Project Design Phase-II
Data Flow Diagram & User Stories

Date	17 JULY 2025
Team ID	PNT2025TMID11128
Project Name	Measuring the Pulse of Prosperity

👉 Table 1: Components & Technologies

S. No	Component	Description	Technology / Tools
1	User Interface	Frontend layer with embedded dashboards	HTML, CSS, JavaScript, Bootstrap
2	Embedding Logic (App Logic)	Embedding Tableau dashboards via API	Tableau Embedding API v3 <tableau-viz>, JS, flerlagetwins.com, help.tableau.com, connectedapps.theinformationlab.io
3	Backend Logic (Optional)	(Optional) For custom filters or view auth	Python + Flask (if extended)
4	Visualization Engine	Dashboard creation and refresh	Tableau Desktop, published to Tableau Public
5	Database (Local)	Store the legacy dataset	MySQL or SQL Server
6	Database (Cloud)	Cloud-hosted or sync data	AWS RDS / Azure SQL or similar
7	File Storage	Static files and assets	GitHub Pages / Netlify static hosting
8	External API	For user authentication or email notifications	OAuth (Gmail, LinkedIn), SMTP email API
9	Machine Learning Model	(Future scope) Score forecasting or clustering	Python ML (scikit-learn / TensorFlow)
10	Hosting Infrastructure	Application deployment and CDN	GitHub Pages / Netlify, optional Flask on Heroku

Table 2: Application Characteristics

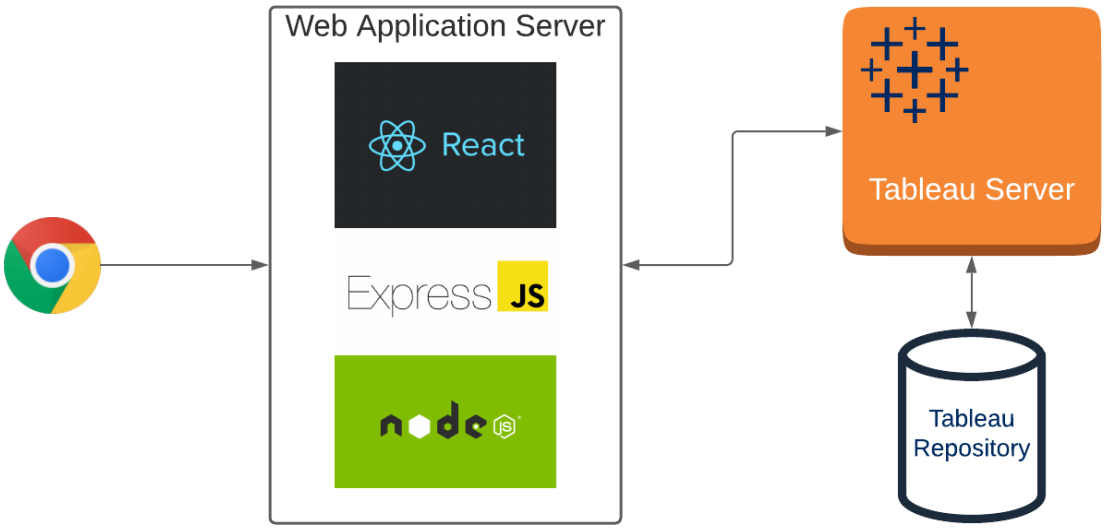
S. No	Characteristic	Description	Technology / Approach
1	Open-Source Frameworks	Use of community-supported tools for frontend and backend	Bootstrap, Python, Flask
2	Security	Data protection and access control	HTTPS/TLS, OAuth, SMTP, Content-Security-Policy
3	Scalable Architecture	Supports adding datasets or heavy traffic seamlessly	Modular frontend, static CDN, cloud database
4	Availability	High uptime; static hosting and cloud databases ensure resilience	GitHub Pages / Netlify + managed DB services
5	Performance	Fast frontend loading and visualization rendering	Tableau extracts, CDN, caching, minified assets
6	Maintainability	Easy enhancements, refresh logic, and version control	Modular codebase (HTML/CSS/JS), CI/CD via GitHub

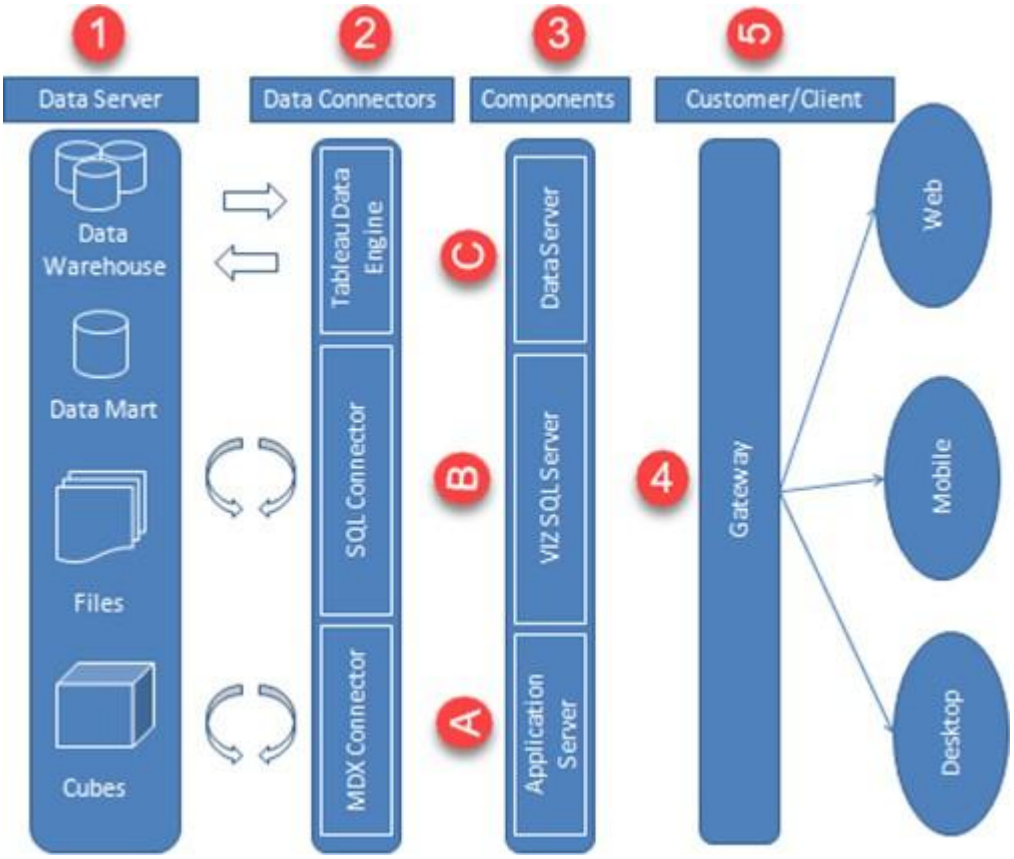
Explanation of Key Points

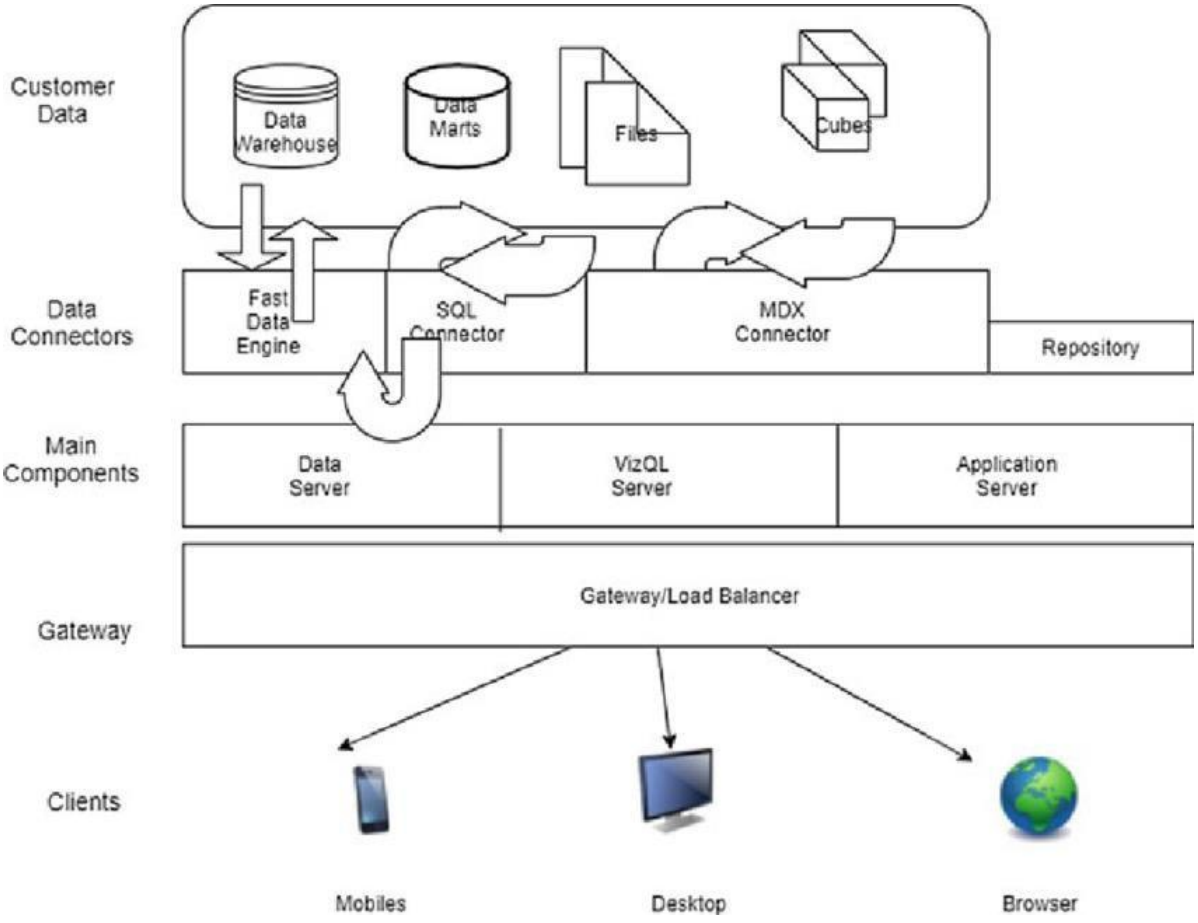
- **Frontend & Embedding**: We use `<tableau-viz>` web components and the Embedding API v3, per Tableau documentation help.tableau.com.

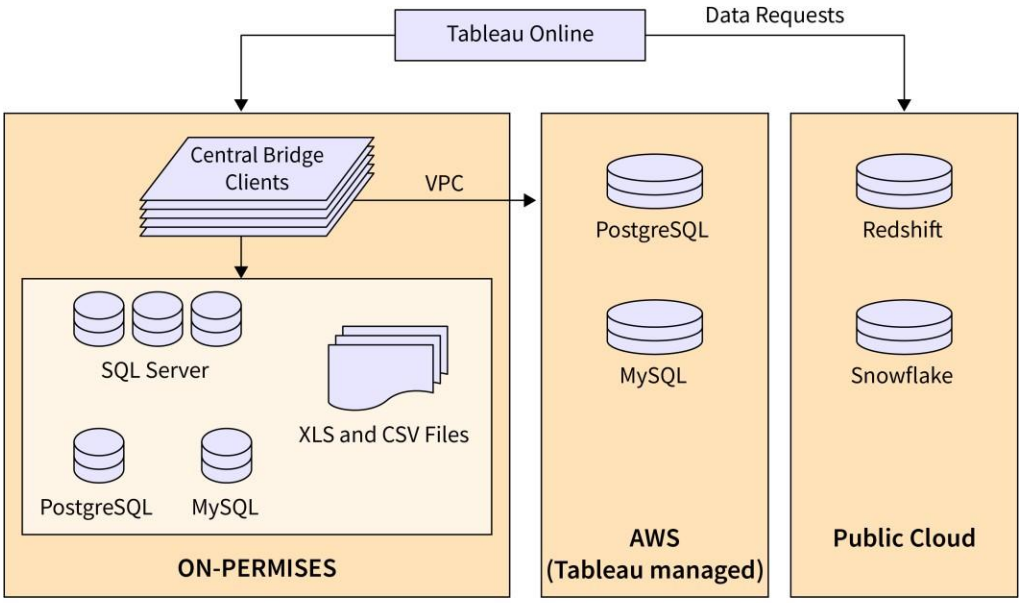
- **Backend (Optional)**: Flask can serve custom APIs, like filter endpoints or authentication proxies.
- **Storage & Hosting**: Static assets are served via GitHub Pages or Netlify for security and speed; MySQL/SQL Server (locally or in the cloud) manages core data.
- **Security Measures**: HTTPS, OAuth logins for OAuth, and secure email confirmation workflows ensure trusted user access.
- **Scalability & Availability**: Static hosting scales automatically; cloud database solutions provide managed availability.

🔧 Suggested Architecture Diagram




1
2
3
5
Data Server
Data Connectors
Components
Customer/Client
Data Warehouse
Data Mart
Files
Cubes
Tableau Data Engine
SQL Connector
MDX Connector
C
B
A
4
Data Server
VIZ SQL Server
Application Server
Gateway
Web
Mobile
Desktop


Customer Data
Data Warehouse
Data Marts
Files
Cubes
Data Connectors
Fast Data Engine
SQL Connector
MDX Connector
Repository
Main Components
Data Server
VizQL Server
Application Server
Gateway
Gateway/Load Balancer
Clients
Mobiles
Desktop
Browser



SCALER Topics

(To be created using tools like draw.io, Lucidchart or C4 modeling)

Container Diagram (C4 - Level 2):

```
pgsql
CopyEdit
[User Browser]
   ↕
[Static Web App (HTML/Bootstrap + Tableau Viz)]
   ↕ (HTTPS requests embed API)
[Tableau Public/Server] ← [Tableau Dashboards from Tableau Desktop]
   ↔
[Database (MySQL / SQL Server)]
```

Optional Additional Components:

- [Flask API Container] ↔ [Database]
- External OAuth providers (Gmail, LinkedIn)
- Email API for confirmation