****

**Software Engineering Department  
Braude College**

Capstone project Phase B

**Automated Medical Diagnosis: Enhancing Chest X-ray Analysis with Convolutional Neural Networks**

**Maintenance Guide**

Project Code: 24-1-D-8

**Supervisor: Dr. Natali Levi**

**Noam Ofir**

**Oded Morag**

GitHub Repo:

<https://github.com/GoodMoodMan/xray_classifier>

1. System Architecture Overview:

The Automated Medical Diagnosis system is built with a separate backend (Node.js and Python) and frontend (React) structure. The server folder contains the Node.js server and Python scripts for machine learning operations, while the client folder houses the React application.

1. Server Requirements
   * Software:
     1. Node.js (version 14.x or later)
     2. Python (version 3.8 or later)
     3. MongoDB (version 4.4 or later)
2. Installation Process:
   * Clone the repository:
     1. git clone https://github.com/GoodMoodMan/xray\_classifier.git
     2. cd xray\_classifier
   * Set up the server:
     1. cd server
     2. npm install
     3. pip install -r requirements.txt
   * Set up the client:
     1. cd ../client
     2. npm install
   * Set up environment variables:
     1. Create a .env file in the server directory
     2. Add necessary environment variables (database URL, API keys, etc.)
   * Build the React frontend:
     1. npm run build
   * Start the server (from the server directory):
     1. node server.js
   * In development, start the React development server (from the client directory):
     1. npm start
3. Updating the ML Model
   * Navigate to the server directory
   * Locate the model files (e.g., best\_chest\_xray\_classifier.pth)
   * Replace with the updated model
   * Update the corresponding Python script (e.g., xray\_classifier.py) if the model architecture has changed
4. Database Management
   * The system uses MongoDB. Ensure regular backups are performed.
   * Database connection is likely managed in server.js or a separate configuration file in the server directory.
5. Monitoring and Logging
   * Implement logging in server.js and other server-side scripts
   * Use a tool like PM2 for monitoring the Node.js application
6. Security Measures
   * Regularly update all dependencies in both server and client directories
   * Implement proper CORS settings in server.js
   * Ensure all API endpoints are properly authenticated and authorized
7. Troubleshooting
   * Check server logs for backend errors
   * For frontend issues, check the browser console and network tab
   * Verify MongoDB connection if database-related issues occur
   * For ML model issues, check the Python environment and model file integrity
8. Customization and Extension
   * To add new features to the frontend, create new React components in the client/src/components directory
   * Backend API endpoints can be extended in server.js or in separate route files
   * For new ML capabilities, add new model files and update the corresponding Python scripts in the server directory