Designing Google calender

Planning

1. Requirement Gathering
    a. Functional
    b. Non-function
2. Component Architecture
3. Data API and Protocol
4. Data Entities
5. Data Store
6. Performance and Optimization
7. Accessibility
8. Security

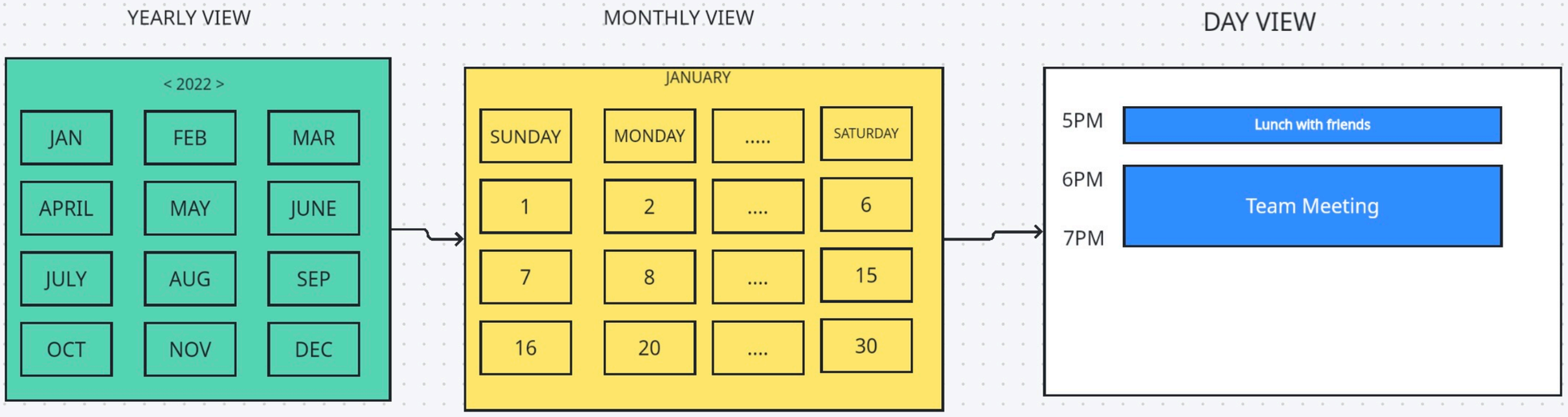
Functional Requirement

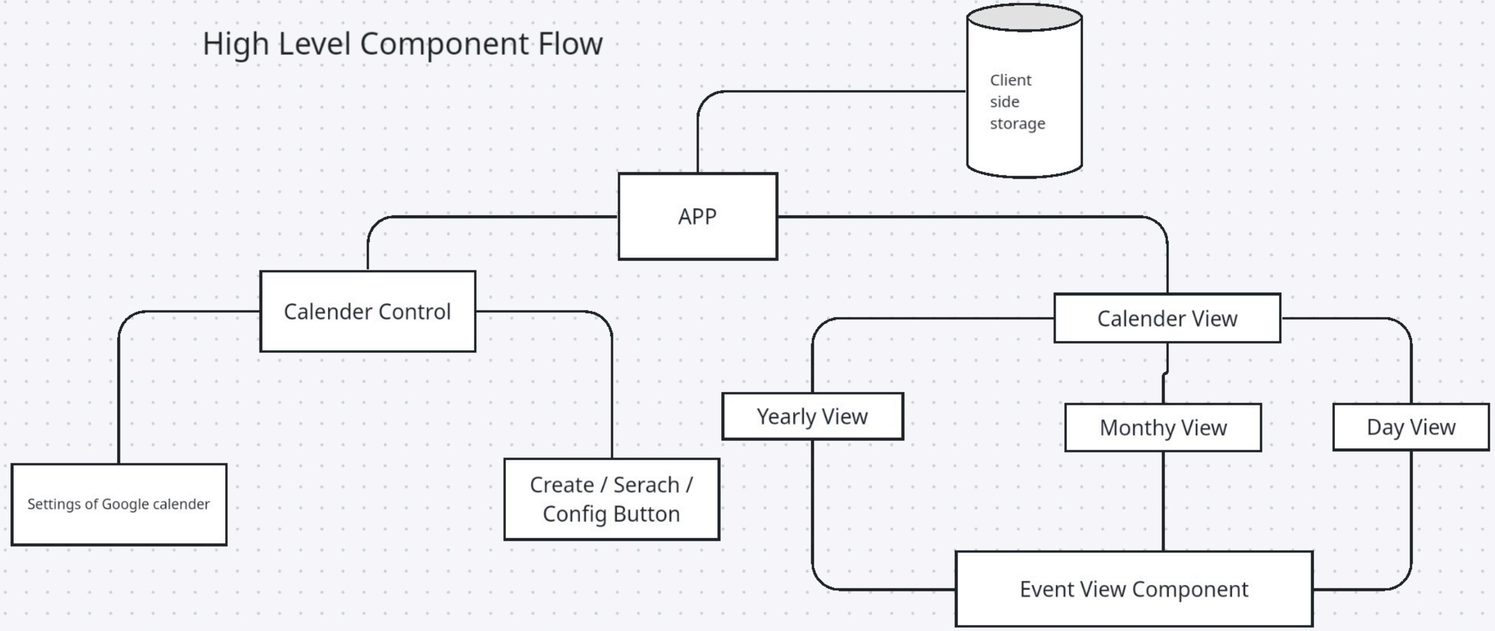
1. User should be able to view the calender in different format
    a. Day Format
    b. Week Format
    c. Monthly Format
    d. Yearly Format
2. User should be able to create / delete events
3. The application should detect and provide options to resolve conflicting events.
4. Offline Support

Non-Functional Requirement

1. It should support a variety of devices,
2. It should be consistent across browsers.
3. Transition should be smooth
4. Web Vitals (LCP / CLS / INP)
5. Logging (storing runtime error)
6. Variety of language (Localization / Internalization)

Component Architecture





1. **We can talk about CSR vs SSR**
2. **We can talk about GraphQL vs Rest Appraoch**

```
Type Event = {
   id: String
   title: String
   description: String
   attchments: [Attachments]
   createdAt: DateTime
   startTime: DateTime;
   endTime: DataTime
   attendes: [Attendes]
}
```

```
Type Attendes = {
   userId: String
   email: String
   name: String
}

Type Attachments = {
   id: String
   url: String(url)
}
```

```
Type Calender = {
   events: [Event]
}
```

Now, we can also talk about how we are going to get the latest data in UI ?

1. Pooling based technique
2. WebSocket
3. Event Based technique

Do we need to discuss about conflicts resolution ?

1. For every event we will send the request to server, so If we got the succesfull event only then we will create
2. Or
3. We can talk a little bit it will provide good impression.

1. **Lock-based mechanisms or versioning**
• If two people edit the same event at the same time (like rescheduling a meeting), you need a way to **prevent or detect conflicting changes**.
2. **Optimistic Concurrency Control**
• Let both users make changes, but **check a version number or lastModified timestamp** when saving. If it has changed, ask for confirmation or retry.

3-Way Merge or Patch Resolution (optional)
• Useful when a user edits while offline and then reconnects.
• Helps decide how to **merge conflicting changes without overwriting important data**.