**Project Report**

**IRIS TUMOR DETECTION**

**Team Members:**

**Himanshi Singh,**

**Arsalan Rafique Shaikh**

**, Deena Grece**

**1. Project Overview**

* **What's the Goal?**
  + The *Iris Tumor Detection Project* aims to leverage *deep learning* for the early diagnosis of tumors in the iris by analyzing high-resolution eye images. The goal is to develop an AI solution using *Convolutional Neural Networks (CNNs)* to support healthcare professionals in making prompt, accurate decisions, thus enhancing patient outcomes.
  + This project includes data collection, preprocessing, model training, and integration into a web-based platform, but it will not include real-time monitoring or integration with existing hospital management systems.
* **Why is it Important?**
  + Early detection of iris tumors can significantly improve treatment outcomes. This project is crucial for enabling quicker diagnoses, reducing the need for invasive procedures, and providing healthcare professionals with a tool to assist in their decision-making process.
  + The specific outcomes include reducing diagnosis time, improving accuracy, and enhancing patient care.
* **Who's Involved?**
  + **Project Leader**: Oversees the development and ensures project goals are met.
  + **Data Scientists**: Responsible for data collection, preprocessing, and model training.
  + **Web Developers**: Develop the user interface using Django, HTML, and JavaScript.
  + **End-Users**: Healthcare professionals who will use the system to analyze eye images.

**2. Requirements Documentation**

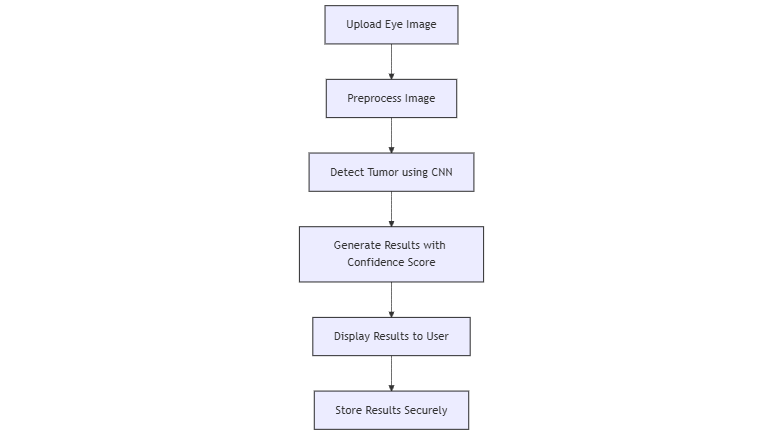
* **What Does it Need to Do?**
  + The system should:
    - Allow users to upload high-resolution iris images.
    - Automatically detect and classify tumors using a trained CNN model.
    - Provide accurate detection results with a confidence score.
    - Securely store user data and results.
* **How Well Does it Need to Work?**
  + The model needs to achieve at least **90% accuracy** on test data.
  + The system should handle multiple concurrent users with minimal latency.
  + Ensure **data security** and comply with healthcare privacy regulations (like HIPAA).
  + Provide a **user-friendly interface** for healthcare professionals with minimal technical knowledge.
* **Why is it Needed?**
  + This project aligns with the healthcare sector's goal of using AI to assist in early diagnostics, improving patient outcomes, and reducing healthcare costs by enabling non-invasive screening methods.
* **How Will Users Use It?**
  + **User Stories**:
    - As a healthcare professional, I want to upload an eye image to quickly detect potential tumors.
    - As an admin, I want to manage user registrations and monitor the system's usage.
    - As a user, I want to receive clear and reliable diagnostic results that can aid in my decision-making process.

**3. Project Plan**

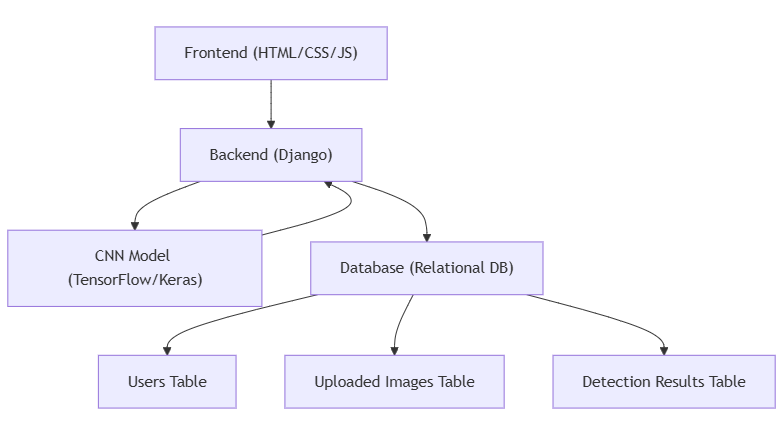
* **When Will it Be Done?**
  + **Timeline**:
    - Phase 1: Data Collection & Preprocessing - *2 weeks*
    - Phase 2: Model Design & Training - *4 weeks*
    - Phase 3: Web Interface Development - *3 weeks*
    - Phase 4: Testing & Optimization - *2 weeks*
    - Phase 5: Deployment & Maintenance - *1 week*
  + **Milestone**: Achieve a functional prototype by the end of *8 weeks*.
* **What Do We Need?**
  + **Resources**:
    - Team: Data scientists, web developers, project manager.
    - Software: Python, Tensor Flow/Keras, Django, HTML/CSS, JavaScript.
    - Hardware: High-performance GPUs for training the CNN model.
    - Dataset: High-resolution iris images.
* **How Much Will it Cost?**
  + **Estimated Budget**:
    - Development & Testing: $ -
    - Deployment & Hosting: $-
    - Maintenance & Support: $ -
    - Total Estimated Cost: $-

**4. Architecture and Design Documentation**

* **How is it Built?**
  + The system uses a **client-server architecture**:
    - **Frontend**: Developed using HTML, CSS, and JavaScript for user interactions.
    - **Backend**: Built using Django to handle data processing, model integration, and user management.
    - **CNN Model**: Trained using TensorFlow/Keras to detect tumors in uploaded iris images.
* **What Does it Look Like?**
  + **Diagrams**:
    - *Flowchart*: Illustrates the user journey from uploading an image to receiving the detection results.



* + - *System Architecture*: Shows the integration of the frontend, backend, and deep learning model.



* **How is Data Stored?**
  + **ER Diagram**:

