🩸 Capstone Project: Blood Bank Management System (Microservices with MERN Stack & PostgreSQL)

A full-stack web application using microservices architecture and PostgreSQL from the start. The system streamlines donor registration, inventory management, appointments, and blood requests while ensuring maintainability, scalability, and deployment readiness for cloud platforms.

✅ Milestone 1: PostgreSQL Integration and Node for Backend Microservices

Goal: Set up PostgreSQL and build initial backend microservices using Node.js for donor and inventory modules. Define schemas, create RESTful APIs, and structure the microservice codebases.

Tasks & Subtasks:

Task	Subtasks
PostgreSQL Setup	Design normalized schemas for Donor, Appointments, Blood Inventory
Node Microservices	Set up DonorService and InventoryService as separate Node.js services
RESTful API Endpoints	Implement basic endpoints for registration, viewing inventory, booking
Project Structure	Use Docker or separate folders for services
GitHub Setup	Initialize version control with README, commit structure

Deliverables:

- PostgreSQL schemas and connection config
- Node-based microservices (Donor, Inventory) with REST APIs
- Modular code structure
- GitHub repository with logs and setup instructions

Rubric (100 Marks):

Criteria	Marks
PostgreSQL Schema Design	20
Microservice Architecture & Structure	20
API Design and Functionality	20
Code Modularity	10
Git Usage & Commit Hygiene	10
Booking Use Case Flow	20

✅ Milestone 2: Backend with Express and API Integration, React Frontend

Goal: Expand backend with Express.js features (authentication, middleware, routing) and connect React frontend via APIs to support complete donor workflows like login, scheduling, and inventory viewing.

Tasks & Subtasks:

Task	Subtasks
Express Enhancements	Add routing, middleware, and security (Helmet, CORS) in each service
JWT Authentication	Role-based (Donor/Admin) auth with secure login APIs
React UI	Forms and components for login, registration, booking, inventory list
API Integration	Connect React to microservices using Axios or Fetch
Input Validations	Client-side + backend validations for all forms

Deliverables:

- Express-based microservices with security and auth
- React frontend with routing and form integration
- Connected and functional API calls
- Postman collection for API testing

Rubric (100 Marks):

Criteria	Marks
JWT Authentication & Role Access	20
Middleware & Security Practices	15
React Form Integration	20
API Integration and Data Sync	20
Postman API Testing	10
Code Cleanliness & Best Practices	15

✅ Milestone 3: Azure Deployment & GitHub Workflow

Goal: Deploy microservices and frontend to Azure using Docker/VM/App Services. Set up GitHub workflows, DevOps practices, and project management tools for collaboration and CI/CD hygiene.

Tasks & Subtasks:

Task	Subtasks
Azure Deployment	Containerize and deploy at least one service (e.g., Inventory) to Azure
Frontend Hosting	Host the React app using Azure Static Web Apps or App Services
GitHub Workflow	Use GitHub Actions for CI/CD (build/test/deploy pipelines)
DevOps Practices	Create issues, manage PRs, define branching strategy
Project Management	Use Jira/Trello to manage tasks, track bugs, and progress
Documentation	Architecture diagram + service README + deployment steps

Deliverables:

- Live deployed service and frontend on Azure
- GitHub repo with Actions and workflow files
- Jira/Trello board screenshot
- Swagger/README docs for APIs
- Architecture and deployment diagram

Rubric (100 Marks):

Criteria	Marks
Successful Azure Deployment	25
GitHub Actions Workflow	20
Architecture Documentation	15
Jira/Trello Usage	10
PR, Branching & Commit Practices	15
Service Availability Testing	15

✅ Milestone 4: PostgreSQL Migration, Python Chatbot Helpdesk & Final Deployment

Goal: Migrate all services and data cleanly to PostgreSQL (if any dev DBs were used), integrate a Python-based GenAI chatbot for donor FAQs, implement notification logic, and finalize the project.

Tasks & Subtasks:

Task	Subtasks
PostgreSQL Migration	Finalize all services with production-ready PostgreSQL setups
Chatbot Integration	Python-based chatbot to answer donor FAQs (embedded or standalone)
Notification System	Send SMS/Email (real or mock) for appointment reminders
Testing	Write unit/integration tests using Jest, Supertest, PyTest
Final Deployment	Deploy full system to Azure/Render with test credentials
Final Presentation	Create PPT, video demo, project report with challenges/insights

Deliverables:

- All microservices with PostgreSQL finalized
- Python chatbot for donor interaction
- Deployed frontend + backend + chatbot
- Complete test suite with results
- Final repo, PPT, demo video

Rubric (100 Marks):

Criteria	Marks
PostgreSQL Final Migration	20
Python Chatbot Helpdesk	15
Notifications Implementation	10
Testing Coverage	15
End-to-End Deployment	15
Presentation (PPT + Demo)	15
Innovation (Extra Features)	10