GEMINDZ-EMS Task2 Solution

Software Requirements:

1. Monitoring System:
   - Collect real-time energy consumption data from the electric panel.
   - Transmit data securely over Wi-Fi to the backend system by a frequency 1 reading / sec.
2. Web Interface Application:
   - Receive and display energy consumption data.
   - Provide visualization tools for users to understand their expenses.
   - Implement budgeting features including setting, tracking, and alerting for overspending.

Recommended Technologies:

1. Monitoring System:
   - Microcontroller (Raspberry Pi)
   - Wi-Fi module for connectivity
2. Backend:
   - Python for server-side logic
   - Flask for RESTful API
   - MySQL for database
3. Frontend:
   - React.js for web interface and dynamic UI
   - HTML,CSS and Javascript for building Frontend Components (eg. Dashboard Component,Budget Tracker Component,Alerts Component and Settings Component)
4. Cloud Services:
   - AWS or Google Cloud for hosting backend services
   - Firebase for authentication and real-time database

Potential Dependencies:

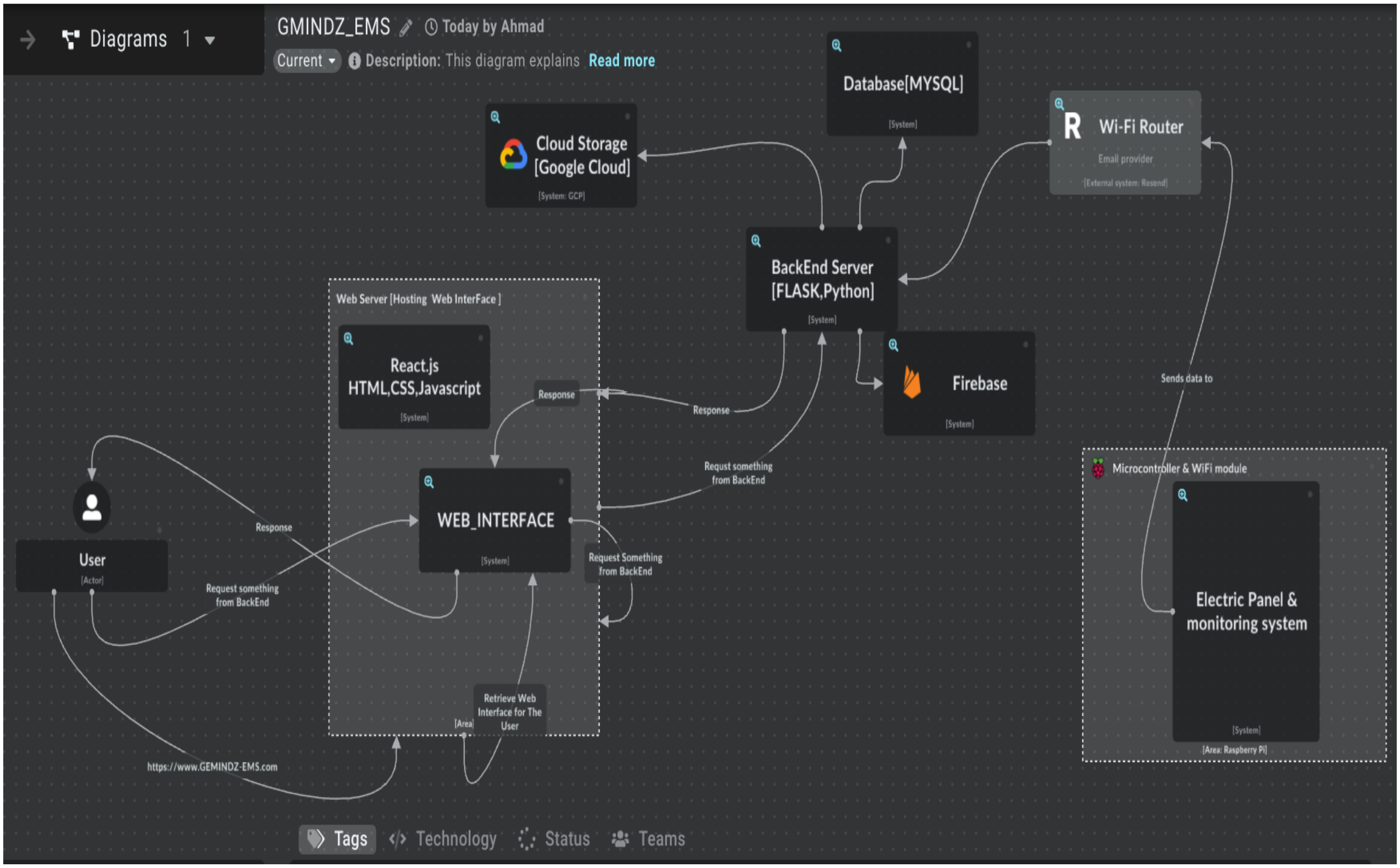
- Availability of Wi-Fi connectivity for data transmission.
- Compatibility of chosen Microcontroller with monitoring system requirements.
- API integration for visualization and budgeting features.
- Security and privacy compliance.

MVP for Frontend:

- Utilize React.js for web interface.
- Implement basic UI components for displaying real-time energy consumption data.
- Develop simple budgeting features such as setting budget limits and displaying current expenses.
- Time needed: 2-3 weeks.

Software Architecture Graph:

Link : https://s.icepanel.io/UEycLc8MgqZDQb/b8y4


Diagrams 1
GMINDZ_EMS
Today by Ahmad
Current
Description: This diagram explains Read more
Database[MYSQL]
[System]
Cloud Storage [Google Cloud]
[System: GCP]
Wi-Fi Router
Email provider
[External system: Resend]
BackEnd Server [FLASK,Python]
[System]
Web Server [Hosting Web InterFace]
React.js HTML,CSS,Javascript
[System]
Response
Response
Firebase
[System]
Sends data to
Requst something from BackEnd
WEB_INTERFACE
[System]
Request Something from BackEnd
Response
User
[Actor]
Request something from BackEnd
Microcontroller & WiFi module
Electric Panel & monitoring system
[System]
[Area: Raspberry Pi]
Retrieve Web Interface for The User
[Area]
https://www.GEMINDZ-EMS.com
Tags
Technology
Status
Teams

Task Breakdown:

1. Backend Development:

- Set up server and database: 1 week
- Implement RESTful API: 2 weeks
- Integrate with monitoring system: 1 week

2. Frontend Development:

- Design UI/UX: 1 week
- Develop real-time data visualization: 2 weeks
- Implement budgeting features: 2 weeks

3. Testing and Deployment:

- Unit testing and debugging: 1 week
- Integration testing: 1 week
- UAT & Deployment to production: 1 week

Timeline and Milestones:

- Week 1-2: Backend setup and API development
- Week 3-4: Frontend UI/UX design and real-time data visualization
- Week 5-6: Implementing budgeting features and testing
- Week 7 : Integration testing and deployment

Milestones:

- Backend API ready for integration(Week 2)
- Frontend UI/UX design finalized (Week 4)
- Budgeting features implemented (Week 6)
- Application deployed to production (Week 7)

Storage Considerations:

- Cloud storage is recommended for scalability and accessibility from web or mobile platforms.
- Local storage can be used for caching data on devices but may not be suitable for long-term storage or cross-platform access.

Security and Privacy Considerations:

- Implement HTTPS for secure data transmission.
- Use encryption for sensitive user data.
- Implement user authentication and authorization mechanisms.
- Regular security audits and updates to address vulnerabilities.

Limitations and Challenges:

- Dependency on stable Wi-Fi connectivity for real-time data transmission.
- Compatibility issues between monitoring hardware and software components.
- Ensuring data accuracy and reliability in real-time monitoring.
- Compliance with privacy regulations (e.g., GDPR) regarding user data collection and storage.

To address limitations and challenges:

1. Stable Wi-Fi Dependency:
   - Implement robust error handling for network disruptions.
   - Provide offline functionality for local data storage and synchronization.
2. Compatibility Issues:
   - Standardize communication protocols and data formats.
   - Conduct thorough compatibility testing between hardware and software.
3. Data Accuracy and Reliability:
   - Implement sensor calibration procedures.
   - Introduce error-checking mechanisms for data transmission.
4. privacy Compliance:
   - Integrate privacy controls into the system architecture.
   - Obtain explicit user consent for data collection.
   - Encrypt sensitive data in transit and at rest.
   - Implement access controls and authentication mechanisms.
   - Conduct regular audits for ongoing compliance.