process by an order of magnitude, requiring little or no manual labor, compared to the complex measurement and travel currently required. The mobile app could increase the convenience factor even

further, allowing technicians to take photos of the camera installation itself, as well as the photos required for structure from motion, annotating and potentially using computer vision to annotate them automatically. These photos will be exported to the remote server for storage, once a network connection is available. This improved flow is outlined in Figure 2 below. Basic marker identification using computer vision is nothing new, and unfortunately is not revolutionary in and of itself, but in areas where it has not yet been implemented, it could be. Application to flood detection is seemingly novel and could greatly benefit people across the globe.

Figure 2 | System Flow Diagram



4 Project Requirements

The following sections will outline the functional, non-functional, and environmental requirements for the HydroCams project. These requirements have been thoughtfully chosen in order to guarantee the functionality, reliability, and accuracy of our software. Our functional requirements, which detail the critical functionalities of our software, have been listed in order of priority. These are followed by our non-functional (performance) requirements, which detail *how* we want our software to operate and perform. Finally, our environmental requirements outline the inherent constraints of our project, limitations imposed by factors beyond our control, along with our plans to mitigate these constraints.

4.1 | Functional Requirements

4.1.1 | Image Workbench

- **Local Image Upload:**
    - Users shall be able to browse their local files for an image to upload. Accepted file types include PNG, JP(E)G, and HEIC.
    - Once selected, the user can select “Upload” to upload the image into the workbench environment.
    - If a user uploads a new image to the workbench, the workbench environment will reset.

- **Marker Identification:**
    - Users shall:
        - be able to use a rectangular selection box tool to manually select markers in their image. The user can then save the selection as a marker.
        - have the option to automatically detect all markers within the image rather than manually selecting them.
        - have access to an RGB color selector to fine tune the color of markers that will be automatically detected. This tuning will involve selecting a “low” and “high” RGB value for each color channel, forming a narrow range of color to detect.

- **Marker Modification:**
    - Saved markers shall automatically be listed for the user in the order they were saved (e.g., Marker 1, Marker 2, Marker 3).

    - Selecting a saved marker shall display the selection box of that marker within the image.
    - Users shall:
        - be able to edit each marker selection individually.
        - be able to delete any marker selection.
        - have the option to mark a single selection as the zero point marker.

- **Measurements:**
    - Users shall:
        - be able to manually enter measurements between any pair of markers.
        - have the option to select an “Auto-Calibrate” option, which will utilize SfM to automatically fill out the measurements between each pair of markers.
    - Measurement units will be in metric as a standard.

- **Image Canvas Navigation:**
    - Users shall be able to zoom in and out of the loaded image as well as pan across it.

- **Calibration File Output:**
    - Users shall:
        - have the option to output a JSON calibration file to either their local system or a remote server.
        - have the option to choose between metric and imperial measurements for output.

- **Image and Calibration File Storage**
    - All annotated images, along with the calibration files they produce, will be stored and organized in the remote server’s file system.
    - These files will be stored in directories organized by camera ID, image status (original or annotated), and date/time. Calibration files will be stored separately from images, but with matching naming conventions.

4.1.2 | Computer Vision Implementation

- **Automatic Marker Identification**
    - Ability to detect objects of a known size, shape, and color
    - Automatically designate detected objects as markers

4.1.3 | Structure from Motion Implementation

- **3D Marker Distance Measurement**
    - Ability to measure distances between markers, taking depth into account for maximum accuracy

4.1.4 | Mobile Application Implementation

- **Camera Functionality:**
    - Users shall be able to open the app and take an image.
    - Once the image is taken then the user can upload the photo to be annotated.

- **Image Upload Functionality:**
    - As mentioned above, the user will take a picture to be uploaded to the server. When using the app offline, the image will be queued so when the user is online once again the images can then be uploaded to the server. Otherwise, they are uploaded instantly.
    - After the upload process, the app should receive both the annotated image and associated calibration file in response. The annotated image will show where the markers have been identified in the image, and ask the user they are correct. If incorrect, it can be sent back to the server to be reanalyzed. In addition to the annotation process, if for some reason the image cannot be processed, the user will be notified.

4.2 | Non-Functional (Performance) Requirements

Now that the core functional requirements of our system have been thoroughly discussed, we will transition into outlining the non-functional (performance) requirements. These requirements detail *how* we expect our functional requirements to perform (including computation times, data storage integrity, and UI performance), along with higher-level requirements of reliability and maintainability to ensure our project is robust and capable of surviving long into the future.

4.2.1 | Computation Times

- **Computer Vision:** The bulk of our project relies on OpenCV to perform computer vision processing. Therefore, optimizing this process is crucial. Our system should be capable of performing these operations within a maximum of 2 minutes to ensure a seamless workflow.

- **Structure from Motion:** All necessary SfM operations (which may involve processing sets of 3 - 10 images at a time) should take no more than 5 minutes.

4.2.2 | Data Storage

- **Image Storage:** All images uploaded by users must be stored in a dedicated directory on the server. These images must retain their original quality and metadata. Furthermore, all annotated images (which include user identifications of markers) must also be stored on the server, separately from the originals. These should be linked to their original counterparts through naming convention (e.g., camera identifier and date/time of image).

- **File Organization:** The server file system must be organized hierarchically to support scalability and ease of access. All files should be categorized first by their camera identifier, then by if it is an original or annotated image, followed by their date/time. As mentioned above, consistent naming conventions must be used to support this organization (e.g., cam17910_orig_7-6-24:17:20.jpg).

- **Security:** It is crucial that we implement appropriate access controls to ensure that only authorized users can access or modify our stored images. This will involve changing our file system permissions so that the image directory may only be accessed by superusers (developers, authorized technicians, etc.).

4.2.3 | UX

- **Responsiveness:** The full webpage UI should load within a maximum of 4 seconds when initially accessed. Further interactions (form submissions, calibration output, etc.) should happen dynamically whenever possible, and should take no longer than 1 second regardless.

- **Efficiency:** Our web application should utilize the minimum amount of resources possible so that it can easily run on outdated hardware.

- **Reliability:** The UI should implement graceful error handling so that users are given clear and informative messages in response to errors or crashes.

- **Modularity:** The UI should be designed with modularity in mind, so that future features may be added (or current features modified) without requiring a full page redesign.

4.2.4 | Reliability

- **Data Backups:** To ensure data integrity and availability, our system will use rolling backups. Rolling backups involve continuously overwriting older backups with newer ones, to limit storage impact.

4.2.5 | Maintainability

- **Clear documentation:** Having abundant documentation for all of our components is paramount in ensuring a maintainable system. To achieve this, we plan to have all code clearly documented (commented) along with high-level documentation in our GitHub repository.

- **Well-defined coding standards:** To ensure readability in our code, our team will stick to a rigid set of coding standards and styles. Our coding standards will also be documented in our GitHub repository.

- **Safe Updates:** To ensure easy and safe maintenance, our system should have a method in place to easily and quickly roll back to previous automated or manual backups. This allows maintainers to not have to worry about potentially breaking the system with each update.

4.3 | Environmental Requirements

4.3.1 | HydroCam Hardware

- **Cameras:** Cameras are placed in locations prone to flooding on sturdy poles or other stable structures and are powered by a small solar system. The camera set-ups are not only able to provide pictures but also data on the power system as well as local temperature. These 8 megapixel cameras need to capture high enough quality images to detect distant small markers in order to calibrate.

- **Gauge Points:** Gauge points are the primary method of allowing the camera to measure water levels. Our testing gauge points are made of sturdy material and are firmly placed in/on the ground or terrain where measurements are needed. They will be painted with stripes of the same color red as the markers that are used for calibrating the cameras.

- **Reliability:** The HydroCam hardware is well set up for reliability between mount points, power, and data. Cameras are mounted to secure poles or at least to a secure structure that should allow them to withstand irregular weather conditions and events. Gauges are also securely placed into the ground or onto a surface to

prevent the risk of them being washed away. The solar power system was designed to allow for reliable 24/7 operation of cameras, ensuring that there is no time when water levels are not being monitored. The ability of the camera systems to provide extra data on top of pictures allows for easy monitoring of current camera reliability.

4.3.2 | Mobile OS Compatibility

- **IOS and Android:** As instructed by our client, we are going to implement the app on both IOS and Android. Android will require an OS version 10 or higher, and IOS will require version 13 or higher. The main functionality on both devices requires camera accessibility and server communication.

4.3.3 | Hardware and Software Used by Technicians

- **Cross-Browser Compatibility:** The web application needs to be accessible and reliably run on all modern browsers such as Chrome, Firefox, Edge, Safari, etc. This will be accomplished by developing the application using established frameworks and cross-platform testing.

- **Outdated Hardware:** To mitigate the impacts of older hardware being used, a web application was chosen. This does require the user to use hardware capable of running a modern web browser, but that limitation should be minimal. This also allows for a variance in operating system without any impact on application usability.

5 Potential Risks

With a project designed to mitigate the damage of pre-existing natural phenomena, risks are bountiful. While there will never be a greater risk due to a failure of our project, those existing risks are still dangerous and should be avoided if possible.

Calculation Inaccuracies

- Our CV and SfM calculations will be used in determining if flooding is taking place in the observed area. If those calculations are incorrect, either from a bug in the SfM or CV algorithm, flooding could possibly go undetected. This could lead to potential damage or loss of property or life. We plan to thoroughly test and validate our software to avoid this risk.

Injury During HydroCam Installation

- While the actual camera installation is out of our project scope, we have a hand in part of it. The need for images from varying distances and angles means that technicians may be moving around unfamiliar terrain, potentially leading to injury. To mitigate this, we plan to develop a robust SfM implementation that minimizes the amount of images required. Lastly, technicians should follow standard safety protocols established by their respective agencies to avoid risk of injury.

Destruction of Markers

- Our final risk involves the possibility of markers being destroyed or otherwise displaced due to natural events, be it wind, rain, wildlife, or even flooding. While the destruction of these markers would require reinstallation and recalibration on-site, this process would still be easier with our system than it would with the traditional approach. For testing purposes, we plan to construct our markers of hard plastic – along with devising a secure mounting system – in order to ensure the reliability of our markers and mitigate the risk of destruction/displacement.

6 Project Plan

Our team is committed to following a well-structured and clearly defined development timeline in order to guarantee the highest quality of our software. To support this, we have established a collection of seven key milestones, each with their own clear deadline. We strategically decided on these milestones so that they build upon one another, ensuring a coherent, progressive development process.

As this project spans across NAU's two-semester capstone sequence (Spring 2024 - Fall 2024), the majority of our development will take place in the Fall, beginning in September. Our team plans to perform further research and prototyping throughout the summer leading up to this. We are confident that this amount of time for research, combined with our thoughtfully developed milestones, will allow us to fulfill all of our project's requirements by December 2024.

Milestone 1: Develop the Front End of the Image Workbench Using Svelte

We chose the development of our front-end interface as our initial milestone for a multitude of reasons. Primarily, this interface will serve as the visual and interactive foundation of our entire project. Developing this component early is essential for shaping the user experience moving forwards. Having a functional interface will also allow for far easier testing throughout the development process. Lastly, the design and implementation of our front end will largely dictate the requirements for our next milestone, the back end of the workbench.

This milestone will consist of designing and developing a clean, responsive, and intuitive user interface. It should also be developed in a modular, adaptive fashion so that adding functionality in the future is a trivial process. It will also involve implementing basic functionalities like input forms and any necessary navigation options. Finally, this milestone will include deciding on an online hosting solution so that our front end is fully online.

We plan to reach this milestone by the final week of August.

Milestone 2: Develop the Back End Using Django

We will approach this milestone, for the most part, concurrently with our front-end development. This step involves developing a robust back-end infrastructure to serve as the backbone for the remainder of our project. The back end will be vital in supporting our previously developed front-end functionality, ensuring reliable data management and facilitating further development and necessary integration capabilities.

This step will involve integrating all necessary APIs to ensure communication between our front and back end, and most importantly, creating an intuitive, scalable architecture that will be prepared to connect to our future computer vision, structure from motion, and mobile application implementations.

We plan to reach this milestone by the end of August.

Milestone 3: Implement Basic Image Handling and Marker Management

At this point, our team will have a foundational front and back end fully implemented. This step will therefore focus primarily on the implementation of core functionality, specifically the ability to upload and annotate images. During this process, we will develop tools to allow the user to select areas on an image to identify as a marker; they will also be able to edit and delete existing markers. Finally, we will implement basic image navigation capabilities, specifically pan and zoom, into our front end.

We plan to reach this milestone by the final week of September.

Milestone 4: Calibration File Output

Our fourth milestone focuses on implementing the functionality for JSON calibration files to be generated based on the input parameters (number of markers, distance between markers, distance from zero point, etc.). During this step, we will focus on ensuring accurate and robust generation of calibration data along with providing the user with flexible output options, including output destination and measurement unit preferences.

When this milestone is achieved, we will have a fully-functioning manual marker identification application; all that remains is to automate it using CV for a more streamlined process, implement SfM for increased accuracy, and design mobile applications for increased ease-of-use.

We plan to reach this milestone by the middle of October.

Milestone 5: Integrate OpenCV for Automatic Marker Detection

After achieving the previous milestone, the core functionalities of the image workbench will have been implemented. This milestone will therefore focus on streamlining the current process by allowing for automatic detection of markers via computer vision, specifically OpenCV. We plan to utilize OpenCV's plethora of image processing libraries to develop an algorithm capable of automatically detecting markers of a specific shape and general color within an image. After successfully implementing this feature, users should no longer need to manually identify any markers (though they are

still able to edit the automated selections). A large portion of this step will involve rigorous testing and validation to ensure accurate tuning of our algorithm.

We plan to reach this milestone by the first week of November.

Milestone 6: Structure from Motion Implementation

Our sixth milestone involves implementing sophisticated structure from motion techniques to automatically calculate the distances between markers and a zero-point on a 3D plane. This functionality will further streamline the user experience by eliminating the need to manually enter measurements. This step will involve developing and integrating SfM algorithms that can accurately measure distances between identified markers, and, similarly to our last milestone, extensive testing and validation to ensure optimal accuracy.

We plan to reach this milestone by the middle of November.

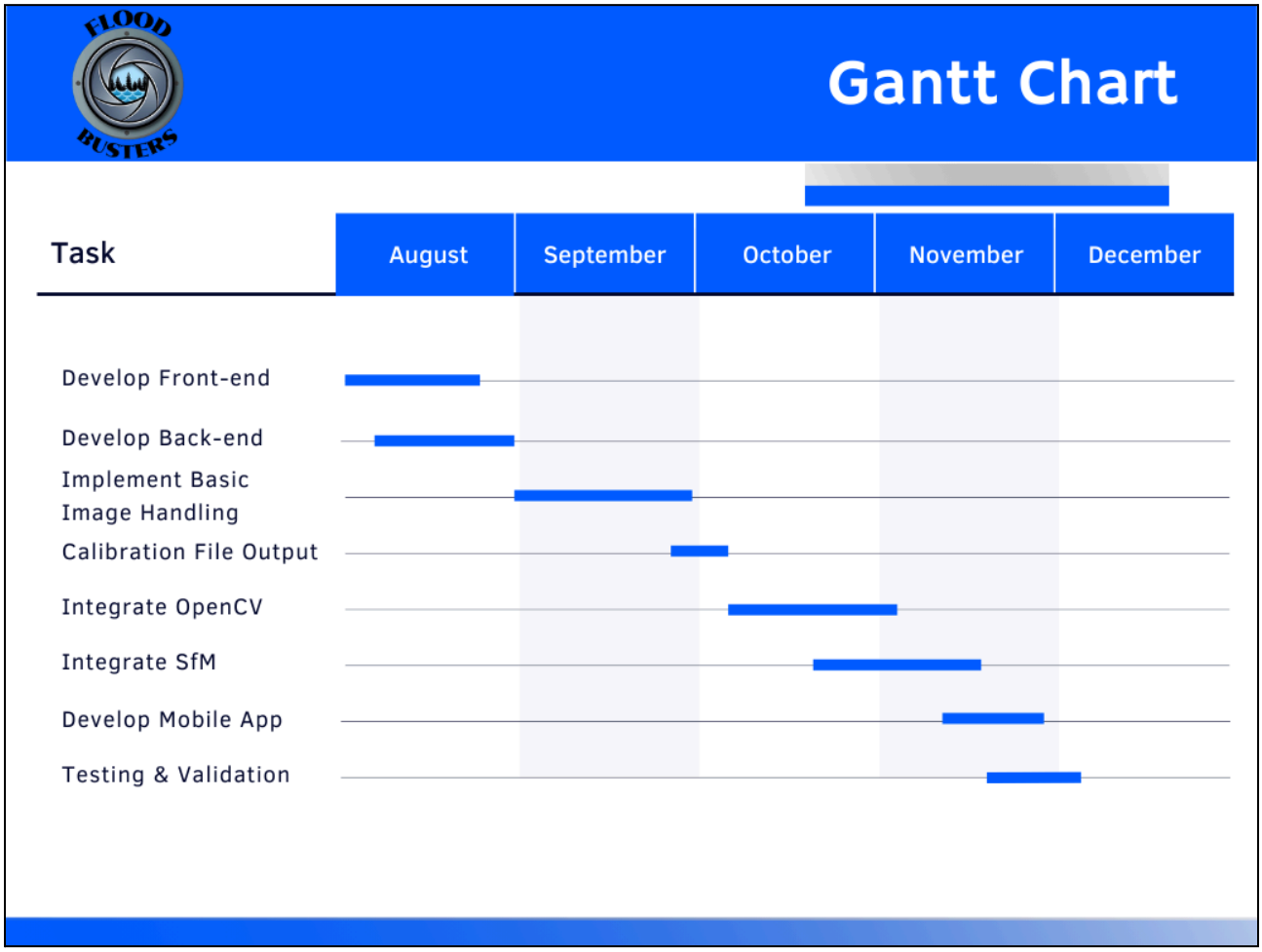
Milestone 7: Develop Mobile Application Using Ionic

With our core software having been fully developed, our final milestone will focus on developing a mobile version of our web application. This will allow technicians to easily take photos of markers from their phones while in the field, which will then be automatically processed by our software to produce a calibration file. Having a mobile app implementation allows technicians to immediately detect and address images whose markers could not be identified – instead of having to travel back to their office, notice the issue, and travel back to the field once again, they should now be able to simply take another set of photos on the spot and retry the calibration.

We plan to reach this milestone by the final week of November.

Figure 3 below depicts our Gantt chart, serving as a visual representation of our project plan.

Figure 3 | Gantt Chart



7 - Conclusion

In conclusion, flooding is a worldwide problem that impacts people's lives significantly every year, and is expected to only grow in frequency. Whether it's property damage, disruption of infrastructure, or even loss of human life, flooding is one of the most destructive natural disasters – due to not only their frequency, but the immense inefficiencies and difficulties involved in modern flood detection systems. The HydroCams project aims to revolutionize traditional flood detection systems with innovative, modern technology, reducing the time and resources required compared to traditional approaches. With the current process requiring skilled personnel, expensive equipment, and archaic methods of data recording requiring excessive man-hours to perform, we feel we can significantly improve this process by implementing our proposed solution.

Throughout this document, we have reintroduced the core problem our project serves to solve, restated our envisioned solution, described the individual components that go into that solution, risks associated with them, and a plan for their development. Our solution involves an online image workbench, computer vision marker identification, 3D marker measurements using structure from motion, and a mobile application to supply the required images and perform computations in the field. Those components can be further described through the functional, performance, and environmental requirements. The functional requirements describe the functionality required of the solution components, the performance requirements describe the efficiency and accuracy of the solution components, and the environmental requirements describe limiting factors for the project itself.

The HydroCams project promises to offer quick and accurate flood monitoring at a fraction of the current cost and complexity, which would serve as a significant leap forward in disaster preparedness. We are confident that HydroCams will become a pivotal tool in safeguarding communities, saving lives and resources along the way.