**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 24 June 2025 |
| Team ID | LTVIP2025TMID35409 |
| Project Name | Hematovision |
| Maximum Marks | 4 Marks |

**Technology Stack & Architecture:**

HematoVision follows a simple and effective three-tier architecture, which includes:

1. User Interface (Frontend) – Where the user uploads a blood cell image through a webpage.
2. Application Logic (Backend) – This part processes the image and runs the deep learning model to identify the blood cell type.
3. Storage & Model – Where the trained model (Blood\_Cell.h5) is used and images are temporarily stored for prediction.

The application uses a lightweight deep learning model (MobileNetV2) to ensure quick predictions and a good user experience. Everything is tied together using Flask, a Python-based web framework.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | |  | | --- | | Let’s users upload blood cell images easily |  |  | | --- | |  | | HTML, CSS, JavaScript |
|  | Application Logic-1 | |  | | --- | | Loads the ML model and preprocesses the image |  |  | | --- | |  | | Python, Flask |
|  | Application Logic-2 | |  | | --- | | Runs the actual blood cell classification |  |  | | --- | |  | | TensorFlow, MobileNetV2 |
|  | Application Logic-3 | |  | | --- | | Handles image reading and rendering results | | OpenCV, Flask |
|  | Database | |  | | --- | | For storing data like past results or logs | | |  | | --- | | SQLite (Optional) | |
|  | Cloud Database | |  | | --- | | If moved to the cloud for scalability |  |  | | --- | |  | | IBM Cloudant / Firebase (Optional) |
|  | File Storage | |  | | --- | | Temporarily holds uploaded images | | |  | | --- | | Local Filesystem | |
|  | External API-1 | |  |  | | --- | --- | | Could fetch real-time health/environment data |  | |  |  |  | |  |  |  | | OpenWeatherMap API |
|  | External API-2 | Could verify patient identity | UIDAI Aadhaar API |
|  | Machine Learning Model | Classifies the image into 4 blood cell types | MobileNetV2 saved as Blood\_Cell.h5 |
|  | Infrastructure (Server / Cloud) | Where the app runs | Localhost / Heroku / AWS (optional) |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | |  |  | | --- | --- | | We’ve used popular open tools for better support and cost-efficiency |  | | Flask, TensorFlow, Keras, OpenCV |
|  | Security Implementations | Validates image file types, prevents bad uploads | Flask built-in methods, path checks |
|  | Scalable Architecture | The modular design lets us move to cloud easily | 3-tier architecture, microservices-ready |
|  | Availability | Can be deployed on platforms with high uptime | Heroku, AWS, Load Balancers (optional) |
|  | Performance | Fast response time using a lightweight model | MobileNetV2, efficient image preprocessing |

**References:**

* [C4 Model for Architecture](https://c4model.com/)
* [IBM Developer Architecture Patterns](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)
* [IBM Cloud Architecture](https://www.ibm.com/cloud/architecture)
* [AWS Architecture Examples](https://aws.amazon.com/architecture)
* [How to Draw Architecture Diagrams](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)