Development Plan
Software Engineering

Team #1, Sanskrit Ciphers
Omar El Aref
Dylan Garner
Muhammad Umar Khan
Aswin Kuganesan
Yousef Shahin

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 16th 2025	Dylan Garner	Added initial POC plan
September 17th 2025	Yousef Shahin	added confidential information, IP to protect, copyright license, team meeting plan, team communication plan, and team member roles sections
September 18th 2025	Umar Khan	Added Workflow Plan and Project Decomposition and Scheduling
September 21st 2025	Umar Khan	Added Expected Technology and Coding Standard
September 21st 2025	Dylan Garner	Updated POC plan with more implementation details
September 22nd 2025	Aswin Kuganesan	Added Team Charter in Appendix

This document presents the development plan for the capstone project *Computational Puzzle-solving the Ancient Textual History of Indian Buddhism.* The plan outlines the organizational and technical strategies our team will follow throughout the project. It includes details on confidentiality and intellectual property considerations, team meeting and communication plans, member roles, workflow, and project scheduling. In addition, it describes our proof-of-concept demonstration strategy, expected technologies, and coding standards. The appendix provides space for reflection and the team charter, which defines expectations, goals, and processes for effective collaboration. Together, these sections serve as a roadmap for ensuring the project is completed in a structured, transparent, and successful manner.

1 Confidential Information?

The image fragments used to train the machine learning models in this project will be treated as confidential. The team has made a verbal agreement not to make the images of the fragments public, although no written agreement will be signed.

2 IP to Protect

Since the team will be using a General Public License (GPL), there will be no IP to protect in this project.

3 Copyright License

For this project, we will be using a General Public License (GPL) which will allow users to copy and modify the software. This will allow for greater collaboration and sharing of ideas within the open-source community.

4 Team Meeting Plan

The team will meet weekly in person on campus. If conflicts arise, the meeting will be held virtually. Meetings will be planned in advance by the organizer and led by the chair. During these meetings, the team will discuss progress, issues, and next steps.

The team will also meet with the industry advisor biweekly, in person, with the option to meet more frequently if needed. These meetings will last 30–60 minutes, be planned by the organizer, and led by the chair. During advisor meetings, the team will ask questions for clarification and receive feedback on project progress.

5 Team Communication Plan

The team will use Discord as the primary platform for communication, including day-to-day discussion, coordination, and sharing of resources. For urgent matters, team members will use SMS messaging to ensure timely responses. Communication with the industry advisor will be conducted through email to maintain a professional channel for updates, questions, and meeting arrangements. In addition, GitHub Issues will be used to track project-related tasks, bugs, and feature requests, ensuring transparency and accountability in the development process.

6 Team Member Roles

Our team will adopt five rotating roles to help structure collaboration and ensure accountability. While specific members will not be permanently assigned, these roles will be shared and will rotate on a biweekly basis to balance responsibilities throughout the project.

- **Leader** – Oversees the overall project, monitors progress against milestones, and ensures that all deliverables are submitted on time.
- **Organizer** – Prepares meeting agendas by identifying key discussion items and tasks that require team input.
- **Meeting Chair** – Facilitates meetings, keeping discussions on track and ensuring all agenda items are addressed.
- **Note Taker** – Records detailed notes during meetings, including decisions, action items, and deadlines, and circulates them to the team afterward.
- **Reviewer** – Reviews the team's work for accuracy, clarity, and completeness before submission or presentation.

These roles are intended to provide structure, improve efficiency, and distribute responsibility fairly among team members.

7 Workflow Plan

- **Version Control & Workflow:**
  - We will be making use of GitHub with a feature-branch workflow. Each task/issue will get its own branch.
  - All code changes require a pull request and at least one approval from a reviewer before a merge to the master branch.
    - All PRs must follow a standardized naming convention:

Issue # Name of feature

  - All PRs must also contain a standardized description outlining the details of the change:
    - What?
    - Why?
    - How?
    - Testing?
    - Screenshots (optional)
    - Anything Else
    - Refer to https://arc.net/l/quote/touaacbp
- Commit messages must contain a short description of the change being added to the feature branch.

- **Issues & Project Tracking Boards:**
  - We will be making use of a Kanban board on GitHub to track our project's progress.
  - This Kanban board will contain Issues and will overall be split up into 5 separate categories:
    - To-Do
    - In Progress
    - Pending Review
    - Reviewed
    - Done
  - Issues will be used with templates for things like bugs and tasks.
  - We will also be making use of Labels:
    - Bug, enhancement, documentation
    - Easy
    - Needs design (design discussion required)
    - Due By ...

- **Milestones:**
  - We will be creating milestones in GitHub corresponding to all major deliverables:
    - POC
    - Rev0
    - Rev1
    - Final
  - Each issue will be linked to a milestone.

- **Continuous Integration / Continuous Deployment:**

- We will be making use of GitHub Actions to trigger a CI pipeline.
- Each push and pull request will trigger:
  - Build automation:
    - Utilizing makefiles to compile all modules
    - **(ALREADY COMPLETED)** automatically build .tex files into PDFs
  - Some sort of linter for style compliance
  - Unit test suite
  - Deployment

8 Project Decomposition and Scheduling

As mentioned previously, we will be making use of a Kanban board through GitHub Projects to manage all of our tasks. Each task will be associated with an Issue and linked to milestones corresponding to all major deliverables (POC, Rev0, Rev1, Final). We plan on dividing tasks based on knowledge built through personal experiences as well as interest; however, each task will be reviewed cross-functionally to ensure shared knowledge.
Each major milestone will contain deliverables within them.

Major Milestones

1. Proof of Concept (POC)
2. Revision 0
3. Revision 1
4. Final

Deliverables from Course Outline

- Team Formed, Project Selected
- Problem Statement, POC Plan, Development Plan
- Requirements Doc and Hazard Analysis Revision 0
- V&V Plan Revision 0
- Design Document Revision -1
- Proof of Concept Demonstration
- Design Document Revision 0
- Revision 0 Demonstration

- V&V Report and Extras Revision 0
- Final Demonstration (Revision 1)
- EXPO Demonstration
- Final Documentation (Revision 1)

Task Decomposition

We plan on breaking down all tasks into features that can be completed in a few hours of work rather than days.

Some examples would be:

- Design database schema draft
- Implement orientation correction function for fragments
- Write test plan for OCR module for Sanskrit Input

We want to make sure that the tasks (Issues on GitHub) we create are specific and not open to much ambiguity. We also want to ensure that they are easy to test and estimated in hours. During our weekly meetings, we will be identifying large chunks of work and decomposing them into digestible chunks. This will be completed through discussion and trying to gain an understanding of the task at hand.

Assigning Work

Initially, task assignment will be based on interest and strengths. However, as we move on, we plan on rotating tasks to prevent overspecialization and to ensure everyone has a chance to explore areas of interest. Each task will have both an assignee and a reviewer.

Additionally, each team member will be expected to commit approximately 10 hours per week of work dedicated to this project. This includes meetings with peers or professors, tutorial time, and actual development time.

9 Proof of Concept Demonstration Plan

9.1 Main Risks for Project Success

The most significant risks for our Buddhist manuscript fragment reconstruction platform are:

- **Inconsistent OCR Performance**: OCR accuracy varies dramatically across different manuscript conditions, script styles, and image quality. Ancient Buddhist texts often contain deteriorated characters, non-standard orthography, and multiple script variations that challenge standard OCR engines.

- **Lack of Positive Training Examples**: The absence of confirmed fragment matches creates a training data challenge. Without verified examples of fragments that belong together, we will need to explore unsupervised learning approaches and develop alternative validation strategies for our matching algorithms.

These are risks because they directly impact the core functionality of our system - the ability to accurately identify which fragments belong together.

9.2 Implementation Challenges

The most challenging aspects of implementation will be:

- **OCR Integration and Consistency**: Implementing OCR engines that can handle Sanskrit manuscript variations while maintaining consistent output across different image conditions and script styles.
- **AI Model Training Without Ground Truth**: Creating and training machine learning models for fragment similarity matching without access to verified positive examples of matched fragments. The final model will need to combine multiple features (OCR text, edge patterns, damage signatures) into confidence scores, which presents challenges for training validation. We plan to explore unsupervised learning approaches to address this challenge.
- **Interactive Canvas**: Building a responsive, intuitive drag-and-drop interface that can handle potentially hundreds of fragment images while maintaining performance.
- **Multi-scale Matching**: Implementing algorithms that can suggest matches at different confidence levels.

9.3 Testing Difficulties

Testing will present unique challenges because:

- **Ground Truth**: We need access to known fragment matches from Buddhist Studies scholars to validate our algorithms, which may be limited initially.
- **Subjective Validation**: Fragment matching often involves scholarly interpretation, making automated testing metrics challenging to define.
- **Performance Testing**: Testing the system with large batches of high-resolution fragment images will require substantial computational resources.
- **User Experience Testing**: The interface must be intuitive for scholars with varying technical backgrounds, requiring extensive usability testing.

9.4 Library and Technology Risks

9.5 Hardware and Infrastructure Concerns

- **Processing Power**: Computer vision and AI algorithms may require substantial CPU/GPU resources for model training.

9.6 Demonstration Plan

To address these risks and demonstrate feasibility, our POC will focus on proving that the core technical challenges can be overcome:

- **OCR Functionality Demonstration**: Show OCR engines successfully extracting text from sample manuscript fragments with varying quality levels. We will demonstrate text extraction from both clear and degraded fragments to validate OCR consistency approaches.
- **Image Segmentation and Normalization**: Demonstrate fragment segmentation algorithms that can isolate individual fragments from composite images and normalize them for consistent analysis (rotation correction, standardization).
- **Basic Feature Extraction**: Show extraction of key features from fragments including edges, text content, and damage patterns that could be used for matching.
- **Prototype Similarity Metrics**: Implement basic similarity algorithms that compare normalized fragments and provide confidence scores, acknowledging the limitation that it will be difficult to validate accuracy without ground truth data.
- **Interactive Workspace**: Create a basic interface where users can upload fragments, view OCR results, and see preliminary similarity suggestions.

This demonstration will prove the technical feasibility of the core components. The POC focuses on demonstrating that our technical approach can extract and compare the right types of features for fragment matching. While the absence of ground truth data presents challenges, we will explore unsupervised learning techniques and develop validation approaches that can work with the available data.

10 Expected Technology

For our project, we will use a combination of programming languages, external libraries, dev tools and infrastructure for fragment processing, matching and analysis. While the exact implementation has not been finalized yet and may evolve, the following technologies are expected to be used:

Programming Languages

- Python – primary language for image preprocessing, data pipelines and model integration
- C++ (if required) – will be utilized for performance-critical modules
- JavaScript / TypeScript – for frontend development

Libraries & Frameworks

- OpenCV for image preprocessing, orientation correction, and edge detection
- PyTorch / TensorFlow for machine learning models for script identification and OCR
- scikit-learn for probabilistic fragment-matching algorithms
- SQLAlchemy for database interaction (PostgreSQL or SQLite)
- React for frontend framework to build the researcher-facing interface
- Tailwind CSS for frontend styling libraries
- Flask, FastAPI or Django for backend API layer to connect frontend and ML/database modules

Pre-trained Models

- We will be exploring OCR models for Sanskrit
- We plan on training a lightweight model in the case that pre-trained models are not sufficient

IDE & Dev Tools

- VSCode as the primary development environment
- LaTeX/Markdown for documentation and deliverables
- Doxygen for auto-generating API documentation
- Postman to test backend endpoints

Testing and Code Quality Tools

- Pytest for backend unit testing
  - Can utilize coverage.py to generate coverage reports
- Combination of MyPy and Ruff for Python linting
- Jest for frontend testing
- ESLint and Prettier for frontend linting
- Google C++ Style Guide if we develop modules in C++
- Valgrind for C++ performance if we develop modules in C++

Build & Automation Tools

- Makefiles for automated builds of modules and LaTeX documents
- GitHub Actions as our CI/CD pipeline:
  - Linting and style checks for frontend and backend
  - Unit and integration tests
  - Docker image builds
  - PDF generation from LaTeX docs

Deployment

- Docker for containerized environments
- Git and GitHub projects for version control and project management

11 Coding Standard

The most important thing that we will be striving for is consistency throughout our project. This can only be possible by highlighting a clear set of rules and standards for development. These rules and standards have been listed below.

General Rules

- Code reviews: 1+ review approval required before merge
  - Approval depends on style as well as correctness of code
- Commits and PRs must follow the standardized template established in the workflow plan
- No sensitive data will be committed to the repository

  - Credentials
  - Datasets with restrictions (fragment images)
- Decisions regarding style, tech stack and others may evolve over time but must be documented in this document (Development Plan)

Python Rules and Standards

- Follow PEP8 guides for formatting and structure
- Use Ruff for linting and rule enforcement
- MyPy for typechecking
- Write NumPy/SciPy-style docstrings for public functions, including data shapes and types
- Implement tests with Pytest and measure coverage with coverage.py
  - Aim to reach 80% coverage

JavaScript/TypeScript Rules and Standards

- Follow Google JavaScript Style Guide with React/JSX convention
- Use ESLint for static analysis and Prettier for consistent formatting
- Standardize styling with Tailwind CSS
- Test frontend code with Jest and aim to reach 80% coverage

API and Documentation Standards

- Define request and response schemas using `pydantic` models and TypeScript interfaces
- Use proper HTTP status codes as well as versioned endpoints
- Use consistent routing for endpoints
- Generate API documentation

CI Enforcement

- Run Ruff, MyPy and Pytest for backend code
- Run ESLint, Prettier and Jest for frontend code
- Block merges on failed lint, typecheck or test results

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

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

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

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

A development plan provides structure and direction before we begin implementation. For a complex and specific project like ours, it ensures that all team members are on the same page and share the same understanding of workflow (GitHub branching, issue tracking, CI/CD, PRs) and milestones. This also helps the team keep on the same page on the project as a whole. That's why having this plan up front reduces ambiguity, prevents duplicated or conflicting work. This also helps keep us accountable as the rules for this project have been decided pretty early on. It also makes it easier to identify risks and dependencies early, so we can adjust scope if needed rather than mid-way through development. Having this as a rough blueprint for the project helps everyone stay on track and should hopefully translate to better team work and dynamic in the future.

The main advantage I find with CI/CD is the fact that everyone can see your work and that you can continuously get feedback and iterate on your work. This is great as the work continuously gets better until it is good enough for submission. Another aspect that is great is that your work is available to everyone so that way you can always see if the team is on track to finish on time or not and you can always check other people's work to give them feedback. It also makes it easier to have a final good version since you can always use a PR for that. With docs I feel like it can get a little complicated with that. Also having different branches is a huge plus to separate the work for each person or for each category of work. The biggest downside I find is that this only works

if everyone does it which isn't always the case. If someone doesn't use it often or update often then it could get a little frustrating especially since you can't see how the actual progress of the project is going so far.

Honestly we haven't had many disagreements so far. Everything has been laid out for us since this is a project from the project list. The biggest pro from that is that we don't have to make up anything ourselves since everything is laid out for us. Also so far everyone has been pretty fair so whenever we come up with a rule to make things as fair as possible everyone has agreed. Everyone has also been responsive to being accountable to their work so again when we make a rule to hold everyone more accountable everyone has agreed. So far we haven't had any disagreement though surely some will pop up later in the project. That is just the way things tend to get. And if no conflicts then for the better.

Appendix — Team Charter

External Goals

Our project will contribute to the field of religious studies by providing valuable manuscripts that have been lost due to the fragmentation of the texts. We will also develop our knowledge of machine learning and computer vision through this project which we will helpful for our future professional development. Our tool will be able to be used by anyone studying religious studies so that they can create their own texts, which will lower the barrier required to provide valuable information to the field.

Attendance

Expectations

Our team expects all team members to arrive on time for team meetings unless they are late due to an acceptable excuse. All team members must be present for the entire meeting unless they have an reason for leaving early. Team members are expected to be prepared for the meeting and participate in the discussions to provide their viewpoint on the topics. Team members are expected to attend every meeting unless an acceptable excuse is given.

Acceptable Excuse

An acceptable excuse for missing meetings or deadlines includes family emergencies such as a death in the family or serious illness. Medical emergencies such as illness or doctor appointments will also be acceptable. If there is a conflicting class or midterm/exam taking place during the meeting, the team member will be allowed to attend the event, but we will try to avoid planning meetings during these events.

Poor time management or missing meetings due to other commitments will not be tolerated, and team members are expected to notify the entire team about their reason for missing the meeting as soon as possible so that the team can plan in advance.

In Case of Emergency

In an emergency situation, team members must notify the team and provide the emergency details. They must coordinate with the team to fill the gap in coverage so that the work will be able to finished. They must let the team know the approximate length of time required for the emergency, or keep the team updated during the emergency so that they can plan ahead if necessary. If possible, provide any work that is in progress to the team so that they can continue working on it. If this emergency happens over a long period of time, we will notify the instructor.

Accountability and Teamwork

Quality

All team members must come to meetings with meeting topics and questions prepared, and all necessary tasks must be completed prior to the meeting. All team members must also read the agenda and meeting notes. Our documentation must satisfy the course requirements and be easy to understand, so that we can easily reference it. We aim for our code to have high readability and maintainability. Furthermore, our code will be tested rigourously through unit testing for basic coverage and regression testing to ensure that the new code does not cause any defects, and all code will be code reviewed to reduce technical debt. We will also ensure that our deliverables are cited properly and peer reviewed.

Attitude

Our team must communicate in a respect manner with each other, and include everyone in group discussions. We will work collaborately by providing feedback to improve the quality of the work and through knowledge sharing. We will also focus on having a positive attitude when dealing with problems and we will be receptive to new ideas to maximize our creative potential.

Code of Conduct:

- Effectively communicate and collaborate with each other
- Be respectful to all team members
- Avoid using personal attacks or harsh language
- Violating the code will result in further consequences

Stay on Track

We will incentivize good performance by keeping track of metrics such as the number of pull requests, so team members feel more motivated to increase their number of pull requests. Furthermore, we will keep track of how team members are performing outside of the milestones which includes team member roles such as creating the agenda and meeting minutes. Some performance targets includes participation in all meetings, completion of tasks before the deadline and participation in code reviewing and giving feedback. If any team members are underperforming, we will first discuss this in a meeting to ensure that they are aware that they are not performing well. If their performance does not improve, we will bring this issue up in a TA meeting so that the TA can give feedback to the team member. If the team member is still not performing well, we will get the instructor involved in the discussion. Finishing your tasks early will give team members the opportunity to provide more feedback to others, and they will be able to get started on the next milestone earlier.

Team Building

We will build team cohesion from establishing a friendly team environment by discussing topics unrelated to the project occasionally and by getting to know each other. Furthermore, we will acknowledge each other accomplishments and promote working together on problems to build team chemistry. Team members will be encouraged to spend time together outside of time spent working on the project. Also, since we are enrolled in the same courses, team members will be encouraged to collaborate on team projects or labs in other courses.

Decision Making

Since our group has 5 team members, we are able to make decisions based on majority vote. The team leader will be responsible for mediating during disagreements, and we will keep voting anonymous to avoid bias affecting the vote. Before the vote, we will have a discussion involving all group members, and we will not be able to vote if all team members are not present, to prevent team members from being left out. All team members will be able to bring up topics for decision making, but the team leader will be responsible for when the vote will be held to avoid spending too much or too little time making a decision.