|  |  |
| --- | --- |
|  | Feasibility Study and Risk Assessment |
|  |  |
|  | Natalia Palej |

A00279259  
Software Design with Artificial Intelligence for Cloud Computing  
Year 4

Contents

[1 Feasibility Study 2](#_Toc183021163)

[1.1 Technical Feasibility 2](#_Toc183021164)

[1.1.1 Evaluation 2](#_Toc183021165)

[1.1.2 Conclusion 2](#_Toc183021166)

[1.2 Operational Feasibility 2](#_Toc183021167)

[1.2.1 Evaluation 2](#_Toc183021168)

[1.2.2 Conclusion 2](#_Toc183021169)

[1.3 Economic Feasibility 3](#_Toc183021170)

[1.3.1 Evaluation 3](#_Toc183021171)

[1.3.2 Conclusion 3](#_Toc183021172)

[1.4 Scheduling Feasibility 3](#_Toc183021173)

[1.4.1 Evaluation 3](#_Toc183021174)

[1.4.2 Conclusion 3](#_Toc183021175)

[2 Risk Assessment 3](#_Toc183021176)

[2.1 Risk Identification 3](#_Toc183021177)

[2.2 Risk Analysis 4](#_Toc183021178)

[2.3 Risk Mitigation Plan 4](#_Toc183021179)

[3 References 5](#_Toc183021180)

# Feasibility Study

## Technical Feasibility

### Evaluation

|  |  |  |
| --- | --- | --- |
| **Resource** | **Function** | **Description** |
| **Python** | Programming Language | Primary language used for development, ensuring compatibility with tools like PyTorch and Flask |
| **PyTorch** | Deep Learning | Framework used for training and deploying YOLOv8 models |
| **YOLOv8** | Object Detection | Prototype has proven to work well with food datasets and supports integration with PyTorch |
| **Flask** | UI Development | Lightweight framework ideal for creating the simple web interface |
| **Groq AI API** | Recipe Generation | Reliable for generating recipes but dependent on its availability |
| **College PCs** | Hardware Resources | High GPU and memory resources available for model training |

### Conclusion

The project is technically feasible. Proof-of-concept tests have shown that using a pre-trained YOLO model works well for detecting food items. By training my own model with a larger dataset, the accuracy and diversity of detected items can be improved. Challenges in dataset preparation will be addressed by carefully labelling and augmenting the dataset as needed. The dataset will be also limited to the most common products and if time allows, it will be expanded further.

## Operational Feasibility

### Evaluation

|  |  |
| --- | --- |
| **Goal** | **Description** |
| **Main Goal** | Reduce food waste by suggesting recipes based on detected food items |
| **Easy to Use** | Designed for users with basic web application knowledge |
| **Time Efficient** | Helps users save time by providing practical and easy-to-make recipes |
| **Reliable** | Ensures consistent and accurate food detection and recipe generation |
| **Scalable** | Designed to scale easily, allowing for future improvements |

### Conclusion

The application aims to provide reliable and time-efficient solution for reducing food waste. To ensure seamless user experience, an input field will allow users to manually add any items that the image recognition model missed. These items will be combined with the detected ones and sent to the Groq API for recipe generation. While user inputs will not be automatically validated, clear instructions will be provided on how to format the entries to o get better results. The system is also designed with scalability in mind, supporting future improvements.

## Economic Feasibility

### Evaluation

The estimated cost is minimal because:

**Open-Source Tools:** All tools (YOLOv8, Flask, PyTorch, Groq AI) are free to use.

**Hardware:** Training the model will be done on college machine, eliminating hardware costs.

### Conclusion

The project remains economically feasible. However, if the college PCs become unavailable, the only alternative is to rely entirely on external AI APIs for both image recognition and recipe generation. This fallback option will be avoided unless absolutely necessary, as it increases dependency on external resources and introduces potential costs.

## Scheduling Feasibility

### Evaluation

The project schedule, detailed in the Gantt chart, allows for sufficient time for all phases, including:

* Dataset preparation and annotation
* Model training and evaluation
* Backend and frontend development
* Testing and refinement

Potential risks, such as overlapping academic responsibilities, have been considered.

### Conclusion

To ensure the project stays on track, I am following a detailed Gantt chart. Weekly meetings with my supervisor will help monitor progress, identify any deviations and address issues promptly. These check-ins also ensure that I stay focused on high-priority tasks and avoid spending too much time on less critical aspects.

# Risk Assessment

## Risk Identification

* Dataset Quality Issues
* Annotation Delays
* Groq API Downtime or Unavailability
* Hardware Limitations During Training
* Project Overlap with Academic Work
* User Errors in Image Uploads

## Risk Analysis

The following table summarizes identified risks, their likelihood, impact and corresponding mitigation strategies:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Likelihood** | **Impact** | **Mitigation Strategy** |
| Dataset Quality Issues | Medium | High | Use multiple datasets, perform data augmentation to improve dataset diversity |
| Annotation Delays | High | Medium | Focus on key food items first, automate annotation by using tools like MakeSense, use pre-annotated datasets |
| Groq API Downtime or Unavailability | Medium | High | Use ChatGPT API as a fallback for recipe generation |
| Hardware Limitations During Training | Low | High | Optimize the model, train in batches to reduce GPU/memory load |
| Project Overlap with Academic Work | High | Medium | Stick to the Gantt chart, prioritize critical tasks |
| User Errors in Image Uploads | Medium | Low | Include image preview, validation and clear instructions |

## Risk Mitigation Plan

|  |  |
| --- | --- |
| **Risk** | **Strategy** |
| **Annotation Delays** | - Use pre-annotated, open-source datasets like Roboflow [[1]](#_References), Labelbox [[2]](#_References) or Kaggle [[3]](#_References)  - Focus on most common food products to ensure the application is functional with smaller, high-priority dataset  - Use automation tool for annotation like MakeSense [[4]](#_References) |
| **API Dependency** | - Test alternative APIs (eg. ChatGPT) to ensure fallback option is ready and working  - Implement the logic to switch between APIs if necessary |
| **Hardware Limitations** | - Have a dedicated machine to train the model in college  - Implement the check to stop training if the validation set reaches 90%  - Implement the check to stop training if the validation set becomes lower |
| **Scheduling Risks** | - Use regular check-ins to monitor progress  - Allocate buffer time for critical phases like data annotation and testing |
| **User Errors in Image Uploads** | - Implement image preview feature, so users can confirm selected image  - Add validation check to ensure file type is correct (.jpg, .png)  - Include clear and easy to follow instruction to guide users through the application |

# References

1. Roboflow. "Roboflow: Streamline Computer Vision Projects." Available: <https://roboflow.com>
2. Labelbox. "Labelbox: The Leading Training Data Platform." Available: <https://labelbox.com>
3. Kaggle. "Kaggle: Your Home for Data Science." Available: <https://www.kaggle.com>
4. MakeSense. "MakeSense: An Open Annotation Tool for Computer Vision." Available: <https://makesense.ai>