**System Requirements Specification**

**D**iabetic Retinopathy

| **Version** | **Date** | **Author** | **Description** | **Approver** |
| --- | --- | --- | --- | --- |
| **1.0** | **9/10/2024** | **Guy Yarhi** | **Initial draft** |  |
| **1.1** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

1. [Introduction](#1-introduction)
   * [1.1 Purpose](#11-purpose)
   * [1.2 Scope](#12-scope)
   * [1.3 Definitions, Acronyms, and Abbreviations](#13-definitions-acronyms-and-abbreviatio)
   * [1.4 References](#14-references)
2. [Overall Description](#2-overall-description)
   * [2.1 Product Perspective](#21-product-perspective)
   * [2.2 Product Functions](#22-product-functions)
   * [2.3 User Characteristics](#23-user-characteristics)
   * [2.4 Constraints](#24-constraints)
   * [2.5 Assumptions and Dependencies](#25-assumptions-and-dependencies)
3. [System Features and Requirements](#3-system-features-and-requirements)
   * [3.1 Functional Requirements](#31-functional-requirements)
   * [3.2 Non-Functional Requirements](#32-non-functional-requirements)
   * [3.3 External Interface Requirements](#33-external-interface-requirements)
4. [System Design and Architecture](#4-system-design-and-architecture)
   * [4.1 Architectural Design](#41-architectural-design)
   * [4.2 Module Design](#42-module-design)
5. [Development and Implementation](#5-development-and-implementation)
   * [5.1 Technology Stack](#51-technology-stack)
   * [5.2 Development Tools](#52-development-tools)
   * [5.3 Coding Standards](#53-coding-standards)
6. [Deep Learning Process](#6-deep-learning-process)
   * [6.1 Data Collection and Preprocessing](#61-data-collection-and-preprocessing)
   * [6.2 Model Selection](#62-model-selection)
   * [6.3 Training and Evaluation](#63-training-and-evaluation)
   * [6.4 Deployment](#64-deployment)
7. [Extension and Scalability](#7-extension-and-scalability)
   * [7.1 Adding New Disease Models](#71-adding-new-disease-models)
   * [7.2 API Integration](#72-api-integration)
8. [Appendix](#8-appendix)
   * [8.1 Timeline](#81-timeline)
   * [8.2 Task List](#82-task-list)
9. [Glossary](#9-glossary)

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the system requirements for an image analysis application designed to detect and classify eye diseases using machine learning models. The application aims to facilitate early diagnosis of conditions such as diabetic retinopathy by analyzing eye images. The system is intended to be developed in Python, managed by Poetry, and later deployed on HuggingFace for broader accessibility.

Based on: <https://www.kaggle.com/datasets/benjaminwarner/resized-2015-2019-blindness-detection-images>

**1.2 Scope**

This system will initially focus on diagnosing diabetic retinopathy using image classification techniques. It will employ deep learning models like ResNet-18 or custom neural networks to classify images into multiple severity levels. The architecture will adhere to SOLID principles to ensure maintainability and extensibility, allowing for the integration of additional disease models in the future.

**1.3 Definitions, Acronyms, and Abbreviations**

* **API**: Application Programming Interface
* **DR**: Diabetic Retinopathy
* **SOLID**: Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion principles
* **ResNet-18**: A convolutional neural network architecture
* **HuggingFace**: A platform for deploying machine learning models
* **APTOS**: Asia Pacific Tele-Ophthalmology Society

**1.4 References**

* **2015 Diabetic Retinopathy Detection Competition Dataset:**
  + **This dataset is likely the one from the Kaggle competition for diabetic retinopathy detection, which contains labeled images of retina scans. This is a reference to the dataset being used in the project.**
* **APTOS 2019 Blindness Detection Competition Dataset:**
  + **This is a dataset from the APTOS competition, also from Kaggle, which focuses on detecting diabetic retinopathy or blindness from eye images. This dataset is referenced similarly to the first one, indicating it will be used for training the model.**
* **ilovescience's image preprocessing code:**
  + **This refers to a source of image preprocessing code that someone (presumably a contributor or user named "ilovescience") wrote. This code would handle the preprocessing of the images (resizing, normalizing, or augmenting them), which would be important before feeding them into a model.**
  + **The reference implies the use of a specific set of preprocessing routines or code that ilovescience developed or published, possibly available on platforms like GitHub or Kaggle.**

**2. Overall Description**

**2.1 Product Perspective**

The application will serve as a diagnostic tool for ophthalmologists and healthcare providers. It will analyze eye images to detect and classify the severity of diabetic retinopathy, with potential extensions to other eye diseases. The system will be developed using Python, leveraging deep learning frameworks and managed with Poetry for dependency management.

**2.2 Product Functions**

* **Image Upload via API**: Receive eye images for analysis.
* **Preprocessing**: Resize and crop images to standard dimensions.
* **Classification**: Use deep learning models to classify images into severity levels.
* **Result Output**: Provide classification results with confidence scores.
* **Extensibility**: Add new disease models without altering the core system.

**2.3 User Characteristics**

* **Primary Users**: Healthcare professionals (ophthalmologists, technicians)
* **Secondary Users**: System administrators and developers
* **Technical Proficiency**: Users should have a basic understanding of API usage and machine learning concepts.

**2.4 Constraints**

* Adherence to SOLID principles for maintainability.
* Limited to command-line interface initially; no UI development in the current phase.
* Timeframe: Development and testing within two months (9 Oct 2024 - 9 Dec 2024).

**2.5 Assumptions and Dependencies**

* Availability of high-quality, labeled datasets for training and testing.
* Access to computational resources for model training and deployment.
* Dependency on external libraries and frameworks managed via Poetry.

**3. System Features and Requirements**

**3.1 Functional Requirements**

1. **Image Upload API**
   * Allow users to send eye images through a RESTful API.
   * Validate image formats and sizes.
2. **Image Preprocessing**
   * Resize images to a maximum of 1024px.
   * Crop images as per preprocessing scripts.
3. **Disease Classification**
   * Implement classification using ResNet-18 or a custom neural network.
   * Classify images into predefined severity levels (0-4).
4. **Result Delivery**
   * Return classification results with confidence scores via API.
   * Log processing details for audit purposes.
5. **Model Management**
   * Facilitate adding new models for different diseases without impacting existing functionality.
6. **Future UI Integration (Planned)**
   * Develop a user interface for image upload and result visualization after initial testing and model validation.

**3.2 Non-Functional Requirements**

* **Performance**
  + The system should process and return results within 5 seconds per image.
* **Scalability**
  + Support concurrent API requests without degradation of performance.
* **Maintainability**
  + Codebase should follow SOLID principles to ensure ease of maintenance and extensibility.
* **Reliability**
  + Ensure high availability of the API with minimal downtime.
* **Security**
  + Implement authentication and authorization for API access.
  + Ensure data privacy and compliance with healthcare regulations.

**3.3 External Interface Requirements**

* **API Interface**
  + RESTful API endpoints for image submission and result retrieval.
* **Data Storage**
  + Store processed images and classification results in a secure database.

**4. System Design and Architecture**

**4.1 Architectural Design**

The system will follow a modular architecture adhering to SOLID principles:

* **Controller Module**: Handles API requests and responses.
* **Service Module**: Contains business logic for preprocessing and classification.
* **Model Module**: Encapsulates the deep learning models.
* **Utility Module**: Provides auxiliary functions such as logging and configuration management.

**4.2 Module Design**

* **API Controller**
  + Endpoints for image upload and result retrieval.
* **Preprocessing Service**
  + Functions for resizing and cropping images.
* **Classification Service**
  + Interfaces with the deep learning models to perform predictions.
* **Model Manager**
  + Handles loading, updating, and switching between different models.

**5. Development and Implementation**

**5.1 Technology Stack**

* **Programming Language**: Python
* **Deep Learning Framework**: PyTorch or TensorFlow
* **Dependency Management**: Poetry
* **Deployment Platform**: HuggingFace

**5.2 Development Tools**

* **Version Control**: Git
* **IDE**: VSCode or PyCharm
* **Testing Framework**: pytest

**5.3 Coding Standards**

* Implement unit and integration tests for all modules.
* Maintain comprehensive documentation for codebase.

**6. Deep Learning Process**

**6.1 Data Collection and Preprocessing**

* **Dataset Description**:
  + **2015 Diabetic Retinopathy Detection Dataset**:
    - Contains images labeled with subject ID and eye side (left/right).
    - Each image is resized and cropped to a maximum size of 1024px using ilovescience's preprocessing code.
  + **APTOS 2019 Blindness Detection Dataset**:
    - Images do not specify subject or eye side.
    - Also resized and cropped to a maximum size of 1024px.
* **Severity Levels**:
  + Each image is rated for diabetic retinopathy severity on a scale of 0 to 4:
    - **0** - No DR
    - **1** - Mild
    - **2** - Moderate
    - **3** - Severe
    - **4** - Proliferative DR
* **Data Handling**:
  + Address noise in data by augmenting images and applying normalization techniques.
  + Split data into training, validation, and testing sets, ensuring balanced representation across severity levels.
  + Manage variations due to different clinics and camera types by implementing robust preprocessing pipelines.

**6.2 Model Selection**

* Evaluate **ResNet-18** for baseline performance.
* Explore custom neural network architectures for potential improvements.

**6.3 Training and Evaluation**

* Train models using appropriate loss functions and optimization algorithms.
* Monitor training with metrics such as accuracy, precision, recall, and F1-score.
* Perform cross-validation to ensure model generalizability.

**6.4 Deployment**

* Package the trained model for deployment on HuggingFace.
* Ensure the deployment pipeline supports seamless updates and scaling.

**7. Extension and Scalability**

**7.1 Adding New Disease Models**

* Design the system to allow integration of additional models without altering existing code.
* Provide interfaces for loading and managing multiple models.

**7.2 API Integration**

* Develop API endpoints to support new functionalities as additional models are integrated.
* Ensure backward compatibility with existing API endpoints.

**8. Appendix**

**8.1 Timeline**

**Project Duration**: 9 October 2024 - 9 December 2024

| **Phase** | **Start Date** | **End Date** | **Tasks** |
| --- | --- | --- | --- |
| **Definition** | 09 Oct 2024 | 11 Oct 2024 | Define project scope, requirements gathering |
| **Design** | 12 Oct 2024 | 18 Oct 2024 | Architectural design, module specification |
| **Development** | 19 Oct 2024 | 25 Nov 2024 | Coding, model training, API development |
| **Testing** | 26 Nov 2024 | 05 Dec 2024 | Unit testing, integration testing, validation |
| **Deployment** | 06 Dec 2024 | 09 Dec 2024 | Deploy on HuggingFace, final review |

**8.2 Task List**

**Preliminary Tasks**

* **Requirement Analysis**
  + Gather detailed requirements and define use cases.
* **System Design**
  + Create architectural diagrams and module designs.
* **Setup Development Environment**
  + Configure Python environment with Poetry.
  + Initialize version control repository.

**Exploratory Tasks**

* **Data Preprocessing**
  + Implement and test image resizing and cropping.
* **Model Development**
  + Train ResNet-18 model on the dataset.
  + Experiment with custom neural network architectures.
* **API Development**
  + Develop endpoints for image upload and result retrieval.
* **Testing**
  + Write and execute unit and integration tests.
* **Deployment**
  + Package and deploy the model on HuggingFace.
  + Monitor and validate deployment.
* **Future UI Development (Post-Deployment)**
  + Design and develop a user interface for enhanced user interaction.
  + Integrate UI with existing API endpoints.

**9. Glossary**

* **ResNet-18**: A type of residual neural network with 18 layers, commonly used for image classification tasks.
* **Poetry**: A dependency management and packaging tool for Python.
* **HuggingFace**: A platform providing tools and infrastructure for deploying machine learning models.
* **APTOS**: Asia Pacific Tele-Ophthalmology Society, associated with the 2019 Blindness Detection competition.

**Future Extensions**

This system is designed with extensibility in mind. Future iterations can incorporate additional models to diagnose other eye diseases by simply integrating new models into the existing architecture. Each new model can be managed independently, ensuring that the core system remains unaffected by extensions. Additionally, the development of a user interface will enhance usability, allowing non-technical users to interact with the system seamlessly.

**Notes**

* This document serves as a foundational blueprint for the development of the image analysis application.
* Detailed technical specifications and implementation guidelines should be developed in subsequent phases.
* Regular reviews and updates to this document are recommended to ensure alignment with project progress and requirements.

**Summary of Additions**

1. **Detailed Dataset Description**: Expanded in **Section 6.1 Data Collection and Preprocessing** to include specifics about the datasets, labeling conventions, severity levels, and data handling strategies.
2. **Future UI Development**: Explicitly mentioned in **Section 3.1 Functional Requirements** and **Appendix 8.2 Task List** as a planned task post-deployment to indicate its scope as a future enhancement.
3. **Glossary Enhancement**: Added the definition for **APTOS** to provide clarity on the dataset source.

These additions ensure that the system requirements document comprehensively covers all aspects of your project as specified.