Child Care Service Application for iOS devices

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

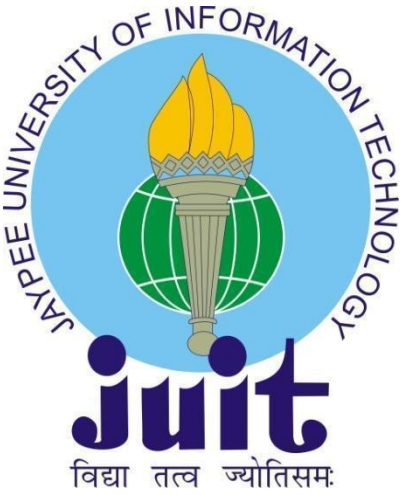
in

Computer Science and Engineering/Information Technology

By

Rijul Sharma (191336)
Under the supervision of

Dr Pradeep Kumar Gupta & Dr. Nafis Uddin Khan
to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Child Care Service Application for iOS Devices "** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Prof. Dr. Pradeep Kumar Gupta** (Professor, Department of Computer Science and Engineering) and **Dr. Nafis Uddin Khan** (Assistant Professor (SG), Department of Electronics and Communication Engineering)
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Rijul Sharma, 191336.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Prof. Dr. Pradeep Kumar Gupta
Professor
Department of Computer Science and Engineering, JUIT
Dated:

(co-Supervisor Signature)
Dr. Nafis Uddin Khan
Assistant Professor (SG)
Department of Electronics and Communication Engineering, JUIT
Dated:

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ __Department: _____ Enrolment No _____

Contact No. _____E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Almighty God for his divine blessing that makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Pradeep Kumar Gupta** , **Professor**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of **"iOS Development"** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Pradeep Kumar Gupta**, Department of CSE, for his kind help to finish my project as well as Dr. Nafis Uddin Khan, Department of ECE for his guidance.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Rijul Sharma (191336)

TABLE OF CONTENTS

Title	Page no.
List of Abbreviations	vii
List of Figures	viii
List of Tables	ix
Abstract	x
Chapter-1 Introduction	1
Chapter-2 Literature Review	8
Chapter-3 System Development	9
Chapter-4 Experiment & Result Analysis	39
Chapter-5 Conclusions	46
References	48

LIST OF ABBREVIATIONS

1. iOS: iPhone Operating System
2. UX: User Experience
3. JRE : Java Runtime Environment
4. MVC: Model View Controller
5. OS: Operating system
6. MVVM: Model View View-Model
7. SQL: Structured Query Language
8. UI: User Interface
9. OTA: Over-the-Air
10. MVC: Model View Controller

LIST OF FIGURES

S No.	Figure No.	Description	Page no.
1	3.1	Parents Panel Path	14
2	3.2	BabySitter Panel Path	14
3	3.3	Admin Panel	15
4	3.4	front end of sign up, verification and sign in screens	16
5	3.5	user authentication diagram	15
6	3.6	Specification requirement screens	18
7	3.7	Payment functionality in customer panel	19
8	3.8	feedback functionality of the application	20
9	3.9	Nanny Panel Location Screen	21
10	3.10	Job card screen in nanny panel	22
11	3.11	Screens that list requests of families as well as their details	23
12	3.12	Add Account screen for babysitter	24

13	3.13	Edit Profile screen for nanny panel	25
14	3.14	StoryBoard Architecture	29
15	3.15	Flow of requests and response from client to server	32
16	3.16	List of Firebase features	36
17	4.1	Running the app on iphone 13 and iphone 14	41
18	4.2	Testing data for nanny profiles	42
19	4.3	Payment Functionality testing	43
20	4.4	Customer Support for admin	44
20	4.5	Customer Support in admin panel	45
21	4.6	Testing dummy data in Admin Panel	46

LIST OF TABLES

Index no.	Title	Page no.
1	Technologies used for application	6
2	List of screens of Parent's panel	11
3	List of screens of nanny panel	12
4	List of screens of admin panel	14

ABSTRACT

This article begins with an introduction to the functionality of the Child Care Service application. Next, it describes the application architecture based on the Model-View-View Model (MVVM). The document explores the two different panel flows of the application, for it’s two different types of users, i.e. parents(clients) and babysitters (service providers).

The document also explores various applications, including components, views, data structures and activities, and the relationships between them.

The following document details the app's features and functions, including user authentication, customer engagement, booking, and payment. The document also describes how to use key libraries and frameworks in app development, including Xcode StoryBoard, CocoaPods, Firebase etc

The document ends with a discussion of the challenges and lessons learned in the application development process. It also identifies the future potential of the app and makes suggestions for future updates and additions.

Chapter - 1
Introduction

1.1 Introduction

In this fast paced society, where the emergence of nuclear families and working women has significantly increased, there is a dire need of effective, reliable and convenient baby-sitting services. Such baby-sitting services can be provided “one click away” to maximise convenience for the parents. This project is an application developed for Apple Devices particularly iPhones that provides these babysitting services to parents both for short term and long term. This application seeks to provide an easy interface where the potential clients (parents) can meet baby-sitters as per their requirements.

The application will hold a basic client-service provider structure where the babysitters can make their own profiles, edit their profiles, search for families near their location, accept or reject babysitting requests from families, get in contact with the clients and much more.

The application for the customers (parents) will allow several functionalities such as setting up location, specifying their needs, matching with the available babysitters near them, getting in contact and make secure payments.

This project has a huge scale and required multiple panels, technologies as well as databases for implementation. The project is inspired by the economic as well as cultural impact of shifting services online for the ease of accessibility for both clients and service providers.

The project explores several aspects ranging from aesthetic user interface designs, technological aspects as well as the economic capacity of Baby Sitting E-services.

1.2 Problem Statement

The challenge and stress parents experience when trying to find a dependable and trustworthy babysitter for their children is the issue we are attempting to solve with our baby-sitting software project. Finding a reliable babysitter on short notice can be difficult and time-consuming for parents who have hectic schedules and need someone to watch their kids.

One of the more traditional methods of finding a babysitter is asking friends or family for recommendations, however this method can be limited and unreliable. Online employment forums and postings can be daunting and often require a full review to ensure children's safety and well-being.
T

1.2 Objectives

The objective of this project is to provide a user-friendly interface and provide a platform where clients (parents) can easily connect with service providers(babysitters). The objective of this application are as follows:

- Assist parents in finding dependable babysitters who adhere to strict requirements and have undergone background checks and safety screening.
- Offer a platform that enables parents to look for and book last-minute babysitters with ease.
- Provide tools for babysitters to manage their schedules and reservations as well as the option for them to create profiles with their qualifications, experience, and availability.
- Offer a rating and review system so that parents can share their opinions about their experiences with babysitters. This can assist the quality of care to be improved over time.

- Make it simple for parents to pay for babysitting services via the app, and give babysitters access to a safe and practical payment method.

1.4 Methodology

The approach employed in the project will be covered in detail in this section. For the front-end mobile application of this application, we used the Swift and Xcode programming languages.

There are many ways to create and distribute iOS applications. The iOS development methodology, however, is what we've opted to employ for our project. Consequently, we are able to exert more control over the creation process and benefit from the rich features and capabilities offered by Apple's iOS platform.

An individualised, scalable mobile application developed with iOS development can satisfy the specific goals and specifications of our project.

The initial step in our project is to gather the relevant data and requirements after choosing the platform and framework. In this phase, user requirements are identified, necessary features are decided upon, and the project's scope is established. The next step is design and wireframing, when we concentrate on developing UI/UX designs for the application. To achieve this, the application must be conceptualised and prototyped, user flows must be defined, interface functionality and user experience must be built, and an aesthetic that meets the needs of the application and its target audience must be developed. We follow a set of criteria to do this, such as identifying target users through user surveys and feedback collection. Additionally, we prefer a straightforward, uncomplicated structure over sophisticated navigation and cluttered interfaces.

We must adhere to the following guidelines in order to do this:

2. Simple Design: It is important to make sure that the design is minimal and simple so that customers of all age groups can easily make.

3. Design consistency: Designing a successful application requires maintaining consistency. We must use the same layout , patterns and colour palettes through the app. Because of the uniformity, users will be able to recognise and operate the programme with ease, feeling at ease and comfortable with the interface.
4. Using the appropriate colours: Selecting the appropriate colours is crucial to developing a user interface that works.

Table 1.1 shows the technologies stacked for implementation of the application.

Table 1.1 Technologies used for application

Swift, Xcode	Front-end framework
Node.js	Back-end
Express.js	API
Stripe	Payment gateway
Firebase	Back-end authentication

Once the interface of the application is created and the user needs are understood, the next step is to determine the technology requirements for the project. This includes selecting the technological stack. Swift is the main technology used for developing the mobile application for iOS users. Additionally, Node.js is used for API connectivity and Firebase is used for backend services. Google's Firebase platform provides a wide range of backend capabilities that enable developers to create apps with greater efficiency and speed. Programmers can leverage real-time cloud communication to distribute

warnings to consumers across numerous platforms, including iOS and Android, the web, and Firebase features like authentication and cloud messaging. The most important phase of writing the code begins after the technologies are chosen. This entails developing the front-end and back-end of the programme, integrating APIs, and testing it for bugs and other difficulties. To make sure the application meets its functionality and operational requirements, testing is done throughout the testing and quality management phase. Documenting all work done throughout this phase and making sure the source code is flexible and managed to handle upcoming updates and improvements are essential.

Chapter-2
Literature Survey

A literature review on babysitting apps for iOS users reveals that there is a growing demand for such apps, as parents look for ways to manage their busy schedules while ensuring their children's safety. A study[1] by InMobi found that over 80% of parents with children aged 5 and under used mobile devices to assist with parenting tasks, including finding babysitters. rely on user ratings and reviews to assess capable babysitters.

Since it's crucial to take into account security, privacy, and dependability, families should only use safe and dependable apps. Numerous studies have emphasised the significance of putting robust protections in place to protect kids and their families, making privacy and security other essential issues to take into account. For instance, researchers have advocated for improved identification verification processes, background checks, and secure messaging technologies to ensure safe and secure communication between parents and babysitters. Babysitter apps offer a lot of promise to enhance the childcare experience by granting parents more convenience and flexibility and by providing sitters with career opportunities and flexibility, according to the research review.

[2] Mrugank Gandhi, Shubham Kothavade, “Development of a iOS Mobile Application for online Babysitters,” IRJMETS [2022]

With the use of this app, parents may connect with nannies, babysitters, and pet sitters. Based on their unique requirements and preferences, parents can use this app to identify and employ caretakers. It allows parents to create an account and post a job listing outlining their childcare needs. They can outline the caregiver's necessary schedule, working environment, and any other requirements they might have. The software then connects a job posting to nearby carers who meet the criteria. On this app, each carer has a personal account with a profile that

details their training, qualifications, background checks, and company endorsements.

Before choosing a nanny, parents can browse these websites, get in touch with potential candidates, and schedule appointments or interviews to verify compatibility. Background checks are among the other services it offers, which may reassure parents.

[3] Yogita Masare, Sneha Mahale, Manjusha Kele, Ashvin Upadhyay, Bhushan R. Nanwalkar, “Subscription-based babysitting service,” IJERT [2021]

Using this mobile app, families may locate babysitters nearby. It is designed to give parents a quick and dependable way to find and book reputable babysitters for their children. Several American cities broadcast it on television. Using smartphone applications, people may create accounts and look for suitable nannies in their area. The application contains information on each nanny's background, experience, degree of experience, and family ratings checks. Parents can find out whether the nanny has received any recommendations from friends or other reliable sources.

Using the software, parents can request applications from potential caretakers. The babysitter can then choose whether to accept or decline the request based on their availability. The app also has features that let parents arrange interviews, make payments, and communicate with babysitters. One of this app's standout features is the social connection functionality. As a result of having the chance to meet babysitters who come highly recommended by their friends, parents can develop confidence and trust in the person who will be looking after their children.

[4] Vrushali C. Waikar, Sheetal Y. Thorat, A. A. Ghute, and Priya P. Rajput, Mahesh S. Shinde, “Connecting Families with Caregivers Application for Android Mobile Application,” IRJET [2020]

Parents can connect with qualified babysitters who have completed rigorous screening using this technological platform. It offers childcare services that are available on demand, making it convenient for parents to find reputable nannies when they need them. Parents who wish to use the application's gateway to book babysitters must pay a monthly or annual subscription fee. The platform aims to give parents peace of mind by doing thorough background checks, reference checks, and interviews with potential babysitters.

[5] Noof A. Al-Safi, Rawan A. Al-Asiri, Malak A. Al-Malki, Sameera Abar, “Fostering Childcare E-Service: Design and Development of a Software Application,” IJITN [2022]

The IJITN released this research paper. In order to give parents access to an online childcare service where they could promptly track, supervise, and take care of their children from a distance, they created a vibrant mobile application for babysitting. Performance evaluation of the integrated features and functionality of the suggested strategy's integrated features and functionality confirms its usefulness and ease in fostering good parenting for young children. Additionally, this research offers helpful references in a variety of sectors for students who are creating their own software applications.

Chapter - 3
System Development

3.1 Analysis

An examination of the project's technological viability should be done before designing a babysitter app. The needs for the app, including its features and functionalities, should be evaluated as part of this analysis. The analysis should also take into account the technological limitations, such as the mobile platforms on which the app will run and the app's device compatibility. The cost of development, including the resources and time needed to create the app, as well as any potential security or data privacy concerns that need to be handled, are additional aspects to take into account in the technical analysis. Developers can detect potential issues or challenges that may arise during development and take action to address them before starting the project by doing a thorough technical analysis. This can ensure that the app is created successfully and efficiently, and that it satisfies the requirements of its consumers.

For handling the backend of a babysitting app, Firebase might be a fantastic choice. The real-time database function would be especially helpful for keeping track of babysitting sessions because it will let parents and sitters see updates right away. A further degree of protection might be added by using the authentication feature to make sure that only allowed users can access the app.

Additionally, the hosting and cloud capabilities of Firebase can aid the app's performance and scalability, enabling it to manage a high amount of requests and users. This is crucial for a babysitting app since it must always be accessible and responsive.

On the other hand, the server-side runtime environment Node.js enables developers to build web apps that are speedy and scalable. It is straightforward

to develop and build a backend system that is adapted to the specific needs of a babysitter app because to its large library of modules and packages.

3.2 Design

This step involves gathering, analysing, identifying, compiling, and evaluating the requirements and functionality for the application. It is necessary to locate the application's essential elements, such as new user registration, logging in as a registered user, identifying the pick-up and delivery points for current users, figuring out the delivery cost, processing payments, and delivery tracking.
Based on these requirements we made a system design that covers the structure, interface for users, and database structure based on the specifications. Make a design paper that specifies the system's technical specifications. We need to create the system with the aid of essential platforms and technologies, including Xcode and Firebase (for backend services). Design the user interfaces, combine the back-end services, and put the system logic into practice. After implementation we have to run tests on our application. The platform's construction utilising Swift, Firebase, and other pertinent resources and structures is the main emphasis of the implementation phase. Firebase is used to create the backend services and offers features like cloud computing, database storage, and authentication.
Firebase is a popular backend-as-a-service (BaaS) platform Firebase offers numerous services for mobile and web applications, such as cloud storage, real-time databases, verification, and more. There are a number of benefits of utilising Firebase in iOS projects.

Numerous advantages of using Firebase in OS projects include simple integration, scalability, real-time updates, strong authentication
This phase involves designing a technological strategy for the item's delivery application as part of the design process. This comprises the database schema, user interface, and architecture. The system's backend services, APIs (Application Programming Interface), and user interface components are all

described in the structure of the system plan along with how they interact. To assure accessibility and ease of use, the user interface layout should adhere to best practices and guidelines for design. The data model, relationships between entities, and data storage techniques should all be specified in the database architecture.

The flow and architecture of the entire project included dividing the model into three sections. The parents/ Customer panel, The Nanny panel and the Admin Panel.
It was important to analyse the needs and requirements of each section and create front end along with functionalities of each module accordingly.

Parents Panel

The customer panel is designed for parents who are looking for a babysitter. It allows them to search and filter through available babysitters, view their profiles, and book a sitter for a specific date and time.
The customer panel also enables parents to view their past bookings, rate and review babysitters, and make payments for their bookings. Figure 3.1 shows the entire path of the application for it’s customers after the home screen.

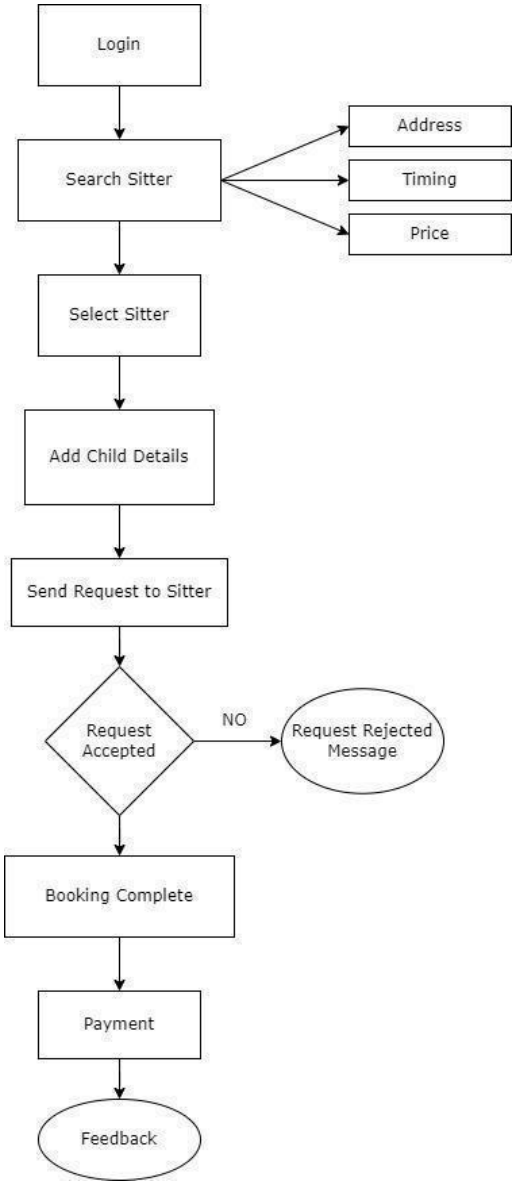


Fig 3.1 Parents panel flow of the application

There are several modules and pages in this project starting from the Launch Screen. We have many, numerous screens or "views" that collectively make up the application's user interface. Each screen serves as an individual component of the mobile application and has various features and functionalities. The parents panel included the following screens.

S. No.	**Screen Title**	**Functionality**
1.	launch Screen	This is the launch screen of the app i.e.

		the initial screen that is displayed when the app is opened by the user.
2.	Welcome Screen	This screen welcomes the user to the app and gives a brief introduction to the product.
2.	Login Screen	In this screen the user needs to enter the registered mobile number and press submit and he will move to the next screen.
3.	Register Screen	If the user is not registered (new user), he has to enter a mobile number to get himself registered.
4.	Verification Screen	In this screen there is a user input of 6 digits in which the user needs to enter the OTP which he got via SMS on the entered mobile number.
5.	Home Screen	This screen lets the user select whether he/she wants to look for a nanny, or babysit. This screen navigates which panel the user will be navigated to.
6.	Location Screen	This screen enables the user to enter the location where they need babysitting services at.
7.	User Profile Screen	This screen gives a personalised information of the account holder and helps user find their account information, notifications, reviews etc
8.	Job Card Screen	This screen allows the user to select their requirements and specifications like timings, price range etc
9.	Select Nanny Screen	This displays all the nannies near the user that have been filtered out by the previous requirements filled by the customer.
10.	Hiring Nanny Screen	Gives insight of the nanny selected, and lets the user hire or invite accordingly.
11.	Payment Screen	After the nanny is hired, the user is redirected to this screen to complete payment.

12.	Ratings and Reviews Screen	This is a feature screen of this application where the user can rate and review the nanny for his/her service.
13.	Notification Screen	This screen shows the notifications for the user.
14.	Edit Profile Screen	The user can edit his/ her personal profile and update his/her details using this screen.

Table 3.1 List of screens of Parent's panel

Nanny Panel

The nanny panel is designed for babysitters who are looking for work. It allows them to create their personal profiles, accept requests and display their qualifications along with several other functionalities.Figure 3.2 describes the flow of the nanny panel of the application.

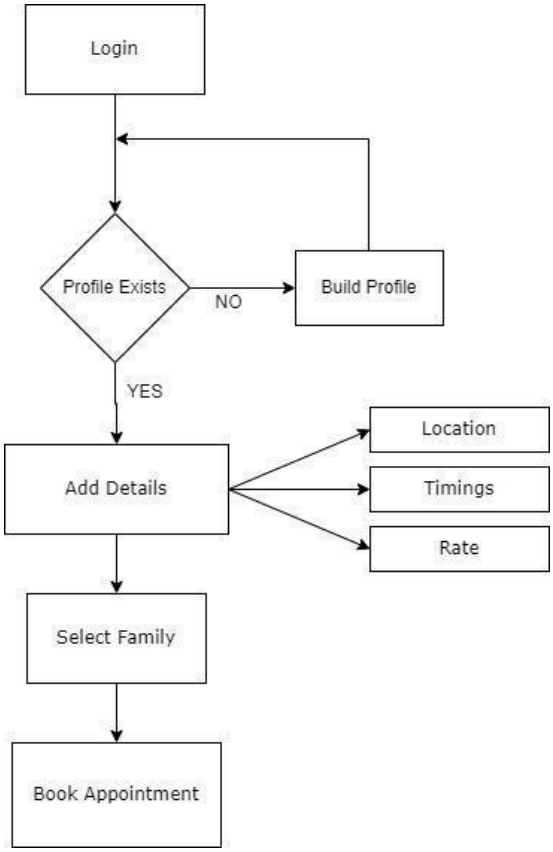


Fig 3.2 Baby sitter panel flow of the application

After getting redirected to the nanny panel from the home screen, table 3.2 lists the screens of the Nanny Panel.

Table 3.2 List of screens of nanny panel

S. No.	**Screen Title**	**Functionality**
1.	Location Screen	In this screen the nanny needs to enter the location where they can provide the babysitting services.
2.	User Profile Screen	This screen gives a personalised information of the account holder and helps user find their account information, notifications, reviews etc
2.	Job Card Screen	This screen allows the nanny to fill in their details like hourly rate, availability etc.
3.	Search Family Screen	Allows the nanny to search families which have registered accounts to provide long term services.
4.	Application Screen	Allows the nanny to apply for a family
5.	Requests Home Screen	Allows the nanny to get a list of the customers along with their details that have invited him/her.
6.	Notification Screen	In this screen the nanny can see all the notifications/ communications
7.	Ratings and Reviews Screen	This screen is for keeping track of the ratings and reviews the nanny has been awarded by previous clients.
8.	User Profile Screen	This screen is the personalised account holder screen
9.	Add account Screen	Let the nanny create an account and add her bank details to receive payments from clients.

10.	Edit Profile Screen	This screen enables the nanny to edit her public profile.
11.	Set Price Screen	Allows the nanny to fix a price which will be publicly displayed for his/her services.

Admin Panel

The admin panel is for application administrators who manage application operations. The admin panel also allows administrators to view analytics and generate reports on app usage, user behaviour, and revenue. Figure 3.3 displays the admin panel of the screen

The Admin Panel included the following screens:

Table 3.3 List of screens of admin panel

S. No.	**Screen Title**	**Functionality**
1.	Manage Users	Lets the admin manage both customers and nanny profiles.
2.	Manage Complaints	Keeps a track of the complaints issued by the customers.
2.	Payments	Keeps a record of all the payments made by the customers.
3.	Customer Support	Keeps track of customer support.
4.	Invoices	Keep a track of invoices.

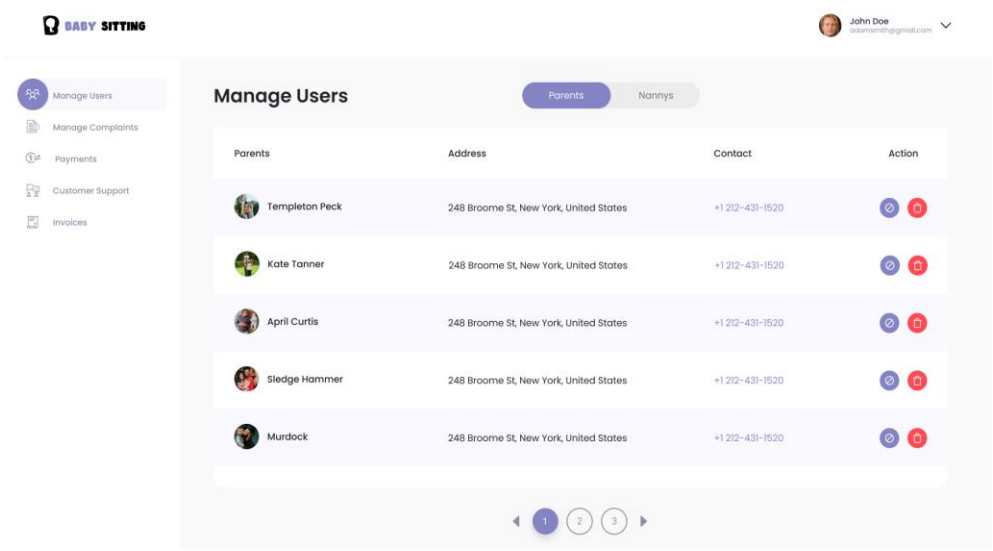


Fig 3.3 admin panel

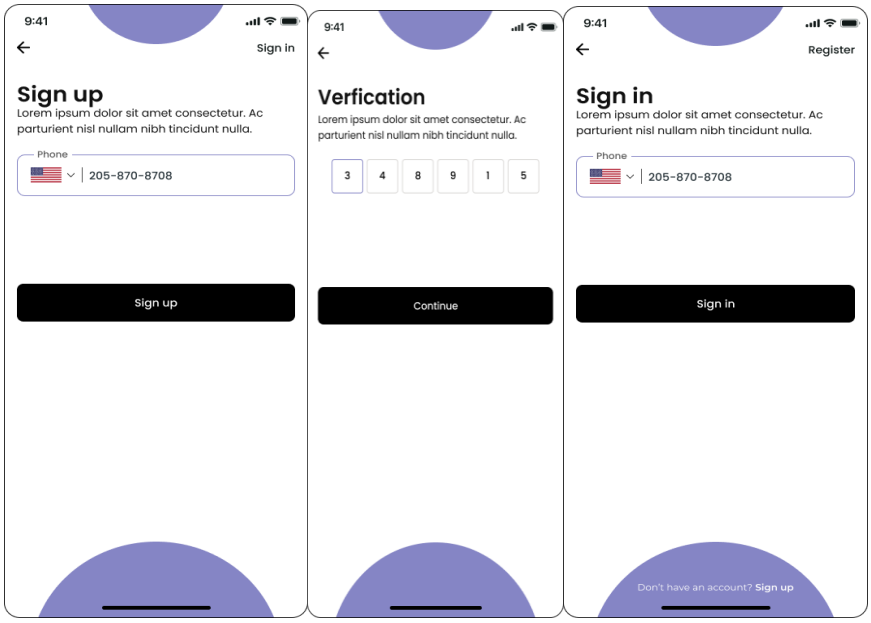


Fig 3.4 front end of sign up, verification and sign in screens

User authentication

It is a common functionality in all three panels. If the user is new to this application he needs to sign up and if an individual is already registered then he has to simply sign in. In both the cases the user can only validate himself with the mobile number. The user gets an OTP on their registered mobile number. Figure 3.5 is a diagrammatic representation of user authentication.

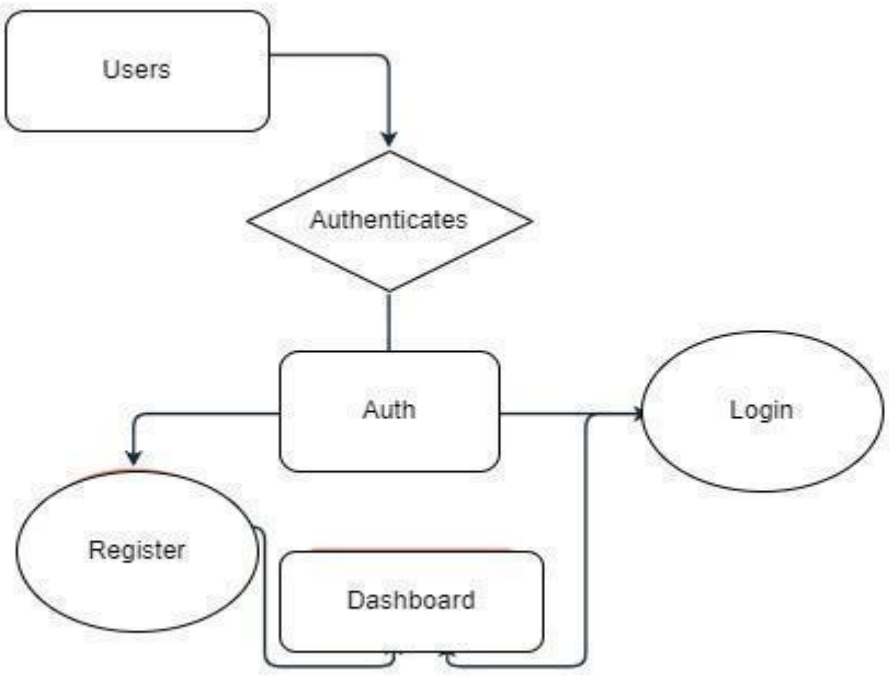


Fig.3.5 User Authentication

Customer Panel Path : Location, Pricing Selection

Moving further in this application after signing-in the user (customer) will get an option of filling out specification details. In this module of our application the user has to fill following details :

- Address
- City
- Timings
- Price Range

Fig 3.6 shows the flow of this process, along with the details selection screen and its consequent screen.

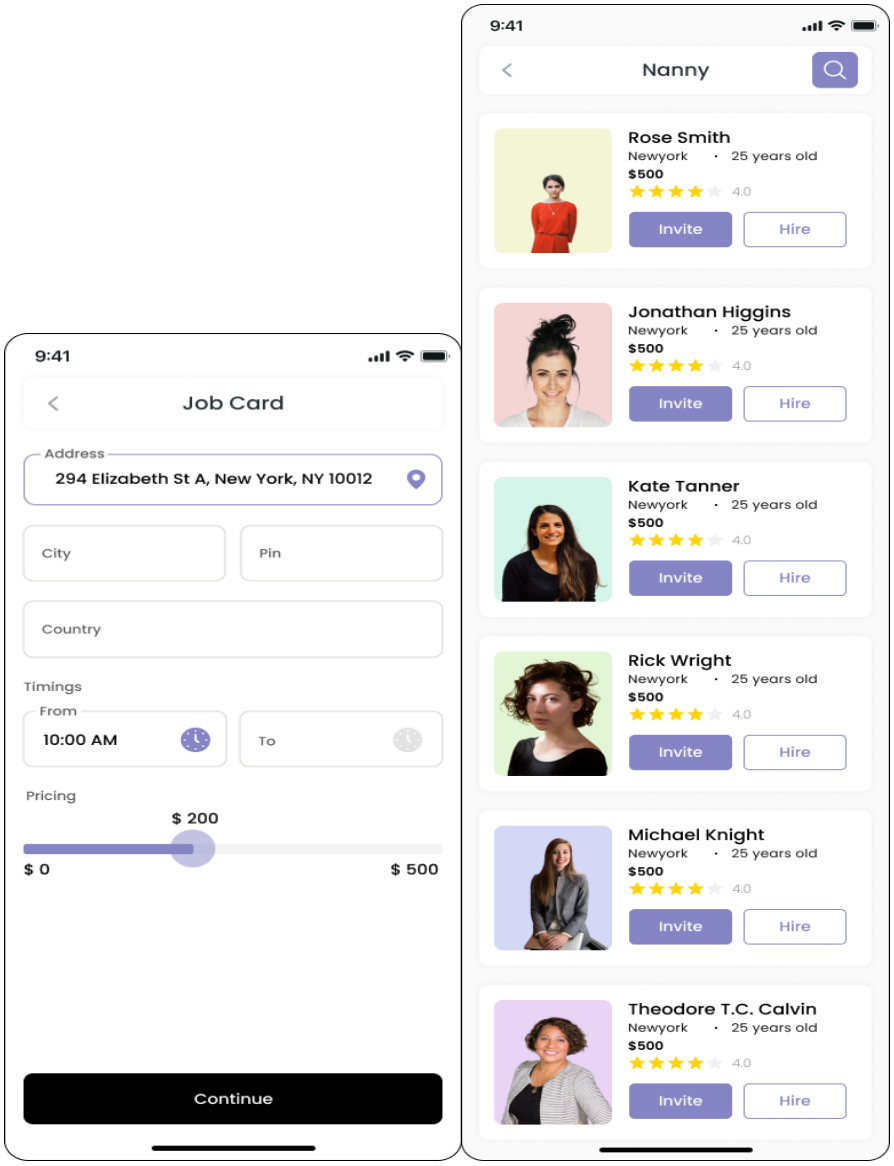


Fig 3.6 Specification requirement screens

After selecting the desired nanny, the user is directed to the make payment page. As the user confirms the source and destination points, he will be getting a detailed bill for the delivery and he will get a make payment page. The user has the functionality to add a new card to make payment. Fig 3.7 displays that functionality,

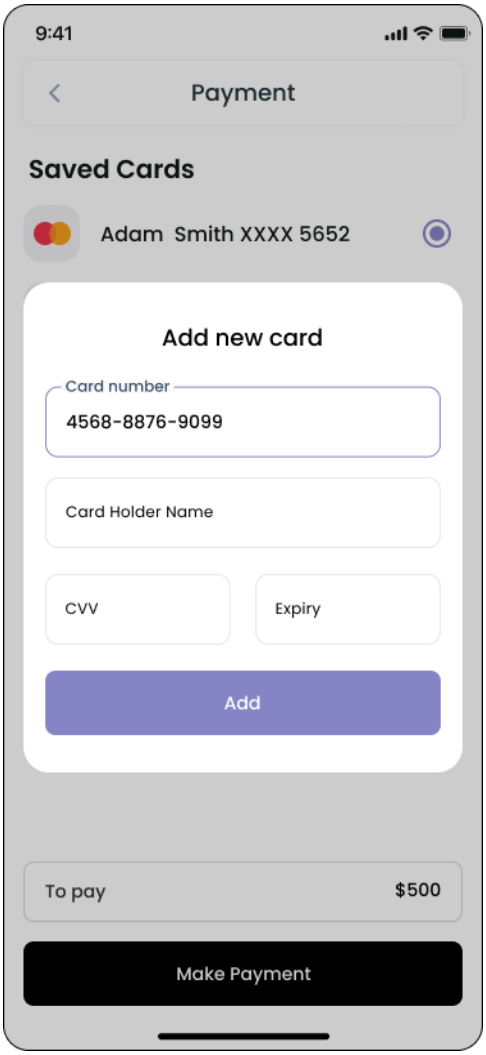


Fig 3.7 payment functionality in customer panel

The user will be choosing each time from which card he wants to make a payment, if he wants to add a new card, a dialogue box will appear in the middle of the screen, where he needs to fill in the card details. There is a proper validation done in the front-end, if the card details are correct or not. The card details whether the card exists or not, the card details match the existing card or not is done on the backend using Stripe, we will be discussing this technology in the further section of this chapter.

The user of this application will also get a number of features like, to see and edit profile, give rating and reviews to the Nanny, order history and notification. Fig 3.8 details the feedback functionality of the application.

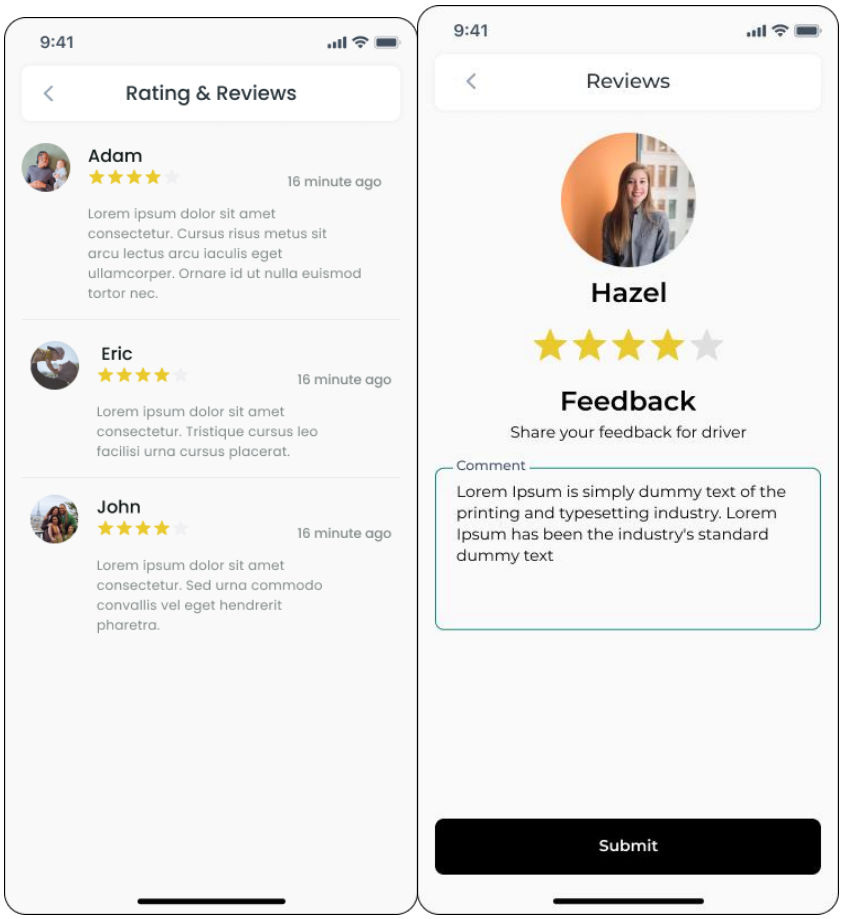


Fig 3.8 feedback functionality of the application

In terms of the path of the Nanny panel, after successful authentication, the user(nanny) will be directed to the location screen where she can set her location. Figure 3.9 shows the service provider's location screen.

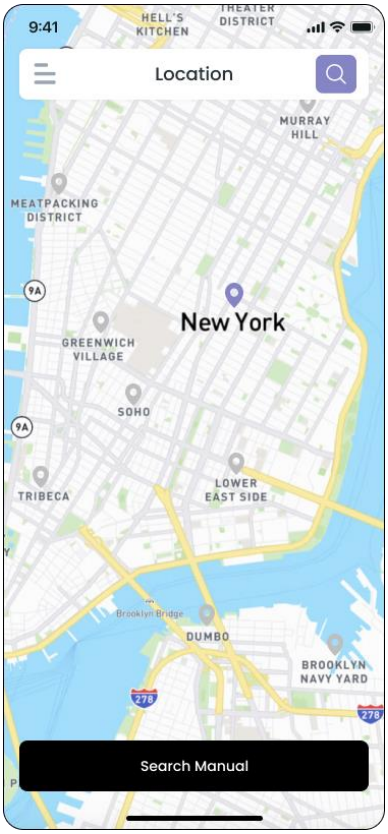


Fig 3.9 Nanny Panel Location Screen

This screen is succeeded by the Job card screen where the user fixes their job card specifications such as address, pricing, availability. This screen is important for the purpose of finding suitable clients for the nanny. Figure 3.10 displays the user interface of this screen. It contains several components including a sliding price range setter, timings of availability text fields among others such as city, country and pin-code.

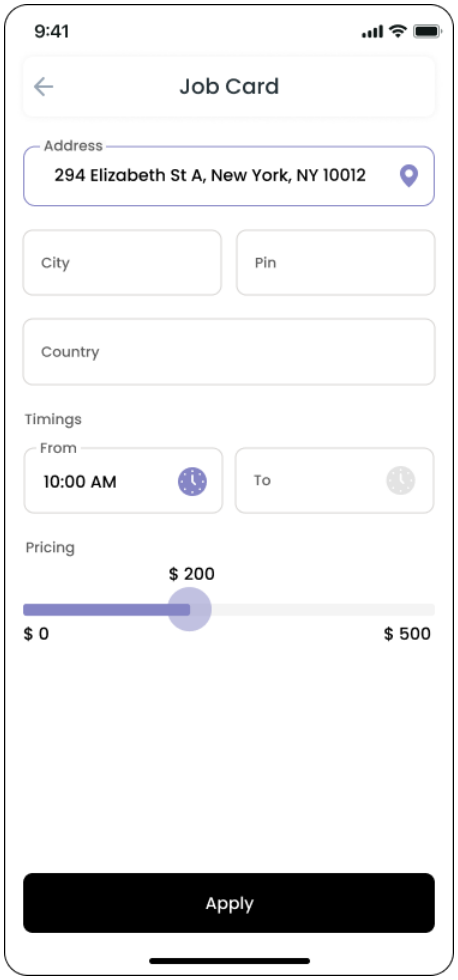


Fig 3.10 Job Card Screen in the nanny panel

After, clicking the apply button, the user is directed further to pages and screens that list the information of families which are potential clients. Based on the response of the nanny, the nanny is directed further in the app.

Figure 3.11 details the screens in which the nanny can apply for families near her locations and also accept or reject requests of clients.

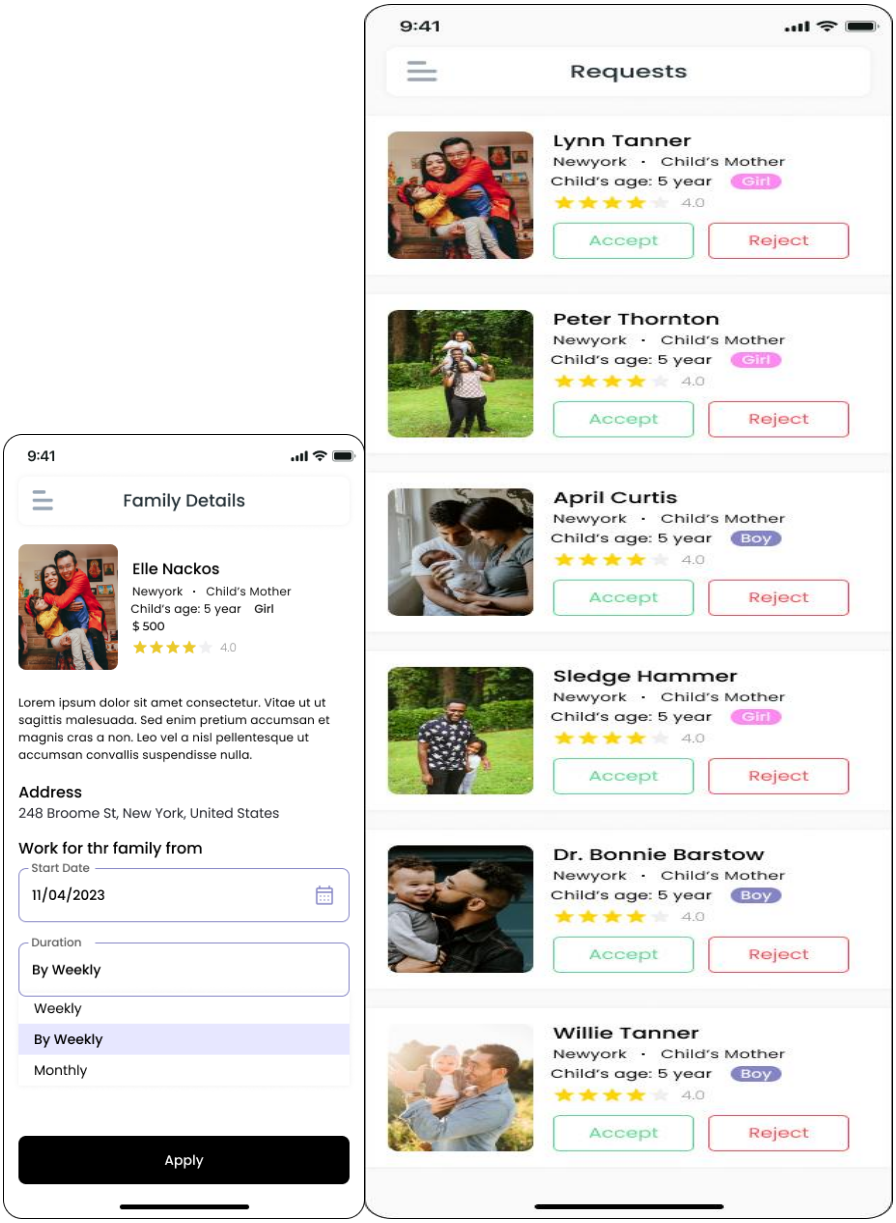


Fig 3.11 Screens that list requests of families as well as their details

The user can then add her bank account after accepting a request. This data is stored in the databases. Figure 3.12 are the payment screens in the nanny panel.

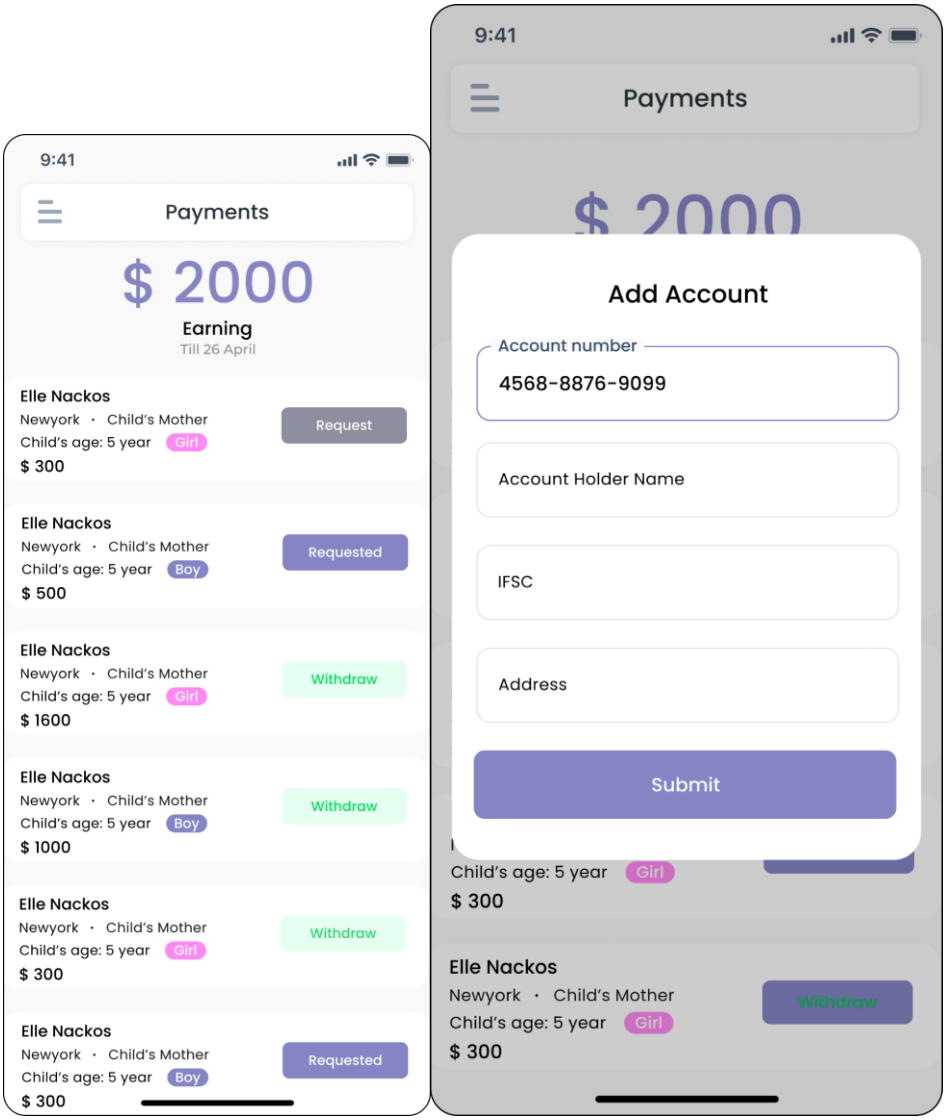


Fig 3.12 Add Account screen

The nanny can also navigate to her personal profile screen and edit her details. Fig 3.13 Shows the front - end of the Edit profile screen.

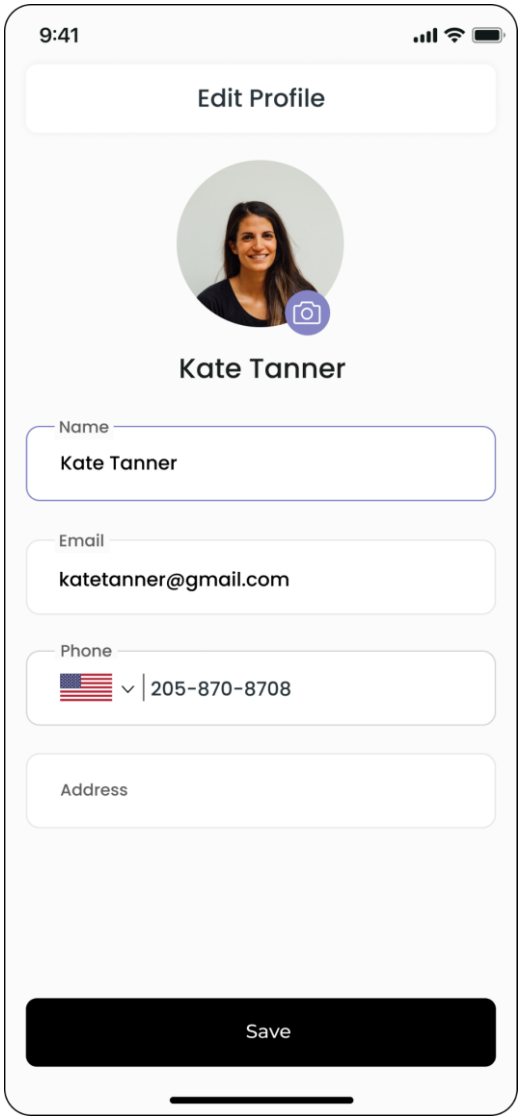


Fig 3.13 Edit Profile screen for nanny panel

3.3 Development

This section focuses on thorough discussion of the building process of our application. It also contains the tools and technologies used, as well as the challenges faced during the creation portion of the project on Xcode application. This contains the detailed information of the technologies used in this project.

- Swift
- Xcode
- CocoaPods
- Node.js
- Express.js
- Firebase

Now, we will discuss each and every technology in detail and how we have used it in our project from initialising our application to adding features to the application and use of these technologies in our application at various stages.

Xcode

Xcode[6]is an integrated development environment (IDE) created by Apple Inc. for developing software applications for macOS, iOS, iPadOS, watchOS, and tvOS platforms. It comes with a set of software development tools that give developers the ability to make, test, and upload programmes to the App Store. The intuitive user interface of Xcode, which makes it simple for developers to navigate and personalise their workplace, is one of its primary characteristics.
Xcode is the basis for development for all Apple devices. Xcode has a robust interface and allows the compilation of swift. It also contains a simulator which is used to simulate our application on different iOS devices. This includes devices ranging from iPhone 8,12,13,14 along with iPad devices.

Swift

Apple Inc. created the open-source, general-purpose programming language Swift[7]. Applications for iOS, macOS, watchOS, and tvOS are created using Swift. The syntax of Swift is simplistic and it is a compiled programming language. Swift is an updated version of objective-C. It is followed by Swift-UI. Swift is used along with storyBoard to create the front end of iOS applications. Swift is used in Xcode to make these iOS applications.

CocoaPods

CocoaPods[8] is a dependency manager for Swift and Objective-C Cocoa projects. CocoaPods can be initialised for our individual iOS projects. Once we initialise, a podfile gets created. This podfile holds all the libraries and modules required in the project. We must write commands such as 'pod install' after every time we edit the podfile. CocoaPods in the context of our application has simplified various tasks such as validation for credit card screens, UI libraries such as MDCtext fields. These are used to create floating labels and create custom text fields. CocoaPods can make sever other tasks such as OTP screen designing easy.

StoryBoard [9]

A storyboard is a graphic representation of an iOS application's front end User Interface. Each scene in a storyboard represents a view controller and its

associated views. Scenes are connected by segue objects, which represent a change in control between two view controllers.

By placing views like buttons, tables, and text views onto scenes, you may layout and design your application's user interface using the visual editor for storyboards provided by Xcode. A storyboard also gives you the ability to govern the data transmission between view controllers and to link a view to its controller object. It is advised to utilise storyboards while creating your application's user interface since they let you see how it will look and function all on one canvas. Figure 3.14 shows the StoryBoard interface.

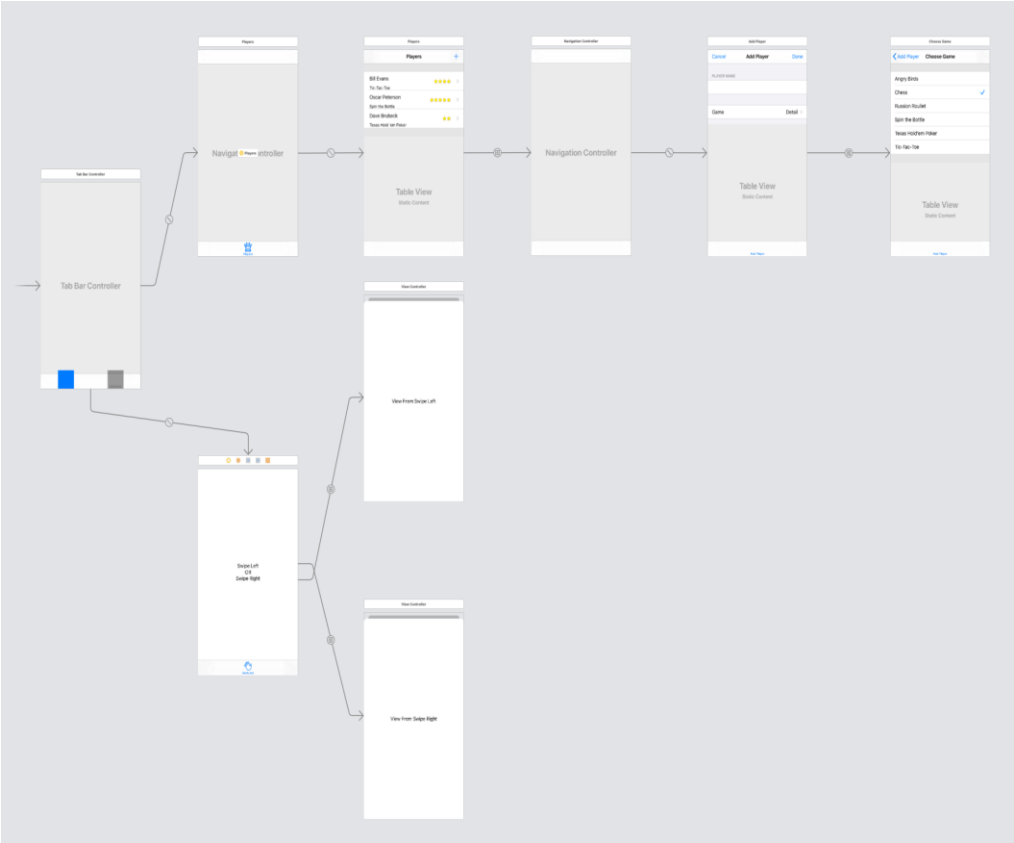


Fig 3.14 StoryBoard architecture

On the iPhone, each scene corresponds to the entirety of the screen; on the iPad, however, numerous scenes can display simultaneously, for instance when utilising popover view controllers. Every scene has a dock that shows symbols

for the scene's top-level elements. The dock is generally used to connect the view controller's views to its outlets and actions.
To complete initialising a view controller loaded from a storyboard, you override awakeFromNib as you would with all objects imported from a storyboard.

Node.js[10]

The open-source, cross-platform server environment Node.js can be used with MacOS, Windows, Linux, and more. Node.js is a JavaScript back-end language which runs within the V8 JavaScript Engine runtime environment.
JavaScript code is not contained within a web browser. JavaScript can be used by developers to create command-line tools with Node.js. scripting on a server. JavaScript code execution on the server is frequently done before the page is given to the browser, to create dynamic web page content online user's browser. As a result, Node.js is a "JavaScript everywhere" paradigm, coordinating the creation of web applications by employing a single programming language rather than utilising multiple Client-side versus server-side programming.
Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimise throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).
The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.

Node.js has been used for building the complete back-end of this project. The features of Node.js for choosing it over other technologies are as follows:

- **JavaScript Runtime:** Using the framework of Node programmers are able to execute the code that uses JavaScript independently of a web

browser. As a result, programmers can apply JavaScript to create the server-side portion apps, command-line tools, and other kinds of software.

- **Event-Driven Architecture:** Node.js's event-driven, autonomous I/O approach enables it to manage many concurrent connections without delaying other requests. Because of this, Node.js is very extensible and ideally suited for creating applications that operate in real time, such as chat, gaming, and other kinds of apps that need data to be processed immediately.
- **Support for Multiple Databases:** Both relational and non-relational databases are supported by Node.js. This makes it simple to select the ideal database for your application in accordance with your unique demands and specifications.
- **Simple to Learn:** For programmers who are already comfortable with JavaScript, for them Node.js is rather simple to learn and implement. This implies that developers won't need to learn an entirely novel programming language or environment in order to get started immediately developing server-side apps leveraging Node.js.
- Node.js was developed on top of the V8 JavaScript engine, that has been substantially optimised for speed and is compact. Node.js is hence able to operate quickly and consume less of the system's resources compared to other server-side platforms. Asynchronous programming, or asynchronous programming, is another feature of Node.js that enables programmers to create quick-running code.

- **Wide-ranging Modules:** Wide range of modules and packages are created and maintained by a huge and active community: Node.js has a tremendous and engaged community of contributors. To offer additional capabilities and features, these extensions may be readily incorporated into applications written in Node.js.

Express.js

Express.js[11] acts as a rapid, lightweight Node.js web framework that has several useful capabilities for building APIs and online applications. It is based on Node.js and provides a straightforward API, simplifying the process of developing web and mobile apps.

The server-side functions offered by Express.js may be utilised to carry out a variety of activities, including processing requests that come in, authorising users, and managing exceptions. Complex request-response flows may be made by connecting these services provided by middleware in a pipeline. Figure 3.5 shows the client-server architecture.

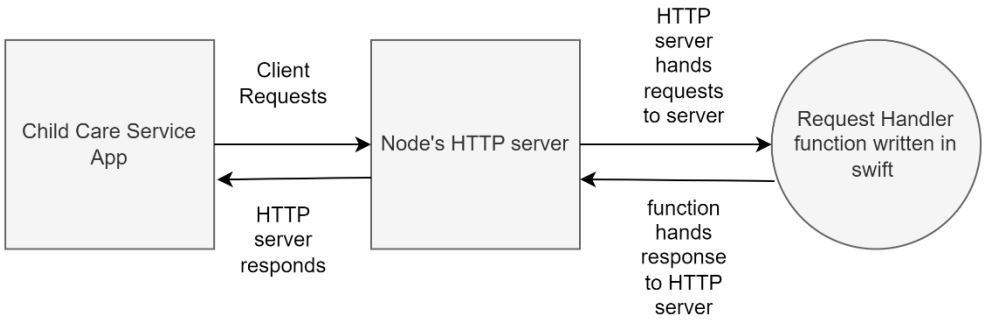


Fig. 3.15 Flow of requests and response from client to server

Express.js is frequently used to create APIs, or interfaces for application programming, which enable interaction between different parts of a programme. A common method for software to communicate data and services is provided by an API, which facilitates the development of sophisticated software platforms. Express.js offers a straightforward and understandable API that makes creating APIs simple. Developers may create functions that handle GET, POST, PUT, and DELETE requests by defining routes that correlate to the various HTTP methods. A variety of intermediate operations, including processing receiving JSON data, authorising users, and managing errors, are available in Express.js. A RESTful API, which is an architectural framework for developing APIs that adheres to a set of restrictions, may also be built using Express.js. RESTful APIs modify resources (like data objects) while offering replies in an accepted format (like JSON) via HTTP methods.

Security and scalability are crucial factors to take into account while developing an API with Express.js. Data in transit may be made more secure with the support for SSL/TLS encryption that Express.js offers. Additionally, it encourages rate limiting along with other attack-prevention strategies. Benefits of using Express.js for building APIs in iOS apps:

- **Scalability:** Express.js is very scalable and can handle many concurrent connections without experiencing any lag. It can be set up on a group of servers or a platform that uses the cloud, like AWS or Google Cloud.
- **Security:** Express.js supports SSL/TLS encryption, which helps to protect data while it is being transmitted. Additionally, it encourages limitation of rate and other attack-prevention strategies.
- **Large Community:** Substantial and prominent developer community: Express.js has a sizable and active developer ecosystem that builds and maintains a variety of modules and libraries that are readily incorporated into Express.js programmes to bring additional features and possibilities. Additionally, the platform has excellent documentation that makes it simple for developers to comprehend and use.

Node has several advantages from the perspective of web server development, including:

Excellent performance: Node is a solid answer for many typical issues with web development (such real-time online applications), as it was created to optimise throughput and scalability in web applications.
Because "plain old JavaScript" is used to write code, dealing with "context shift" between languages when developing both client-side and server-side code takes up less time.

When compared to other conventional web-server languages (such as Python, PHP, etc.), JavaScript is a relatively young computer language that benefits from advancements in language design. We can also use TypeScript,

CoffeeScript, ClojureScript, Scala, LiveScript, and many more modern and widely used languages because they compile or convert to JavaScript.

Numerous reusable packages are accessible through the node package manager (npm). Additionally, the majority of the build toolchain can be automated using it, and it features best-in-class dependency resolution.

Node.js is transportable. Microsoft Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS, and NonStop OS are all supported. It is also highly supported by a large number of web hosts, who frequently offer specialised infrastructure and documentation for hosting Node sites.
It has a thriving development community and third party ecosystem with tonnes of helpful people.

Express.js provides techniques for:

- Write handlers to reply to requests on distinct URL paths (routes) using different HTTP verbs.
- Create responses by filling up templates by integrating with "view" rendering engines.
- Set up common web application parameters like the connection port to use and the location of templates used for displaying responses.
- Add more "middleware" to process requests at any point in the pipeline for handling requests.

Express is a relatively straightforward framework, but developers have created middleware packages that integrate with it to address almost any web development challenge. There are libraries that may be used to work with cookies, sessions, user logins, URL parameters, POST data, security headers, and many other things.

Google Firebase[12]

A powerful and feature-rich development platform is more vital than anything else when it comes to producing reliable and high-quality mobile applications.

One such platform that has gained the favour of developers all across the world is Firebase from the Google stable.

Google has created a platform called Firebase that offers an array of tools for developing and expanding mobile and web applications. Due to this it provides a variety of services and tools that can enable development and quicken time-to-market, it is a well-liked option for iOS developers. Fig 3.16 shows the list of features provided by the firebase in the built section.

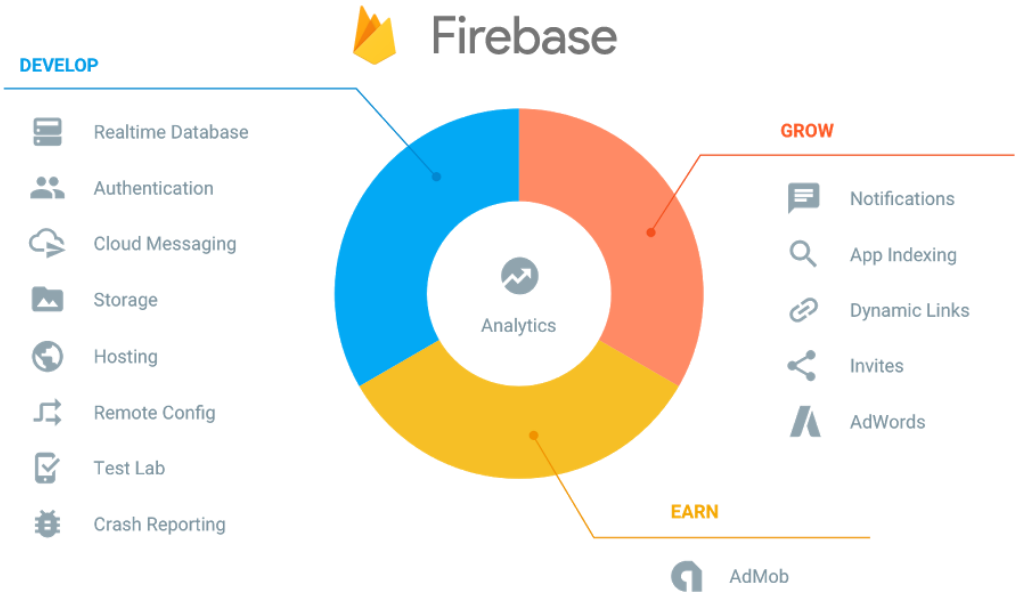


Fig 3.16 List of Firebase features

Key details regarding Firebase for iOS applications is provided below:

1. One choice is to use Firebase Realtime Database, a cloud-hosted database, to rapidly store and sync data. It offers a NoSQL database, making data management and upkeep simple in a flexible, scalable

approach. Applications that respond immediately to data changes can be built using the Firebase Live Database, increasing their interactivity and user appeal.

2. User-generated material with a large size, such photographs and videos, may be easily stored and served with Firebase cloud-based storage. It allows for both local and remote storage, allowing access to material from any location in the globe. For a full storage solution for your iOS application, Firebase Cloud Storage interfaces with other Firebase services like Authorization and Dynamic Database.

3. Google Firebase also provides a feature called FCM (Firebase cloud messaging). Delivering messages to clients on Android, iOS, and the web is made possible by the dependable and scalable Firebase cloud-based messaging service. It offers a variety of capabilities, including targeting, planning, and statistical analysis, making it simple to give people timely and relevant communications. It is a popular option for iOS due to its simplicity of use, scalability, and interaction with other Google services.

Stripe for Payment

A service for handling payments called Stripe makes it possible for companies of any kind to receive and handle payments online. It delivers a variety of services and applications that enable businesses to make payments easily and effectively. Many companies, which include entrepreneurs, small enterprises, and large corporations, use Stripe on a global scale. Stripe works for all kinds of payments starting from card payment to UPI payments, card payments also includes all kinds of cards, i.e. master-card, visa-card, in debit card and credit card options. Some key features of Stripe are given below:

- **Payment methods :** Online payment processing is made simple and safe by Stripe. Numerous payment options, such as debit and credit card transactions, and payments via mobile devices are supported. All aspects of payment processing, such as identifying fraudulent transactions, reimbursements, and currency conversions, are handled by Stripe. Additionally, it supports memberships and periodic payments, making it simple for businesses to handle customers' payments over time.

- **Security :** Stripe prioritises safety and delivers a variety of tools to support organisations in keeping their transactions safe. The greatest level of validation accessible to payment processors is the PCI Level 1 Service Provider accreditation that it possesses.
- Additionally, Stripe's platform offers fraud identification and avoidance tools like multiple-factor authentication, real-time risk assessment, and machine learning algorithms.

- **Global Reach :** With around 135 recognised denominations and over 40 operating nations, Stripe makes it simple for business establishments to take payments through clients all around the globe. International payments are handled entirely by Stripe, involving conversions of currencies and observance of regional laws.

- **Billing :** A set of software programmes called invoicing makes it simple for companies to handle membership and payments that recur. It offers capabilities for creating price structures, maintaining users and payments, and responding to unsuccessful payments. To offer a complete billing solution, Stripe Billing connects with other Stripe services like Payment Processing and Hawkeye.

Organisations can handle and accept payments via the internet with ease thanks to a variety of products and services offered by Stripe. Stripe offers a complete platform for managing payments in your iOS application with payment

processing, developer tools, security, global reach, billing purposes, and detection of fraudulent transactions solutions. Businesses across all kinds choose it because of its versatility, protection, and simplicity of use.

Chapter - 4
Experiments and Result Analysis

When we tested the b app on dummy data, we tried to maintain the environment as close to the real world as possible. In this section, we'll outline some steps for testing the babysitting app on dummy data.

Setting up the test environment:

We created a separate test environment that mirrored the actual app. This included creating an admin panel with dummy data. This was done by manually entering test data into the system. To prevent any potential data corruption or other concerns, it's crucial to ensure the test environment is entirely distinct from the live app.

Testing the user interface:

We need to check the User Interface to make sure that all buttons, tables, constraints are displaying and functioning correctly. We need to verify that the UI is consistent across all screens and devices. To achieve this, we test-run the app on different Apple devices such as iPhone12, iPhone 13, iPad etc.

Testing user authentication and authorization:

Another key aspect of testing the babysitting app on dummy data was to ensure that user authentication and authorization worked correctly. In the testing phase, we made several dummy accounts both of the parents/customers as well as babysitters. We tested this on our database in firebase. We needed to make sure that the app was performing tasks like signing-in, signing-up and OTP verification without any errors.

Testing data entry and retrieval:

The admin panel is responsible for managing all the data within the app, including babysitters, parents, and job listings. We verified that all data could be entered and retrieved correctly. Several queries are performed to make sure data entry and retrieval is possible in the back-end. It was also important to make sure validation was being performed correctly in several pages such as Signing-In, Signing-up, Adding credit card details etc.

Testing system performance:

Finally, it's important to test the performance of the system under different loads. In the future phase of testing, our app will be tested to see if it can withstand high volumes of traffic and user activity. It is important to test the app for stability. Load-balancing and Caching are some techniques to maintain the stability of the app. Monitoring of system resources like memory and CPU usage was done to ensure that the app could scale appropriately. Several CocoaPods files that were not required were deleted in order to optimise space. It is important that the app does not take too much space and memory.

Several changes were made along the process of development of the app. Experimenting with different User Interface designs, testing with different types of dummy data and running the app on multiple iOS devices of different screen constraints was done. After multiple rounds of quality testing and checks, a conclusive flow of the application architecture was formed.
API integration was completed and the app was found compatible with all iOS devices. The app was simulated and was tested on iPhone 14 pro, iPhone 14 plus, iPhone 13, iPhone 13 pro, iPhone 8 plus, iPhone SE (1st Generation), iPhone SE (2nd Generation) and iPhone 11 pro Max.
The app performed well on all devices with minor adjustments.
Fig 4.1 displays the testing of the app on iPhone 13 and iPhone 14 devices.

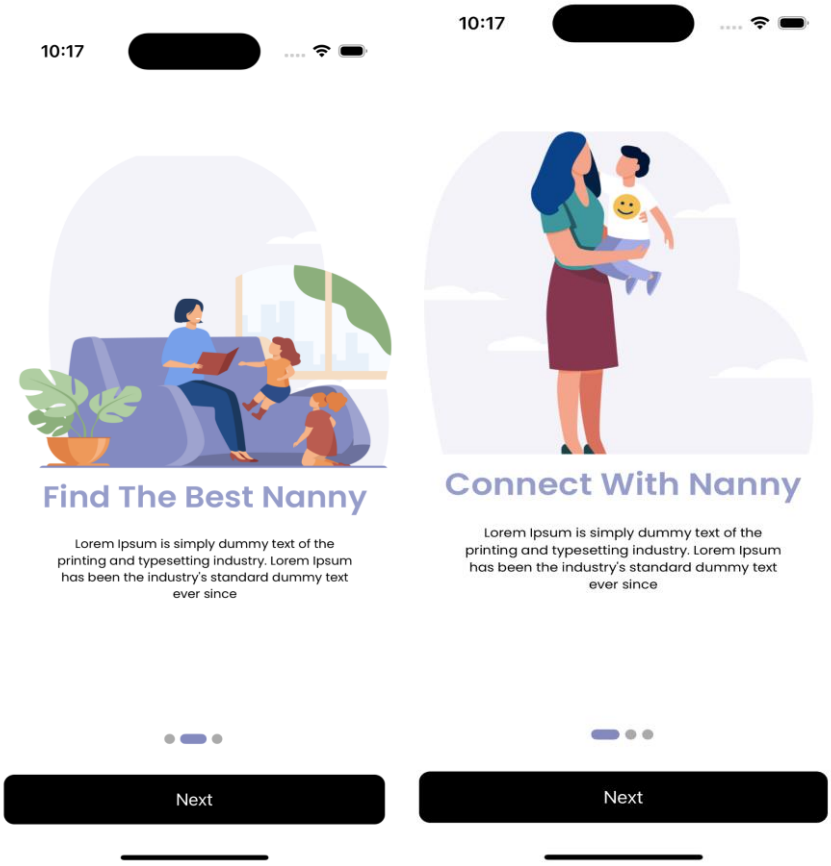


Fig 4.1 Running the app on iphone 13 and iphone 14

Testing the app on more dummy data, and multiple nanny profiles were created.

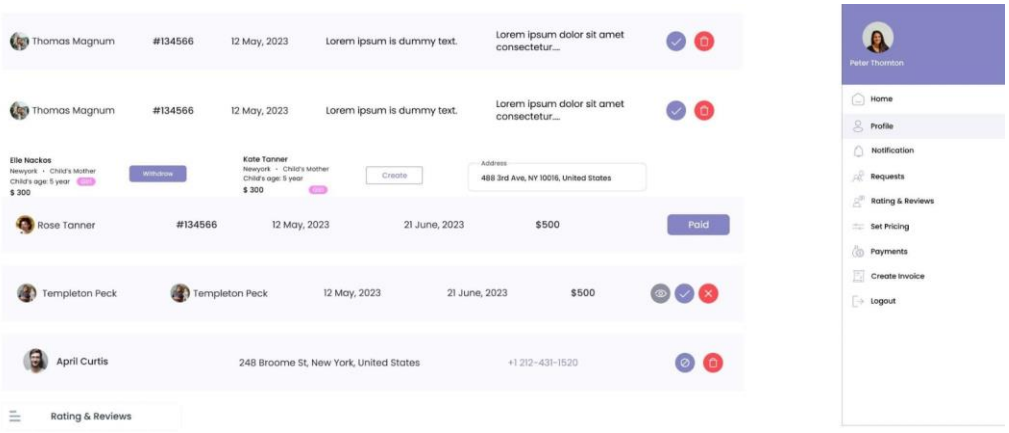


Fig 4.2 Testing data for nanny profiles

Testing the payment screens on dummy data in the admin panel for a babysitting app server is an essential part of ensuring the app's overall functionality. It is important to simulate different payment scenarios, such as different payment methods, currency types, and amounts, to ensure that all payment functions are working correctly. Verifying that all payment information is accurately captured and saved in the app's database should also be part of testing.

Before the app is made available to users, any potential payment-related problems can be found and fixed by thoroughly testing the payment screens using dummy data in the admin panel. This guarantees that both parents and babysitters using the app will have a seamless and secure payment experience. Figure 4.4 details the testing of this payment functionality.

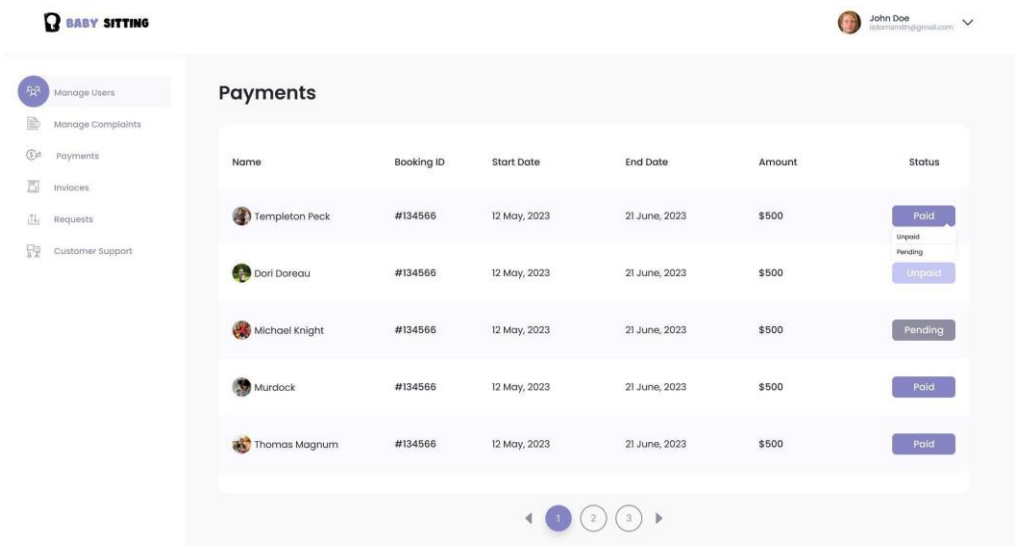


Fig 4.3 Payment functionality testing

Simulating various support scenarios, such as evaluating the capacity to submit support tickets, the response of the support team, and the efficacy of the remedies offered, should be part of the testing of customer support capability. The capacity to track support tickets and keep track of response times should also be covered during testing. Before the app is made available to customers, any potential problems can be found and fixed by extensively testing the customer support capability on fictitious data in the admin panel. This helps to guarantee a positive user experience and high levels of customer satisfaction.

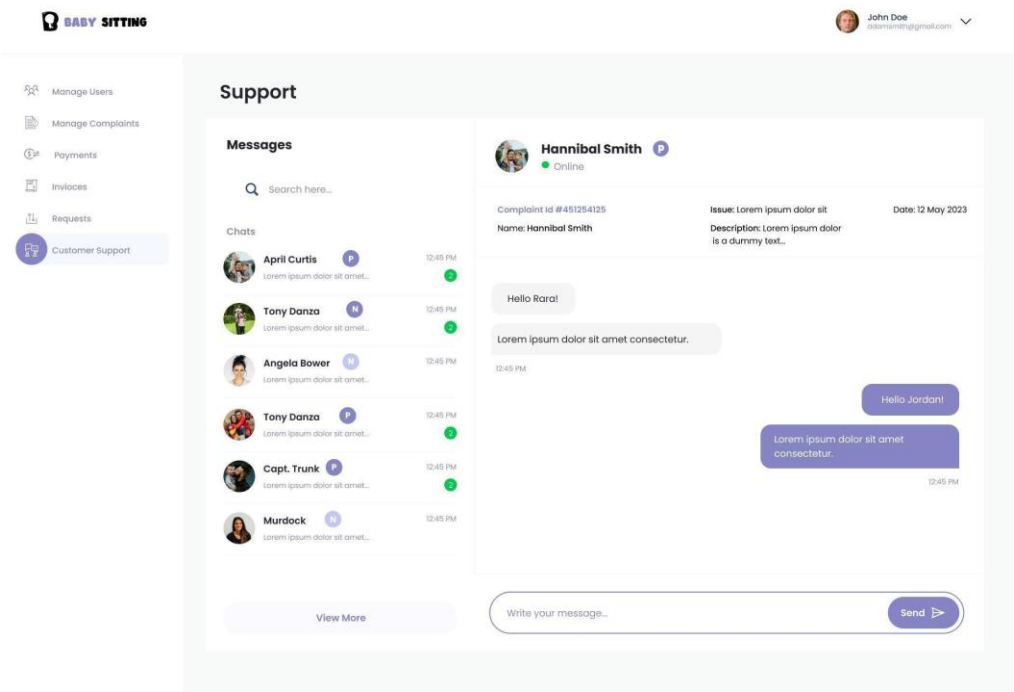


Fig 4.4 Customer Support for admin

Complaint Management functionality is a crucial aspect of any service providing application. Fig 4.6 shows the testing of complaint management on dummy data

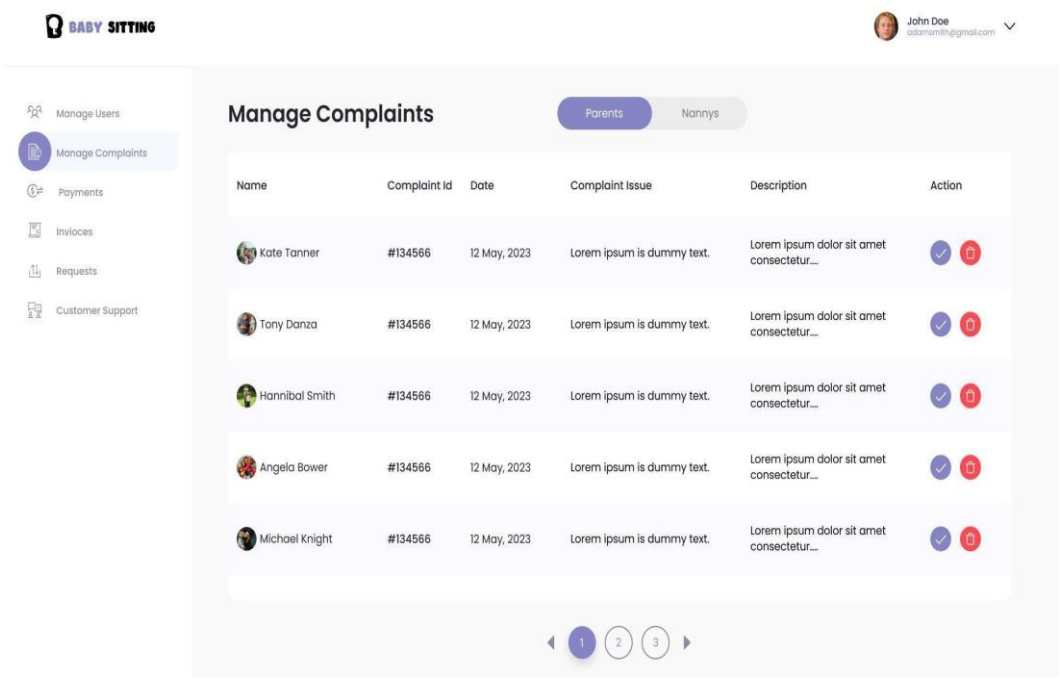


Fig 4.5 Customer Support in admin panel

A babysitting app server's admin panel manages users screens should be tested to make sure all user management features are functioning properly. This involves testing user-adding, user-profile-editing, and user-delete functions. The testing should also include the different roles and permissions that users, such babysitters, parents, and administrators, can have. Before the app is made available to users, any potential problems can be found and fixed by extensively testing the managed users screens in the admin panel. This will guarantee a seamless and obstacle-free experience for all app users.

Figure 4.8 shows the corresponding screen.

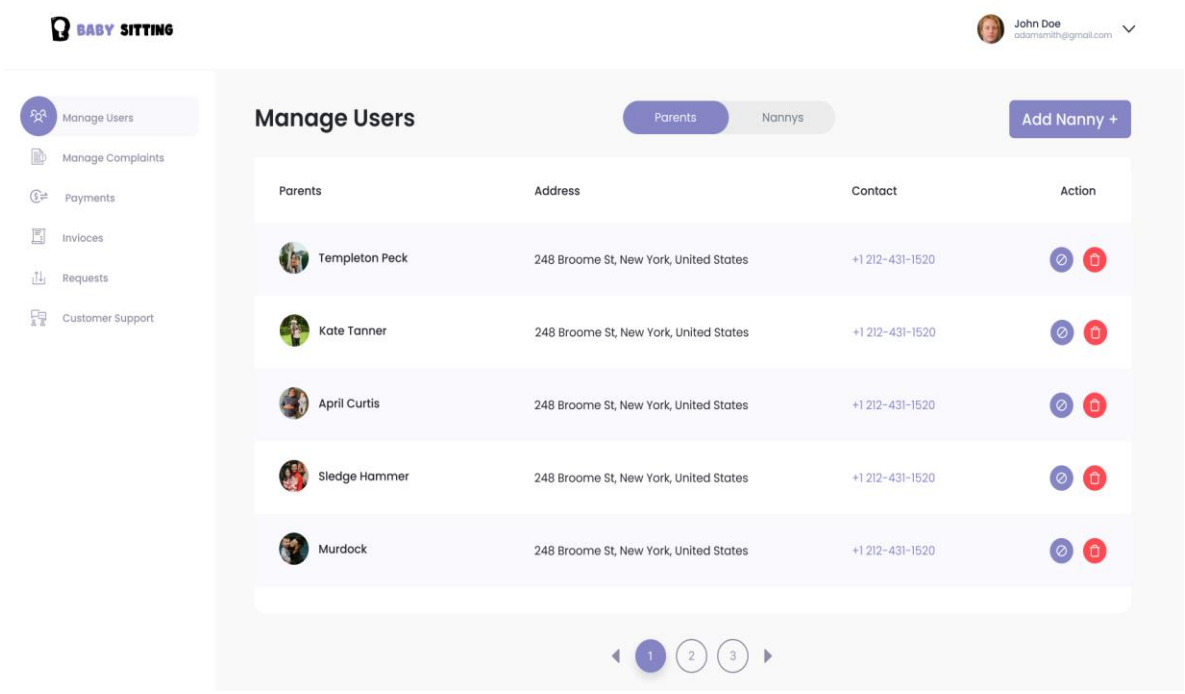


Fig 4.6 Testing dummy data in Admin Panel

Chapter - 5
Conclusion

5.1 Conclusion

In conclusion, a babysitting app can be a great tool for parents and babysitters alike. As more and more Services shift online, the Baby-Sitting app holds great economic value in future markets. It's important to build an app that can sustain a good design flow. Since applications have a high churn rate it is important to have a good user Interface which matches the brand- value. Moreover, the User Interface should be clear for the user to understand. It is important for the user to learn the User Interface of the application. For this, User engagement time should be high. This can only be achieved if the app's interface is intuitive and easy to understand. Design consistency is also needed across typography, buttons, icons, and other branding components.

A fast and responsive loading time is important in a good app, and leads to better user experiences, higher user retention and high user engagement.

A good app must not take more than 5 seconds to load. It is important to minimise the loading as much as possible. An app should be stable and should not crash. It should also have a good and fast content delivery network (CDN).

Another very important feature that needs to be taken care of is an application's security. Since the BabySitting app has the potential to hold large quantities of private data such as location, address of users etc, There are some practices that are required to be implemented such that the app's security can be taken care of:

- creating secure code that is simple to patch and upgrade.
- Using code signing and code hardening.
- all data being encrypted.
- use only approved APIs.
- establishing various session expiration periods for token validation.

- multi-factor authentication being necessary.
- investing in penetration testing and threat modelling to find app vulnerabilities.

Regular security testing is necessary to identify vulnerabilities and assess any gaps in your data protection because new threats are constantly developing. Sensitive information will remain secure if these problems are resolved quickly, and doing so will also boost consumer loyalty and brand trust.
Another key- component in making a good application is built-in user support. Since an application's performance is highly dependant on the user's retention, It is important to increase a user's comfort. Due to this, an application must hold excellent user support. This can be achieved through features such as AI – driven chatbots and other AI driven customer support services. Proving a helpdesk or an FAQ section is also useful for a smooth experience.

Other connectivity capabilities, like as in-app messaging or customer service integration, can hasten the feedback process, enhance internal collaboration, and assist teams in finding quicker solutions to business issues.

5.2 Future Scope

As societies become increasingly advanced, the market for at-home services continues to expand. There are numerous ways to upgrade and improve this app. Here are some potential future scope points the app:

1. Make the app cross-platform: The current implementation of this application is only in swift and hence this application can only be run in Apple devices. As the scope of this project expands, it is crucial to make the application. This can be achieved by developing the app using React Native.

2. Adding an In-app video conferencing feature: A useful functionality that can be added in the application can be of in-app video calls such that the parents can contact and see the babysitter and their child at anytime.

3. AI-powered matching: Deep Learning and Machine Learning Algorithms can make the process of matching of nanny and the client much more optimised at the backend. This is especially beneficial for large amounts of data and cases of high traffic in the app.

4. Integration with smart home devices: A potential additional feature of the Baby-Sitting app can be the integration of smart devices like cameras, sensors etc. This can help the parents to keep a better watch.

5. Chat services: Chat functionality can be added into the app so that the parents can chat with the babysitter beforehand. This can help maintain privacy for the babysitter who may want to keep their phone numbers private from the clients.

6. Virtual activities: As more activities move online, babysitting apps could offer virtual activities for children to do with their sitters, such as interactive games, educational activities, sleep-time stories.

7. Adding features to track children's health: Features that can track a child's academic performance or health can be useful for busy parents who use babysitters regularly or have employed full time nannies.

REFERENCES

[1] S.K. Meena, S. l. N. Reddy, B. Anand, M.P. Roy , “Mobile media uses and circumstances in which parents offer these devices to child: a cross sectional study in North India”, International Journal of contemporary paediatrics ,Vol 7, pp2312-2315 Dec 2020.

[2] Mrugank Gandhi, Shubham Kothavade, “Development of a iOS Mobile Application for online Babysitters,” IRJMETS [2022]

[3] Yogita Masare, Sneha Mahale, Manjusha Kele, Ashvin Upadhyay, Bhushan R. Nanwalkar, “Subscription-based babysitting service,” IJERT [2021]

[4] Vrushali C. Waikar, Sheetal Y. Thorat, A. A. Ghute, and Priya P. Rajput, Mahesh S. Shinde, “Connecting Families with Caregivers Application for Android Mobile Application,” IRJET [2020]

[5] Noof A. Al-Safi, Rawan A. Al-Asiri, Malak A. Al-Malki, Sameera Abar, “Fostering Childcare E-Service: Design and Development of a Software Application,” IJITN [2022]

[6] Wikipedia contributors, "Xcode," Wikipedia,TheFree Encyclopedia, https://en.wikipedia.org/w/index.php?title=Xcode&oldid=1156534824

[7] Wikipedia contributors. "Swift (programming language)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 May. 2023. Web. 23 May. 2023.

[8]https://developer.apple.com/library/archive/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html

[9] https://developer.apple.com/documentation/uikit/uistoryboard

[10] Wikipedia contributors, "Node.js," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Node.js&oldid=1155877079

[11] https://expressjs.com/

[12] Wikipedia contributors, "Firebase," Wikipedia ,TheFree Encyclopedia https://en.wikipedia.org/wiki/Firebase