Project Design Documents

Team JAHMS (Group 5) - Event Planner Application

Alish Jain, Hannah Raju, Jinita Bhatt, Micheal Rawlings, Sach Denuwara

GITHUB REPO - https://github.com/HannahSRaju/ITCS6112_MDSP24_Group5

1. Project Overview:

Event Planner is a centralized platform designed to streamline event organization. It offers tools for creating, managing, and customizing events, along with features for discovering and booking vendors, tracking tasks and budgets, and communicating with guests. With its user-friendly interface and comprehensive functionalities, the application simplifies the complexities of event planning, saving time and resources for organizers while ensuring a seamless experience for all stakeholders. The goal of this project is to develop a comprehensive and user-friendly event planner web application that simplifies the process of organizing and managing any event.

This application aims to solve the below mentioned problem with an online solution.

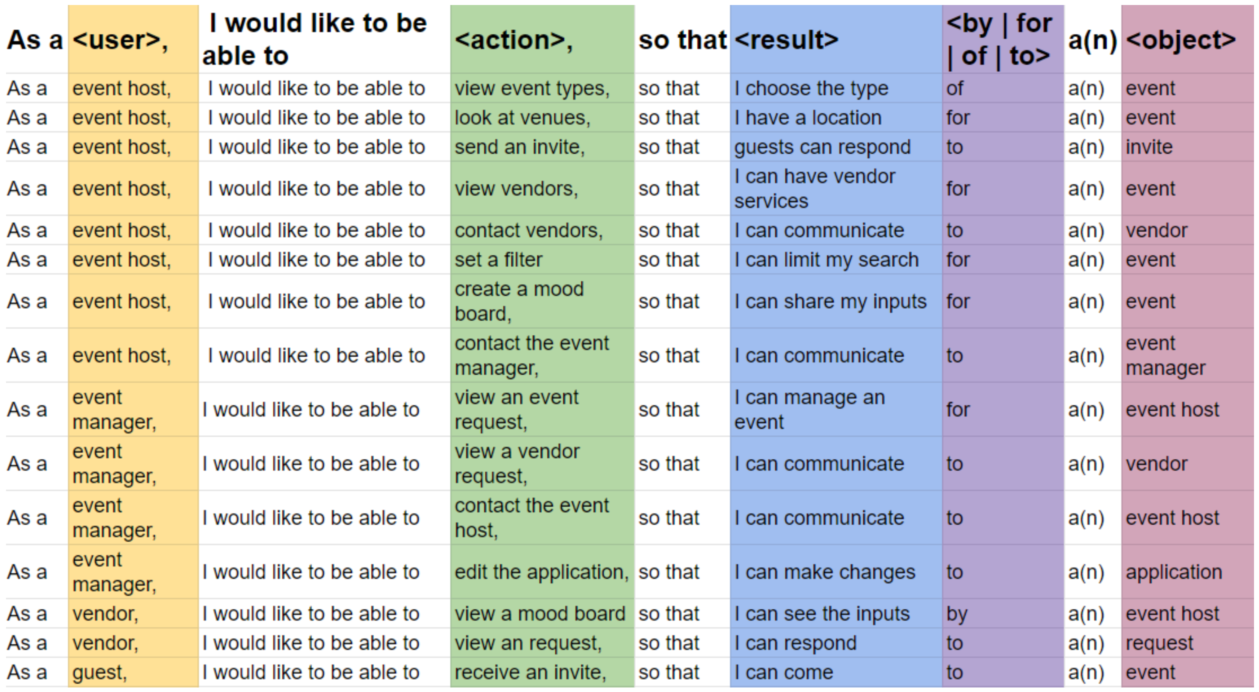
Problem Statement -

Organizing events, from small gatherings to large-scale weddings, involves complex coordination of multiple tasks including venue selection, photographer selection, vendor management, budget tracking, and effective communication with guests. Event organizers often struggle with the disparate tools and manual processes required to manage these components efficiently, leading to increased time, costs, and the potential for oversights that can impact the event's success. All the vendor selection for any event by the event host is a manual process which involves searching for good Vendors for catering, venue places, Decorators, etc.

Solution -

The Event Planner application aims to address these challenges by providing a centralized platform for the end-to-end management of events. It integrates tools for creating and customizing events, vendor discovery and booking, task and budget tracking, and guest communication into a single user-friendly interface. This comprehensive approach simplifies the event planning process, saves time and resources for organizers, creates opportunities for vendors, and enhances the experience for hosts.

User Stories -



As a	<user>,	I would like to be able to	<action>,	so that	<result>	<by | for | of | to>	a(n)	<object>
As a	event host,	I would like to be able to	view event types,	so that	I choose the type	of	a(n)	event
As a	event host,	I would like to be able to	look at venues,	so that	I have a location	for	a(n)	event
As a	event host,	I would like to be able to	send an invite,	so that	guests can respond	to	a(n)	invite
As a	event host,	I would like to be able to	view vendors,	so that	I can have vendor services	for	a(n)	event
As a	event host,	I would like to be able to	contact vendors,	so that	I can communicate	to	a(n)	vendor
As a	event host,	I would like to be able to	set a filter	so that	I can limit my search	for	a(n)	event
As a	event host,	I would like to be able to	create a mood board,	so that	I can share my inputs	for	a(n)	event
As a	event host,	I would like to be able to	contact the event manager,	so that	I can communicate	to	a(n)	event manager
As a	event manager,	I would like to be able to	view an event request,	so that	I can manage an event	for	a(n)	event host
As a	event manager,	I would like to be able to	view a vendor request,	so that	I can communicate	to	a(n)	vendor
As a	event manager,	I would like to be able to	contact the event host,	so that	I can communicate	to	a(n)	event host
As a	event manager,	I would like to be able to	edit the application,	so that	I can make changes	to	a(n)	application
As a	vendor,	I would like to be able to	view a mood board	so that	I can see the inputs	by	a(n)	event host
As a	vendor,	I would like to be able to	view an request,	so that	I can respond	to	a(n)	request
As a	guest,	I would like to be able to	receive an invite,	so that	I can come	to	a(n)	event

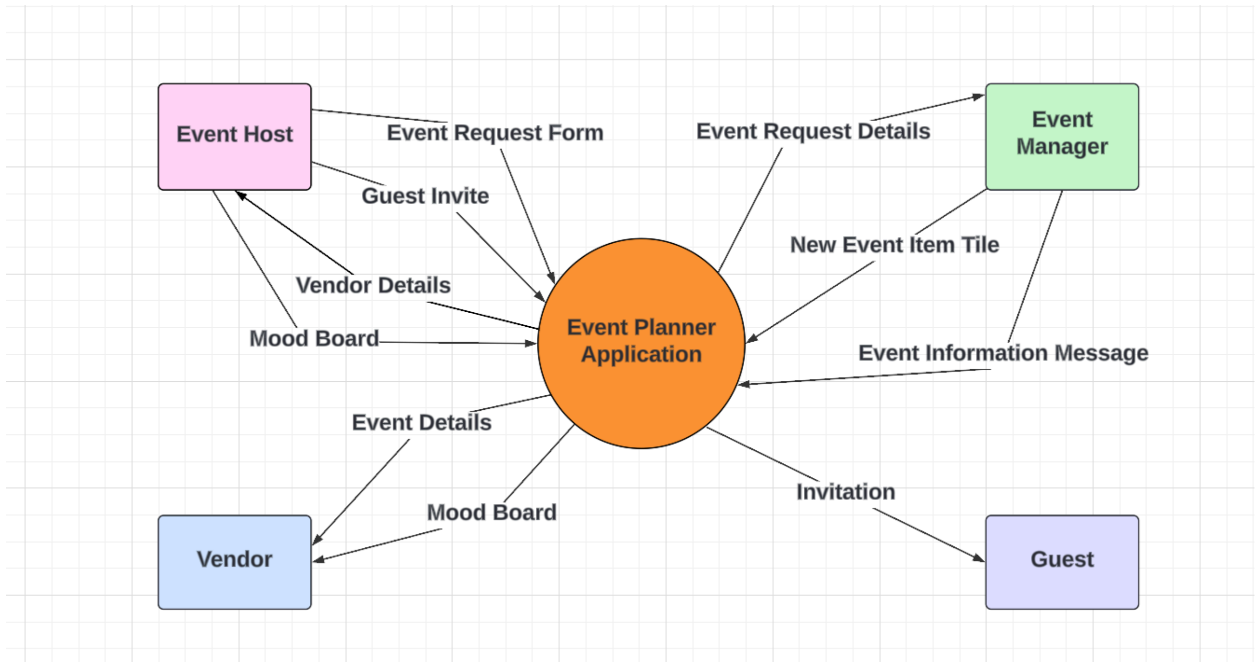
Product Backlog

- More Event creation, editing, and deletion.
- Vendor profile management and service listing.
- Budget and task management tool.
- Guest invitation and RSVP tracking.

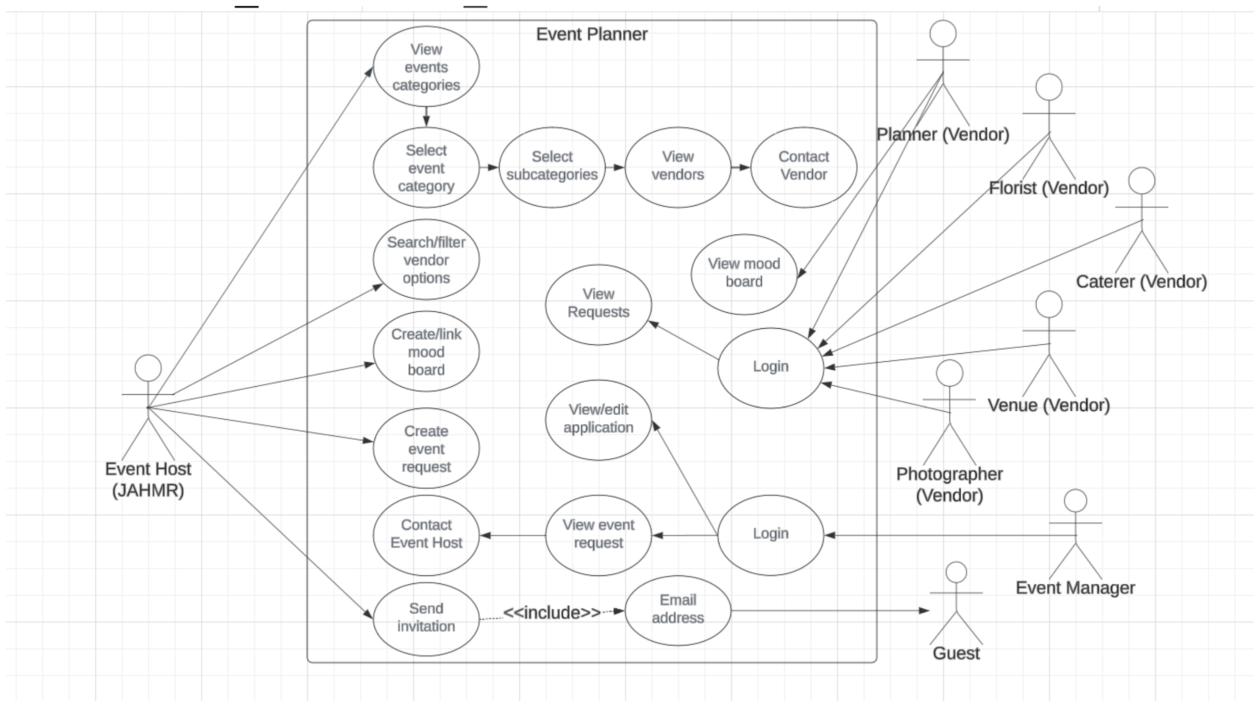
Main Use-Cases and Technical Details

- Event Creation and Management: Utilizes a database to store event details, with a front-end interface allowing application managers to input and modify data. Technical aspects include form handling, data validation, and user authorization.
- Vendor Looking and Contacting: Vendors for all events can be searched and contacted through the application by the Event host for various events that are mentioned in the application.
- Requesting for event Plan - Event host can request the Manager to plan and suggest event vendors for various subcategories like Photographer, Florists, etc. It involves filling out the google form with all the details.
- Communication Platform: A messaging system that supports email features to multiple guests.

Context Diagram:



Use Case Diagram -



2. Architectural Overview:

In the early design stage, we assessed a classic MVC architecture in addition to taking serverless and microservices designs into account. Because of the robust separation of responsibilities, which makes testing, maintenance, and scaling easier within the confines of a single application deployment environment, MVC was chosen. While microservices necessitate intricate coordination among several autonomous services, Model-View-Controller (MVC) provides a simplified method that supports swift web application development and implementation cycles.

2.1 Subsystem Architecture

Under an MVC framework, the subsystem architecture is reduced to three main parts that operate together in the server-side environment of the web application:

Model: This layer manages data logic and symbolizes the dynamic data structure of the programme. It directly oversees the application's logic, rules, and data. Models for the Event Planner application comprise vendor information, event data, and event planning information.

View: The View component is in charge of giving the user access to data in a certain format. It converts models into user-interactive UI components. This layer includes the structure and user interface for elements related to event management, vendor selection, and event planning.

Controller: The Controller functions as a go-between for the Model and the View, handling all business logic and incoming requests, working with the Model component to alter data, and coordinating with the Views to produce the final product. For instance, the event management controller would be in charge of organizing the views for these operations in addition to generating, editing, and removing events.

A UML package diagram is an ideal way to represent all the dependencies and relationships between the layers and components that constitute the application's architecture. Below is the breakdown of what each layer typically contains and their responsibilities in the system:

Four Layers in a Subsystem Dependency Diagram

1. Application-Specific Layer

- Responsibility: Includes the portions of the programme that were specifically created for it and are exclusive to its domain. This covers unique controllers, views, models, services, and utilities required especially for the Event Planner programme.
- Architectural Styles: Primarily influenced by MVC for separation of concerns—models handle the data logic, views manage the presentation logic, and controllers deal with application logic. The choice of MVC here addresses the need for maintainability, scalability, and clear separation between user interface and business logic.

2. Application-Generic Layer

- Responsibility: Consists of general-purpose libraries and modules that are utilized across the application and are not particular to any one application domain. This might include utilities that offer cross-cutting functions, such as logging services, data validation, and authentication libraries.
- Architectural Styles: Makes frequent use of shared libraries or the concepts of service-oriented architecture (SOA), which promotes reuse and modularity. The architecture of this layer facilitates efficiency and minimizes redundancy by offering shared services and functions that the application-specific layer may utilize.

3. Middleware Layer

- Responsibility: Acts as the intermediary layer that provides communication and data management services between the application and the operating system or network services. This includes web servers, application servers, message brokers, and database management systems.
- Architectural Styles: Often based on client-server and request-response patterns to support the application's needs for data exchange, session management, and communication. Middleware abstraction facilitates scalability and integration by offering a uniform interface to diverse underlying technologies.

4. System-Software Layer

- Responsibility: Comprises the operating systems, database servers, and network software that provide foundational services and resources for the application. This layer is responsible for managing hardware resources, data storage, and low-level network communications.
- Architectural Styles: Influenced by layered architecture, ensuring that higher-level layers abstract away the complexities of hardware and system software interactions. This separation ensures application portability and simplifies deployment across different environments.

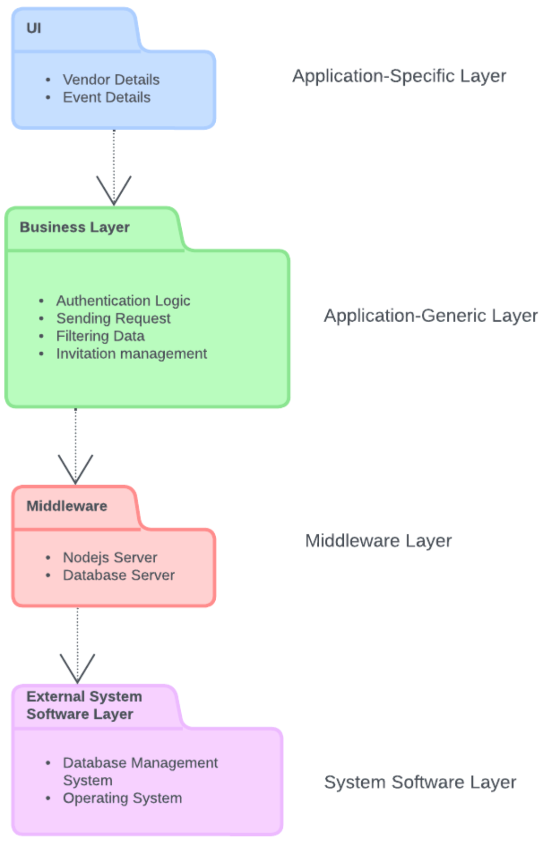
Architectural Styles Applied

- Model-View-Controller (MVC): Selected for its capacity to divide issues, making maintenance and scalability simpler. It facilitates a clear division between the backend functionality and the user interface and increases developer efficiency.
- Service-Oriented Architecture (SOA): Used at the application-generic layer to handle the requirement for efficient development processes and the ease of replacing or updating common services. It allows for the modularity and reuse of generic services throughout

the application.

- Layered Architecture: Managing dependencies and interactions between the application and the system environment requires the use of layered architecture. By dividing the programme into discrete layers with defined roles, it reduces complexity and improves security and maintainability.

UML Package Diagram -

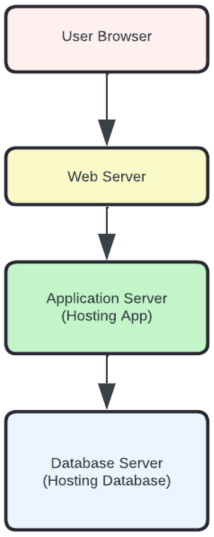


2.2 Deployment Architecture:

Deployment Components:

User's Browser: Represents the client-side interface(UI) through which users interact with the event application. *This acts as a Presentation layer.*

Web Server (Load Balancer): This component acts as a load balancer, distributing incoming client requests across multiple application server instances for improved performance and reliability.


User Browser
Web Server
Application Server (Hosting App)
Database Server (Hosting Database)

Application Server: Hosts the MVC (Model-View-Controller) application, including the application-specific and application-generic layers, responsible for handling business logic, data processing, and user interactions. *This acts as an Application Logic.*

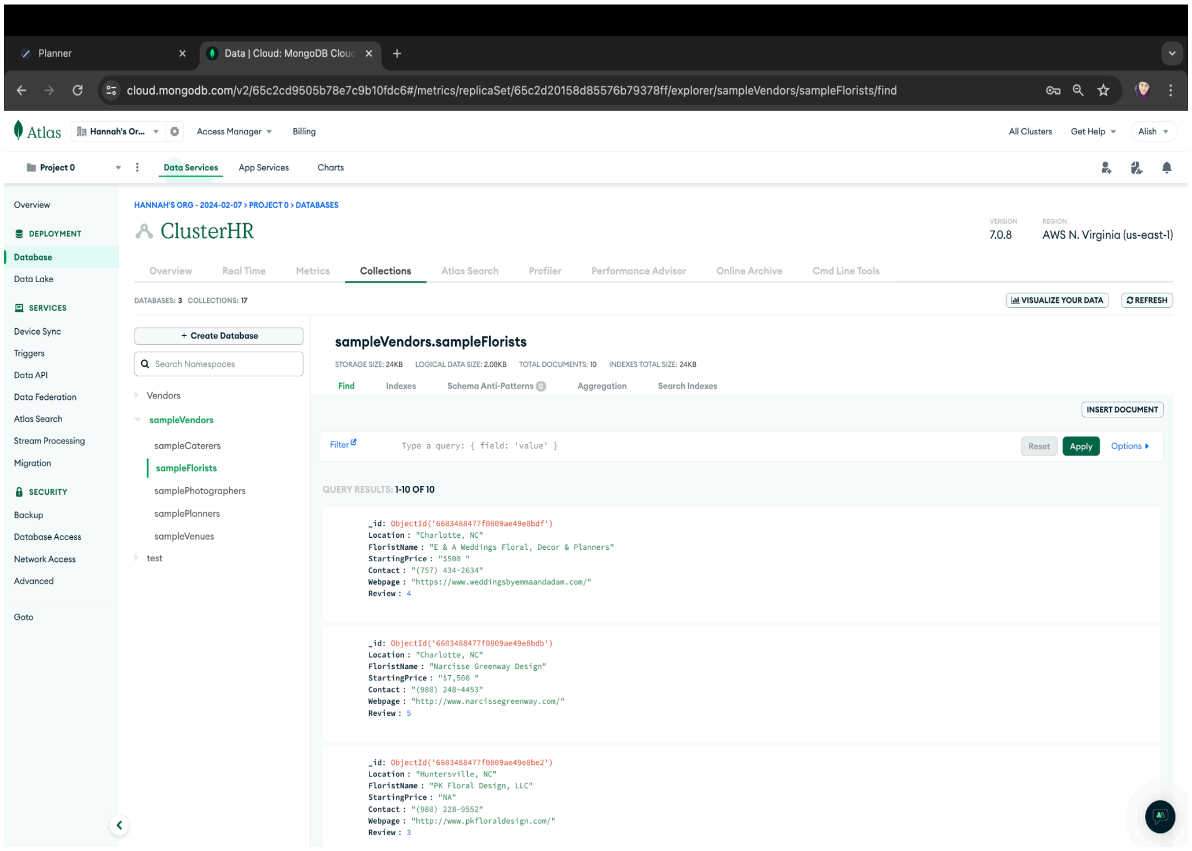
Database Server: Hosts the database management system (DBMS), housing the application's data. This includes data related to events, vendors, tasks, budgets, users, and other relevant entities. *This acts as a Data Storage layer.*

Communication Protocol:

- HTTP/HTTPS: The communication protocol used between the User's Browser, Web Server, and Application Server is HTTP or its secure variant HTTPS. This protocol is chosen for its simplicity, wide support, and compatibility with web browsers.
- Database Connectivity: For the event application, we have used MongoDB for database connection. Within the application server, communication with the database server typically occurs over a database-specific protocol such as TCP/IP. This protocol enables efficient data retrieval and manipulation, ensuring seamless interaction between the application and the database.

2.3 Persistent Data Storage:

We are using MongoDB as our Database System. We have made collections for all the Vendors and for now have considered 4 major Event Categories - Wedding, Birthday, Sports, Corporate Events. Below is our schema -



sampleVendors.sampleFlorists

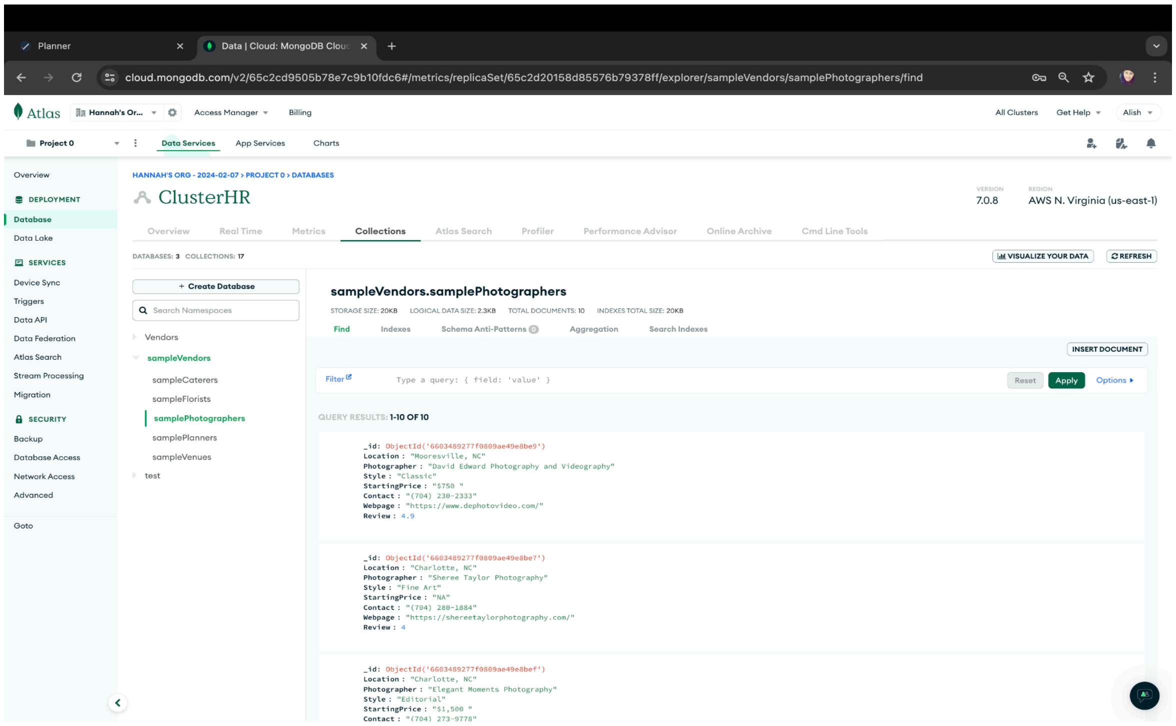
STORAGE SIZE: 24KB LOGICAL DATA SIZE: 2.08KB TOTAL DOCUMENTS: 10 INDEXES TOTAL SIZE: 24KB

QUERY RESULTS: 1-10 OF 10

```
_id: ObjectId('6603488477f0809ae49e8bdf')
Location : "Charlotte, NC"
FloristName : "E & A Weddings Floral, Decor & Planners"
StartingPrice : "$500 "
Contact : "(757) 434-2634"
Webpage : "https://www.weddingsbyemmaandadam.com/"
Review : 4

_id: ObjectId('6603488477f0809ae49e8bdb')
Location : "Charlotte, NC"
FloristName : "Narcisse Greenway Design"
StartingPrice : "$7,500 "
Contact : "(980) 248-4453"
Webpage : "http://www.narcissegreenway.com/"
Review : 5

_id: ObjectId('6603488477f0809ae49e8be2')
Location : "Huntersville, NC"
FloristName : "PK Floral Design, LLC"
StartingPrice : "NA"
Contact : "(980) 228-9552"
Webpage : "http://www.pkfloraldesign.com/"
Review : 3
```



sampleVendors.samplePhotographers

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 2.3KB TOTAL DOCUMENTS: 10 INDEXES TOTAL SIZE: 20KB

QUERY RESULTS: 1-10 OF 10

```
_id: ObjectId('6603489277f0809ae49e8be9')
Location : "Mooresville, NC"
Photographer : "David Edward Photography and Videography"
Style : "Classic"
StartingPrice : "$750 "
Contact : "(704) 230-2333"
Webpage : "https://www.dephotovideo.com/"
Review : 4.9

_id: ObjectId('6603489277f0809ae49e8be7')
Location : "Charlotte, NC"
Photographer : "Sheree Taylor Photography"
Style : "Fine Art"
StartingPrice : "NA"
Contact : "(704) 280-1884"
Webpage : "https://shereetaylorphotography.com/"
Review : 4

_id: ObjectId('6603489277f0809ae49e8bef')
Location : "Charlotte, NC"
Photographer : "Elegant Moments Photography"
Style : "Editorial"
StartingPrice : "$1,500 "
Contact : "(704) 273-9778"
```

2.4 Global control flow:

The online Event Planner application system is a centralized platform to search and send the request to the event manager to plan any event. The system primarily operates in an event-driven manner, where actions are triggered by user interactions, system events, or external stimuli. For example, user actions such as creating an event, searching vendor details, or sending notifications to guests are initiated by specific events rather than following a predefined linear sequence. This approach offers flexibility and allows users to perform actions in a non-linear order based on their needs and preferences. The majority of an event-driven control flow in an event management system enables adaptable user interactions and responsive behavior.

3. System Design

The system Design of any software development project refers to the process of creating a high level architecture to meet the requirements and achieve desired objectives. For system design we need to first understand the functional and non-functional requirements. Moreover, we need to analyze the static and dynamic semantics, quality and syntax required by the project.

System Requirements -

FUNCTIONAL REQUIREMENTS -

- Create a Dashboard to show various cards for event planning categories (Wedding, Birthdays, Corporate
- Event, Sports)
- Create Inventory of Venue, Photographers, Caterers, Planning & Decoration, Checklist & Invitation
- Template to show on UI for each category.
- Create a Login Page for Admins and End Users to enter or request any operation.
- Create a Request button for end users to get customized event plan by admins and attach a form to it.
- Create Alert for Admins and end users for any message received.
- Create a Filter option for all users so as to filter the options by cost/location/category.

External Interface Requirements:

User Interface: The application interface will be designed to afford end users convenient and user-friendly access. It will present a cohesive and easily navigable view of the entire system, facilitating straightforward execution of action-driven processes through clearly labeled, function-specific buttons. Through these design principles, the application aims to ensure that users, particularly the target audience in event planning will experience a high level of convenience and ease of use.

Hardware Requirements:

To be able to run the system, the only requirement should be a Desktop/Laptop/Mobile phone to open the application to browse. The hardware used must have a competent firewall to secure the data in the system (Security - To be implemented later)

Software Requirements:

1. Language Javascript, HTML, CSS, React.
2. Database MongoDb/MySQL.
3. Server NodeJs.
4. Editor Visual Studio.

NON-FUNCTIONAL REQUIREMENTS -

● Performance - When submitting a request by the end user to suggest event details, an alert in the form of a message will be sent to the admin of the application.
● Usability - Ease of use and clarity of the application.
● Scalability - The application is scalable and modifiable by the Administrator.
● Interoperability - The program is based on multiple vendors and their locations
● Reliability - The program is reliable during use and browsing inside.
● Maintainability - The program is maintainable when there is any problem.
● Serviceability - Users can communicate with the administrator when there is any inquiry or suggestion.
● Security - The program will be secured from hacking and viruses.(To be implemented in future)
● Regulatory - Using the application is regular and easy to use.
● Manageability - There is an administrator who manages and supervises the application.

3.1 Static view semantics:

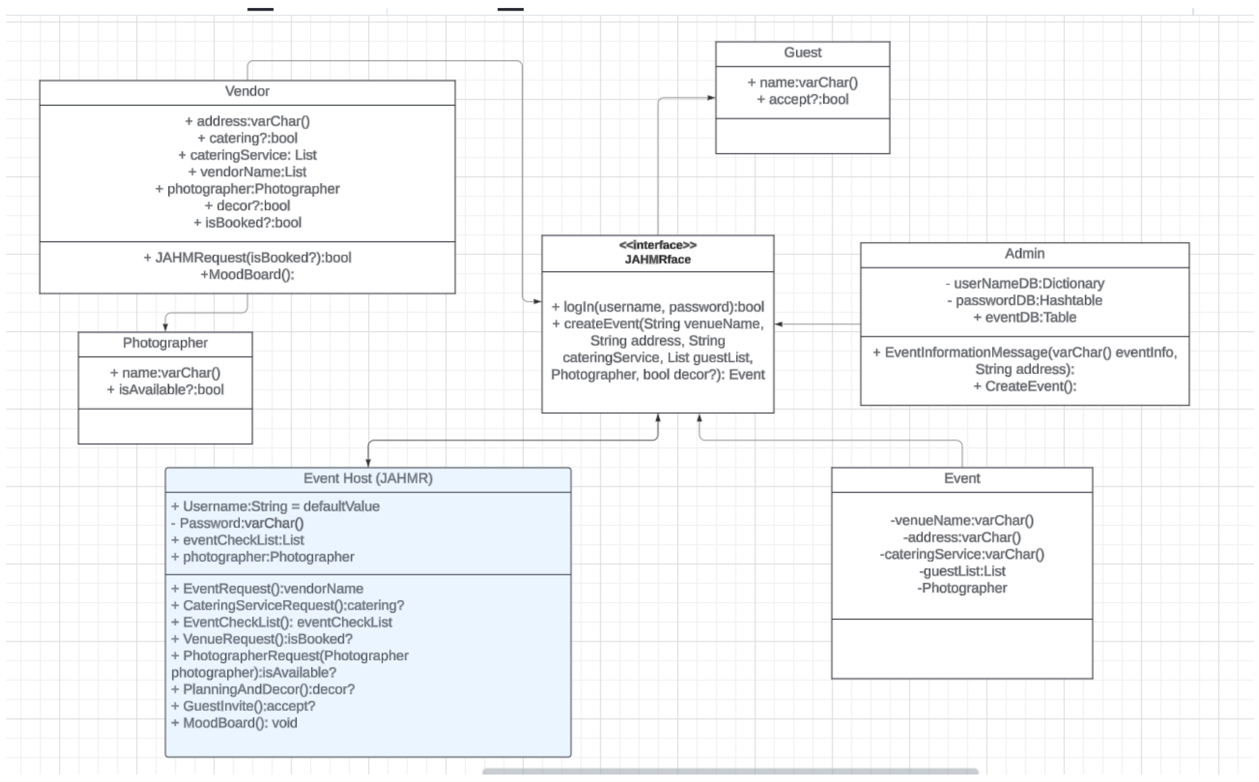
Our system begins with a clear entry point by first clicking on the ‘Online Planner’ link that directs you to each category of event. Through choosing the event type users are presented with a sample list of vendors, florists, caterers, photographers and wedding planners. Each category also has a clickable tile that can show users a detailed list of options to explore. Our system also has contact options for planning services and a review link to further enhance user interaction and feedback. Through the use of modularity we have created a cohesive set of responsibilities as outlined in our UML diagrams ensuring a clear delineation of services. Moreover, the relationships between these classes facilitate a seamless navigation experience for users.

Classes: Venues, Caterers, Florists, Photographers, Planners

Venue Attributes: id, location, name, guest capacity, service type, facility type, starting price, contact, webpage, review.

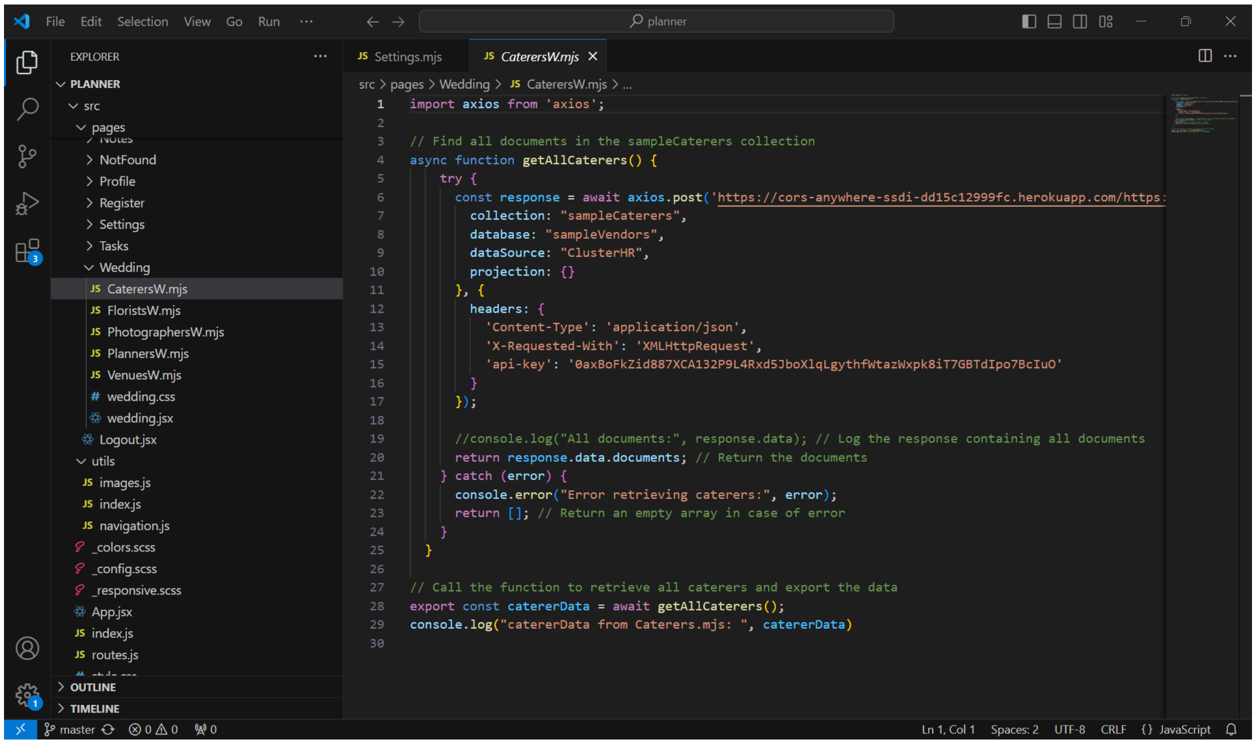
Caterer Attributes: id, location, name, starting price, contact, webpage, review.
Florist Attributes: id, location, name, starting price, contact, webpage, review.
Photographer Attributes: id, location, photographer, style, starting price, contact, webpage, review.
Planners: id, location, name, num services, starting price, contact, webpage, review.

Class Diagram:


Vendor
+ address:varChar()
+ catering?:bool
+ cateringService: List
+ vendorName:List
+ photographer:Photographer
+ decor?:bool
+ isBooked?:bool
+ JAHMRequest(isBooked?):bool
+MoodBoard():
Guest
+ name:varChar()
+ accept?:bool
<<interface>>
JAHMRface
+ logIn(username, password):bool
+ createEvent(String venueName, String address, String cateringService, List guestList, Photographer, bool decor?): Event
Admin
- userNameDB:Dictionary
- passwordDB:Hashtable
+ eventDB:Table
+ EventInformationMessage(varChar() eventInfo, String address):
+ CreateEvent():
Photographer
+ name:varChar()
+ isAvailable?:bool
Event Host (JAHMR)
+ Username:String = defaultValue
- Password:varChar()
+ eventCheckList:List
+ photographer:Photographer
+ EventRequest():vendorName
+ CateringServiceRequest():catering?
+ EventCheckList(): eventCheckList
+ VenueRequest():isBooked?
+ PhotographerRequest(Photographer photographer):isAvailable?
+ PlanningAndDecor():decor?
+ GuestInvite():accept?
+ MoodBoard(): void
Event
-venueName:varChar()
-address:varChar()
-cateringService:varChar()
-guestList:List
-Photographer

3.2 Static view quality:

Static view Quality refers to the clarity , readability and maintainability of the code. When it is not in the changing or executing phase.
Code Readability is clear as the code for our Event Planner Application is written in React.
Code Modularity is well organized as the different modules like Event page, Wedding page and other vendor pages are written separately and then collaborated with the mainframe. Below shows one of the vendor code Modularity and clarity how the data is fetched from the MongoDB. This maintains the quality of our Application.



```
import axios from 'axios';

// Find all documents in the sampleCaterers collection
async function getAllCaterers() {
    try {
      const response = await axios.post('https://cors-anywhere-ssdi-dd15c12999fc.herokuapp.com/https:
        collection: "sampleCaterers",
        database: "sampleVendors",
        dataSource: "ClusterHR",
        projection: {}
      }, {
        headers: {
          'Content-Type': 'application/json',
          'X-Requested-With': 'XMLHttpRequest',
          'api-key': '0axBoFkZid887XCA132P9L4Rxd5JboXlqLgythfWtazWxpk8iT7GBTdIpo7BcIuO'
        }
      });

      //console.log("All documents:", response.data); // Log the response containing all documents
      return response.data.documents; // Return the documents
    } catch (error) {
      console.error("Error retrieving caterers:", error);
      return []; // Return an empty array in case of error
    }
  }

// Call the function to retrieve all caterers and export the data
export const catererData = await getAllCaterers();
console.log("catererData from Caterers.mjs: ", catererData)
```

3.3 Static view syntax:

For all major parts described our static view syntax is correct. In our class diagram all classes and attributes are described and their visibility is shown. Methods are shown also in the diagram with their return types and visibility as well. Relationships between classes are also shown within the class diagram.

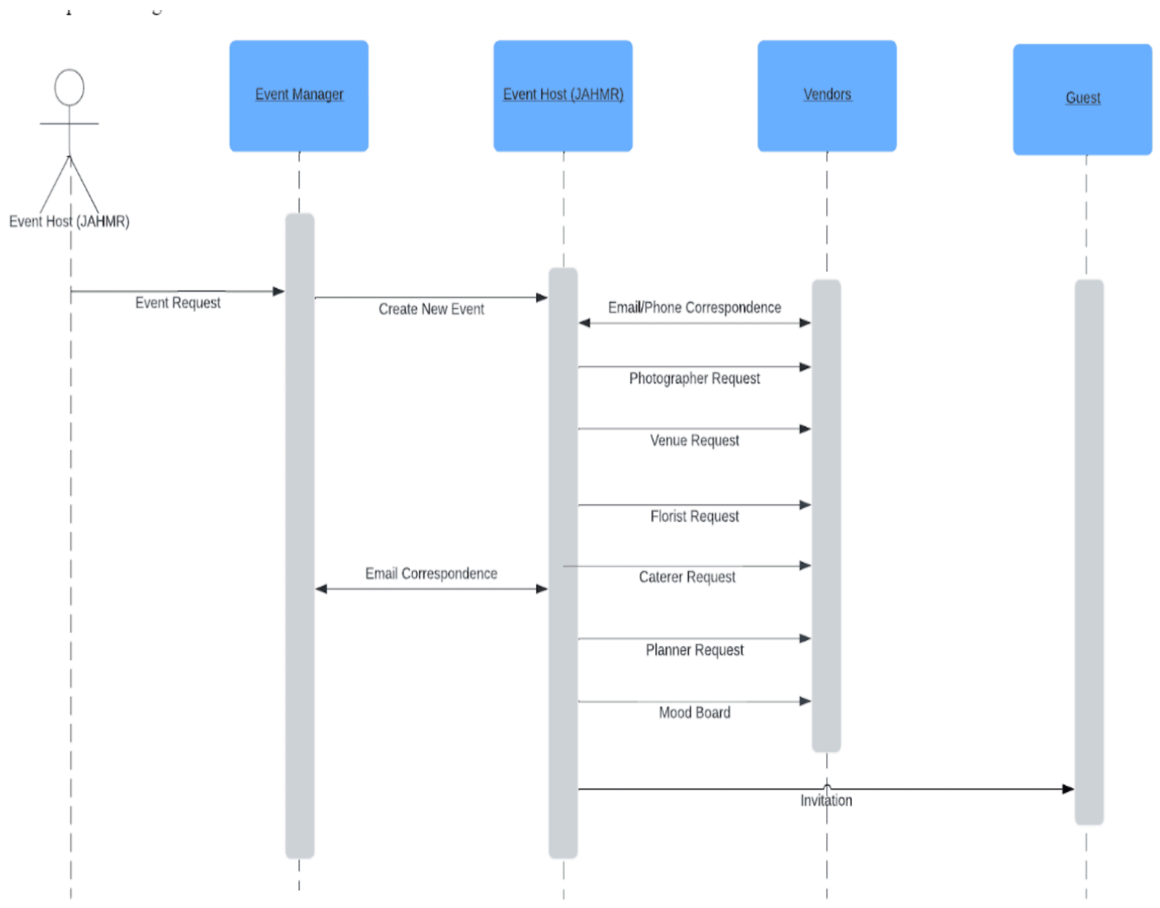
3.4 Dynamic view semantics:

As mentioned, our system begins with a clear entry point by first clicking on the 'Online Planner' link that directs you to each category of event. Through choosing the event type users are presented with a sample list of vendors, florists, caterers, photographers and wedding planners. Each component is responsive and the runtime of the execution of any activity is not more than 10 sec. Also, the responsiveness of the login and signup page is quick and easy.
The Event Handling is proper as the modules fetch from the database and also our application is interactive so that any new event is requested, there is proper consistency and synchronization. The concurrency of the Application is matched and Events work parallel to each other. Our Application fetches the data currently restricted to North Carolina Vendors and the database can adapt the information provided and also it dynamically fetches it from the DB.
Classes: Venues, Caterers, Florists, Photographers, Planners

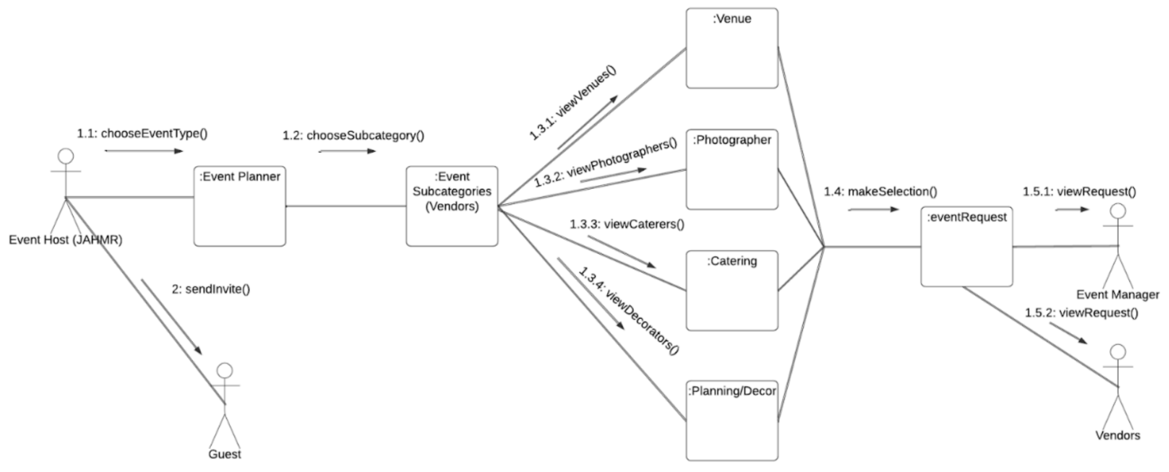
Venue Attributes: id, location, name, guest capacity, service type, facility type, starting price, contact, webpage, review.
Caterer Attributes: id, location, name, starting price, contact, webpage, review.
Florist Attributes: id, location, name, starting price, contact, webpage, review.
Photographer Attributes: id, location, photographer, style, starting price, contact, webpage, review.
Planners: id, location, name, num services, starting price, contact, webpage, review.

For more information, refer below sequence Diagram:

Sequence Diagram



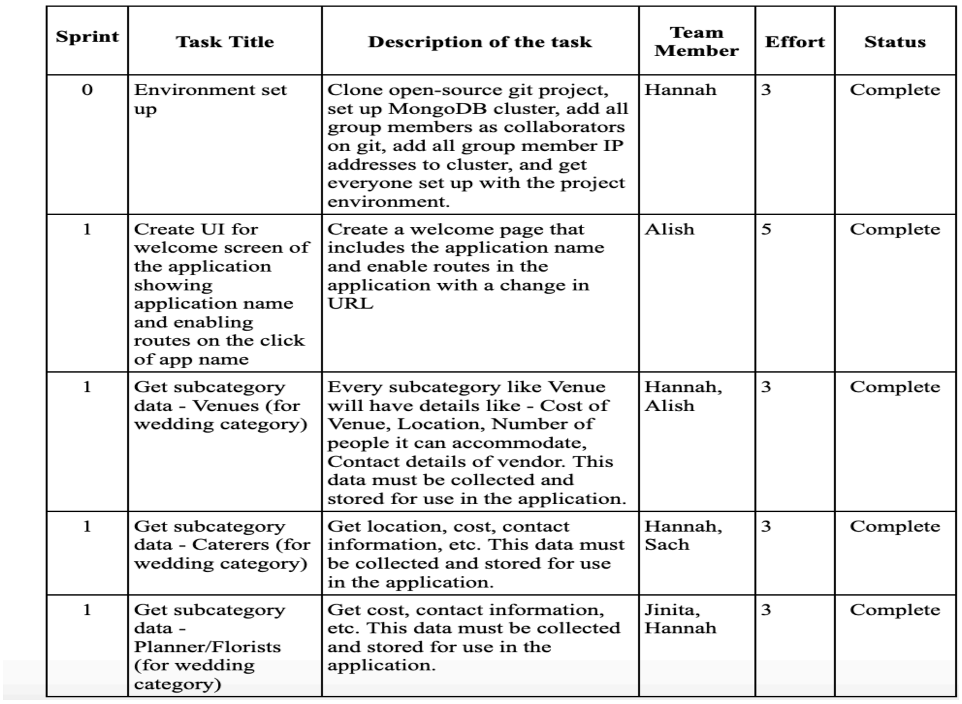
Communication Diagram:



3.5 Dynamic view syntax:

All the major parts of our system are syntactically correct. Our sequence diagram uses the correct arrows, execution occurrence boxes, and messages that correspond with the interface. All classes are present from the static view as well.

Previous and Current Sprints -



Sprint	Task Title	Description of the task	Team Member	Effort	Status
0	Environment set up	Clone open-source git project, set up MongoDB cluster, add all group members as collaborators on git, add all group member IP addresses to cluster, and get everyone set up with the project environment.	Hannah	3	Complete
1	Create UI for welcome screen of the application showing application name and enabling routes on the click of app name	Create a welcome page that includes the application name and enable routes in the application with a change in URL	Alish	5	Complete
1	Get subcategory data - Venues (for wedding category)	Every subcategory like Venue will have details like - Cost of Venue, Location, Number of people it can accommodate, Contact details of vendor. This data must be collected and stored for use in the application.	Hannah, Alish	3	Complete
1	Get subcategory data - Caterers (for wedding category)	Get location, cost, contact information, etc. This data must be collected and stored for use in the application.	Hannah, Sach	3	Complete
1	Get subcategory data - Planner/Florists (for wedding category)	Get cost, contact information, etc. This data must be collected and stored for use in the application.	Jinita, Hannah	3	Complete

1	Get subcategory data - Photographers (for wedding category)	Get photo style, costs, contact information, location, etc. This data must be collected and stored for use in the application.	Micheal, Hannah	3	Complete
2	Create UI for displaying events (clickable cards for all 4 categories)	The homepage of the event section should include 4 tiles displaying each event type: Weddings, Birthdays, Sports, and Corporate Events. Clicking on the wedding tile should bring the Event Host to a more specific page of event details for that type of event.	Alish	5	Complete
2	Create UI for displaying subcategory details (for wedding category)	This is the next page the Event Host will see once selecting an event type. This page needs to have the venue, caterer, planner, florist, and photographer tabs that can be selected and explored further to make selections.	Jinita	8	In Progress
2	Create an updated Google form	If the Event Host does not want to look through vendor details and wants suggestions from the Event Manager, they can fill out a Google form. They can choose which vendors they need from a checkbox and specify any details/criteria such as budget, cuisine, etc.	Hannah	3	Complete
2	Mood board: give the Event Host an option to paste a link in Google form	Allow the Event Host to link a Pinterest board of the event aesthetic they like. This can then be sent to the decor vendor to create a look that will meet the Event Host's expectations.	Micheal	3	Complete
2	Create a Login Page for Event Managers only.	This is where Event Managers can see and manage the application.	Sach	3	Complete
3	Starting middleware integration	Connecting the UI and database via middleware	Jinita, Hannah	8	New
3	Create a 'Request Event Plan' button for the Event Host.	Event Hosts need to get customized event plans from Event Managers. The request button should open up the Google form. The Event Manager will be able to see the Event Host's choices, contact vendors, and begin planning the event based on the Event Host's specifications when they receive the Google form in the Event Manager email.	Micheal	5	New
3	Create UI for displaying subcategory details (for wedding category)	This is the next page the Event Host will see once selecting an event type. This page needs to have the venue, caterer, planner, florist, and photographer tabs that can be selected and explored further to make selections.	Alish/Sach	8	New
3	Create a subsample of data	Create a subsample of around 10 database entries for each table to test the complete application	Hannah	3	New
4	Create a search on UI to search for the specific name of the vendor	Create filters to search for any vendor-specific information like name	Alish	5	New
4	Complete documentation	Create overall documentation to show the functionality	Jinita	3	New
4	Work on cosmetic changes on UI	Work on pending UI changes to give a better look and feel to the application	Sach	5	New
4	Allow the Event Host to Leave a Review	Create a "Post a Review" button for the Event Hosts who have already successfully completed the event and want to leave their feedback for Event Manager/Vendors/Application	Micheal	5	New

Sprint Review - In every Sprint Review, we as a team work together on below points -

- Work upon the Backlog item to take in next sprint
- Recheck the story estimations and efforts for pending stories.
- Check upon the code quality for achieved user stories.
- Check if all the user stories for current and previous sprint have been done and marked.
- Check if the requirements and software that we are building is in sync.
- Update the documentation if required.
- Discuss about the feedback that we have received in the current sprint from TA's and how to implement it.

3.6 Design Rationale:

When designing the Wedding Planning Application other architectures were considered such as Microservices, Layered and Serverless. However MVC was our clear choice because of its clear separation of concerns and its suitability for our web-based application. The separation of Model, View and Controller allowed us to maintain modularity making it easier to manage different aspects of the application independently. MVC provided us with a structured and standardized approach to handle different components efficiently, ensure code consistency, and since we were familiar with this architecture it allowed us to streamline the development process.