Introduction and Goals

The product being developed is to be an MVP (Minimum Viable Product) of an app for Children Learning and Wellness Centers. Further, the product is to be run on easily accessible platforms like Android and iOS and is supposed to be easy to use for an everyday/casual user (parent). The requirements and driving forces of the software to be considered include the following:

- The main goals of the products are:
  - Reach a larger customer base
  - Increase in sales of services
  - Increase the customer base via ease of access
- Essential features include:
  - Login
  - Search
  - Do a booking
  - Pay
- Essential functional requirements include:
  - Authentication using email & password as the (key, value) pair
  - Editing details in the profile & updating in database
  - The payment is integrated with an external payment gateway
  - Check for availability of centers in entered time slot
- Quality goals for the architecture include:
  - Functional Stability
  - Operability (Useability)
  - Security
  - Maintainability
- Relevant stakeholders and their expectations:
  - **Business Franchise**: Their expectation of the product is to serve as another form of booking means. They would expect this to be easy to manage as well as to work alongside their already existing means of booking. They would also want a robust app to appeal to the customers and increase their sales.
  - **Parents (User)**: Their expectation from the product is to make it easy for themselves to do bookings of centers. They would want the process to be smooth, robust, and secure.

Requirements Overview

Contents

Functional requirements include the following:

- Safe Authentication
  This includes storing the username and password of the user in our database and

validating the same before letting a user login. To make the process "safe," we store the password in the database after encrypting it. We use the safe and standard **bcrypt** library for doing this. The API calls are made only after verifying the source from where the call is being made, hence making this process of interacting with the database more secure.

- Editing and updating details
  The user must be allowed to edit and update their details after creating an account on our platform. Apart from essential details like email, which is used to uniquely identify a user, most details about an account need to be updateable, especially features regarding adding and removing children under a user's account.

- External payment gateway
  The payment feature must be integrated with an external payment gateway for handling transactions. The app receives response about payment and decides its course of action accordingly depending on whether the payment was a success or not. The reason for using an external payment gateway is because we want the process to be robust and reliable.

Motivation

The motivation behind this product is to make the system of booking learning and wellness centers smoother and easier for customers. At the same time, the product also increases the customer base for the franchise by appealing to them with a more accessible and flexible way of making bookings.

Form

Please refer to the following documentation for specific use cases and descriptions:
Software Requirements Specification

An exhaustive set of use cases has been given in the last 2 pages of the above linked document.

Quality Goals

Contents

The most important quality goals for the architecture of this product are:

- Functional Stability
  This is a very core requirement expected from the app. We are required to ensure the stability of our functional aspects as this is an app built to be used by the masses. As an app used for providing service, stability must be ensured to make it a worthwhile experience for the user.

- Usability
  As a system that is used to attract customers, usability is necessary. Ease of access and simplicity in learning must be guaranteed to appeal to a large customer base.

- Security
  As several sensitive information like family details, locations, and passwords are involved, security must be guaranteed by the app so that this information may not be used by a third party for malicious use. Security and robustness must be guaranteed for the payment aspect of the app as well.

- Maintainability
  Modularity and modifiability of the source code must be guaranteed, this is because eventually some functionality might need to be modified based on user feedback/for introducing new features.

Motivation

It is important that complete and valid feedback is taken from all the stakeholders of this application. Below are the listed motivations for each stakeholder as to what they require and expect out of this project

- Franchise
  The franchise is concerned with the quality goals mentioned above to ensure customer satisfaction. Maintainability is a unique quality goal that they care about unlike users, as it involves rolling out more updates and modifying the current software.

- Parents (Users)
  The parents are concerned with functional stability, usability, and security of their information. They expect the software to be stable in its performance, providing the services without running into unexpected issues. A general user also wants the app to be easy to learn and use, so that they can use it. Security is a quality goal that is a concern of most users because their information can be easily misused by a third-party attacker.

Form

A table with quality goals and concrete scenarios, ordered by priorities

Quality Goals	**Concrete Scenario**	**System Response**
Functional Stability	There is an issue with the database/backend server and the sent request fails.	The system considers the request has been made and changes its course of action accordingly to ensure the functionality does not have abrupt results.

Usability	A new user installs the app. The user is not very literate when it comes to using digital services like apps.	The app guides the user properly around the app with a simplistic approach and tabbed view.
Security	A third-party attacker tries to make API calls to the backend for fetching data of other users.	The backend validates the source of API call and only lets valid and authenticated sources access the database.
Maintainability	The franchise wants to roll out an update introducing a new feature in the app.	Given the modularity of the system, code is very readable and modifiable. It becomes easy for the franchise to add a new feature and/or update an older feature.

Stakeholders

Contents

Stakeholder overview:

- Client:
  First and foremost, we must keep our client in the loop about our architecture. Since the product is an MVP, we must first convince our client of the robustness and working of our architecture as they will be taking it forward.

- Franchise:
  The Franchise for which this app is being developed needs to be kept in know regarding the software architecture. As they are the main group who will be dealing with the app after being deployed, their approval is important for the architecture as well.

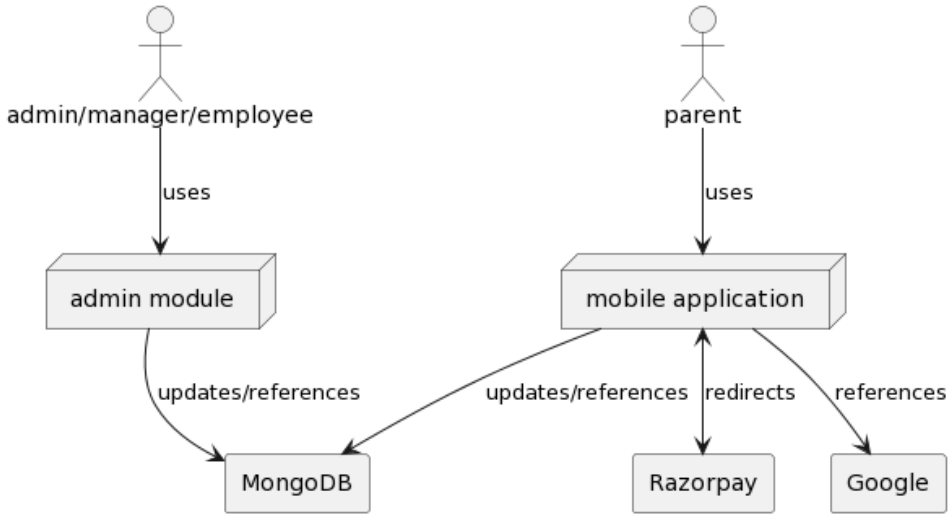
- Development Team:
  The App Development Team needs to be very well versed with architecture as well so that they can provide their feedback from a developer's point of view. Also, since they are the ones who will be implementing the architecture, they need to be aware of that.

- Users:
  While not involved in app development, they are directly affected by the architecture of the system. The architecture must guarantee that the interests of the user are fulfilled properly.

Architecture Constraints

Constraint	Explanation
No outside access to backend	Since we are dealing with sensitive data, it is imperative that the customer data is protected.
Cross-platform over IOS and android	The client specifically asks us that they require the application to be cross platform to service as many customers as possible
Built using open-source or free technologies	Since this is project is primarily for educational purposes, we will not be able to spend money on licenses
Minimal backend and server hosting	Due to less budget, we need the website to be able to be hosted on a personal computer

System Scope and Context



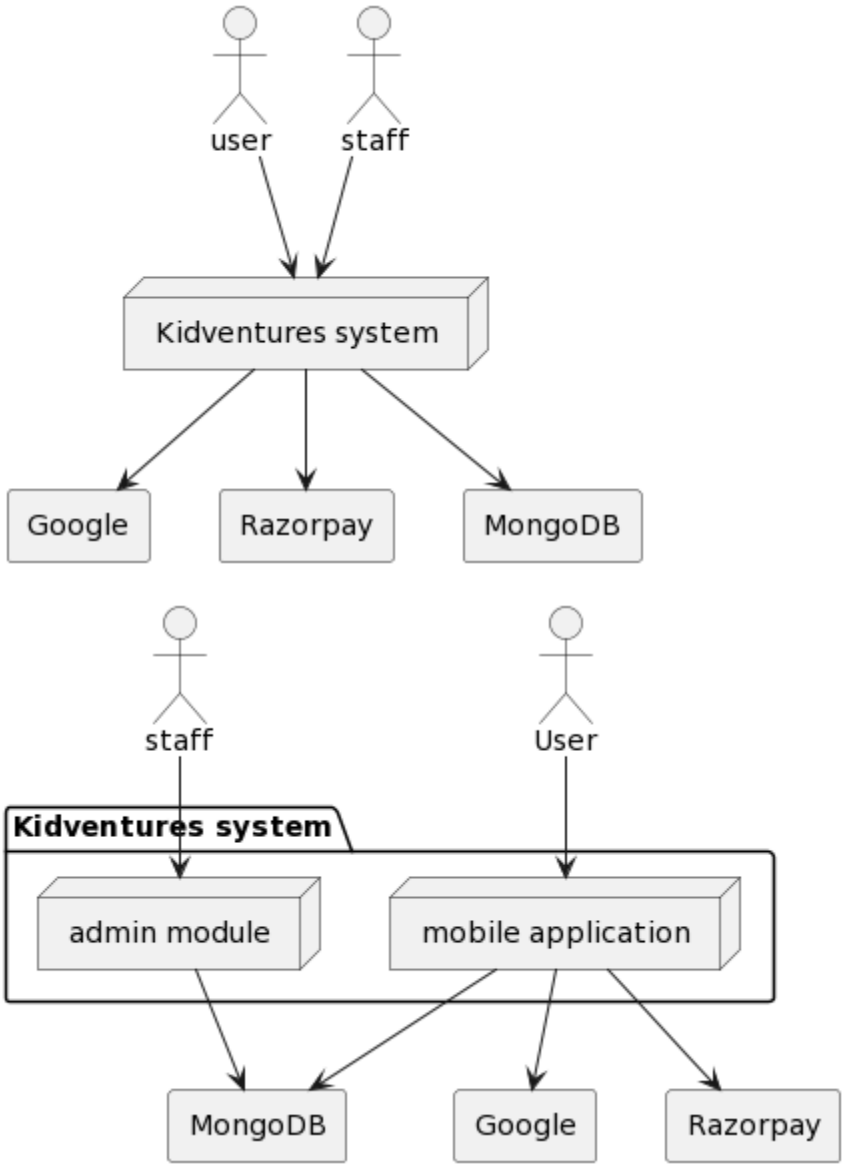
Neighbour	Description
User (parent)	Makes bookings and add details of children
Admin	Administers the complete system
Manager	Manages a particular centers detail
Employee	Uses system to gather information
Mongo DB	Acts as primary data base for our system
Razor Pay	Acts as primary transaction handler
Google maps API	Used to determine distance of user from the centers

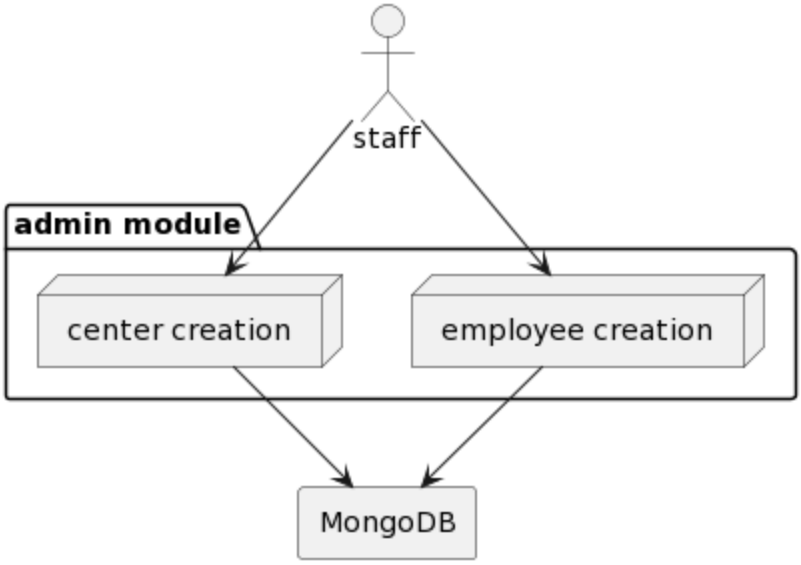
Solution Strategy

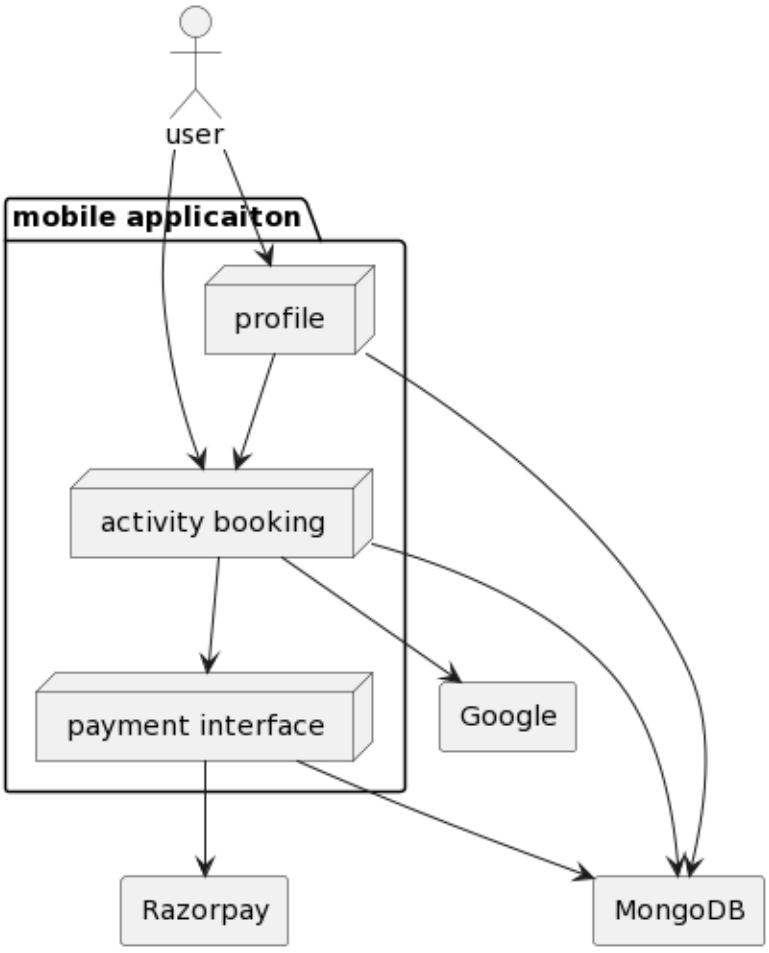
1. Implementing the administration interface as a website. This was done primarily because of its convenience to the office/franchise workers as it is easier to manage databases on a larger screen as opposed to a handheld device

2. Using MongoDB instead of firebase, primarily due to the experience of the dev team with this technology and for its convenient JSON data structure and compatibility of MongoDB with Web Development.
3. Delegating the payment interface and operations to Razor pay API as it is a more secure and reliable source to handle transactions
4. Encrypting passwords and introducing middleware between frontend and backend as our system deals with very personal and confidential information, thus it is important that data breaches are minimal

Building Block View







Level 1

Admin module

- The purpose of the admin module is to provide a convenient graphical interface for the staff of Kid Ventures franchise to manage their database and add or remove information
- It only interacts with the MongoDB database. Thus, it indirectly interacts with the mobile application through the database

Mobile application

- This is the main module through which the users interact with our system. It provides functionality of looking at previous bookings and making new bookings
- It interfaces with many external applications and APIs like Razor pay, Google maps and mongo DB
-

Level 2

Breakdown for Mobile application

- Payment module
    - This module is responsible for providing the user details of the bill and the subsequent breakdown along with providing a secure payment interface
    - This module interacts with the Razor pay API and is accessed by going through the activity module
- Profile
    - The responsibility of this part is to display information about the user and allow them to edit/add details
    - It interacts with the MongoDB database and is used directly by the users
- Activity module
    - This is responsible for taking the user through the process of making a booking through the app and then redirecting them to the payment interface
    - It interacts directly with the user and references the mongo DB database. It also uses the Google maps API to provide location information in showing centers. It goes into the payment module

Breakdown for admin module

- Employee creation

    - This module is broken down into 2 parts. One for creating managers and one for creating normal employees

- Centre creation
    - This is responsible for allowing the staff for entering the details of a new location to be added to the database
    - It uses the mongo DB cluster and is used directly by the staff

Runtime View

We describe the following 4 use cases

- Making a booking for children
    1. Parent selects the children he wants to make booking for
    2. They select the date and time and duration of the booking
    3. They select the activities for that booking.
    4. They select the center
    5. They are redirected to the razor Pay interface to confirm payment
    6. They are redirected to a payment confirmation page on successful payment
- Admin wants to add a center
    1. Administrator logs into admin module with proper credentials
    2. They add details of the center
    3. They assign a manager for managing this center
- Booking a party
    1. Parents choose the number of adults and children attending the party.
    2. They select the date, time, and duration of the party.
    3. They pick a center from the available list sorted based on distance from current location.
    4. They are redirected to the razor Pay interface to confirm payment
    5. They are redirected to a payment confirmation page on successful payment
- Admin wants to update the activity prices

1. Logs into the administration website with required credentials.
2. Open the Edit screen for prices.
3. Update the prices for each activity.

Contents

1. Making a booking for children
   This is executed by fetching the required details (like list of children, price of activities etc.) from the database and serving it on the app so that user can select easily. This is followed by another call to a Google Geolocator API which calculates distance between center and current user location to serve the list of centers on app from which choice can be made. App asks users to co-operate by giving permission for location services if not already given at this point. Some errors and exceptional scenarios can be because of failed API calls which do not fetch the required data. This can lead to wrong billing.
2. Admin wants to add a center
   The system accepts input from the admin, and then makes a direct API call to the backend to add that data to the database. This database is also used to serve data on frontend app, so it makes sure the center is also up and visible after being added.
3. Booking a party
   Similar to Activity Booking in point 1, this does not require the system to fetch the list of children. However, some adults and children must be given. An exception can be made if the user can more than set the value (which is 10), in the number of adults/children somehow. In this scenario, data present in the database does not adhere to the expected requirement data anymore and may cause issues.
4. Admin wants to update activity prices
   Similar to (2), admin enters the details which are then posted in an API call to the backend. This data is then pushed to the database and the required fields are updated. Since this is a sensitive area as it affects the billing, a huge case of exception can be made when price is shown to be updated on admin side but is not so on the user side.

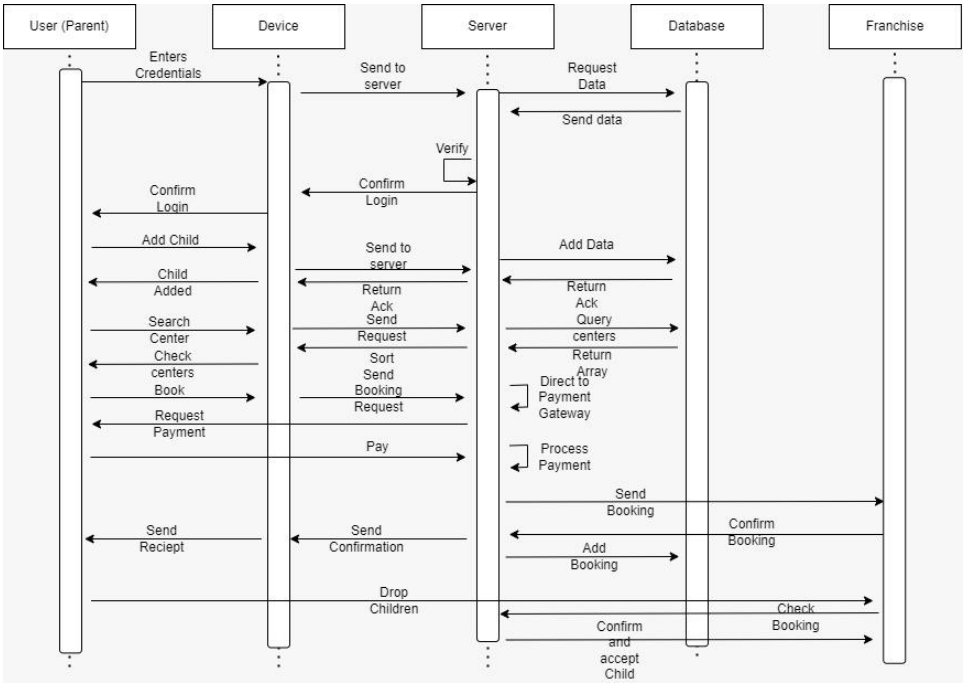
Motivation

Motivation between architecture behind runtime scenarios is that changes and updates need to be guaranteed when shown successful and the data input/fetched should be in the required constraints (like max 10 is constraint for number of adult/children). Amongst all the scenarios, another similarity that can be drawn is the requirement of security which now becomes a huge concern because issues regarding money transfer and personal data should be handled carefully. One other thing is that neither the average staff at a center nor the user go out of their way to learn how to use the app. The app is required to be simplistic and ask for as little from the user as possible in this case.

Cross-cutting Concepts

1. Security must be embedded in the system. The architecture of the product must ensure that a breach of security does not happen.

2. Our domain model is available here :



3. The design pattern caters to the idea that the app must be simplistic, intuitive, and visually appealing. In some cases, design choices to make it simpler might be made at the cost of some level of flexibility of functionality.