Development Plan
Software Engineering

Team #11, OKKM Insights
Mathew Petronilho
Oleg Glotov
Kyle McMaster
Kartik Chaudhari

Table 1: Revision History

Date	**Developer(s)**	**Change**
9/24/2024	Oleg, Kartik, Kyle, Mathew	First Revision

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

1 Confidential Information

There is no confidential information to protect.

2 IP to Protect

Currently we chose not to protect our IP as we are aiming for a commercial product. Down the line we most likely will have to opt in for IP protection.

3 Copyright License

We opted to go with the Apache License 2.0. It allows anyone to use, modify, and distribute the software for free as long as they include the license and copyright notice. However, we are entitled to payments if it is used commercially.

https://github.com/OKKM-insights/OKKM.insights/blob/main/LICENSE

4 Team Meeting Plan

The team will meet weekly on Mondays from 11:30 until 13:20, in a study room booked by Kyle McMaster. The schedule to meet with our supervisor will be determined when our supervisor is confirmed.

Weekly meetings will be chaired by Kyle, who will prepare an agenda to be sent in advance of team meetings. Team members can add additional agenda items as needed. Team members are also able to request additional meetings if necessary. In this case, it is their responsibility to chair the meeting, organize a meeting location, and provide an agenda. Following all meetings, the meeting chair will prepare a list of action items.

5 Communication Plan

1. Primary Liaison for External Stakeholders: Kartik will act as the main point of contact for external stakeholders, ensuring clear communication with groups like Air Rescue Services and other key parties.

2. Managing Cross-Team Communication: Responsible for ensuring seamless communication between team members and supervisor during any conflicts of interest, Kartik will coordinate updates, ensuring everyone is aligned with project objectives.

3. Issue Escalation and Resolution: Monitoring GitHub issues, any critical roadblocks will be escalated and resolved through team discussions during meetings or via Teams, ensuring the project stays on track.

4. Feedback Coordination: Feedback from external stakeholders will be compiled and tracked to ensure it is clearly communicated to the development team and integrated into project improvements.

5. Regular Stakeholder Updates: Team members will manage the creation and dissemination of concise updates to stakeholders, keeping them informed on project progress, timelines, and deliverables.

6. Documentation of Key Communications: All significant external communications and feedback will be documented for future reference, maintaining transparency and accountability across the project.

6 Team Member Roles

Member Name	**Roles**	
Mathew Petronilho	Document Manager, Front-End Development Expert	Developer, Tester, PR Reviewer, Issue Creator, Meeting Participant, and Note Taker.
Oleg Glotov	Team Lead, Project Manager	
Kyle McMaster	Meeting Chair, Back-End Development & AI Expert	
Kartik Chaudhari	Customer Relations Manager, Product & Deployment Manager, ML, & Git Expert	

Table 2: Team Roles

- **All Team Members:** Every team member is responsible for developing code and creating tests for the code. Everyone is also responsible for reviewing open pull requests and providing feedback if necessary. Additionally, all members will be tasked with creating issues using the appropriate templates, tracking issue status, and updating issues assigned to them with relevant information. Each member is expected to attend meetings punctually and contribute ideas to discussions, while maintaining respect-

ful and concise communication. The role of meeting note taker will rotate among members in each meeting. The note taker should keep track of meeting attendance, issues discussed, decisions made, and action items. These notes should be well-maintained and easily accessible to all team members.
If we encounter challenges, we may consider switching roles to maintain progress and improve team performance. More specific roles can be assigned as the project evolves and implementation details become clearer.

- **Mathew Petronilho:** Responsible for ensuring that all documents are formatted consistently, that all necessary components are included, and that there are no grammatical or spelling errors. Also responsible for assisting team members with the front-end and taking the lead in implementing this component of the project.
- **Oleg Glotov:** Responsible for liaising with the supervisor, teaching assistant, and professor. Coordinates project tasks among team members, organizes meetings, ensures equitable distribution of work, and monitors deadlines to ensure they are met.
- **Kyle McMaster:** Responsible for creating meeting agendas, guiding discussions, managing meeting time, and resolving conflicts. Also responsible for assisting team members with the application's back-end logic, deploying services, and contributing expertise in machine learning and artificial intelligence to integrate advanced data processing, and intelligent system functionalities.
- **Kartik Chaudhari:** Responsible for managing relationships with clients and stakeholders, ensuring that project requirements are understood and met. Leads the deployment and integration of the product by overseeing infrastructure, pipelines, and cloud environments. As the team's Git expert, Kartik ensures best practices for branching, merging, and resolving conflicts, while protecting the integrity of the main branch by managing required approvals. Additionally, Kartik applies his expertise in machine learning to guide the team in developing and optimizing models, ensuring the accuracy and efficiency of the platform's AI capabilities.

7 Workflow Plan

7.1 Git Workflow Plan

7.1.1 Git Workflow Oversight

Kartik will oversee the entire Git workflow to ensure the adoption of best practices. This includes enforcing branching and merging strategies, such as creating feature branches for new tasks and only merging them into the main branch after passing all necessary reviews. Protection of the main branch will be enforced by requiring approvals from 1-2 team members before any merges.

7.1.2 Approval Process

Before any branch is merged into the main branch, at least one or two team members must approve the changes. Code reviews will focus on checking for code quality, adherence to project standards, and testing coverage. Pull requests will include detailed comments describing the changes made, associated issue numbers, and any potential impacts.

7.1.3 Branch and Issue Management

Each branch will be associated with a specific GitHub issue to ensure tasks are clearly delineated. Issues will be small, focused tasks with clear descriptions. Each task will be broken into manageable parts, helping the team stay organized and providing clarity on the state of the project at all times. This will also simplify issue tracking and debugging when needed.

7.1.4 Milestone and Deliverable Tracking

GitHub's milestone feature will be used to track project deliverables. Each issue will be assigned to specific milestones, which are aligned with project deadlines. Regular monitoring of milestones will help identify blockers or delays early. Team members will be responsible for updating their progress and addressing blockers as soon as they arise.

7.1.5 Consistent Naming Conventions

A clear and consistent naming convention will be followed for branches, commits, and issues.

- **Branch names:** Should follow the format `<issue number>-<short-description>` (e.g., `23-fix-navbar` or `45-add-login-feature`).
- **Commit messages:** Should be concise but descriptive, explaining the purpose of the change (e.g., *Fix navbar alignment issue* or *Implement user login feature*).
- **Issue titles:** Will follow a similar format to describe the nature of the task (e.g., *Fix alignment on homepage navbar*).

7.1.6 Tagging and Labeling Issues

GitHub Issues will be categorized and labeled for better tracking and prioritization:

- **Priority:** `high-priority`, `medium-priority`, `low-priority` to signify the urgency of tasks.
- **Type:** `bug`, `enhancement`, `documentation`, `meeting` (for tracking meeting notes or action items).

- **Status:** `in-progress`, `blocked`, `completed` to provide a quick visual status of each issue.
- **Special tags:** `documentation` (for tasks related to docs) and `meeting` (for tracking meetings) to keep these types of work organized.

7.1.7 Integration of GitHub Issues and Projects

GitHub Issues will be the core tool for managing the project's progress. Each issue will be assigned to a team member and associated with a branch. The team will use GitHub Projects (or similar) for Kanban-style task management, keeping the workflow efficient and transparent.

7.2 Usage of CI/CD

7.2.1 Tex Files

To ensure the PDFs found in our repository are consistent with the tex files they are generated from, we have developed a CI workflow in GitHub actions. This workflow detects when a push has been made to the *docs* folder in the *main* branch. When this happens, the workflow automatically regenerates the relevant documents and pushes the new PDFs to the repository. This will ensure that our PDFs are always the most recent version.

7.2.2 Linting & Static Checks

As discussed in sections 10 and 11, we expect we will use Python to develop the backend components of our software. To improve the clarity of our code, especially when collaborating, we have decided to use type hinting. To enfore this requirement and reap further benefits of type hinting, we will use a static type checker as part of our CI/CD workflow. This is possible with a package called mypy.

For both the front and back ends of the project, we have selected coding standards as described in section 11. To ensure these standards are met, we will include a linting step before pull requests can be merged into the main branch.

7.2.3 Testing

We will use a suite of tests to ensure our code base continues to satisfy our requirements before integrating changes into the code base. These tests will be run when a new pull request is created, before a code review is conducted.

7.2.4 Deployment

Depending on how we decided to host our server, we will also investigate if it would be beneficial to develop a continuous deployment workflow when changes

are pushed to the main branch. This will be determined in the future when our requirements are more clear.

8 Project Decomposition and Scheduling

We will use a Kanban board to track tasks for each deliverable. First, work to be completed will be identified and added to the project backlog. Then, team members will be assigned tasks. Once assigned, they will be moved to the 'Assigned' section of the board. When actively being worked on, they will move to the 'inflight' section. After completion, and when a pull request is made, they will be moved to the 'for review' board. Finally, when the review is complete, they are moved to the archive. This organization will allow us to track how different components of the project are progressing, and identify if a team member is overwhelmed before it becomes a catastrophic issue.

The project can be found at
https://github.com/orgs/OKKM-insights/projects/1/

Phase	Tasks	Start Date	End Date	Milestone	Progress (%)	Expected Days	Hardest Challenge
Team Formation & Initial Planning	Team Formed, Project Selected	16-Sep-24	16-Sep-24	Team Formed	0%	1	Aligning everyone on the project scope
Problem Definition & POC Plan	Problem Statement, POC Plan, Development Plan	17-Sep-24	23-Sep-24	Submission of Problem Statement & POC Plan	2%	7	Defining a clear and achievable proof of concept
Requirements Gathering	Draft Requirements Document (Backend & Frontend)	24-Sep-24	08-Oct-24	Requirements Document Revision 0	5%	15	Gathering comprehensive and precise requirements
Hazard Analysis	Identify Risks (Data quality, server security, etc.)	10-Oct-24	23-Oct-24	Submission of Hazard Analysis	3%	14	Identifying all possible hazards in satellite images
V&V Plan	Verification & Validation Plan (Test Plans, QA Plans)	24-Oct-24	01-Nov-24	V&V Plan Revision 0	5%	9	Designing a comprehensive testing plan for image models
Proof of Concept (POC)	Backend setup (database, server), Frontend skeleton	01-Nov-24	10-Nov-24	Basic functionality demonstrated	5%	10	Integrating satellite data into the backend system
Proof of Concept Demo	Present basic working model, get feedback	11-Nov-24	22-Nov-24	Proof of Concept Demonstration	5%	12	Gathering meaningful feedback and adjusting the system
Backend Development	API development, satellite image integration	23-Nov-24	05-Jan-25	Functional Backend for satellite data	Ongoing	44	Handling large amounts of satellite data efficiently
Frontend Development	UI/UX Design, Integrate APIs, Map Rendering	23-Nov-24	05-Jan-25	Functional Frontend	Ongoing	44	Building a user-friendly and interactive map interface
Design Document Revision	Revision of Design Document with final system architecture	15-Dec-24	15-Jan-25	Design Document Revision 0	5%	31	Creating detailed technical architecture and design
Mid-Project Demonstration	Functional prototype of backend, frontend, and APIs	03-Feb-25	14-Feb-25	Revision 0 Demonstration	10%	12	Demonstrating a fully integrated prototype
Verification & Validation (V&V)	Test satellite image data processing, system reliability	15-Feb-25	07-Mar-25	V&V Report Revision 0	5%	21	Ensuring accuracy and consistency of the data processing
Final Project Demo (Revision 1)	Full system demo (backend, frontend, datasets, etc.)	24-Mar-25	30-Mar-25	Final Demonstration (Revision 1)	20%	7	Coordinating all components for a seamless demo
EXPO Preparation	Prepare for Expo, polishing the UI, final system tweaks	01-Apr-25	Apr TBD, 2025	EXPO Demonstration	10%	15	Finalizing and polishing all aspects of the project
Final Documentation	Compile final reports, code, guides (Problem Statement to Final Report)	01-Apr-25	15-Apr-25	Final Submission	30%	15	Completing all documentation to a high standard

Table 3: Project Schedule Overview

9 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

Our project consists of two main components: the computer vision algorithm and the labeling tool. We intend to present both in our upcoming demo to illustrate the complete pipeline—from data labeling to model training and application on new data.

9.1 Computer vision demo:

We will simulate a specific business case that involves searching for a particular object within an image, using the "Where's Waldo?" book series as our dataset. In these books, readers are challenged to find the character Waldo, who is cleverly hidden among numerous other characters and objects. This scenario mirrors real-world challenges in satellite imagery analysis, where identifying specific targets within complex visuals is essential.

Demo Objectives:

- Model Training: Train a computer vision model using the labeled data to recognize and locate Waldo.
- Out-of-Sample Testing: Apply the trained model to new, unlabeled images to assess its ability to find Waldo without prior hints.

This demonstration will showcase our ability to create a functional model capable of detecting specific objects in complex images.

9.2 Web application demo:

Our web application serves as the primary labeling tool within our platform. We aim to demonstrate a basic yet functional implementation that walks through all the steps involved in the labeling process.

Demo objectives (Frontend):

- Client Upload Page: A dedicated interface where clients can upload a labeling "Job," consisting of images requiring annotation.
- Job Selection Screen: A portal where users (labelers) can view available jobs and select tasks they wish to work on.
- Labeling Interface: A user-friendly page where labelers can annotate images according to the job specifications.

Demo objectives (Backend):

- Image Management: Accept and securely store incoming images from clients.

- Task Distribution: Break down images into manageable labeling tasks and assign them to users.
- Data Integration: Ensure the labeled data seamlessly feeds into the computer vision model for training.

By implementing these features, we aim to demonstrate how users can contribute to the labeling process and how clients can initiate jobs, demonstrating the full experience of our platform.

For both demos, we will utilize images from the "Where's Waldo?" book series. This choice offers several advantages:

- Clarity: The task of finding Waldo is straightforward and easily understood by all audiences.
- Complexity: The crowded scenes provide a realistic challenge similar to locating objects in satellite imagery.

10 Expected Technology

10.1 Frontend

- JavaScript: The primary language for the frontend development. It will handle user interactions and dynamic content updates on the web application.
- React: A popular JavaScript library used to build the user interface (UI). It will manage the frontend components and allow for a dynamic, responsive user experience.

10.2 Backend

- Node.js: Used for the server-side development, Node.js provides an asynchronous event-driven architecture, making it suitable for handling multiple client requests efficiently.
- Python: Plays a major role in machine learning (ML) tasks, image processing, and backend scripting.
- Flask: A lightweight Python framework used to build the backend server. It will handle API requests, manage server routing, and integrate with the database.

10.3 Machine Learning & Data Processing

- TensorFlow (TF): The main machine learning framework to train or fine-tune models for tasks such as image classification and segmentation. We may also leverage pre-trained models, like Meta's Segment Anything Model for segmentation tasks.
- Numpy: A core library for numerical computing in Python, essential for handling arrays and performing mathematical operations efficiently.
- PILLOW: A Python library used for image processing, enabling tasks like image resizing, format conversion, and basic manipulations.

10.4 Database

- SQLAlchemy: A powerful ORM (Object Relational Mapper) for Python that will facilitate database connections and work with both SQL and NoSQL databases.
- SQL (PostgreSQL): Suitable for handling structured data and relationships. There is a program called PostGIS which extendes the capabilities of PostgreSQL database by allowing us to store geospatial data. Ideal if we require ACID compliance and structured querying.

10.5 Cloud

- AWS/Azure: Used for hosting, storage, and machine learning models. AWS/Azure provides scalable infrastructure for both our backend and frontend.

10.6 Linting and Static Type Checking:

- Mypy: A static type checker for Python that will ensure the correctness of types throughout the code, helping prevent runtime errors.
- ESLint: The linter for JavaScript code, ensuring code quality and consistency in the frontend. It enforces best practices and helps catch errors early in development.

10.7 Model Training

- The project will involve training custom models or fine-tuning existing ones, particularly focusing on computer vision tasks. We will be working with Meta's Segment Anything Model for fine-tuning and extending its capabilities to fit our satellite image use case.

11 Coding Standard

Our back-end code, written in Python, will adhere to PEP 8 for code formatting and PEP 484 for type annotations. For our front-end code, written in JavaScript, we will follow the JavaScript Standard Style.

Appendix — Reflection

1. Why is it important to create a development plan prior to starting the project?

   Starting with a development plan was useful for our team for two reasons. First, it helped us develop a clearer picture of what our project will look like. In order to make development plan decisions, we had to discuss the actual project itself. This gave us a chance to clear up misconceptions we were having, and discuss what our overall mission is. More importantly though, working on this document (and problem statement/goals) was a dry run for future deliverables. As we worked on this doc, we were able to put our plans of how we thought we would work together to the test. We found that our git workflow needed a couple of refinements and clarifications. For example, we needed to create more descriptive pull request titles and descriptions so that reviewers know what they are looking to review before approving. Otherwise, it was very hard to get feedback on the *Team Charter* section if the pull review was called '8-expectations'.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

   We are already feeling the benefits of CI/CD in our workflows. Kyle built a tex workflow to compile our latex documents every time there is a push to the main branch. This makes sure that the .tex file and its associated PDF never become out of sync. In the past, we have all experienced challenges working on text files in git, as it is very easy to upload a code change and much harder to upload a pdf (usually requires fixing a merge conflict). This means that we probably just would do one big compile and upload when the document is 'finished'. The problem with that becomes when you are creating living documents and it never really gets finished and the PDF then becomes hard to trust as a source of truth.

   This workflow also exposed us to a disadvantage of CI/CD. It is very easy to spend a lot of time on it. Kyle spent close to 3 hours just getting the git workflow to work and that is not even a very complicated deployment. It is exciting to have such a hands on and interactive part of the code base, where changes are easy to see in real time. This makes it risky that we dedicate a lot of time to CI/CD, when it's not really necessary. We also intend to use CI/CD for regression testing. We have to be careful that we don't let the lovely green check mark showing that a PR passes all the tests, replace thorough code review. When people are pressed for time, it's easy to be tempted to skip reading the code if the tests pass. Although we will try to make a comprehensive test suite, there is always value to reading the code too.

3. What disagreements did your group have in this deliverable, if any, and

how did you resolve them?

We only really had one disagreement for this deliverable and it had to do with our collaboration style on documentation. Some members of the team would prefer to first write sections in a shared google doc, and then upload all of the changes at once when complete and reviewed. This would make it easier to contribute to other sections, not bog us down with pull requests, and allow others to read the document in its entirety. The other members prefer using git issues and pull requests to improve the traceability of contributions to the project. We have had a few discussions regarding this and in general they have been productive. We conduct brainstorming and task allocation in a shared google doc, so that everyone has access to the task delineation. Before discussing the issue further as a group, we decided to confirm with Dr. Smith whether the first option is even permissible. Following this response, we will meet again to finalize our approach. For now, we are using the git issues and pull requests to ensure traceability, as a small sacrifice to the ease of collaboration.

Appendix — Team Charter

External Goals

As a team, we all see this project as an opportunity to demonstrate the extent of our skills and knowledge after several years of training. As such, our main goal is to develop a project we can talk with excitement to future recruiters and for some of us, graduate school admissions committees. We intend to work hard to make a project we are very proud of. We all expect this effort will translate into an A+ in the course, although that is somewhat secondary. Our effort will not be limited by the expectations of the capstone course, and instead by the limits of our abilities.

Attendance

Expectations

- Team members are expected to attend every meeting in person. If attendance in person is not possible, they will notify the team at least 24 hours in advance so the team may discuss mkaing the meeting virtual or rescheduling.
- If attendance is not possible in person or virtually, they will notify the team at least 24 hours in advance. Each team member will be permitted to have 2 absences per semester without penalty. Additional absences, or absences without proper warning, will have the following diciplinary action:

  1-2 Bring snacks to following meeting.

  3-4 Bring snacks to following meeting & a message will be sent to TA.

  5+ Message is sent to course instructor. Meeting is scheduled with team to discuss absences. If meeting is not attended, student receives 2.5% deduction in final course grade per offence.

Acceptable Excuse

- Acceptable excuses are typically limited to exceptional circumstances that are urgent and could not have been foreseen. Examples of such events are medical emergencies of oneself or of family members, severe weather, or other risks for personal safety.
- Unacceptable excuses can be reasonably foreseen or expected by the member of the group. Examples of unacceptable excuses are getting busy with other courses, missing the bus, or over sleeping.
- If a team member is unable to meet a deadline due to other responsibilities, they are to alert the group with at least 7 days of notice. Each team

member will be permitted to request 1 reduction of work per semester, provided other team members are able to take over the task and 7 days of notice is provided.

- Excuses are to be judged by group consensus on a case by case basis. If an excuse for absence is deemed unacceptable, penalties described above will be applied.
- Missing a deadline with an unacceptable excuse will be met with an immediate penalty of a message sent to the TA describing the situation for the first offence. All other missed deadlines will result in a message sent to the instructor and a 2.5% penalty on course grade.

In Case of Emergency

- In the event of an emergency, team members must contact group as soon as reasonably possible if a meeting or deadline might be missed. Team members should avoid completing work at the last minute to lessen the effect of emergencies on the group.

Accountability and Teamwork

Quality

Based on our expectations for the project, the team is required to meet a very high standard of quality. Each addition to the project should be crafted with high attention to detail, and reviewed thoroughly by the creator. We will expect that every pull request is adding code or text that is ready to be submitted to the best of our abilities. To help each other ensure we meet this standard, we will require every addition to be reviewed by at least one other team member. This reviewer will approach the review as if they are the last line of defense for the quality of our codebase.

Every team member will complete expected action items and review the provided agenda before each meeting. They will provide additional agenda items when necessary.

Attitude

- Team members will treat other with respect at all times
- Team members will be friendly and collaborative
- Team members will remember that we are all working towards a common goal, to develop a high quality product
- Team members will not act hostily towards eachother. Offenders to this will face diciplinary action as outlined in the absence penalty structure

- Team members will openly communicate
- Team members will be open to team member ideas. If they do not agree, they are welcome to provide reasons as to why not. It is not permitted to decline an idea because it is 'not the way we do things'
- Team members will try to work together to make work load equitable
- Team members will try to accommodate external responsibilities, but understand it might not always be possible around large deadlines
- Team members will answer questions and provide additional information to team members who are interested in developing skills others are proficient in
- **We will be kind**

Stay on Track

The team will work well in advance to assign work to team members. The team will schedule time in weekly meeting to have a 'stand up' meeting, where they discuss progress and blockers. This will ensure there are no surprises when a deadline is approaching. We will also define earlier internal deadlines to provide enough time for someone to cover in the unlikely event that a team member is not able to meet a deadline.

We will track team member progress by the amount of deliverables they complete in each milestone. Typically, one deliverable will correspond to one or two GitHub issues. We believe this metric will be more useful than lines of code or commits, both of which can be manipulated and do not always capture the amount of value contributed. If planned correctly, each issue should contain roughly the same amount of effort to complete (although this won't always be the case).

We will also track attendance at meetings and lecture. Attendance at meetings is very important and penalties for absence are discussed above. Attending lecture is also important to the team, but less so. As long as we are able to find one person to attend, we will not worry much about lecture attendance. We will track lecture attendance so if we need to force someone to attend, we can choose the person with worst attendance.

Team Building

We already have a strong start with building rapport within the team. Everyone is very friendly towards each other and is aligned to the mission. We tend to sit together in lecture and are active in our group chats. After a large deliverable, the team has agreed to celebrate at a location of their choosing. Kyle might even host a party if there is interest within the team.

Decision Making

In general, we will strive for team consensus. If it is not possible, opposing parties will prepare arguments for their proposal. Other members of the team can then ask further questions, or announce their preference. To make a change to the current path, a majority is needed. Ties will result in the status quo. If at any time a group member has *new* arguments, they may bring the issue up again. It is important that team members are poilte and rational, but also provide a realistic indication of how deeply they feel about their arguments. If someone could go either way on an issue, and we are stuck in a stalemate, learning that they do not have a deep stake might be enough to help the group move forward.

Regardless of how passionate team members are about certain issues, they will follow our code of conduct and continue to be kind.