**System Design Document**

**Medical AI - Disease diagnostic tool**

<https://github.com/comp195/senior-project-spring-2023-medical-ai-disease-diagnostic-tool>

**By**

**Mansoor Haidari:** [**m\_haidari@u.pacific.edu**](mailto:m_haidari@u.pacific.edu)

**Korie Westbrook:** [**k\_westbrook@u.pacific.edu**](mailto:k_westbrook@u.pacific.edu)

**Kelvin Luk:** [**k\_luk2@u.pacific.edu**](mailto:k_luk2@u.pacific.edu)

Date of last revision to the document

**02/04/2023**

1. **Table of Contents**

Front Cover ---------------------------------------------------------------------------------------------------1

Table of Contents -------------------------------------------------------------------------------------------2

System Architecture----------------------------------------------------------------------------------------3

Hardware, Software and System Requirements-----------------------------------------------------6

External Interfaces-----------------------------------------------------------------------------------------10

Software Design --------------------------------------------------------------------------------------------11

Class Diagram ----------------------------------------------------------------------------------------------11

Class Specifications----------------------------------------------------------------------------------------11

Interaction Diagrams--------------------------------------------------------------------------------------12

Design Considerations -----------------------------------------------------------------------------------12

User Interface Design-------------------------------------------------------------------------------------13

Glossary of Terms------------------------------------------------------------------------------------------14

Reference ---------------------------------------------------------------------------------------------------16

**Project Overviews**

Medical AI is a critical tool for disease diagnosis because it can help healthcare professionals make more accurate and faster diagnoses. By using machine learning algorithms, medical AI can analyze vast amounts of medical data, including patient histories, test results, and imaging studies, to identify patterns and make predictions about the presence of specific diseases. This helps to improve the speed and accuracy of diagnoses, which can lead to earlier treatment and better health outcomes for patients. Additionally, medical AI can assist healthcare professionals in making complex diagnoses that would otherwise be challenging to identify. Overall, medical AI has the potential to significantly improve patient outcomes and reduce healthcare costs.

**System Architecture**

The high-level system architecture of the Medical AI - Disease diagnostic tool will include the following components:

1. **Software modules:**

* Image Processing: To process the medical images such as MRIs and X-rays and extract relevant information.
* Machine Learning: To develop and train the algorithms to predict diseases based on the processed images.”
* User Interface: To provide a simple and intuitive interface for users to access and use the tool.

1. **Hardware components:**

* Server: To host the software modules and process the images and predictions.
* Storage: To store the medical images and algorithms.

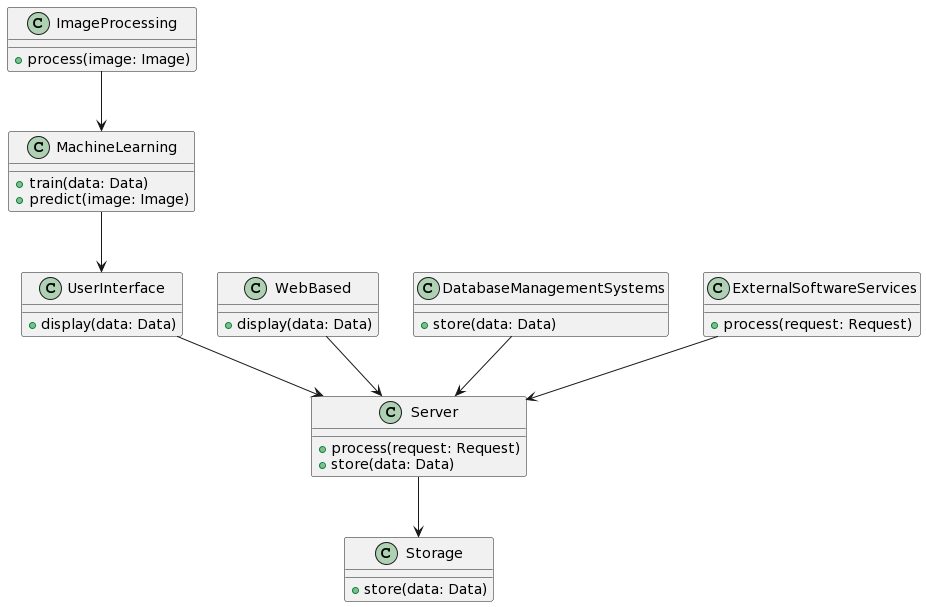
1. **User interface:**

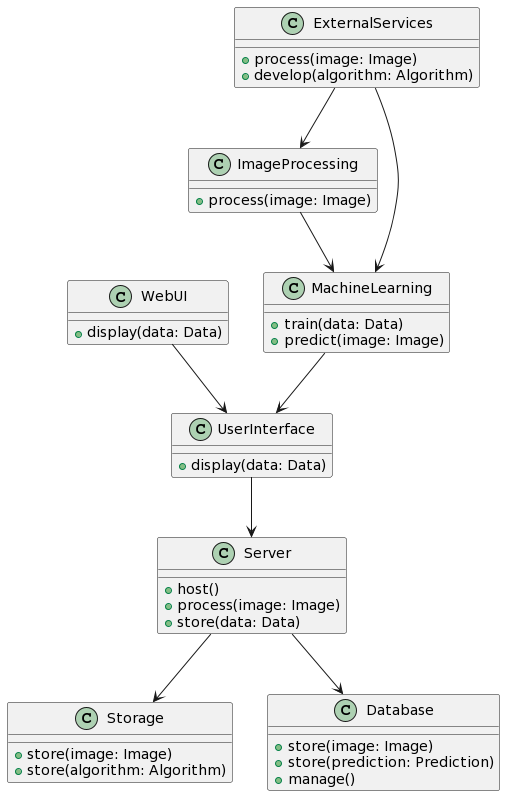
* Web-based: To provide a user-friendly interface for users to access and use the tool.

1. **Interfaces to external systems:**

* Database management systems: To store and manage the medical images and predictions.
* External software services: To access and process the medical images and use them to develop the algorithms.

The software modules and the user interface will run on the server while the storage will be separate from the server. The server will be connected to the database management system and external software services through the network.





**Hardware, Software and System Requirements**

**Hardware Requirements:**

• High-end computer with a multi-core CPU with a minimum clock speed of 2.0 GHz and 8 GB of RAM

• Graphics card with at least 2 GB of VRAM

• 1TB Hard disk drive

• 17 inch or larger display monitor

• A keyboard and mouse

**Software Requirements:**

• Python 3.x

• TensorFlow 2.x

• Keras API for building neural networks

• OpenCV for image processing

• Matplotlib for data visualization

• Sklearn for machine learning algorithms

• Jupyter Notebook for interactive programming

**System Requirements:**

• Windows 10, MacOS High Sierra or later, or a recent version of Linux

• Internet connection for downloading libraries and accessing Kaggle data

**APIs:**

• Google Maps API for geolocation and mapping

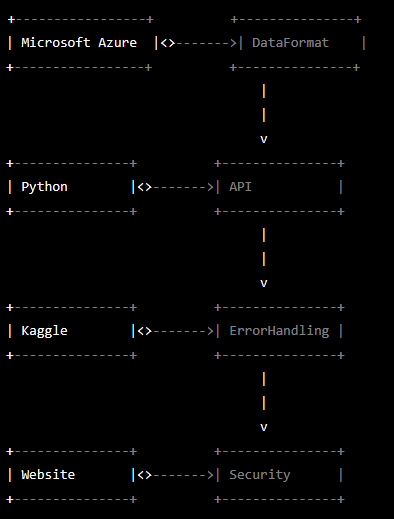
• Kaggle API for accessing data sets

**Comment:** The specific requirements may change based on the size and complexity of the data set being used and the model being built.

**External Interfaces**

This section describes the communication interfaces between the system and external components. It include the following details:

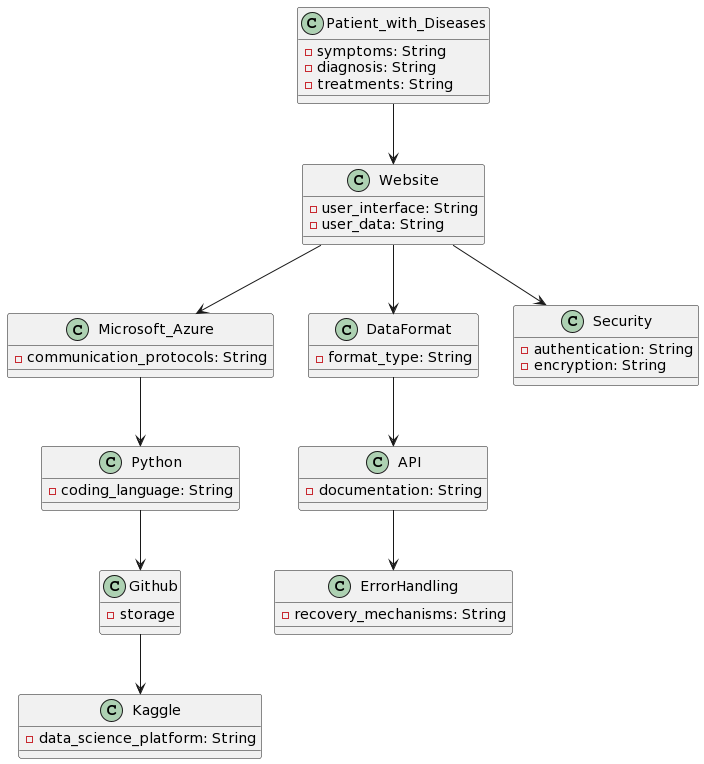
* Protocols used for communication
* Microsoft azure
* Python
* Kaggle
* Data formats (e.g. JSON, XML)
* API documentation (if available)
* Error handling and recovery mechanisms
* Security considerations, such as authentication and encryption



This diagram shows the associations between the classes "Microsoft Azure", "Python", "Kaggle", "Website", "DataFormat", "API", "ErrorHandling", and "Security". The arrows represent the relationships between classes, where the arrow points from the class that uses the other class to the class that is being used. The association between "DataFormat" and "API" represents the fact that the API uses data formats like JSON and XML, while the association between "API" and "ErrorHandling" represents the fact that the API handles errors. Similarly, the association between "Website" and "Security" represents the fact that the website considers security when handling user data.

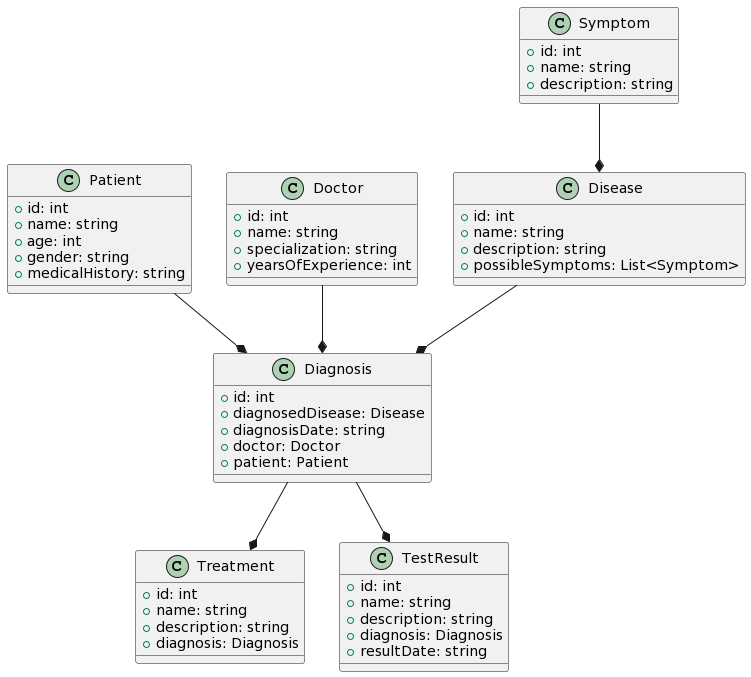
**Comment:** If the external component has already been documented, then the interface requirements can be referred to in the document. This section can be omitted if the system has no interface to external components.

**Software Design**



**Class Diagram and Class Specifications**

The class diagram for a medical disease diagnostic tool would depend on the specific requirements and functionalities of the tool. It could potentially include classes such as "Patient", "Doctor", "Symptom", "Disease", "Diagnosis", "Treatment", "Test Result", etc. The classes would have attributes and methods to represent the data and behavior of each object and would have associations to model the relationships between objects. The exact details of the class diagram would need to be determined based on the specific requirements and design considerations for the medical disease diagnostic tool.



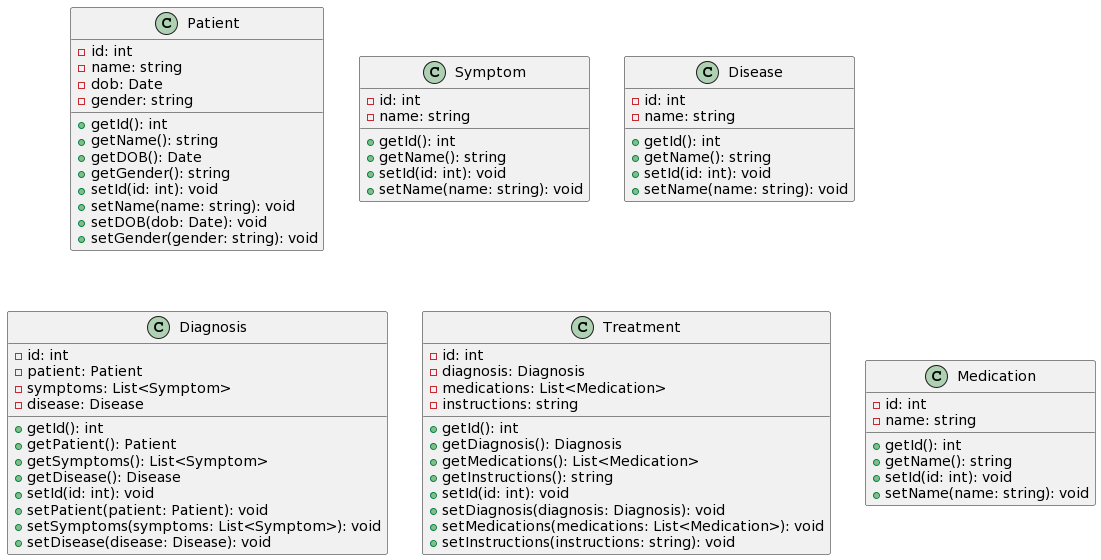
**Interaction Diagrams**

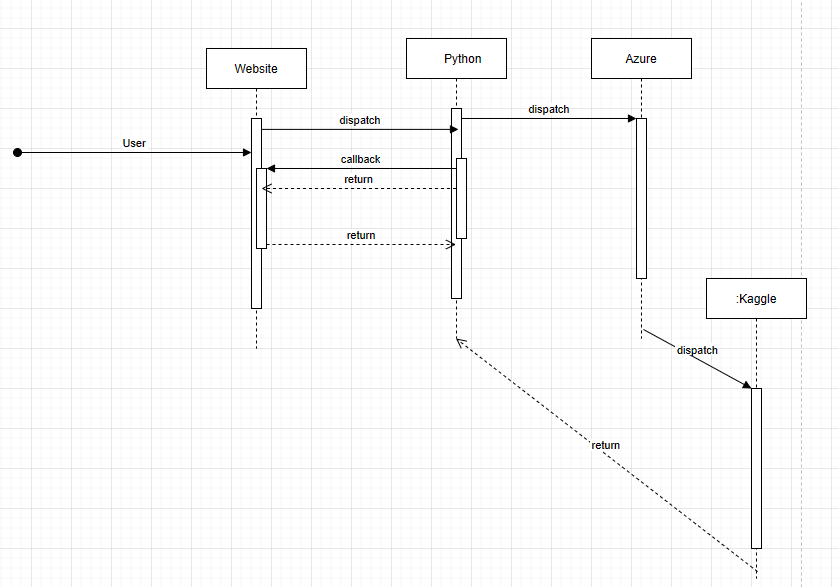
A medical disease diagnostic tool would typically work by allowing users, such as patients, to input their symptoms and personal information into the system. The tool would then process this information and use algorithms based on machine learning and image processing to make a diagnosis. The diagnosis would be based on patterns and correlations in medical data and images, such as X-rays or MRIs, stored in a database.

The user interface for the tool would be designed to be simple and intuitive, allowing users to easily input their information and receive a diagnosis. The interface could be web-based or a standalone application and could include features such as a symptom checker, appointment scheduling, and medical history tracking.

The tool would communicate with external systems, such as database management systems, to access and process medical images and information. It would also interface with external software services to develop and train the algorithms used to make diagnoses.

The tool would be hosted on a server and the medical images and algorithms would be stored in a storage component. The server would process the images and predictions, allowing the tool to provide quick and accurate diagnoses to users.

****



**Design Considerations**

Design considerations for a medical disease diagnostic tool would include the following:

1. User-centered Design: The tool should be designed with the end-user in mind, taking into account their experience, needs, and expectations.
2. Data Privacy and Security: The tool should ensure the privacy and security of sensitive patient data, such as medical history and test results.
3. Reliable Diagnostic Algorithms: The diagnostic algorithms should be based on current best practices and medical guidelines, and should be validated and tested for accuracy.
4. Ease of Use: The tool should be easy to use, with a user-friendly interface and clear, concise instructions.
5. Integration with Electronic Health Records (EHRs): The tool should be able to integrate with existing EHR systems to facilitate data sharing and analysis.
6. Compliance with Regulatory Standards: The tool should comply with relevant regulatory standards, such as HIPAA and GDPR.
7. Scalability: The tool should be designed to handle a large volume of patients and data, and should be scalable to accommodate future growth.

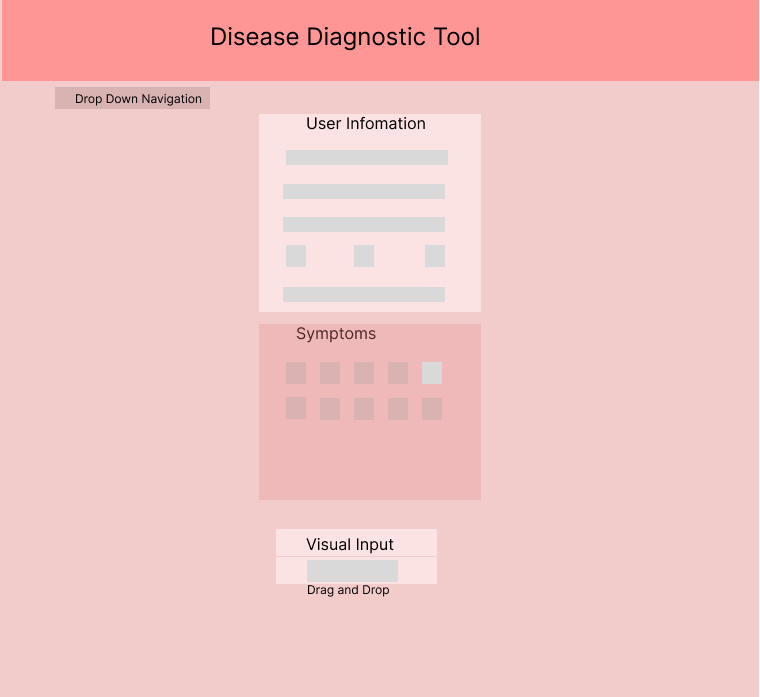
**User Interface Design**

In the user interface design for a medical disease diagnostic tool, we are considering the following aspects:

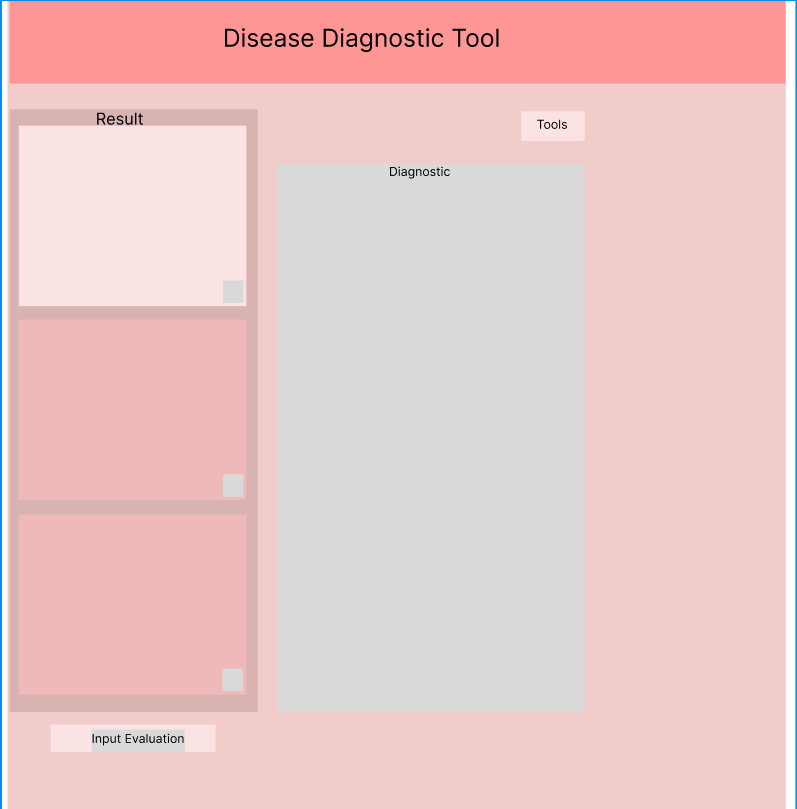
1. User-centered design: The UI should be designed with the user's needs and goals in mind, providing an intuitive and efficient experience for the user.
2. Data Input: The tool should provide an easy way for the user to enter patient information and symptoms.
3. Display of Results: The tool should present the diagnostic results in a clear and concise manner, making it easy for the user to understand.
4. User Feedback: The UI should provide appropriate feedback to the user, indicating if the input is correct, or if there are any errors.
5. Interactivity: The tool should provide interactive elements such as buttons, drop-down menus, and forms, to help the user navigate the diagnostic process.
6. Visual Aesthetics: The UI should be visually appealing, providing an aesthetically pleasing experience for the user.

The sketches or mock-ups of the major UI screens should include a visual representation of the user interface elements, such as buttons, forms, and displays, and the layout of the UI elements. This will help to guide the development of the code and ensure that the final product meets the intended user requirements.

1. Center UI with patient data information and symptoms.



B) Display of Result and Evaluation with Tool to navigate



**Glossary of Terms**

A glossary of terms can be a helpful tool in a System Design Document to provide a common understanding of key terms and concepts used throughout the document. This helps to ensure consistency and avoid ambiguity in the system specification. Some examples of terms that could be included in a glossary for our medical disease diagnostic tool are:

• Disease: A pathological condition or disorder of an organism, often characterized by specific symptoms and signs.

• Diagnosis: The process of determining the nature of a disease or disorder by examination of the symptoms and signs.

• Symptom: A subjective or observable evidence of a disease or disorder.

• Sign: An objective evidence of a disease or disorder, often revealed by medical examination or laboratory tests.

• Medical History: A record of the patient's medical background, including previous illnesses, treatments, and medical conditions.

• Risk Factors: Factors that increase the likelihood of developing a particular disease or condition.

• Treatment: A process for addressing a disease or disorder, often involving medications, therapies, or surgery.

• Algorithm: A set of steps or rules that can be followed to solve a problem or make a decision.

• Machine Learning: A subfield of artificial intelligence that focuses on the development of algorithms that can learn from and make predictions based on data.

• Artificial Intelligence: The simulation of human intelligence in machines, especially computers, that are designed to perform tasks that typically require human intelligence.

**References**

The list of references should contain exact references and/or URLs of any material that is cited in the analysis document. The references should be formatted consistently using ACM, IEEE or APA style. Do not mix citation styles. The following sites may be helpful for formatting your references.

[Kaggle - Wikipedia](https://en.wikipedia.org/wiki/Kaggle)

[Kaggle: Your Machine Learning and Data Science Community](https://www.kaggle.com/)

[Kaggle Data Science Projects for Beginners | Another Techs](https://anothertechs.com/programming/machine-learning/kaggle-data-science-projects/)

• Pacific Library citation tools, includes **APA** citation style:

<http://www.pacific.edu/Library/Get-Help/Citation-Tools.html>

• **ACM** citation style:

<http://www.acm.org/publications/article-templates/acm-latex-style-guide>

• IEEE citation style:

<http://www.ieee.org/documents/ieeecitationref.pdf>

• Examples in **ACM, APA** and **IEEE** styles:

<http://dal.ca.libguides.com/content.php?pid=860&sid=11>