Optimization and Efficiency Research Document

---

1. Optimizing Speed and Efficiency of the App

Resources:

- Optimizing Web App Performance
- Efficient Frontend Rendering Techniques

Code Example:

Debouncing the inputs so that we are not updating on every mouse update, would be very resource heavy. Log mouse inputs locally for 2 seconds e.g. then send to the database etc.

```
const debounce = (func, delay) => {
  let timeout;
  return (...args) => {
    clearTimeout(timeout);
    timeout = setTimeout(() => func.apply(this, args), delay);
  };
};

const fetchData = debounce(() => {
  fetch('/api/data')
    .then(response => response.json())
    .then(data => console.log(data));
}, 300);

// Usage: Call fetchData on user input
inputElement.addEventListener('input', fetchData);
```

---

2. Efficiently Storing Large Amounts of Data in PostgreSQL

Resources:

- PostgreSQL Performance Tuning
- Indexing Best Practices

Code Example:

```
-- Creating an index for faster query performance
```

```
CREATE INDEX idx_user_email ON users(email);

-- Partitioning a large table by date
CREATE TABLE user_logs (
    id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    log_time TIMESTAMP NOT NULL,
    activity TEXT
) PARTITION BY RANGE (log_time);

-- Creating partitions
CREATE TABLE user_logs_2023 PARTITION OF user_logs
    FOR VALUES FROM ('2023-01-01') TO ('2024-01-01');
```

---

3. Optimizing API Communication

Resources:

- REST API Best Practices
- Efficient API Communication Techniques

Code Example:

```
// Using async/await for efficient API handling
async function getData(endpoint) {
  try {
    const response = await fetch(endpoint);
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Error fetching data:', error);
  }
}

// Example API call
getData('/api/v1/resources');
```

---

4. PreFast Methods to Store Large Amounts of Data

Resources:

- Efficient Data Storage Techniques
- Bulk Data Insertion in PostgreSQL

Code Example:

```
-- Using COPY command for bulk data insertion
COPY users(id, name, email)
FROM '/path/to/data.csv'
DELIMITER ','
CSV HEADER;

-- Using JSONB for flexible data storage
CREATE TABLE user_data (
   id SERIAL PRIMARY KEY,
   user_info JSONB
);

-- Inserting JSON data
INSERT INTO user_data(user_info)
VALUES ('{"name": "John Doe", "email": "john@example.com"}');
```

---

5. Final Integration and Testing

Resources:

- Performance Testing Tools
- Database Performance Monitoring

Action Items:

- Monitor app performance post-optimization.
- Run load tests on API endpoints to ensure stability under high traffic.
- Continuously refine PostgreSQL queries and indexing based on monitoring feedback.
