**Project Report**

**IRIS TUMOR DETECTION**

**Team Members:**

* Himanshi Singh
* Arsalan Rafique Shaikh
* Deena Grace

**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**

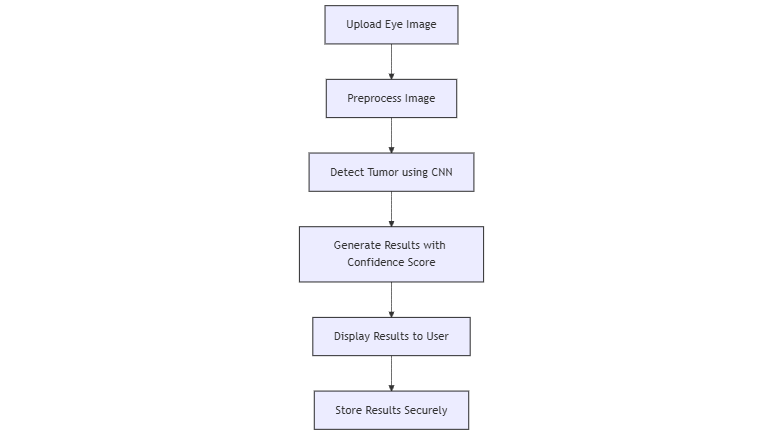
* **What Does it Need to Do?**
  + The system should:
    - Allow users to upload high-resolution iris images.
    - Automatically detect and classify tumors using a trained CNN model.
    - Provide accurate detection results with a confidence score.
    - Securely store user data and results.
* **How 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**

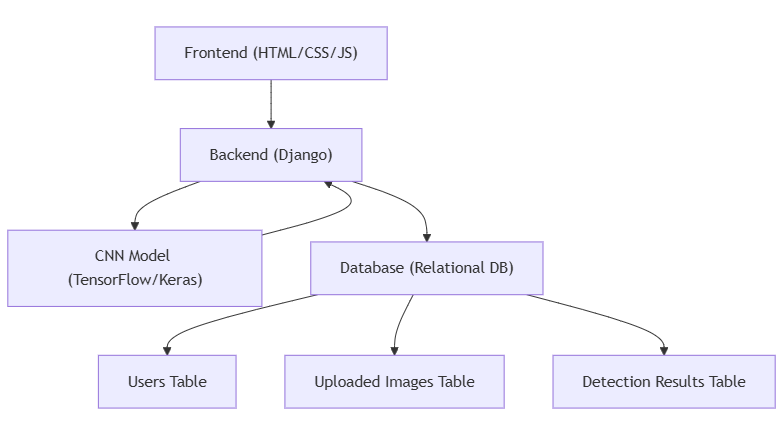
* **When Will it Be Done?**
  + **Timeline**:
    - Phase 1: Data Collection & Preprocessing - *2 weeks*
    - Phase 2: Model Design & Training - *4 weeks*
    - Phase 3: Web Interface Development - *3 weeks*
    - Phase 4: Testing & Optimization - *2 weeks*
    - Phase 5: Deployment & Maintenance - *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**

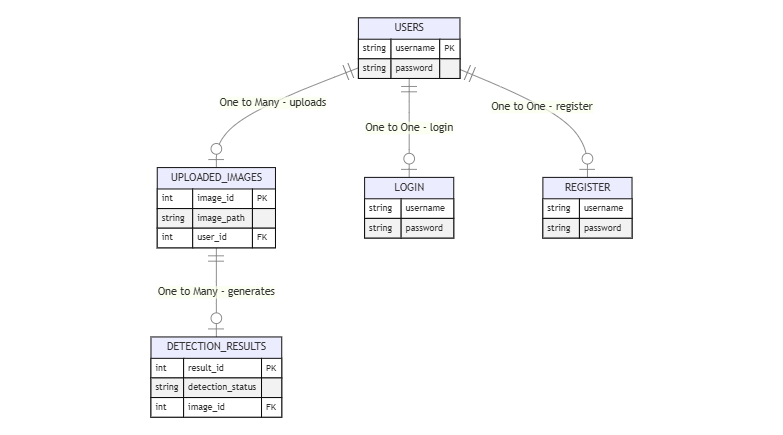
* **How is it Built?**
  + The system uses a **client-server architecture**:
    - **Frontend**: Developed using HTML, CSS, and JavaScript for user interactions.
    - **Backend**: Built using Django to handle data processing, model integration, and user management.
    - **CNN Model**: Trained using 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**:



**5. Testing and Quality Assurance**

**How will we ensure it works?**  
Our testing strategy includes multiple levels of testing to ensure the system functions correctly:

* **Unit Tests**: Focus on individual components, such as the CNN model and Django views, to validate their functionality.
* **Integration Tests**: Ensure that the components work seamlessly together, like the communication between the model and the web interface.
* **System Tests**: Test the complete application in an environment similar to production to verify all features.
* **Tools Used**: PyTest, Selenium for automated testing, and Postman for API testing.

**What makes it complete?**  
The project will be considered complete when:

* The model achieves a detection accuracy of at least 90% on the test dataset.
* All critical features (uploading images, user registration, tumor detection) are fully functional.
* All reported bugs are resolved, and the system passes acceptance testing by stakeholders.

**When will we test?**

* **Phase 1**: Initial unit tests during model training and feature development.
* **Phase 2**: Integration testing once the web interface is connected to the model.
* **Phase 3**: System testing before deployment.
* **CI/CD Integration**: Automated testing pipelines using GitHub Actions to run tests on each push.

**What if something goes wrong?**

* **Issue Tracking**: We will use JIRA and GitHub Issues for reporting and tracking bugs.
* **Fixing Issues**: The development team will prioritize and address issues based on severity and impact on users.

**6. Deployment and Implementation Plan**

**Where will it live?**  
The system will be deployed on a cloud-based platform such as **AWS** or **Google Cloud** for scalability and reliability. This setup will allow future expansion as user demand grows.

**How will we get it there?**

* **Deployment Strategy**: A phased rollout approach to ensure stability. We will first deploy on a staging environment, followed by a production environment after successful testing.
* **Tools**: Docker for containerization, Jenkins for continuous integration, and Nginx as a reverse proxy.

**What if something goes wrong?**

* **Backup Plans**: Automated database backups and version control via GitHub for code. If deployment issues occur, we'll implement a rollback to the previous stable version.
* **User Training**: We will provide a user guide and tutorial videos to help users navigate the system effectively.

**7. Maintenance and Support**

**Who's in charge of keeping it running?**  
The **Development Team** will manage ongoing support and maintenance. A dedicated support engineer will handle user issues post-deployment.

**What needs to be done to keep it running?**

* **Regular Maintenance Tasks**:
  + Monthly software updates.
  + Quarterly model retraining with new data to improve accuracy.
  + Continuous monitoring of system performance and uptime.

**How will we know what users think?**  
We will collect feedback through the platform’s user feedback form and monitor analytics for insights on user interactions.

**What are our service commitments?**

* **Service-Level Agreements (SLAs)**:
  + Response time: Within 24 hours for critical issues.
  + Resolution time: 48 hours for high-priority bugs.

**8. Risk Management**

**What could go wrong?**

* **Technical Risks**: Model overfitting, integration issues, cloud server downtime.
* **Operational Risks**: Inaccurate predictions leading to misdiagnosis, security vulnerabilities in user data.
* **Financial Risks**: Budget overruns due to unexpected technical challenges.

**How can we prevent problems?**

* **Mitigation Strategies**:
  + Regular model validation to detect overfitting.
  + Security audits for the Django web application.
  + Cost tracking to avoid budget overruns.

**What's our backup plan?**  
In case of major issues, we will have:

* A backup server for critical services.
* A manual process for handling user requests in case the automated system fails.

**Who's watching for problems?**  
The **Project Manager** and a dedicated **Risk Management Team** will be responsible for identifying, tracking, and mitigating risks throughout the project lifecycle.

**9. Security and Privacy**

**Data Protection**: We will use \*\*encryption\*\* to keep all data (like iris images and results) safe. Only authorized users can access the data. We'll also follow privacy laws like \*\*GDPR\*\* to protect user rights.

**Security Measures**: The system will have ***firewalls*** to block unwanted access, ***intrusion*** ***detection*** to spot suspicious activity, and regular ***security checks*** to fix any weaknesses.

**Incident Response Plan**: If there's a data breach, we’ll quickly respond by identifying the issue, stopping it, notifying users, and fixing any problems. Activity logs will help us trace any suspicious behavior.

**10. Legal and Compliance**

**Licensing**: We’ll use ***open-source software*** like Django and Python libraries, making sure we follow their license rules. We'll also get any required licenses for hosting and deployment.

**Regulatory Compliance:** We will comply with healthcare data laws like ***HIPAA*** and ***GDPR*** to securely handle sensitive health information.

**Intellectual Property:** The CNN model and code we develop will be protected under ***copyrights.*** We might also consider patents for any unique algorithms used in tumor detection.

**11. Environmental Impact Assessment**

**Sustainability**: To reduce our impact on the environment, we'll use ***cloud-based systems*** that save energy and avoid using physical servers.

**Green IT Practices:** We will use energy-efficient cloud services and optimize our code to use fewer resources. This will help reduce waste and lower energy consumption.

**12. User Documentation**

**User Manuals:** We’ll provide simple guides on how to use the system, upload images, and understand the results.

**Online Help and Tutorials:** Users will have access to ***FAQs***, step-by-step tutorials, and videos to help them use the platform confidently.

**User Interface Design:** The website will be easy to use with a clear layout, large buttons, and simple forms to make it user-friendly for everyone.

**13. Project Management and Monitoring**

**Project Planning:**  
Develop a comprehensive project plan to manage the project effectively. This includes breaking down tasks into phases, setting realistic deadlines, allocating resources, and establishing clear milestones. Regular reviews will ensure that we stay on track and can adjust as necessary.

* **Tasks and Timelines:**
  + Data Collection & Preprocessing: 2 weeks
  + Model Design & Training: 4 weeks
  + Web Interface Development: 3 weeks
  + Testing & Optimization: 2 weeks
  + Deployment & Maintenance: 1 week
* **Resource Allocation:**
  + **Data Scientists:** Responsible for data preprocessing and model training.
  + **Web Developers:** Focused on frontend and backend development.
  + **Project Manager:** Oversees the entire project, ensuring adherence to the timeline.

**Risk Management:**  
We will identify potential risks throughout the project lifecycle and proactively develop mitigation strategies to address them.

* **Risks Identified:**
  + **Technical Risks:** Model overfitting, integration challenges, cloud server downtime.
  + **Operational Risks:** Misdiagnosis due to inaccurate predictions, potential data breaches.
  + **Financial Risks:** Unexpected costs impacting the budget.
* **Mitigation Strategies:**
  + Conduct regular model evaluations to detect overfitting.
  + Perform security audits on the Django application.
  + Track expenditures to avoid budget overruns.

**14. Testing and Quality Assurance**

To ensure the system is robust, accurate, and user-friendly, a multi-tiered testing strategy will be employed.

**Unit Testing:**  
We will perform tests on individual components, such as the CNN model, Django views, and database functionality. This ensures each module operates as expected.

* **Tools Used:** PyTest for Python