**CS673 Software Engineering (PAT)**

**Team 5 - Fafi**

**Project Proposal and Planning**

| **Team Member** | **Role(s)** | **Signature** | **Date** |
| --- | --- | --- | --- |
|  |  |  |  |
| **Matthew Kluska** | Overall team leader | *Matthew Kluska* | 09/10/2022 |
| **Aidan Chang** | Design and Implementation leader | *Aidan Chang* | 09/10/2022 |
| **Derric Syme** | Configuration leader | *Derric Syme* | 09/10/2022 |
| **Brendan Truong** | QA leader | *Brendan Truong* | 09/10/2022 |
| **Patounezambo OUEDRAOGO** | Requirement leader | ***Patounezambo*** | 09/10/2022 |
| **Zengrui Luo** | Security leader | *Zengrui Luo* | 09/10/2022 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1** | **Patounezambo OUEDRAOGO** | **9/12/2022** | - |
| **2** | **Patounezambo Ouedraogo** | **9/26/2022** | Main functionality change from cat identification to facial identification |
| **3** | **Derric Syme** | **10/11/2022** | Updated Configuration Management Plan to reflect new git pulling strategy |
| **4** | **Patounezambo Ouedraogo** | **10/11/2022** | No major requirement change. Only implementation of iteration 2 requirements and features defined in the user stories |
| **5** | **Derric Syme** | **10/18/2022** | Quality Assurance metrics simplified. Basic syntax/grammar correction. |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# Overview

* Our project is about the creation of an face recognition application leveraging technologies such as PyTorch for a ML framework, and providing user experience through a Flask web framework. Specifically, we are working on a face detection web application called **Fafi,** which stands for **Face Finder**.
* Computers’ ability to accurately recognize images will play a key role in many sectors of innovation. For instance, many tech companies are determined to hasten the day when self-driving cars are ubiquitous. This will only be possible if cars can avoid being involved in accidents by accurately identifying images of people, animals and objects such as other cars, etc. Similarly the vast amount of funds that are being spent on robot development with the aim of automating many tasks in various fields such as business, defense, etc., will only be worth it if those robots can successfully make “intelligent” decisions based on accurate recognition of various forms of images.
* Potential users of our face recognition software could use it to detect a person’s face on a video feed for example. Additionally, other developers can also use our code and expand its use case to include more images for diverse purposes. For example, our framework can be repurposed to quickly plug in other ML models and provide a front end framework for users of diverse services.
* We plan to use Python libraries in our tech stack. Flask to create a front end, and Pytorch to utilize pretrained models for identifying faces. Git and Github will be used for version control. Potentially, Github Actions for automation, and AWS services such as EC2, Cloudwatch, and Elastic Beanstalk will be used to deploy our application.

# Related Work

* **Amazon Rekognition** (<https://aws.amazon.com/rekognition/> )

Amazon Rekognition allows for easily adding image and video analysis to applications. It has the ability to identify the “objects, people, text, scenes, and activities, or any inappropriate content from an image or video”. Our program on the other hand will for now be limited to identifying cats with the potential for other elements in the future when expanded.

* **Syte**

Syte is described as being the “world’s first product discovery platform”. Syte leverages AI to revolutionize ecommerce by allowing retail businesses and brands to seamlessly connect with their products, thus driving conversion rates up in sales([www.syte.ai](http://www.syte.ai)). Our software is similar to Syte but it differs in that it is a mini version of it.

# Proposed High level Requirements

1. **Front-end website**
2. Essential Features:
   1. The website consists of three tabs:
      1. A **Main Tab**: This tab is where an end user can upload and submit an image from their local machine. Along with radio buttons on the correct category of the image (person’s face or not a person’s face). Once submitted, after a potential loading screen, the user will then be alerted on the ML model’s results and statistics comparing it against the radio button selection.
      2. A **History Tab**: This tab is where we will display historic submission events which consist of reduced size images, the prediction, and an identifier correlating to a correct or incorrect prediction.
      3. A **Notification Tab**: This is where a user can input their email, and toggle email alerts when a face is identified.
3. Desirable Features:
   1. The website can access the webcam to capture live photos.
   2. User account with SSO, MFA, and other security features.
   3. Filters for searching and historical management in the history tab.
4. Optional Features:
   1. Build the app on phone applications such as Android and ios to access phone’s cameras
   2. Have the WebUI be mobile browser friendly and retain a good experience cross platform.
   3. The ability to upload and predict on a directory of images or multiple images.
5. **ML Algorithm**
6. Essential Features:
   1. We will be utilizing a pre-trained face detection model called Yolov5, implemented by pytorch.
   2. Returns confidence rate alone with predictions.
7. Desirable Features:
8. Enable a flexible framework where we can easily substitute the ML model and UI text to accommodate for other identifiers (interchangeably identify animals or objects)
   1. This could be configurable through a file configuration or by highlighting key directions to fork and replace essential modules.
9. **Nonfunctional Requirements**

* **Usability requirement** User-friendly User Interface
* **Reliability requirement** The application needs to be reliable at >95%
* **Compatibility requirement**The application takes live video from the default webcam.
* **Responsiveness requirement**Responsive UI (That is, the web UI should be responsive and intuitive to user interaction)
* **Security requirement**The application should be secure especially using back-end security measures such that no unauthorized data type can be used as input. Similarly, unauthorized users should not have access to the service.
* **Delivery Requirement**The software should be ready for use and delivered by the deadline
* **Implementation Requirement**The software should be be written in Python: Flask, Pytorch

# Management Plan

1. Objectives and Priorities
   1. Identify the ML model and test it in local environment
   2. Establish APIs and set up back-end server
   3. Establish front-end architecture and connects the ML API
   4. Fine tune the system and ensure highest reliability is achieved
2. Risk Management
   1. The main risks in this week are:
      1. Harmonize all codebase to run seamless across all platforms
      2. Integrate security with the login system
      3. Duplicate work by team members working on the same functionality using different technologies.

To manage those risks, we’ll have meetings of team members that work on the same functionality in order to come to agreement on the framework and implementation approach that will be integrated in the final project.

.

* 1. [**Risk Management Sheet Link:**](https://docs.google.com/spreadsheets/d/1pPxfI9J5PbmwEcyH4_ezYLgKswIMzvFdm_CG_4NBFPs/edit#gid=0)

1. Timeline

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | Main tab | Implement an interface for image submission | 35 hours |
| 2 | History tab | To be expressed in future iterations | 35 hours |
| 3 | ML algorithm | Implement pre-trained model | 40 hours |
| 4 | Login functionality | Implement an interface for login | 10 hours |

# **Configuration Management Plan** -

1. **Tools**

For version control, our team will be using Git and Github.

For IDE tools, our team will flexibly use either a Python IDE (i.e. PyCharm, Spyder) or a text editor (Visual Studio Code) of choice depending on each member’s personal experience and preference.

For CI/CD automation, our team has configurations for Github Actions to automate testing and linting of the code. Currently this is disabled due to private repository limitations.

For DevOps, our team will be using Pivotal Tracker to manage an Agile methodology project management approach.

1. **Code Commit Guideline and Git Branching Strategy**

Our branching strategy of choice is Github Flow, as it is ideal for our use case. As this project spans only six weeks and is rather small in terms of development team size, Github Flow lets us simply create feature branches in quick succession as we develop our application. To avoid pulling directly into our Main branch, we have created a Development branch to directly merge our progress into. We then pull that Dev branch into Main weekly, supply continuous CI/CD, and maintain our code in a constantly improving state.

A feature branch should be created for each new feature. That feature branch can be pushed to Development, and a pull request to merge this dev branch into main should be opened when the feature is complete, or weekly. At least one team member should approve a pull request before it is merged, but the entire time is encouraged to review for major changes.

A pull request template will be created to add documented descriptions, testing notes, and checklist best practices.

Each commit should be reviewed by 2 team members prior to merging. When impending deadlines demand it, a single team member may review and approve a merge request, but this should be avoided.

## Deployment Plan if applicable

Our minimal goal would be to develop our web framework with Flask and have it functional, fully tested, and performing to standard locally.

Our stretch goal will be to use AWS Elastic Beanstalk to deploy our application onto an AWS EC2 instance, set up security groups for inbound traffic, configure load balancers to distribute requests, store our code on AWS S3, and set up CloudWatch alarms to monitor the load of the instance, and configure an auto scaling group in case of instance failure/termination.

# **Quality Assurance Plan** -

## Metrics

| Metric Name | Description |
| --- | --- |
| # of Test Cases | The amount of test cases. |
| Test Case Pass Rate | The percentage of test cases that pass. |
| Test Code Coverage | The percentage of our code that our test cases cover. Ideally we would have tests that check all our methods. |
| Confidence Rate | How confident (%) the algorithm is that the presented image contains a person. |
| Accuracy | How accurate our detection model is. |
| Cost | The amount of person hours invested by the team. |

* 1. **Coding Standard**

Our code will follow the PEP 8 coding convention. (<https://peps.python.org/pep-0008/>)

This coding standard is enforced with a pre-commit configuration that enforces flake8 styling guidelines.

## Code Review Process

Pull requests will be used for the code review. A pull request must be opened and will be required to merge changes to our main branch. At Least two peers must review a pull request before it can be merged (enforced by the repository setting). All team members will be involved in the code review process.

Pull requests are templated with sections for description, type of change, any testing notes, and a checklist of coding practices to review. This will serve to help in documentation of the project along with enforcing best practices.

## Testing

A collection of unit tests can be integrated for feature testing.

The QA leader will provide special focus in integration testing of front end UI and the backend along with performance testing of the end to end application from a user experience perspective.

The test suite can be automated through Github Actions, to run on each Pull Request and ensure both code coverage and pass percentages are met.

## Defect Management

The team will use GitHub issues for defect management. Issues will be raised whenever a team member discovers a defect or potential enhancement in their work. This may occur during the development of a feature or during testing. If the issue is minor, it will be raised and left until there is capacity to address the issue, or until the next iteration. If the issue is major and requires immediate attention, it will be taken on by the team based on capacity.

Types of defects may include arithmetic defects in our ML model, logic defects in our code, user interface defects in our front end UI, or performance defects.

The QA leader will be in charge of reviewing any issues, measuring urgency, and solutioning.

Issue requests are templated to remind users, to detail the type of bug, what the issue pertains to, steps on how to reproduce the defect, any alternative expected behaviors, screenshots, and additional context.

# References

Mishra, Abhishek. “Machine Learning in the AWS Cloud: Add Intelligence to Applications with Amazon Sagemaker and Amazon Rekognition.” *Amazon*, John Wiley & Sons, 2019, https://aws.amazon.com/rekognition/.

“The World's First Product Discovery Platform.” *Syte*, 4 May 2022, https://www.syte.ai/.

“GitHub Flow.” *GitHub Docs*, https://docs.github.com/en/get-started/quickstart/github-flow.

“Python Enhancement Proposals.” *PEP 8 – Style Guide for Python Code*, https://peps.python.org/pep-0008/.

# Glossary

Fafi: Face Finder

AWS: Amazon Web Services

QA: Quality Assurance

ML: Machine Learning

UI: User Interface

CI/CD: Continuous Integration/Continuous Deployment

IDE: Integrated Development Environment