**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

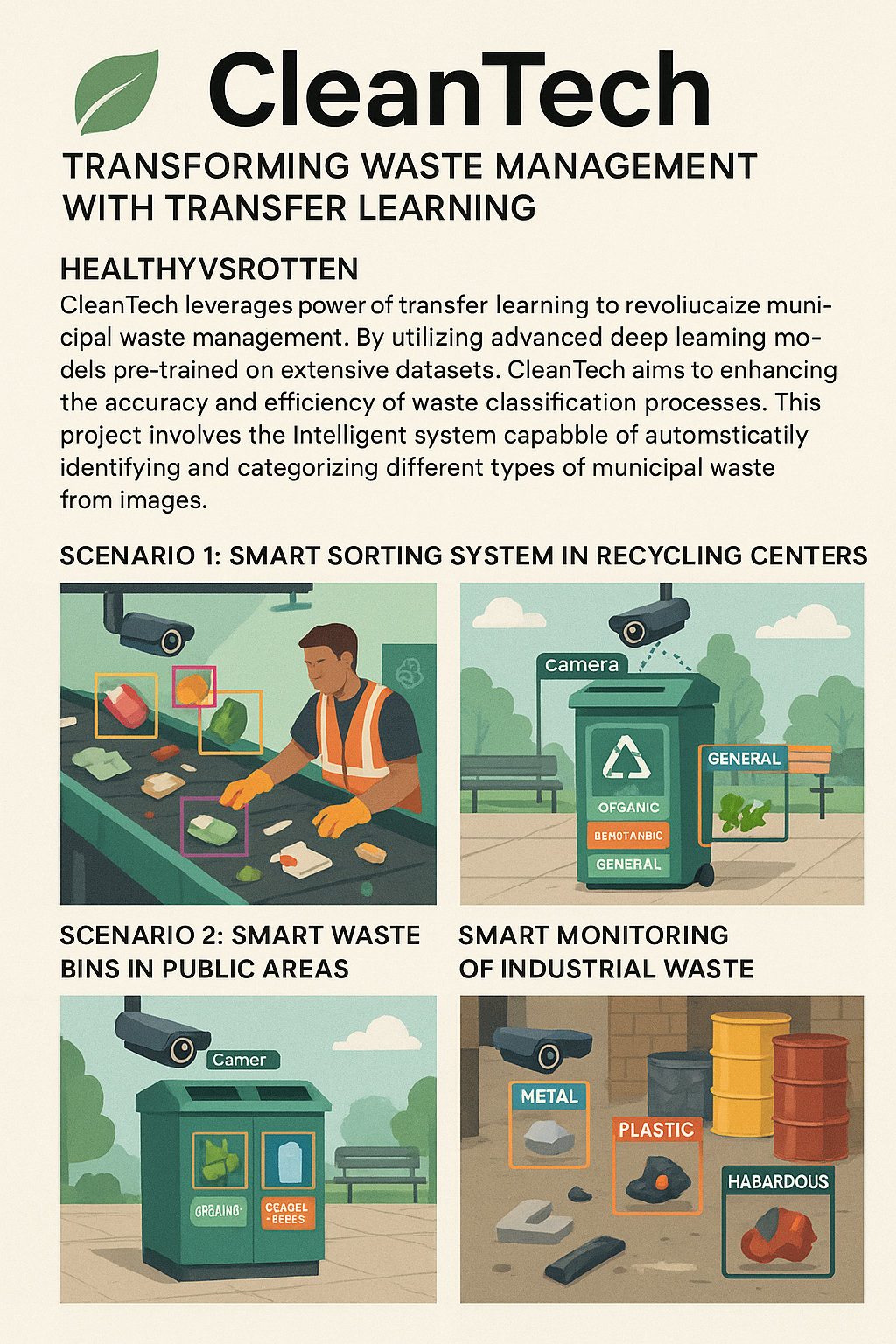
|  |  |
| --- | --- |
| Date | 24 June 3035 |
| LTVIP2025TMID32454 | LTVIP2025TMID32454 |
| Project Name | Clean Tech:Transforming Waste Management with Transfer learning |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)



Guidelines:

1. Use pre-trained models with transfer learning for accuracy.
2. Collect high-quality waste images from multiple sources.
3. Train on diverse datasets (organic, recyclable, hazardous).
4. Deploy cameras in bins, factories, and conveyor systems.
5. Ensure real-time image processing for quick classification.
6. Store classified waste data for reporting and analytics.
7. Regularly retrain the model using user feedback.
8. Implement dashboards for city officials and factory admins.
9. Maintain data privacy and image security.
10. Optimize for speed, accuracy, and scalability.

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web dashboard for monitoring classification results and system status | ReactJS, HTML5, CSS3 |
|  | Application Logic-1 | Waste image acquisition and preprocessing pipeline | Python, OpenCV |
|  | Application Logic-2 | Transfer learning for classifying waste images | TensorFlow, PyTorch, Keras |
|  | Application Logic-3 | Real-time decision system for sorting, alerting, or dashboard update | Flask / FastAPI |
|  | Database | Stores logs, image metadata, and classification output | MongoDB (NoSQL), SQLite for edge devices |
|  | Cloud Database | Centralized storage for analysis and reporting | Firebase, AWS DynamoDB |
|  | File Storage | Storage for training data, image logs | AWS S3, Local Filesystem for edge |
|  | External API-1 | Email/notification system for alerts | SendGrid API, Twilio |
|  | External API-2 | Location-based waste data mapping (smart cities) | Google Maps API |
|  | Machine Learning Model | Transfer Learning Model for waste classification | Pre-trained ResNet50, MobileNetV2 |
|  | Infrastructure (Server / Cloud) | Hybrid: edge for bins/factories, cloud for training and dashboards: | NVIDIA Jetson, AWS EC2, Docker, Kubernetes |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks and libraries used in development | TensorFlow, PyTorch, Flask, React |
|  | Security Implementations | Secure model endpoints and data access using encryption and IAM | JWT, HTTPS, SHA-256, Firebase Auth |
|  | Scalable Architecture | Microservices-based containerized deployment for scale | Docker, Kubernetes, RESTful APIs |
|  | Availability | 24/7 access ensured via cloud deployment and failover support | Load Balancer, Multi-zone AWS Deployment |
|  | Performance | Real-time image inference, caching frequently used model results | TensorRT (for edge), Redis, CDN |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)